



UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

**IN MEMORY DATA WAREHOUSING TO IMPROVE ON DATA
ANALYTICS LATENCY**

MBURU JOHN KAMAU

**A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS OF THE DEGREE OF MASTER OF SCIENCE IN COMPUTATIONAL
INTELLIGENCE OF THE UNIVERSITY OF NAIROBI.**

DECEMBER 2017.

Declaration

This project report is my original work and has not been presented for any other award in any university.

Signature: _____

Date: _____

MBURU JOHN KAMAU

P52/72897/2014

Project Supervisor

This project report has been submitted in partial fulfillment of requirements of the award of Master of Science degree in Computer Science of the School of computing and Informatics of the University of Nairobi, with my approval as the University supervisor.

Signature: _____

Date: _____

Ms. PAULINE WAMBUI

School of Computing and Informatics

Dedication

I dedicate this project to my family who were very supportive throughout the project for allowing me to stay up late to push through the tight timelines. Special thanks to my wife and kids Zahari, Shanice for being part of the motivation to this project thesis.

Acknowledgements

I would like to acknowledge Ms. Pauline Wambui and Dr. Elisha O. Opiyo for their great assistance and direction. Ms. Pauline's recommendations and suggestions led to a large extent, the realization of this project report.

Abstract

Most of the banks in the Kenyan banking industry are still using traditional reporting methods that involves querying data from different transactional databases that are not interconnected. Some business teams download reports from core transaction processing systems (TPS). The data is then subjected to tools such as Microsoft excel to analyze the raw data and use of various excel functions to report what the top business executives need. Every time these executives request for a report, the reporting team, if any, must put a request to the technical information technology teams to generate the raw data with the requested fields and forward it to possibly some shared folder where the reporting team can access in preparation for data analysis and reporting.

This is time consuming, prone to errors and it requires the availability of the technical teams to ensure the process is complete. Some of the results that are produced using such traditional methods are not accurate and are at times wrong. There is also no way to combine results from all systems to get a complete view of a concept (a customer for example). Use of the entire databases for reporting instead of some few attributes of the database in form of a modelled fact/dimension means it takes more time to process or mine data.

The research was aimed at establishing if this problem could be solved by replacing the traditional reporting mechanisms with a modern in-memory data warehousing prototype. This involved development of various data analytics components that created a new set of processes of data collection, cleaning/transformation and analysis. Apache Spark in-memory solution was used to store more summarized data by organizing data into dimension and measures for easier and faster retrieval. At the presentation layer, Apache Zeppelin was used to showcase data visualization.

Sampling method was based on purposive sampling technique by studying the way data analysis is done by a selected banking analytics professionals. The research was done using the case study approach where data was collected from the business reporting, technical and management teams that were involved in the reporting process. The prototype was evaluated to show what improvements were realized using the new in-memory system by using sample data that was large enough to compare with other options.

Table of Contents

Dedication.....	iii
Acknowledgements	iv
Abstract.....	v
Table of Contents.....	vi
Table of Figures.....	ix
Table of Tables	x
CHAPTER 1: INTRODUCTION	1
1.1 Background.....	1
1.2 Business intelligence and data analytics.....	2
1.3 Need for Emerging Analytics.....	2
1.4 Problem Statement.....	4
1.5 General Objective	5
1.6 Specific Objectives	5
1.7 Research Questions.....	5
1.8 Significance of the Study.....	5
1.9 Scope of the Study	5
1.10 Contributions.....	6
CHAPTER 2: LITERATURE REVIEW	7
2.1 Introduction.....	7
2.2 Business Intelligence.....	7
2.3 Business Intelligence Maturity Models	7
2.3.1 IT-Centric	7
2.3.2 Information Management.....	7
2.3.3 Predictive Insight.....	8
2.4 Business Intelligence Vs Decision Support	9
2.5 Steps to achieving a Business Intelligence System.....	9
2.5.1 Transaction processing systems as a BI data source.....	9
2.5.2 Data Warehouses.....	10
2.5.3 Extracting, Transforming and Loading (ETL) Data	10
2.5.4 Multidimensional Data Analyze.....	11
2.6 Review of Business Intelligence, Data warehousing in Banking	11
2.7 In-Memory Analytics	13
2.7.1 In-Memory Technological Advances	13

2.7.2	Benefits of In-Memory Analytics.....	13
2.7.3	Apache Spark as In-Memory Analytics Solution	14
2.7.4	Architecture of In-Memory Analytics Systems	14
2.8	Proposed Architecture for BI.....	15
2.9	Related work on the Problem of Latency	16
2.10	Technological Gap	16
2.11	Overall System Architecture/Model for a Business Intelligence System.....	17
CHAPTER 3: RESEARCH METHODOLOGY		18
3.1	Introduction.....	18
3.2	Research Design	18
3.3	Research Population.....	18
3.4	Sampling.....	18
3.5	Data Collection Methods	18
3.3.1	Prototype Data Sources.....	18
3.3.2	Research Data	19
3.6	Relating Research Objectives to Research Methodology.....	19
3.7	Data Analysis	21
3.4	Prototype Development Methodology	24
3.4.1	Planning phase	24
3.4.2	Discovering phase	28
3.5	Prototype Implementation and Evaluation	30
3.5.1	Configuration Phase.....	30
3.5.2	Data Validation and Simulation Phase.....	35
3.5.3	Customization Phase.....	37
3.5.4	Deployment and Evaluation	37
CHAPTER 4: RESULTS.....		39
4.1	Process Evaluation	39
4.1.1	Multi User Data Access Improvement.....	39
4.1.2	Open Source Technology	39
4.1.3	Flexible Visualization Options	40
4.2	Performance Improvement	40
4.2.1	Data Loading.....	40
4.2.2	Prototype Evaluation.....	41

CHAPTER 5 CONCLUSIONS.....	45
5.1 Latency Reduction	45
5.2 Generalization to other Industries.....	45
CHAPTER 6: RECOMMENDATIONS	46
6.1 Recommendations.....	46
6.2 Limitations of the Study	46
6.3 Further Study.....	46
REFERENCES.....	47
APPENDICES.....	50
7.1 Appendix I: User Manual	50
7.2 Appendix II: Prototype Code	56
7.3 Appendix III: Observation Sheet.....	103
7.3.1 General Information.....	103
7.3.2 Data Sources	103
7.3.3 Data Cleanup.....	103
7.3.4 Data Analysis	103
7.3.5 Data Visualization	104
7.3.6 Dashboards produced.....	104

Table of Figures

Figure 1 Business Intelligence Model, Gartner (2015)	9
Figure 2 Benefits of In-Memory Analytics, Nayem et al (2014)	13
Figure 3 Apache Spark Computing Architechure, Depali (2016)	14
Figure 4 In-Memory Architecture, source: (Garg, at al 2013).	15
Figure 5 Business Intelligence Architecture/ Model, Nguyen Manh et al. (2005)	17
Figure 6 Data Analysis, Years in Analytics.....	21
Figure 7 Data Analysis, Respondents by gender.....	21
Figure 8 Data Analysis, Respondents per job role	22
Figure 9 Data Analysis, Job Role Analytics	22
Figure 11 BIM Methodology phases, BI Minds (2014).....	24
Figure 12 Traditional Data Analytics Process	25
Figure 13 Project Gantt chart.....	27
Figure 14 Architecture of the in-memory analytics engine, E Bay Technologies (2014).....	29
Figure 15 Current Data Analytics Process	30
Figure 16 Apache Spark Home Page.....	33
Figure 17Apache Zeppelin Home Page	35
Figure 18 Apache Spark's - Zeppelin Scala Sample code	37
Figure 30 DBMS Data Load.....	40
Figure 31 Apache Zeppelin/Spark Data Load.....	41
Figure 32 Zeppelin Load	41
Figure 33 DBMS Analysis	42
Figure 34 Zeppelin Analysis.....	42
Figure 35 Zeppelin Analysis Details	43
Figure 36 Performance Comparison Chat	44
Figure 19 Zeppelin Banking Analytics Notebook – Loan fact per branch dimension analysis.....	50
Figure 20 Zeppelin Banking Analytics Notebook – Loan fact per branch dimension analysis.....	51
Figure 21 Zeppelin Banking Analytics Notebook – Loan fact per branch, productype, and currency dimensions analysis.	51
Figure 22 Zeppelin Banking Analytics Notebook – Measuring transactions fact by channel type dimension.	52
Figure 23 Zeppelin Banking Analytics Notebook – HR.....	52
Figure 24 Zeppelin banking Analytics Notebook – HR.....	53
Figure 25 Zeppelin Banking Analytics Notebook - HR.....	53
Figure 26 Spark Executors.....	53
Figure 27 Spark SQL Completed Queries.....	54
Figure 28 Spark - Zeppelin Stages	54
Figure 29 Spark - Zeppelin Jobs	55

Table of Tables

Table 1 Relating Research Objectives to Research methodology	20
Table 2 Raw Data from Observation sheets	20
Table 3 Data Analysis, Classification	21
Table 4 Data Analysis, Data Observation Summary	23
Table 5 List of activities and their time lines for the project.....	26
Table 6 Proposed Budget	27
Table 7 Prototype Requirements, List of Dimensions	28
Table 8 Prototype Requirements, List of Measures.....	28
Table 9 Apache Spark Configuration	32
Table 10 Apache Zeppelin Configuration	34
Table 11 Data Validation and Simulation Using R	36
Table 12 Process Improvement Evaluation	39
Table 13 Performance Comparison	44

CHAPTER 1: INTRODUCTION

1.1 Background

Business intelligence can be looked at as analytical, technology supported tools and processes that are used in gathering, transforming and analyzing large volumes of data. These processes changes data to information and information to knowledge about the opportunities, objectives and the position that an organization is compared to the competition. It supports the entire business decision making process by the management team through insights into data (Bernhard and Maria ,2015).

Data analytics which comes under business intelligence can be defined as use of sophisticated techniques and tools which are more advanced than the traditional business intelligence to examine data with the aim of discovering deeper insights, make predictions or to generate a recommendation (Gartner, 2017). These techniques include data forecasting, semantic analysis, sentiment analysis, pattern matching, visualization, neural networks and graph analysis.

Big data comes in to resolve the velocity, volume and variety issue that large amounts of data presents. Velocity issue comes when there are no machines that are fast enough to gather and store the huge amounts of data while volume problem is when there are no machines that can handle the volume of data being pushed into the storage systems. Variety on the other hand is when the data comes in many types (Chun et al, 2015).

In memory analytics has been developed and is driven by the need to do more analytics that are beyond OLAP analysis by exploiting the potential of hardware innovations whose prices have gone down in recent years. This falls under big data analytics which enables organizations to quickly access and track real time data. In memory analytics as studied by Santhosh et al (2013) will be the next frontier in business intelligence and analytics.

Kenyan financial industry has over the past few years enjoyed exponential growth in deposits, loans, assets, profitability, regional expansion and came up with innovative mobile products for their customers. Due to stiff competition among banks, we have seen that banks are now innovating around customer needs instead of traditional banking products. This means banks have had to invest in digital systems such as mobile banking, internet banking and other alternative channel systems that generate a lot of data. These systems generate a lot of data but they are operated as separate databases posing a need to apply new data warehousing techniques

in solving analytical problems associated with data for decision making by using in memory data warehouse.

1.2 Business intelligence and data analytics

Data analytics, business intelligence and big data analytics have been identified as very crucial in both the academic research and the business communities over the past two decades. Industry studies have highlighted this significant development (Hsinchun, Roger and Veda 2012). A survey of over 4,000 information technology professionals from 93 countries and 25 industries by the International Business Machines (IBM) Tech Trends Report (2011) supports the importance of these areas by identifying business analytics as one of the four major technology trends in the 21st century.

Some of the ways that big data analytics can be used in the banking sector as listed by Utkarsh and Santosh (2015) to include analysis of channels usage, customer segmentation and profiling, sentiment and customer feedback analysis, security and fraud management, customer spending patterns. Their journal article established the that fact that it is possible to use analytics to answer a question such as why there was a drop in-terms of customer satisfaction index by analyzing the data that customers had submitted.

1.3 Need for Emerging Analytics

There are about 13 capabilities that can be achieved using modern business intelligence and analytics platforms (Hsinchun, Roger and Veda 2012). These capabilities make emerging business intelligence and analytics field important to modern organizations; dashboards, ad hoc queries, search-based business intelligence, reporting, online analytical processing (OLAP), predictive modeling, interactive visualization, data mining and scorecards. A few business intelligence and analytics areas are under research and development based on a report from Gartner Business Intelligence Hype Cycle (2015) analysis for upcoming business intelligence trends. These include data mining workbenches, column-based database management systems (DBMS), in-memory database management systems (DBMS) and real-time decision support tools.

1.4 Problem Statement

Most banks have core banking systems, digital banking/omni channels systems, customer relationship systems, enterprise resource planning systems and credit scoring systems that run on different databases. A journal article by Xinhui et al (2015) show latency in financial organizations as a major problem that such organizations face when processing large volumes of data. Their research indicated some traditional solutions that these financial organizations implement such as high-performance machines with more processing capabilities to achieve low latency. They also indicated that adding more central processing units or addition of more memory to scale up the performance of computers. Scaling using traditional methods involved adding more computational nodes that have high performance networking devices. Over the last few years, the article by Xinhui et al (2015) show that there has been a lot of growth in terms of the amount of data to be analyzed which directly impacts the performance of traditional storage systems meaning that latency is significantly impacted by huge volume of data that is being processed. This is called a big data problem which asks for new tools that can handle the huge and complex for traditional tools.

Another problem on dirty data has been studied by Nikhil, Gautam, Hillol (2013). They have defined this problem as any kind of impurity or anomaly in data which could affect the effective utilization of data by slowing down performance or reducing the accuracy of final analysis which is in turn used to make management decisions. This would mean that direct data interpretation and analysis could lead to faulty decisions if the data is polluted with bad data. The problem on inaccurate reporting has been illustrated by The Data Warehousing Institute (2010) that while IT shops are reeling under demands for real-time, continuous availability of data and systems, most internal data quality remains poor.

1.5 General Objective

The general objective of this research was to develop an enhanced data warehousing prototype that would improve on data visualization, speed of reporting and accuracy of conclusions by exploiting in-memory data mining algorithms.

1.6 Specific Objectives

The specific objectives that would assist in delivering the general objectives were as follows:

1. To develop an in-memory data warehousing prototype for use in banking analytics.
2. To evaluate the developed data warehouse prototype and show evidence how banking analytics are improved.

1.7 Research Questions

1. What state of the art systems are available for use in banking analytics?
2. Which technologies can be used to incorporate in-memory algorithms in a modern data warehouse?
3. Which methodology can be used to develop an enhanced data warehouse using in-memory data mining algorithms?
4. How can an in-memory data warehouse prototype performance and accuracy of conclusions arrived using it be evaluated to show improvement in banking analytics?

1.8 Significance of the Study

This research was considered important because it would give important results regarding in-memory data warehouse by exploit various in-memory algorithms to show if there would be improvements in the current state of banking analytics and decision making in the banking industry. These results can be generalized to other fields as well since the underlying architecture of IT systems in the banking industry is like other fields.

1.9 Scope of the Study

The study involved an investigation into modern data analytics tools that can be applied in the banking sector. A prototype would be developed to show how in-memory algorithms can be used to solve performance related issues when handling data warehouse setups. This was to be evaluated by selected sample users in the banking industry to confirm if the performance was improved. The study did not involve developing an actual data warehouse due to privacy and ethical issues that would come up when handling banking industry data.

1.10 Contributions

This study contributed its findings to the field of in-memory analytics by showing evidence that there was an improvement in performance when data was stored in memory compared to traditional disks storage systems. The research was aiming at helping organizations make informed decisions when implementing data analytics initiatives by designing their analytics solutions based on performance and accuracy of analytical results.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter is about related work on business intelligence and data warehousing by looking at research that has been done into these fields. The chapter also considers some of the recent developments in in-memory analytics.

2.2 Business Intelligence

Business intelligence is an analytical, technology supported process that is used in gathering and transforming fragmented data for an enterprise. BI changes data to information and later to knowledge about the opportunities, objectives and the position that an organization is among the competition. Business intelligence is not merely about the applications since it supports the entire decision-making process of management. There is a clear distinction between software, tools a BI solution. A software can be readily available from different software vendors as a standard ready to use solution. A firms' applications and tools on the other hand are software products which have been installed, configured and are used to meet a purpose such as business planning. A business intelligence solution can be viewed as a collection of applications which would include the underlying IT infrastructure in terms of servers, integration platforms, operating systems and networking appliances (Bernhard and Maria, 2015).

2.3 Business Intelligence Maturity Models

There are three stages through which organizations go through in business analytics maturity as their business needs and demands continue to evolve (Gartner, 2015):

2.3.1 IT-Centric

This is the lowest level where organizations usually approach business intelligence as an IT driven project that, mainly focused on selection of analytical tools and data collection. Questions at this level would be 'What happened?'. The key focus is analysis of historical data to assist in making better business decisions.

2.3.2 Information Management

This is the middle level which involves checking an organization's growth against some preset parameters. A key question would be 'what can be changed for us to meet the target or how are we doing as compared to the strategic objective?' Organizations at this level are aware that better business decisions can be reached by using business intelligence solutions that can push

information to people who utilize them to make informed decisions. Most IT teams would be integrating several internal and external systems such as online order processing solutions, CRM systems and ERP systems.

2.3.3 Predictive Insight

There are very few organizations that have managed to reach this level where businesses deploy advanced solutions that can deliver predictive modeling and advanced analytics to anticipate likely outcomes of future business outcome. This is achieved by modeling the expected trends or future market opportunities. A key question at this level would be ‘what outcome will be present at a future date and how do can a firm use the results for maximum returns on investment?’.

To implement a business intelligence solution, it is necessary to first establish where an organization is on the maturity model which in turn helps to understand the likely efforts that need to be made for maximum value of BI to be realized by the organization. For mid-sized companies who are in their early stages of BI they can start by showing how the business is performing using some simple reports and dashboards. Go to the next level of trying to gain insight into why certain events or conditions are occurring through addition of analysis capabilities. Third level is trying to link the insights gained from analysis by incorporating some planning functionality. Lastly is to ensure that action is immediate across the organization by integrating what-if scenario modeling to the planning and analysis capabilities in the second and third levels.

The figure below shows these levels.

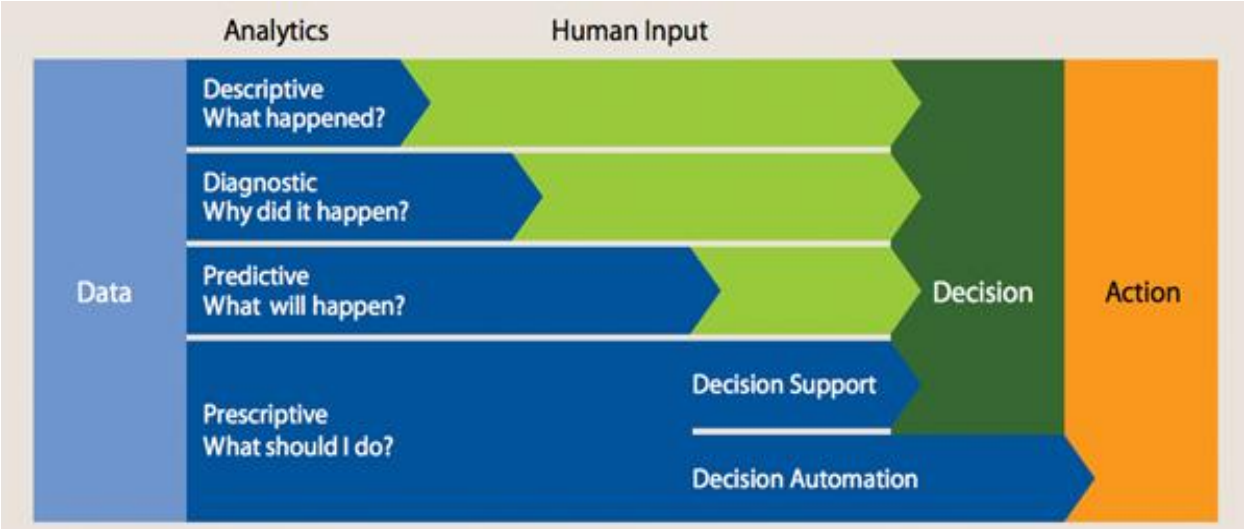


Figure 1 Business Intelligence Model, Gartner (2015)

2.4 Business Intelligence Vs Decision Support

Traditional business intelligence systems are different from earlier types of Decision Support System in some of the following ways (Bernhard and Maria, 2015):

- They enable the development of DSS that are based on facts by integrating, aggregating and managing any form of data in a near real-time data warehouse.
- BI solutions have led to creation of new openings such as data mining by utilizing algorithms that deal with huge data volumes and use in-memory technologies to deliver high processing capacities.
- BI solutions can achieve automated reports/dashboard delivery to or self-service from several devices by utilizing modern ways of information delivery and data interrogation.

Bernhard and Maria (2015) considers all these advances and claims that the new generation of DSS can be compared to modern BI solutions if they solve some of the issues that were associated with previous forms of management information systems. Their research point to questions like: How can BI solutions be used to overcome the issue of getting lost in data while trying to understand the underlying patterns in data? Do BI solutions lead to effective decision making? What components of a BI solution are essential to ensure a BI project success?

2.5 Steps to achieving a Business Intelligence System

Bogdan (2013) says that a company needs to know that BI platform implementations are very long and time consuming given the kind of technology that is involved. The analysis stage alone can take around one year depending on the size of the organization and process complexity in its data generating operations.

2.5.1 Transaction processing systems as a BI data source

This is usually the initial stage for most BI systems and it's called the requirements analysis stage which ensures that the objectives of the business intelligence system are captured. Online transactional processing systems that maintain day to day records of transactions are usually maintained by most data centric organizations. These systems are used to track the day to day transactions of an organization such as managing clients, sales or suppliers, processing loans and other day to day automated processes. The setup is usually that these transactional systems

maintain the data using different database technologies which are usually pushed to a backup location for archival.

Bogdan (2013) study shows that use of transactional data yields important data for business analysis which shows that this data becomes a major source for the data warehouse which is a key component of the business intelligence solution. Some External sources of data such as competition data or data about a regulator can be considered as reliable data source but it varies from one organization to another.

2.5.2 Data Warehouses

This is the most important phase when developing a business intelligence system. This phase seeks to establish how the extraction, transformation and loading processes will be achieved. A data warehouse development phase describes what technology will be used to come up with the reports. It is also a database but it usually implemented differently than the traditional transaction oriented database since it is meant to contain historical data of a certain interest. Most designs involve data warehouses that are divided into an enterprise warehouse, a virtual warehouse and a data mart. The enterprise warehouse will be for collecting all the information for the entire organization. This means it provides very big volumes of data. The grouping can be done into detailed data, aggregate data, and dimensions. A data concentration or data mart contain usually contains a subset of the entire data warehouse's data from the organization. In other cases, a data mart is designed to hold data for a certain department such as sales unit which means it's possible for a data mart to be limited to specific subjects. The virtual warehouse on the other hand is a set of perspectives from the operational databases that are usually simple to develop but would require additional capacities on the storage infrastructure. Aggregate data in a data warehouse is essential because it can be used to improve the average response time even though most people argue that they lead to redundancy in data. They have features such as for data summarization, consolidation, totalization.

2.5.3 Extracting, Transforming and Loading (ETL) Data

The ETL (Extract, Load and Transform) represents a core process or component of a business intelligence solution. This process is initiated by extracting data from the various sources that an organization needs to analyze. Most researchers say this process can be labor intensive and

challenging to many developers depending on the variety of the data sources. Data that is extracted in this stage is usually pushed to a staging database for clean up or directly to the data warehouse. The data extraction stage it is very important since data must be correctly retrieved from the source files for the other phases of the data warehouse to be correct. Extraction of data is usually done from multiples data sources which can be in different formats such as flat files, relational databases. Another major challenge is the transformation process where it is expected that all the data resources will be transformed into one single type which can be easily integrated into the data warehouse. The ETL process usually resolves this problem by inspecting the data files, confirming if they are in a certain format, loading the data in a staging database and automatically removing all the incorrect data. There is an initial data filter at this extraction stage to eliminate some minor data issues. Some logic that is then applied to the data that was extracted in the first step to make up the data transformation phase. This ensures that the data is ready for the loading stage. This stage can lead to very complex transformations of the extracted data depending on the business intelligence solution needs. The last of the three ETL subsystem stages which entails loading data in the data warehouse tables is called the data loading stage. Data loading can be made at certain time frequencies such daily, monthly, quarterly or even annually depending on organization's needs. Most common data warehouse designs allow for an initial loading or only pushing of updates to some few tables.

2.5.4 Multidimensional Data Analyze

Online Analytical Processing (OLAP) is a common term in business intelligence referring to they are a collection of hierarchical, dimensions, and interrelated measures by aggregating data as a multidimensional cube. OLAP helps in achieving complex analysis and processing of data without compromising on system performances. Another key feature of OLAP is ability to present analysis instruments which allow for explanation of complex reports.

2.6 Review of Business Intelligence, Data warehousing in Banking

Adrian and Ovidiu (2012) suggested that several activities, procedures and applications makes up banking intelligence. These include tools for extraction, transformation, and loading (ETL) of data, data warehousing, information portals, OLAP tools, data Marts, business modeling and data mining. Their article explain that the concept of data warehousing came up from the activities to collect and unify the data from disparate sources. Given the contribution BI gives to banking

executives to make decisions, banking intelligence has been identified as one of the most developing fields in information technology. Bank executives need to analyze details such as the bank's customers in different dimensions before they propose a product to them and to also assist in managing various banking risks and to ensure they can compete effectively. Some of these challenges have led to an area called banking intelligence. Adrian and Ovidiu (2012) defines banking intelligence as a set of applications, technologies and the processes of collecting, integrating, analyzing and presenting data patterns. Banking intelligence can use data that has been gathered into a data warehouse or a data mart to provide historical, current and predictive views of banking operations. Some of the key tasks that most information technology professionals go through when building banking intelligence systems can be summed up into intelligent exploration (data extraction), integration, aggregation (data transformation) and a multidimensional analysis of data that originates from various data sources (Mansmann et al 2007).

A study done by Srimani Et al (2014) to show some of the problems that would arise if reporting designs in banks report involve reporting directly from the transactional systems: Analysis time was greatly increased, analysis options were limited, few report generation options, insufficient reports lacking key features such as business forecasting and reports could not support some applications. Their research led to recommendation of having of big data to improve on performance in the analytical process of banking analytics. In their research, they implemented a data warehouse architecture which involved integration of different formats of data sets. They developed a data warehouse, some few dimensions, fact tables and then mapped these with the source and target data marts.

Sunil et al (2015) did some research on banking intelligence and established that banks need a diversity of solutions ranging from a fully integrated global data warehouse to integrated data marts for the rising needs of the divisions. They mention that both extreme solutions have very important disadvantages along with their advantages. Their research looked at the various data sources (RDMS and flat files), a staging area, a warehouse, some data marts and the various users that would mine the data as the main elements of a banking business intelligence support system.

2.7 In-Memory Analytics

Physical hard disk drives in a traditional database is used to store data. The data from physical hard drive is sent to main memory (RAM) for further processing when a query is requested. On the other hand, the in-memory DBMS usually store data permanently in the main memory (RAM) of the available hardware. A central processing unit can directly access the random-access memory content which is made up of some summarized data, stored in RAM. Development of Multidimensional cubes is avoided for In-Memory analytics. Compared to caching, data in in-memory databases are usually huge, at times growing to be the same size as an entire data mart (Garg, et al 2013).

2.7.1 In-Memory Technological Advances

Some of the hardware and software technological advances that have led to in-memory technology can be summarized by looking at each of them individually. Hardware developments include new multi-core central processing units and parallel scaling across blades. Software developments are row and column store technology, data compression, partitioning, removal of aggregate tables, use of insert only data design patterns and on the fly extensibility. Others include bulk loading, use of any attribute as index, active/passive data store among others.

2.7.2 Benefits of In-Memory Analytics

Other than direct improvement in performance and latency reduction, there are other benefits that can be achieved using in-memory technologies as compared to traditional disk storage systems. Some of these are mentioned by Nayem, et al (2014) in their research on emerging BI and advanced analytics. Figure below summarizes these benefits as shown below.

Read and Write Capabilities
Centrally Managed Data, Business Hierarchies, Rules and Calculations
Empower Business Users to Analyze any Combination of Data
High Impact Visualizations
Extend and Transform Excel
Designed for Modern 64 bit Architectures
Easy to Install and Easy to Use

Figure 2 Benefits of In-Memory Analytics, Nayem et al (2014)

2.7.3 Apache Spark as In-Memory Analytics Solution

Apache Spark is an open source tool that can be used for data analytics or big data analytics by using in-memory cluster computing technology. It is built from Hadoop distributed file system which uses map reduce algorithm. The tool supports any platform (Windows, Linux and Mac OS) and can be used to develop apps through use of Java, Python, R and Scala interpreters. Depali (2016) lists a number of case studies where analytics have been done using Apache Spark to include use of cluster technology in computing to map brain activity, engaging shoppers real-time at a shopping counter, analyzing the occupancy of retail shelf on a real-time basis and use of wearable monitors that could be used to track body activities.

The figure below shows Apache Spark's modules and general computing architecture.

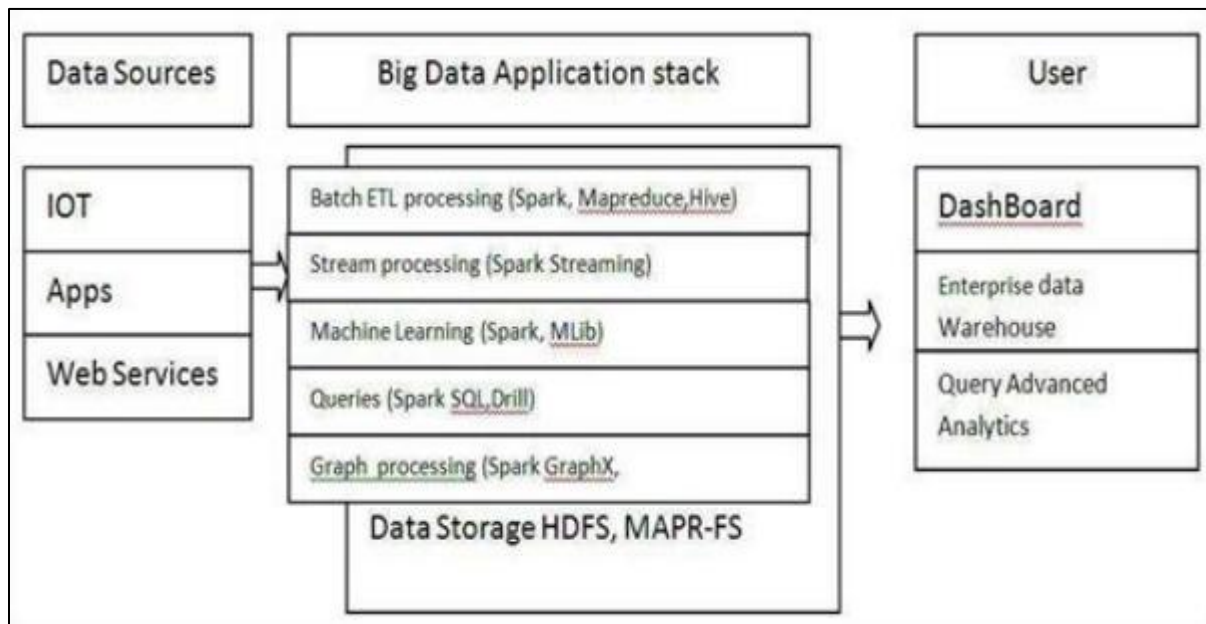


Figure 3 Apache Spark Computing Architecture, Depali (2016)

2.7.4 Architecture of In-Memory Analytics Systems

Garg et al (2013) shows that there are different approaches that can be considered when developing an architecture for in-memory analytics. These are: in-memory OLAP, excel in-memory add-in, associative model, in-memory visual analytics and in-memory accelerator. Some of the main differences between the current/traditional BI systems in relation to the futuristic in-memory systems can be looked at by establishing the storage methods. Whereas the conventional BI tools query data that is residing on disk, in-memory systems on the other hand query data

directly from random access memory (RAM). When a user executes a query in a traditional data warehouse which usually holds data from several databases whose storage is in disk. All information is initially loaded into random-access memory for an in-memory database from where users can directly interact with the data by pushing queries into the machine’s memory. Comparing performance of these two shows exponential improvements when using data that is stored in the random-access memory as opposed to accessing that same data from disk. The figure below shows a sample in-memory architecture.

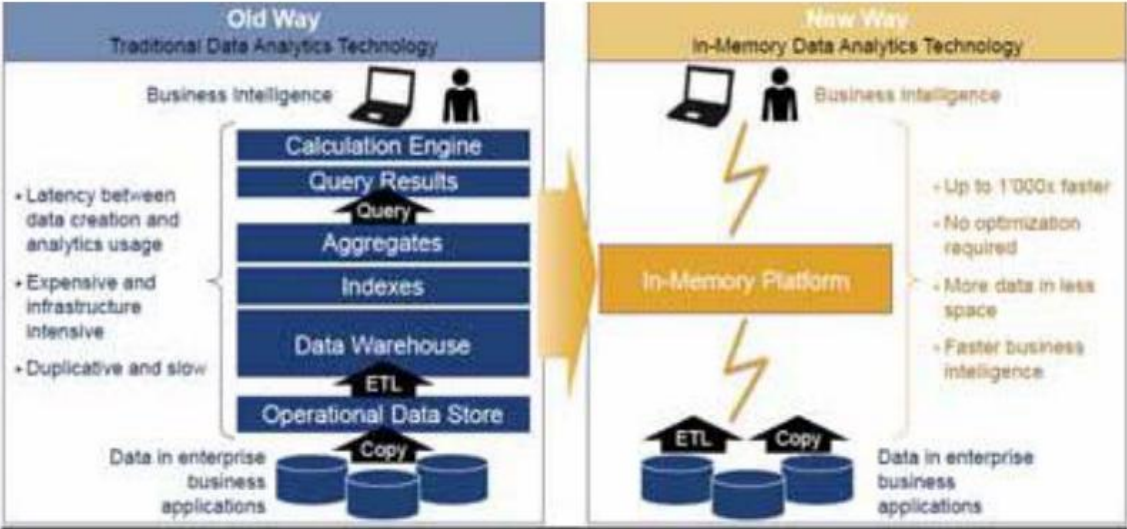


Figure 4 In-Memory Architecture, source: (Garg, et al 2013).

2.8 Proposed Architecture for BI

For a near real-time BI system to be complete, a researcher needs a detailed analysis of the requirement which is required to determine the level of response time that would best serve a given business purpose. This is referred to as right BI in which case a delay of even a day will not adversely affect the quality of business decisions (Dale, Tara, and Nayem, 2012). A near real time business intelligence that can be achieved using in-memory algorithms is described as any of the following (Azvine, Cui and Nauck, 2005):

- ✓ That access to information by a given process is guaranteed at any time.
- ✓ That whenever information is required by management, the process can provide.
- ✓ That a process can have a reduced latency as a requirement.

- ✓ Ability to provide key performance measures based on not only historic data but also current data.

Janina (2012) defines real-time business intelligence as the process of delivering insights about certain business concepts as they occur with minimum latency. This therefore means that an in-memory data warehouse can provide the same functionality as a traditional business intelligence solution. The key difference is that in-memory data warehouse operates on data that is extracted from operational data sources with zero or minimal latency. In-memory solutions can also provide a means to propagate actions back into business processes in real time.

2.9 Related work on the Problem of Latency

Latency is the time taken from something happening or changing to the moment when we can do something about it (Zeljko, 2007). He declares that this issue is a major problem that many organizations face today when they are designing BI and data warehouse systems. This is a critical concept in BI and in-memory analytics since it can be grouped into analysis latency, data latency, action latency and decision latency. Most studies geared towards in-memory research has been directed towards data latency by feeding data faster into the data warehouse. Storing real-time data in the same location that the historical data is stored in the fact tables is also another way to handle the problem of latency.

2.10 Technological Gap

The Gartner BI Hype Cycle for 2016 analysis for emerging business intelligence trends shows that most areas in business intelligence and analytics are still under active research and development. These are summarized under data mining workbenches, column-based database management systems, in-memory database management systems and real-time decision tools. A few companies have developed workbenches for their customer to allow data modelling and other configurations that are necessary when developing data warehousing solutions. Column-based database management systems are used to improve on performance when read data from the hard disk and are more suitable for OLAP based work which is related to data warehousing. In-memory database management systems store data in memory instead of disk which improves performance of data access. These areas have not been studied to evaluate how these technologies can enhance the data analytics in the organizations such as the banking industry. Most of the commonly used algorithms are selection scans, hash tables, linear probing, double hashing, cuckoo hashing and bloom filters (Orestis Et. al, 2015)

2.11 Overall System Architecture/Model for a Business Intelligence System

Figure 2 below shows a sample BI architecture which is adopted from Robinson (2002) and it generally shows a three-tier frame. A centralized data integration platform in tier one is realized in real time by having a collections of real time ETL tools that can collect operational data from different heterogeneous sources. For tier 3, the business rules are analyzed using real time query and reporting tools. Another proposal of handling real time BI by Nguyen Manh et al. (2005) is based on service-oriented architecture. Heavy investments in hardware is required to achieve requirements for high availability, scalability and performance for organizations seeking to incorporate intelligence into business processes.

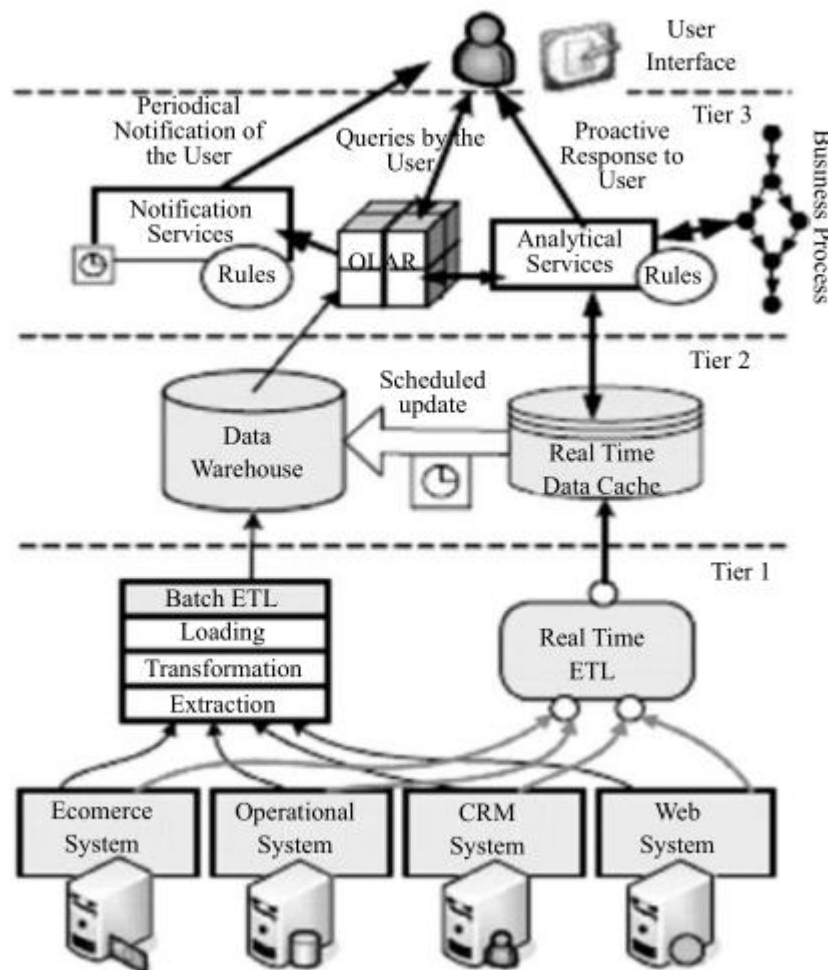


Figure 5 Business Intelligence Architecture/ Model, Nguyen Manh et al. (2005)

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

This chapter covers the methodology that was used to carry out the research and development of the prototype for an in-memory data warehouse. The research methodology used combined both the research elements and the prototype development work of the project.

3.2 Research Design

The research was done using the case study approach where data was collected from the business reporting, technical and management teams that were involved in the reporting process.

3.3 Research Population

The research population included banking sector professionals that are involved in day to day reporting.

3.4 Sampling

The study used purposeful sampling method to get a sample that was used for the study by observing several banking professionals that were involved in banking analytics. Convenience sampling was used to determine the sample by getting all the users that are involved in data analysis by either directly analyzing the data, supporting the analyst or by consuming the results of the analysis.

Observation to collect the functional, analytical and reporting requirements from the banking reporting teams. Most of the functional requirements led to the development of data cubes, data marts by exploiting the dimensions and measures that were also identified in this stage. Since the main purpose of the project was to show that performance would be improved after implementing the new tools as compared to the traditional setup, the project then narrowed the requirements to a subset that could form part of the prototype.

3.5 Data Collection Methods

3.3.1 Prototype Data Sources

Data was available from sample data that had been stored in the databases of a sample bank over a period of about 5 years and from Lending Club statistics website which offer data freely for research purposes. This provided enough data that was used in building the prototype for the in-memory warehouse support system.

3.3.2 Research Data

The second part of the project which involved carrying out research to uncover the current state (challenges) of analytics in the banking sector. The sample population covered professionals who were working as business analysts, technical users and top executives. Filled observations sheets are in the appendix section of this report. The sample was selected based on purposive sampling technique which represented the entire population of data analytics team from banking institutions.

The researcher made a visit to the premises of the target audience to observe and record the different processes that formed data analytics in those organizations. This was after the necessary approval was obtained from the management to allow the study with instructions on how to ensure privacy of the customers and institutions involved. The observation sheets aimed at getting the following information:

- A list of data sources that the banks had and the data formats for each of the sources identified.
- A detailed explanation of the extraction process from the sources identified.
- A list of data cleanup tools and a description of the process for cleaning up data.
- A list of data visualization tools and a description of the process of visualizing data.
- A list of dashboards that the management team get from the whole data analytics process.
- The timelines that the data analytics process take for each of the processes above.

3.6 Relating Research Objectives to Research Methodology

Research Objective	How it was achieved
1. What state of the art systems are available for use in banking analytics?	Literature review
2. Which technologies can be used to incorporate in-memory algorithms in a modern data warehouse?	Literature review
3. Which methodology can be used to develop an enhanced data warehouse using in-memory data	Literature review Prototype development

mining algorithms?	
4. How can an in-memory data warehouse prototype performance and accuracy of conclusions arrived using it be evaluated to show improvement in banking analytics?	Prototype development Evaluation of the prototype Report on results

Table 1 Relating Research Objectives to Research methodology

The raw data that was collected can be summarized into the table below to show how the various respondents were observed and the various variables that were recorded by the researcher.

Job Role	Years in Analytics	Analysis time Per Report	No. of Reports Analyzed	Gender
CHANNELS	5	8	13	F
IT	7	6	8	M
MANAGEMENT	6	7	6	M
CHANNELS	4	8	16	F
HR	3	10	11	F
HR	3	9	10	M
LOANS	1	14	10	F
RECOVERY	3	10	9	M
BRANCH	4	9	12	F
LEASING	6	8	11	M
MANAGEMENT	1	11	10	M
LEASING	4	9	13	F
BUSINESS	1	14	9	M
IT	3	9	8	M
RECOVERY	5	8	10	F
TEEASURY	1.5	12	13	M
LOANS	3	9	9	M
LOANS	4	8	7	M
MANAGEMENT	4	9	9	M
IT	3	10	13	M
BRANCH	5.5	7	16	M
MANAGEMENT	7	6	12	M
TEEASURY	3	9	10	M
BRANCH	4	8	9	M
MANAGEMENT	2	11	11	M

Table 2 Raw Data from Observation sheets

3.7 Data Analysis

The Data that was collected had to be analyzed so that various elements of the study can be easily interpreted using Excel. The data was analyzed by classification of the data into groups with similar characteristics. The table and figures that follow show how the data from the observation sheets was analyzed.

	1-3	3-5	>5
Years in Analytics	12	8	5
	Male		Female
Gender	18		7
	Average number of Reports		Average time per report
Data analysis timelines	10		9

Table 3 Data Analysis, Classification

Figure 6 shows how the count of experience in years in analytics from the respondents with classification into 1-3, 3-5, 5-8 classes to make the data easy to interpret.

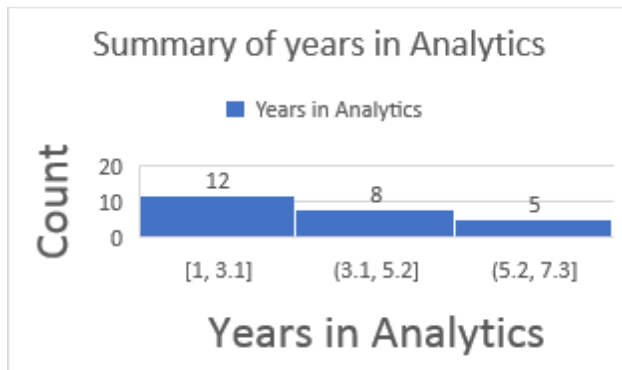


Figure 6 Data Analysis, Years in Analytics

Figure 7 shows some analysis of the data that was collected bases on the gender of the respondents.

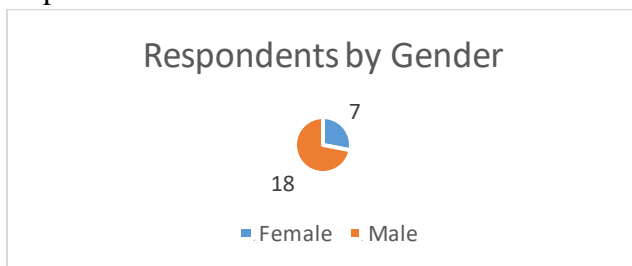


Figure 7 Data Analysis, Respondents by gender

Figure 8 shows the composition of the departments from where the respondents came from.

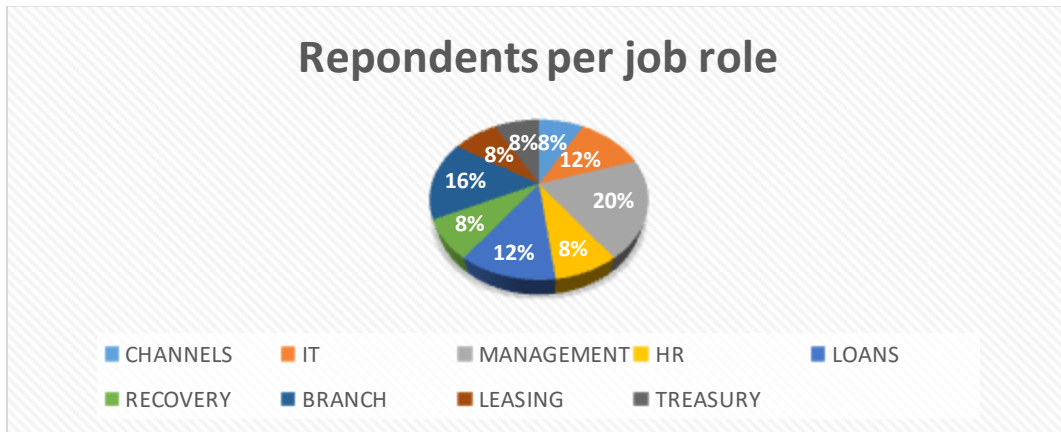


Figure 8 Data Analysis, Respondents per job role

Figure 9 shows how various the average experience that various departments have been handling data analytics, the analysis time per report and the number of reports they analyzed.

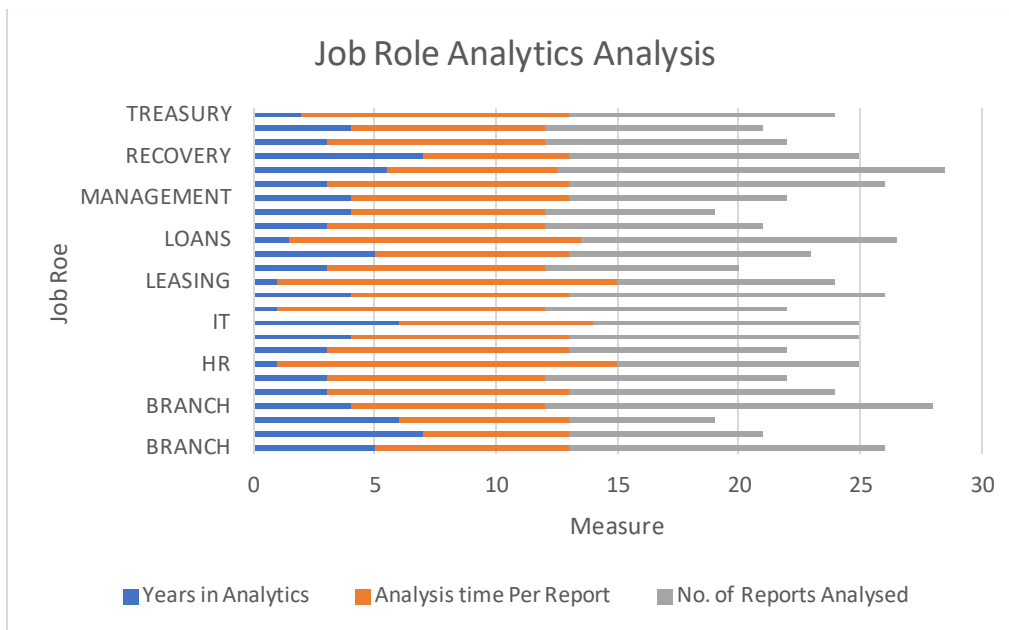


Figure 9 Data Analysis, Job Role Analytics

In summary, the observation sheets showed the following information regarding the banks that were involved in the study.

Item	Observation Results
Data Sources	Oracle/SQL server/PostgreSQL – core banking system, human resources data, Channels. Excel

	<p>Credit/Loan origination/approval Data Share Point Data</p> <p>Most of the data was stored in structured database management systems such as Oracle, SQL server and PostgreSQL. This represented transactional processing systems data. Excel was also used to store data that was obtained from systems that had not been automated. Share point was used to store data that was from workflow based systems such as employee scorecards for performance management.</p>
Data Extraction Process	The technical team were required to login into the databases in each of the transactional systems to run complex SQL queries that would generate data required. The data was then saved in excel sheets and placed in shared folders to allow the analysis team to access.
Data Cleanup	<p>SQL Query Filtering</p> <p>Excel Filtering</p> <p>Manual Editing</p>
Data Cleanup Process	Data extracted was cleaned by filtering SQL using where clauses before extracting into excel. A second step involved manual editing and using various excel tools to remove what was looked at as dirty data.
Data Analysis	<p>Time per report: 15 Minutes</p> <p>Reports Analyzed: 20</p> <p>Analysis Tools: Excel (Charts, Pivot table)</p>
Data Analysis Process	Analysts were required to clean up the data by implementing the data transformation rules identified under data cleanup section. Analysis was done by using various excel built-in tools that allow various analysis to be done on data such as power pivot and pivot table.
Data Visualization	Tools Identified: Excel charts, Power Point
Data Visualization Process	The clean/transformed data in excel sheets was ready to be applied various excel formulas which could assist in arriving at results desired. For example, they used pivot table to get value of loans that had gone to each customer sectors which in turn produced charts.

Table 4 Data Analysis, Data Observation Summary

3.4 Prototype Development Methodology

The prototype was developed using the business intelligence model (BIM) methodology that is broken into 6 phases and is best suited for analytics applications development. Since most analytics applications come with prebuilt designs, they are designed to maximize their value by avoiding development from scratch but instead reusing some existing designs with minimal customizations during early development phases. All other customizations are deferred to later phases of the implementation when it is clear if the application will meet the expected benefits.

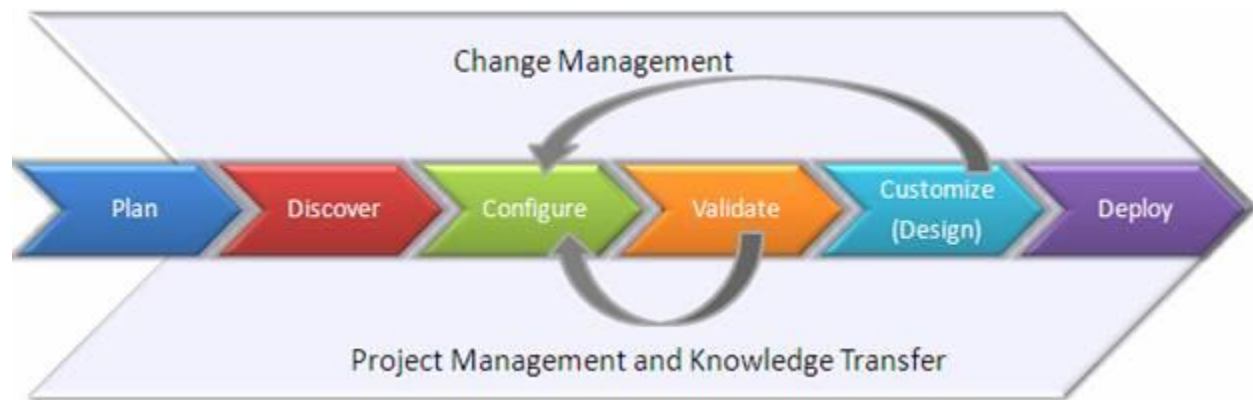


Figure 10 BIM Methodology phases, BI Minds (2014)

The phases covered when developing the prototype were as follows:

- Planning phase
- Discovering phase
- Configuration phase
- Data validation phase
- Customization phase
- Deployment phase

3.4.1 Planning phase

The objectives of the planning phase included:

- ✓ Getting an understanding and evaluation of the current state

To get a clear understanding of the current state, an observation was done into the processes that are involved in current banking analytics. The data was collected by filling in observation sheets showing the steps and procedures performed in each stage. These sheets are as attached in the appendix section of this report. Some of the banking professionals involved in the study showed

that they did not have any business intelligence solution in place while others had solutions in place but were not meeting the expected business objectives. The data collected also indicated the time it took to complete the analysis process using the current tools. From the observation in a selected institution, the data collected can be summarized as shown below.

The Processes in table 1 for data extraction, data cleanup, data analysis and data visualization were also explained and their visual representations drawn as shown below.

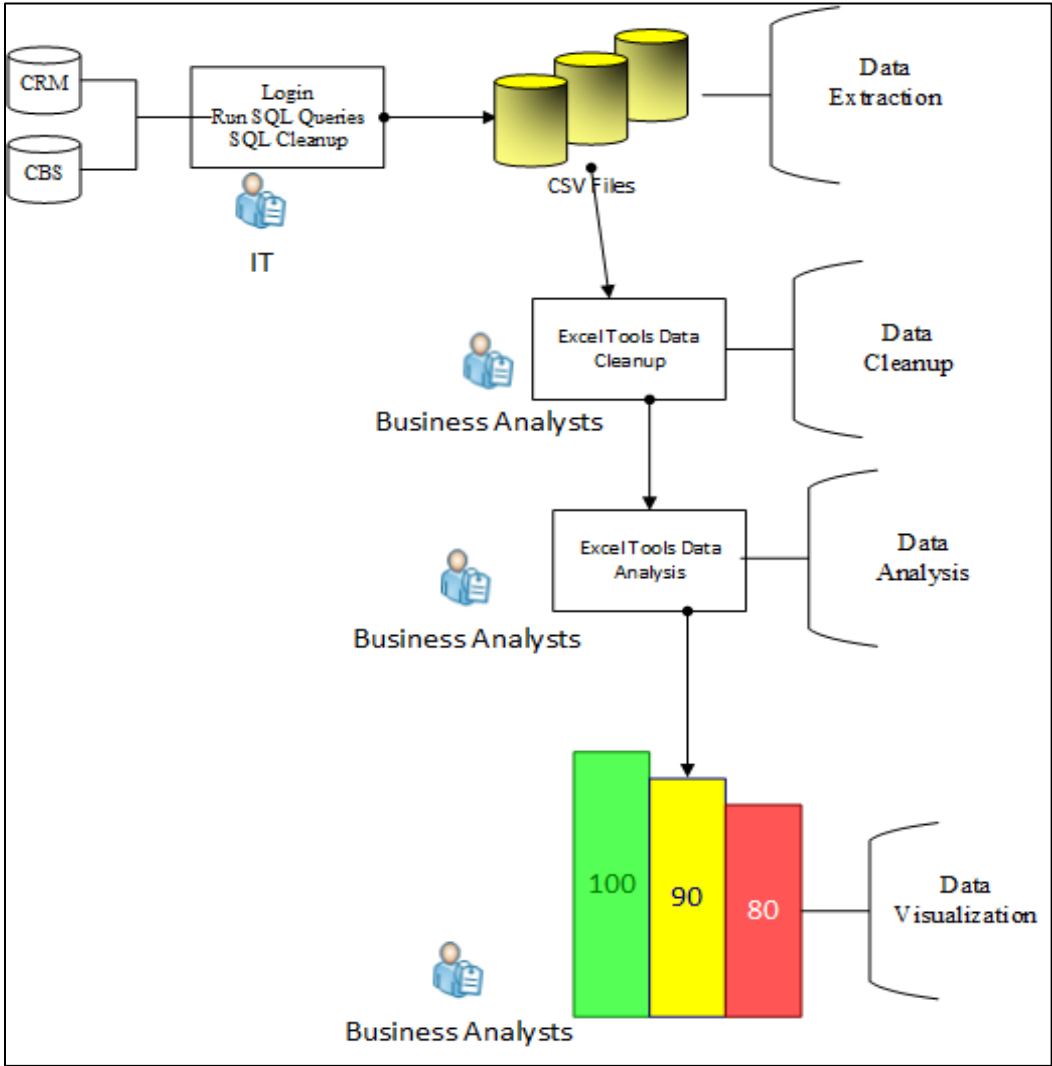


Figure 11 Traditional Data Analytics Process
 From the figure, the processes involved getting data from the different source systems such as CRM and CBS using SQL queries and saving them to some flat files. The data was then subjected to data cleanup tools and finally analyzed by the business analysts.

- ✓ Defining the project scope and the overall project approach

The project scope involved getting an understanding of the issues that banking analytics professionals faced and developing a prototype that would resolve most of these issues. The scope also covered evaluation of the prototype which was done to check if there would be any improvements into the banking analytics process.

The prototype was developed using in-memory stools which was expected to show performance improvements on time taken to carryout data analytics by showing how in-memory algorithms in Apache Spark can lead to lower data analytics latency.

For visualization, Apache Zeppelin which is also open source was picked since it can integrate seamlessly with Spark. The data warehouse prototype involved looking at the various dimensions and measures in a typical banking setup.

- ✓ Developing the project implementation and resource plans

The project implementation plan was developed using Microsoft project planning tool that showed the different activities that were covered during the project along with their timelines.

Table 1 below shows the list of activities and their timelines for the project.

Task Name	Start	Finish	Duration(days)
Requirements Review and Environment Setup	7/14/17	7/25/17	11
Review Current Reporting Environment	7/14/17	7/18/17	4
Prepare MIS Requirements	7/21/17	7/25/17	4
Build Data Integration Engine - ETL	7/21/17	9/27/17	68
Incorporate in-memory Engine	8/29/17	10/9/17	41
Prototype Testing	9/9/17	9/20/17	11
Production Server setup	9/2/17	9/13/17	11
Go live	9/30/17	10/24/17	24
Normal Data Analysis Process Evaluation	9/30/17	10/24/17	24
In memory prototype Evaluation	9/30/17	10/24/17	24
Project Closure	9/30/17	10/24/17	24
Documentation	7/14/17	10/24/17	102

Table 5 List of activities and their timelines for the project

The figure below shows the project Gantt chart that was used to plan the various activities of the project.

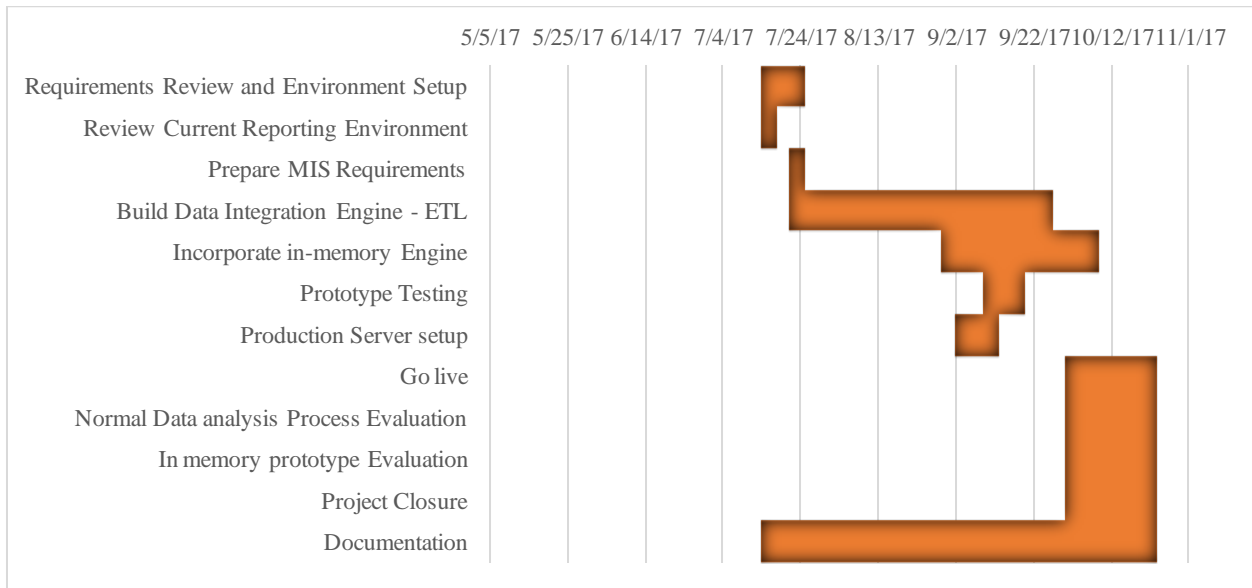


Figure 12 Project Gantt chart

There were no costs for the items/resources since all the tools selected were open source as shown below.

Resource Name	Pricing
Apache Spark 2	Open Source Tools
Hadoop	
Hive	
Apache Zeppelin – Data visualization tool	
Development Server/PC	

Table 6 Proposed Budget

- ✓ Collecting the functional, analytical and reporting requirements.

Table 7 and 8 represents the requirements that were to be met by the prototype. This means that outstanding balance could be looked at in terms of date, branch, product, currency and so on from the list of dimensions. Apache Zeppelin was used to visualize the dimension-measure relationship of the loan module.

No	DIMENSION	PERSPECTIVES(FACT)						
		Loan Book	Deposits	Budget	Transaction	GL	Channel	Staff
1	Date	√	√	√	√	√	√	
3	Branch	√	√	√		√	√	
5	Product	√	√					
6	Relationship Officer / Staff	√	√	√	√	√		
8	Product Type		√					
10	Currency	√						
11	Loan Classification	√						
12	Channel Type						√	
13	Gender							√
14	Education							√

Table 7 Prototype Requirements, List of Dimensions

Measure Name	Description
Outstanding Balance	Total Loan Amount Due
Arrears Amount	Loan Amount that is in Arrears
Arrears Days	Number of days Loan is in Arrears
Non-Performing Book Balance	Loan Amount that is in Arrears > 60 Days
No of Loans	Number of Loans
Average Salary	Average salary amount
Value of Loan Payments	Sum of Loan Re-Payments

Table 8 Prototype Requirements, List of Measures

3.4.2 Discovering phase

The phase involved listing of the business requirements/rules such as ETL logic changes or identifying the modifications that had to be made to the basic functionality. These included addition of more dimensional attributes, definition of the transactional systems of data which included loan origination, CRM, share point, core banking and other digital banking systems that all needed to be integrated into one central database. A look at the historical data requirements

was handled by the data that had been availed for use in this study from the identified transactional systems and online from sites that provide banking data for analysis.

The architecture of the in-memory analytics engine that was developed had the components shown in figure 14 below. This utilize the RDDs (resilient distributed datasets) which are fault-tolerant, many server machines in a cluster to allow programmers to persist intermediate results in memory by controlling how they are partitioned. This leads to an optimized data placement and manipulation using a list of operators. The use of RDDs has become more popular based on two issues that other computing frameworks are not able to resolve: interactive data mining tools and iterative algorithms (Matei et al,).

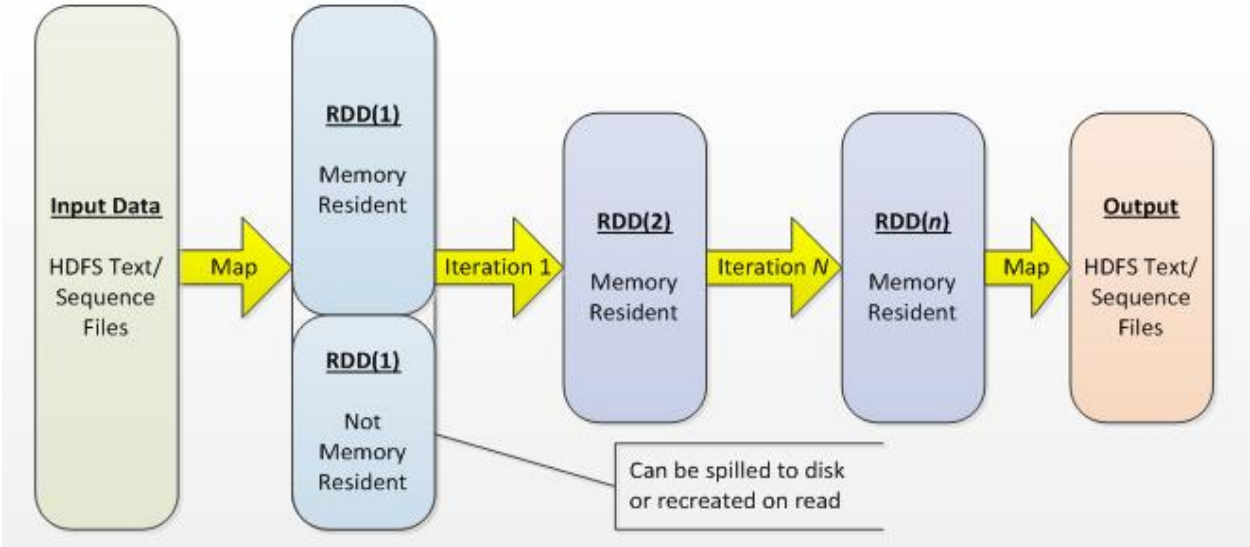


Figure 13 Architecture of the in-memory analytics engine, E Bay Technologies (2014)

After gaining a full understanding of the requirements, the data flow design of the data analytics prototype was changed to show the improvements that were made as shown in figure 15.

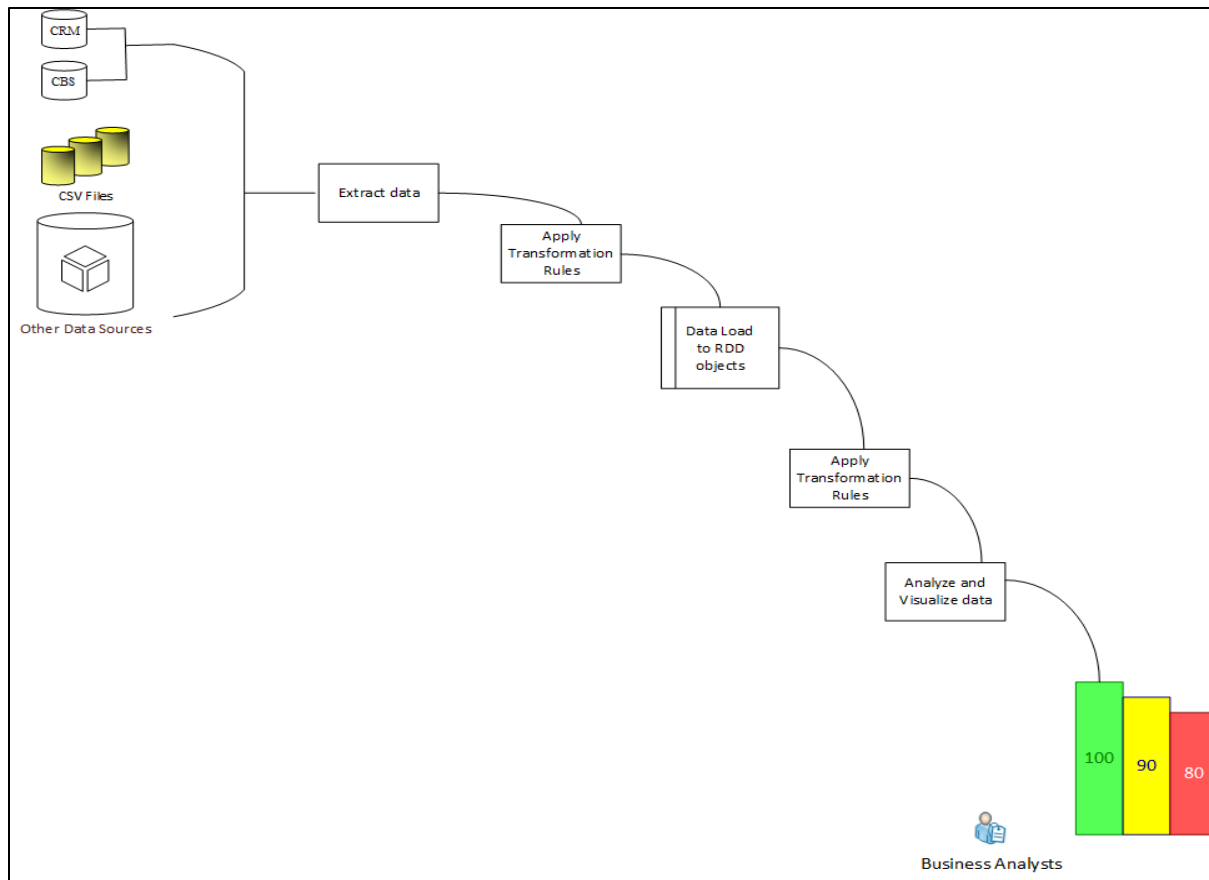


Figure 14 Current Data Analytics Process

From the figure, the processes involved getting data from the different source systems such as CRM and CBS using code that is written on Apache Zeppelin directly. RDD objects are then created from the data. This was then subjected to data any transformational rules. The results are finally analyzed by the business from the Apache Zeppelin interface.

3.5 Prototype Implementation and Evaluation

This section covers the activities that were done during the prototype implementation and evaluation.

3.5.1 Configuration Phase

This phase was focused on setting up the analytics environment by installing the required components and configuring the basic analytics tools that Apache Spark and Apache Zeppelin need. The following tools were installed for the in-memory platform to be fully operational:

3.5.1.1 Java Development Kit (JDK)

Apache Spark is built on Java and so it had to be installed for Spark to work. Java can also be used to develop applications that are based on Spark. For this project, java is used only to

support Spark but no application was developed using it. After installation, there was a need to setup its access from the environmental variables.

3.5.1.2 Scala

Scala is provided as one of the options that one can use to build analytics applications. Some of the items developed and presented on Zeppelin were done using Scala.

3.5.1.3 Apache Spark

Apache Spark is the core engine that hosts the in-memory logic that all the other components rely on. It also has its own built-in Hadoop File System which meant that there was no need to install and configure a standalone Hadoop system. Apache Spark was selected as a better choice due to its ability to process in memory compared to MapReduce which is strictly based on the disk. It required setting up environmental variables for it to work as expected. The table below shows the configurations done to Apache Spark.

Name	Value
spark.app.id	local-1497418834330
spark.app.name	Spark shell
spark.driver.host	10.0.1.237
spark.driver.port	1706
spark.executor.id	driver
spark.home	D:\Research_ML\Tools\spark\spark\bin\.
spark.master	local[*]
spark.repl.class.outputDir	C:\Users\john.mburu\AppData\Local\Temp\spark-bb2fc74c-993a-4686-ab2e-bb5fd59f6694\repl-99205864-e957-4fae-82f6-34fe64c8e741
spark.repl.class.uri	spark://localhost:1706/classes
spark.scheduler.mode	FIFO
spark.sql.catalogImplementation	hive

spark.submit.deployMode	client
SPARK_SUBMIT	true
awt.toolkit	sun.awt.windows.WToolkit
file.encoding	Cp1252
file.encoding.pkg	sun.io
file.separator	\
io.netty.maxDirectMemory	0
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.version	52.0
java.endorsed.dirs	C:\Java\jre\lib\endorsed

Table 9 Apache Spark Configuration

The figure below shows the home page after configuring the Spark instance.

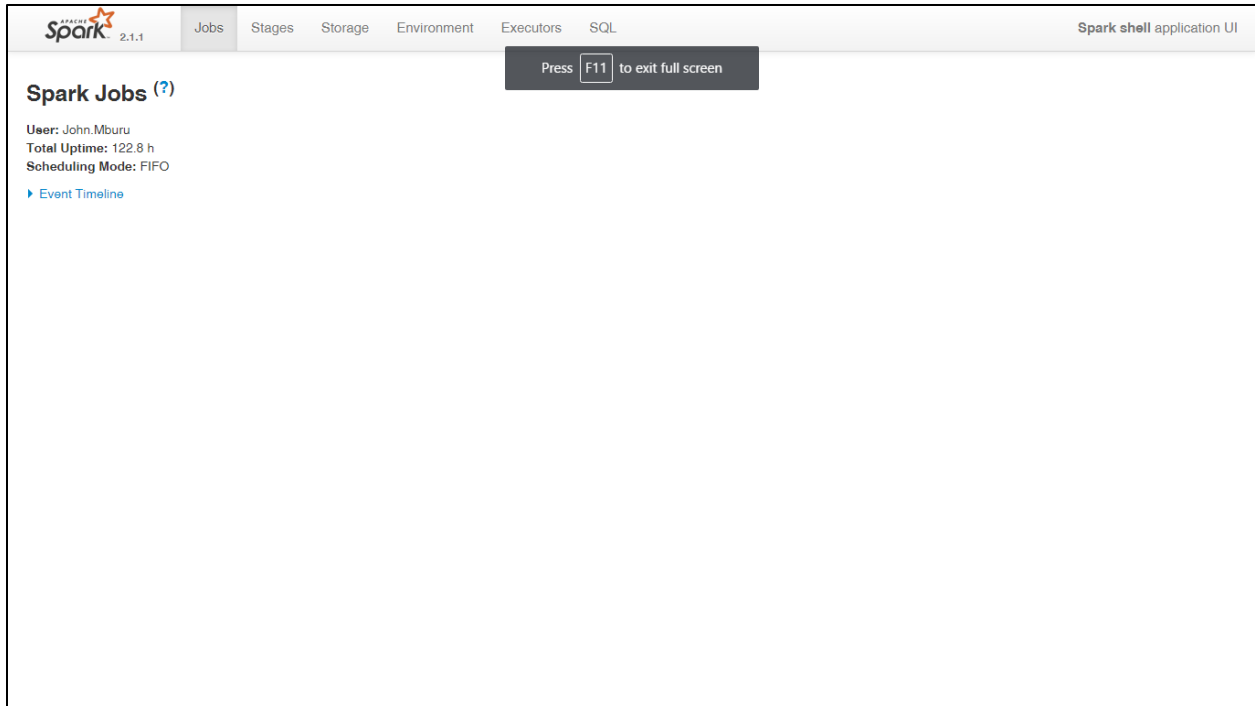


Figure 15 Apache Spark Home Page

3.5.1.4 Apache Zeppelin

This is a close relative of Apache Spark since they are both provided as open source products by Apache software foundation. It was primarily used to achieve data visualization in an in-memory environment. It works by having interpreters that work like compilers. Some of the supported platforms are Scala, R, Python, file among other platforms. It required setting their environmental variables for it to work.

The table below shows the configurations that were done to Zeppelin to allow it to work as a visualization tool for Apache Spark in-memory system.

Name	Value
namevaluezeppelin.anonymous.allowed	TRUE
zeppelin.conf.dir	C:\zeppelin-0.7.0-bin-all\conf
zeppelin.credentials.persist	TRUE
zeppelin.dep.localrepo	local-repo
zeppelin.encoding	UTF-8
zeppelin.helium.localregistry.default	helium
zeppelin.home	C:\zeppelin-0.7.0-bin-all
zeppelin.interpreter.connect.	30000

timeout	
zeppelin.interpreter.dir	interpreter
zeppelin.interpreter.group.or der	spark, md, angular, sh, livy, alluxio, file, psql, flink, python, ignite, lens, Cassandra, geode, kylin, elasticsearch, scalding, JDBC, HBase, big query, beam, pig, Scio
zeppelin.interpreter.localRe po	local-repo
zeppelin.interpreter.max.po olsize	10
zeppelin.interpreter.remoter unner	bin\interpreter.cmd
zeppelin.notebook.autoInter preterBinding	TRUE
zeppelin.notebook.azure.sha re	zeppelin
zeppelin.notebook.azure.use r	user
zeppelin.notebook.dir	notebook
zeppelin.notebook.homescre en.hide	FALSE
zeppelin.notebook.one.way. sync	FALSE
zeppelin.notebook.public	TRUE
zeppelin.server.addr	0.0.0.0
zeppelin.server.allowed.orig ins	*
zeppelin.server.context.path	/
zeppelin.server.port	8080
zeppelin.server.ssl.port	8443
zeppelin.ssl	FALSE
zeppelin.ssl.client.auth	FALSE
zeppelin.ssl.keystore.path	keystore
zeppelin.ssl.keystore.type	JKS
zeppelin.war	C:\zeppelin-0.7.0-bin-all\zeppelin-web-0.7.0.war
zeppelin.war.tempdir	webapps
zeppelin.websocket.max.tex t.message.size	1024000

Table 10 Apache Zeppelin Configuration

The home page for Apache Zeppelin showing a few notebooks after the configurations was done, is shown in the figure below.

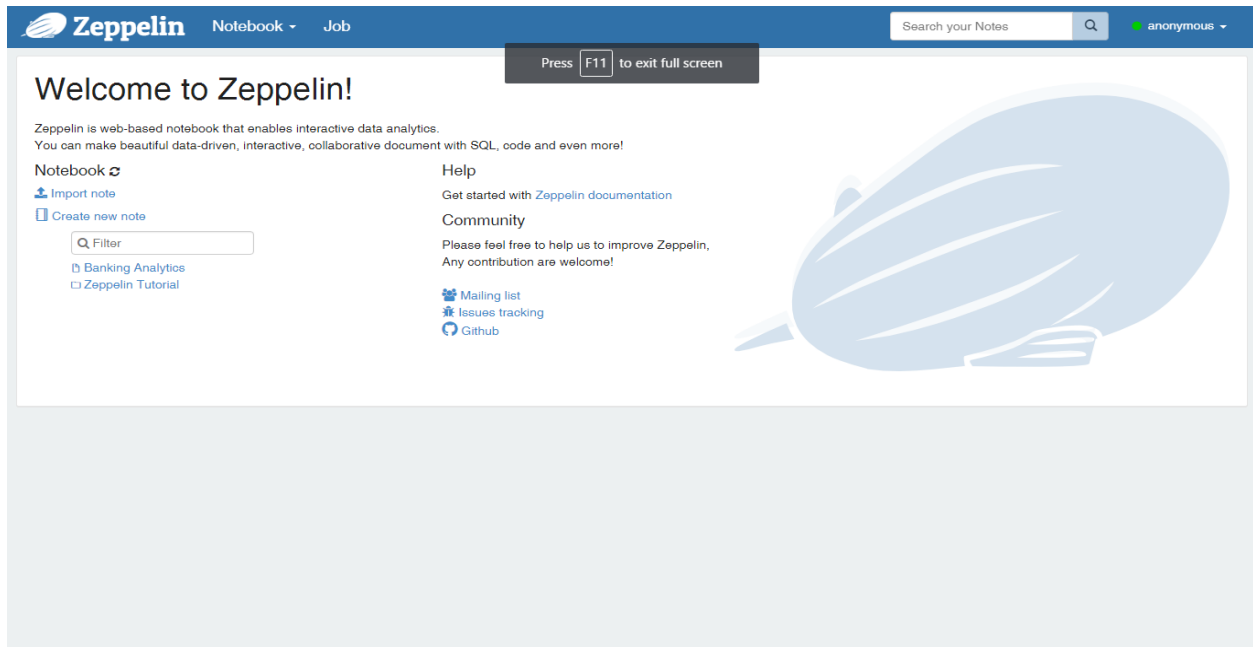


Figure 16 Apache Zeppelin Home Page

3.5.1.5 Python

Python was supported as one of the programming languages in Zeppelin that can allow custom development of visualization items on Zeppelin notebooks.

3.5.1.6 Simple Build Tool (SBT)

It was used to allow compiling of java and Scala code. It required setting its environmental variable for it to work.

3.5.1.7 Win Utils

Apache Spark by default works better on Linux. Win Utils was required to allow a user to run Linux commands on windows.

3.5.2 Data Validation and Simulation Phase

Some of the activities that were done in this phase include testing and validation of the data that was being populated, presenting the solution to the target population to ensure they developed interest. This also allowed them to discover any customizations that were not unearthed in the early stages.

Most tests focused on verifying if all the data that was to be moved from a source to the target had been done and if the transformation rules had been applied as expected. This formed the ETL/warehouse testing. Since Apache Spark supports R, it was used to test that data was clean by implementing various R packages that are used to cleanup data. Some of these tests include removal of NA's or NaN's from the raw data which would have otherwise led to misleading results.

The second data validation technique involved using simple calculations on the data by writing some R codes to calculate the count of loans, average loan amounts for a specific period from which results were compared to what was expected. The table below gives a summary of these tests.

Data Validation Exercise	R Commands issued	Expected Results	Actual Results
Remove NA/NaNs and Confirm Record count	<pre>>testData<-read.csv("C:/zeppelin-0.7.0-bin-all/bin/loans.csv") >sum(sapply(testData,length)) >completeData<-complete.cases(testData) >sum(sapply(completeData,length))</pre>	Before cleanup: 429114 After Cleanup: 25242	Before cleanup: 429114 After Cleanup: 25242

Table 11 Data Validation and Simulation Using R

Load/performance testing which was the main objective of this project formed the last form of validation done. The results or output was used to gauge the performance, scalability of the system under different loads.

For the implementation stage, loans data that had already been transformed and loaded into Apache Spark for use by Zeppelin to implement various analysis to the data. The sample code on figure 18 was used to extract the data for the loans using Apache Spark's Scala interpreter. More code can be found under the appendix section of this report.

```

val bankText = sc.textFile("loans.csv")

case class Bank(OurBranchID:String, AccountID: String, ProductID:String, CurrencyID:String, ExchangeRate:String, Classification:String, AmountFinanced:Double,
  InstallmentAmount:Double, Frequency:String, OutstandingBalance:Double, OutstandingBalance_Local:Double, ArrearsAmount:Double, ArrearsAmount_Local:Double, ArrearsDays
  :Integer, OfficerName:String, ProductTypeID:String, LastCreditDate:String
)

val bank = bankText.map(s=>s.split(",")).filter(s=>s(0)!="OurBranchID").map(
  s=>Bank(s(0).replaceAll(" ", ""),
    s(1).replaceAll(" ", ""),
    s(2).replaceAll(" ", ""),
    s(3).replaceAll(" ", ""),
    s(4).replaceAll(" ", ""),
    s(5).replaceAll(" ", ""),
    s(6).toString.toDouble,
    s(7).toString.toDouble,
    s(8).replaceAll(" ", ""),
    s(9).toString.toDouble,
    s(10).toString.toDouble,
    s(11).toString.toDouble,
    s(12).toString.toDouble,
    s(13).toInt,
    s(14).replaceAll(" ", ""),
    s(15).replaceAll(" ", ""),
    s(16).replaceAll(" ", ""))
)

// convert to DataFrame and create temporal table
bank.toDF().createTempTable("bank")
bankText: org.apache.spark.rdd.RDD[String] = loans.csv MapPartitionsRDD[1294] at textFile at <console>:49
defined class Bank
bank: org.apache.spark.rdd.RDD[Bank] = MapPartitionsRDD[1297] at map at <console>:54
warning: there was one deprecation warning; re-run with -deprecation for details
Took 3 sec. Last updated by anonymous at June 20 2017, 1:41:20 PM.

```

Figure 17 Apache Spark's - Zeppelin Scala Sample code

3.5.3 Customization Phase

During the plan, discovery, and validation phases, additional requirements that could be identified but were not achieved on earlier stages were handled on this phase by extending the solution through customizations. Some new dimensions or facts were added to the list at this stage to allow more visualization options in Apache Zeppelin. Another activity in this phase involved conclusion into the development of the final prototype that was to confirm that the in-memory data mining algorithms are in place and working as per the expectation. Customizations were also done to allow the control environment to be the same as the project setup by ensuring that the amount of memory that was used in both setups was the same. This therefore could allow an evaluation to be done to the prototype by pushing the same job to the control environment and the prototype.

3.5.4 Deployment and Evaluation

This phase was focused on migrating all the components to the live environment, creation of users and their role profiles, establishing data recovery plans and finally conducting training sessions to the end users and IT support users. The Spark production engine was as shown in the figures that follow that show the executors, completed queries, stages and jobs that were done pushed by Zeppelin to Apache Spark.

Since this was a prototype, there was no need to create users after the deployment was done. The visualization platform allowed anonymous login of users if they could be able to access the

server that was hosting the notebooks from a browser. The Apache Zeppelin was accessible on port 8080 from the server that was picked as the live environment.

This also marked the evaluation stage where analytics users would get to evaluate if there will be any performance improvements to their analytics process.

CHAPTER 4: RESULTS

This section is about the results of the evaluation that was done by comparing prototype against the traditional processes that were used for data analytics processes.

4.1 Process Evaluation

This was done by first comparing the processes that were involved in both cases (traditional and prototype) to achieve data analysis from the raw data. The table below was used to compare these processes.

Traditional Processes	Prototype Processes
1. Fetch data from transactional system 2. Apply cleanup and transformation 3. Apply visualization	1. Refresh on Visualization page to update/refresh all the data.

Table 12 Process Improvement Evaluation

From table 12, there was notable process improvement since the number of processes had reduced from a total of three to only one after automation. This therefore directly translated to a reduction in the amount of time it took to understand the pattern that is hidden in a data sample.

4.1.1 Multi User Data Access Improvement

The prototype developed used a web based visualization framework that can be accessed on any browser in a networked environment. In the traditional setup where data was accessed using excel sheets means that this was an improvement since excel sheets are not accessible on the web. Controlled access to Zeppelin environment can be done by ensuring that only authenticated users can access the dashboards. Changes to the data that is being visualized can lead to different presentations to consumers of the data in the traditional setup. Using Zeppelin this is eliminated directly by using temporary data that is deleted after a session has been terminated which means there is no way that users can be able to make changes to the data without the necessary organizational procedures.

4.1.2 Open Source Technology

All the tools that were used to develop this prototype are open source which means if a company was to decide on them as an in-memory analytics solution, they would be of less cost. The costs will mostly go to the developers who will be paid to develop the solution in accordance with the requirements of that company.

4.1.3 Flexible Visualization Options

Apache Zeppelin allowed built-in support for many languages and options compared to what was available in the traditional setup using excel. It allowed ODBC connection to databases, R, Spark using Scala and many others.

4.2 Performance Improvement

Testing the performance of the prototype involved setting up various test options. First one was loading data that was in a CSV format of about 26 million rows.

4.2.1 Data Loading

To start with, the data loading was done to a database management system and logged to compare the time it took to load the data. The logs are as shown on figure 30.

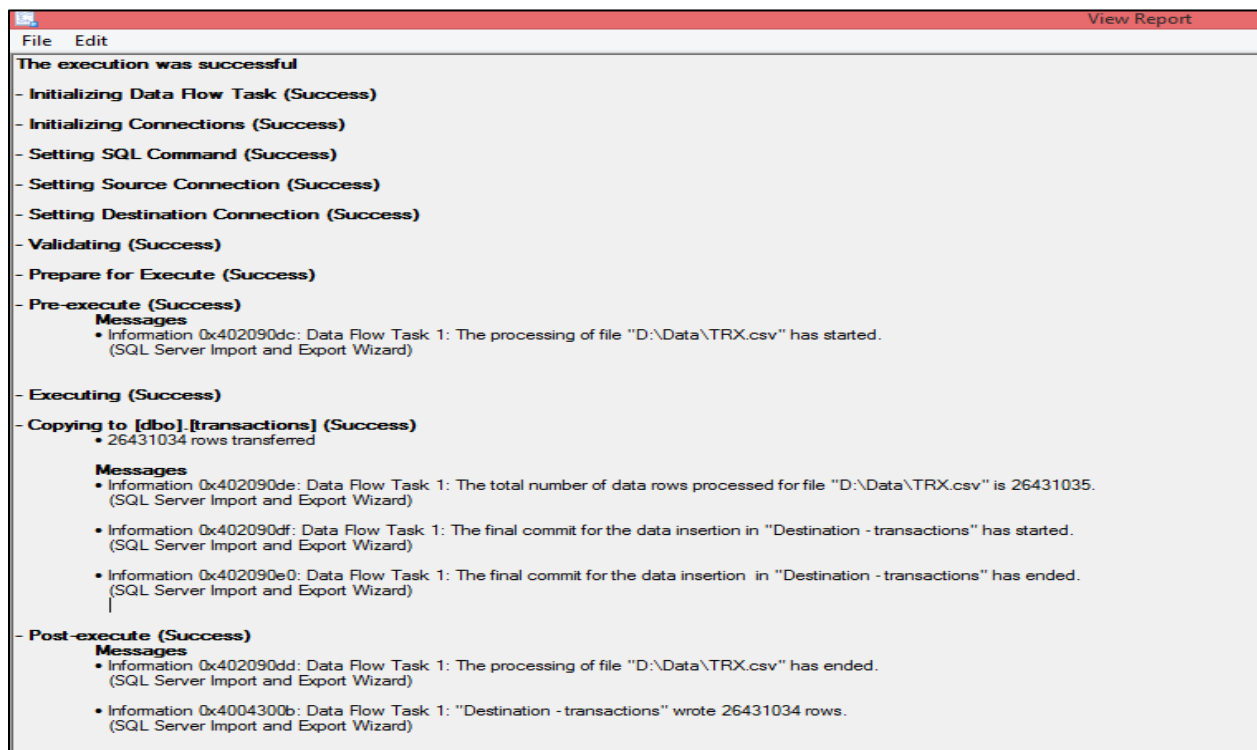


Figure 18 DBMS Data Load

The process took about 4 minutes and 34 seconds.

The same data was loaded and converted into Spark, converted into RDD by running Scala code on Apache Zeppelin notebook's page. After loading the RDD object, it was converted to a data frame from where a table was created. Data loading was as shown on the figure 31 below.

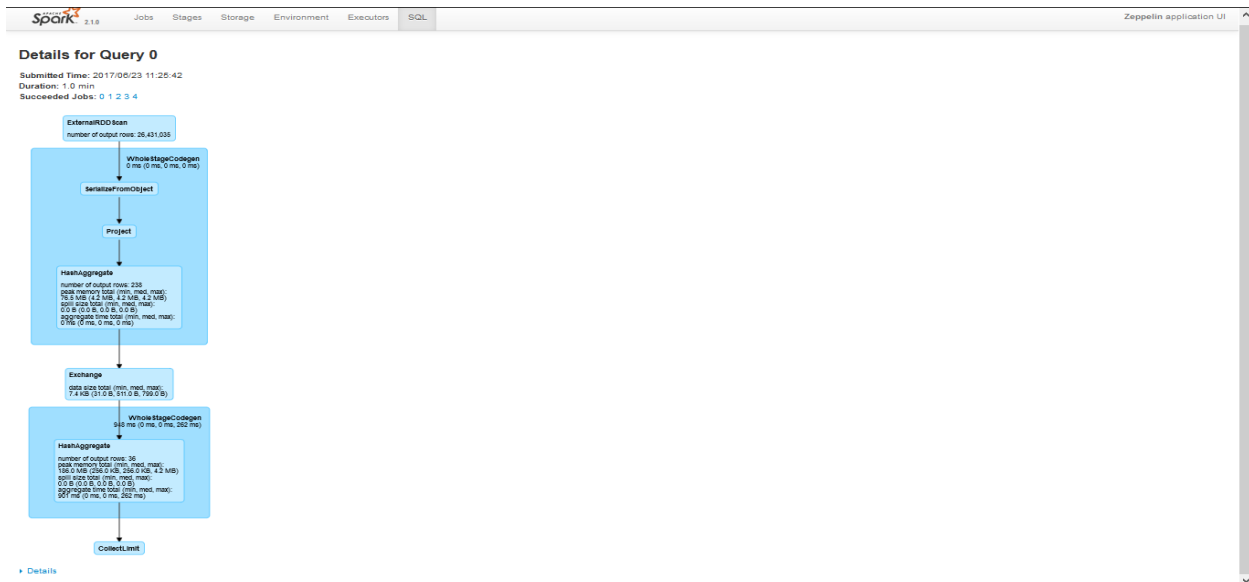


Figure 19 Apache Zeppelin/Spark Data Load

It is from the table that we could be able to run various analytic commands. The process took about 1 minute 38 seconds. Figure 32 below captured this.

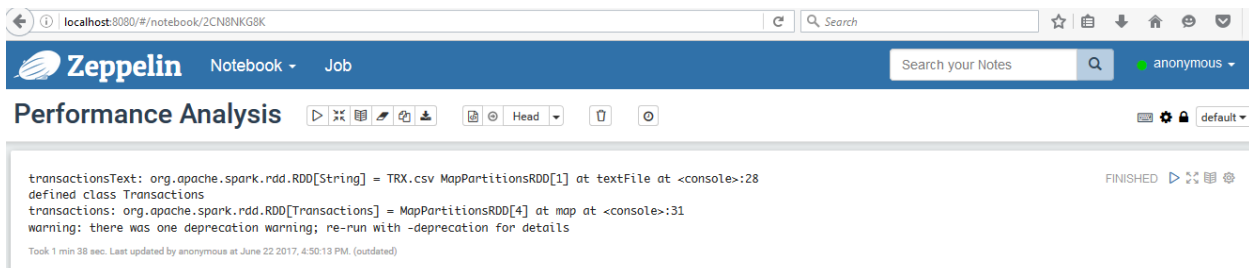


Figure 20 Zeppelin Load

4.2.2 Prototype Evaluation

The next level of test was on the analysis engine by pushing some queries to analyze the data using a selected dimension from the dimension list. At first the data that was already loaded into a table in a database was queried to get the total amounts per branch for the 26 million rows. The results showed that it took SQL server about 3 minutes and 38 seconds. The results are as shown on figure 33.

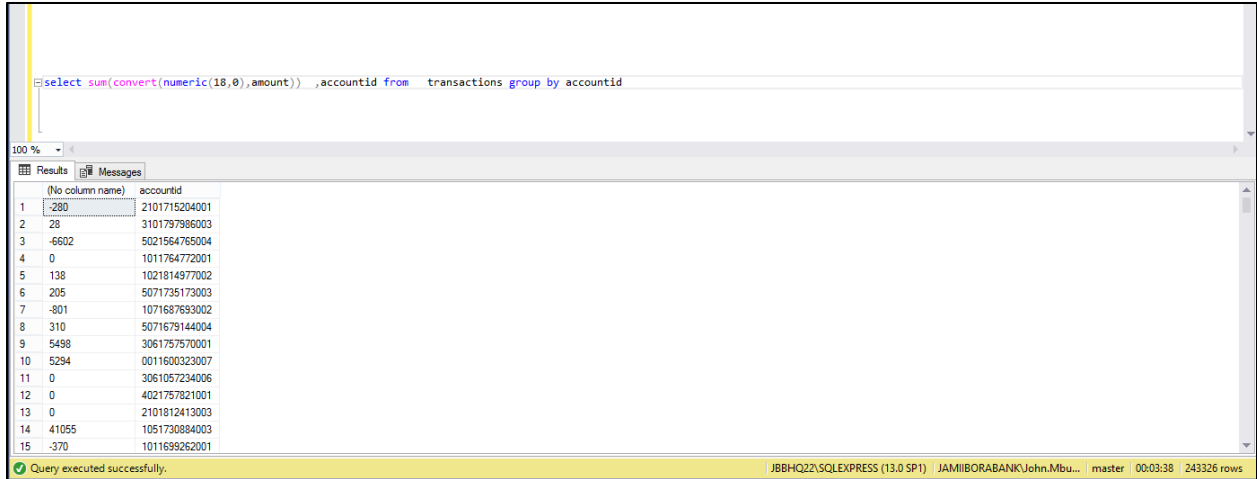


Figure 21 DBMS Analysis

The same analysis was done on Apache Zeppelin notebook and the results showed that it took about 1 minute and 3 seconds to execute. The output is as shown on figure 34 below.

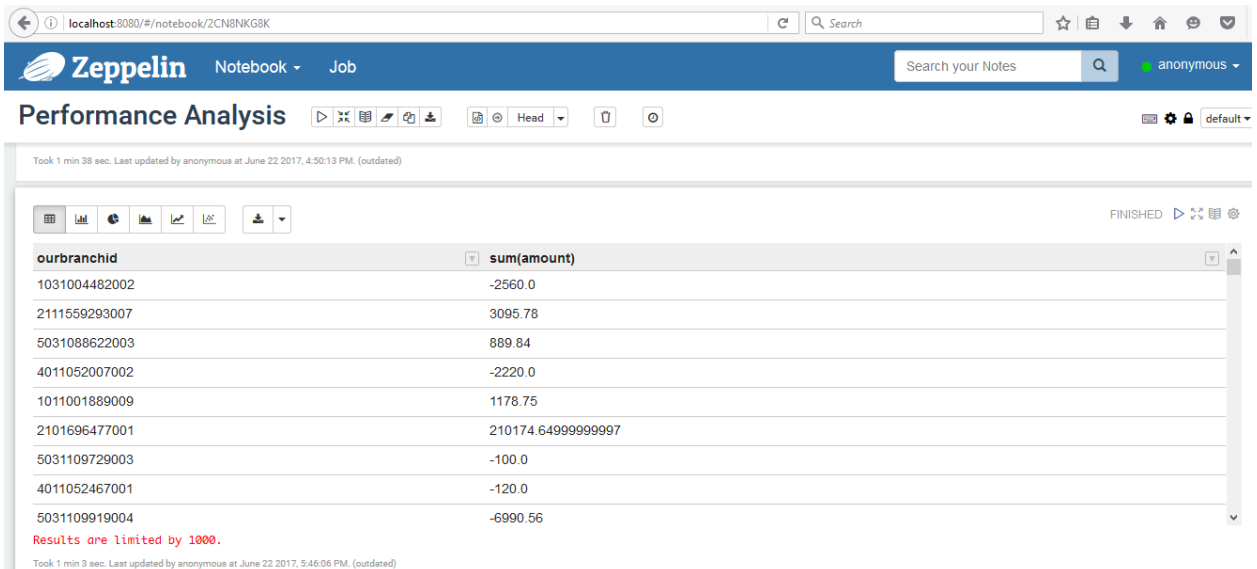


Figure 22 Zeppelin Analysis

The code behind the analysis on figure 34 is as shown on figure 35 below.

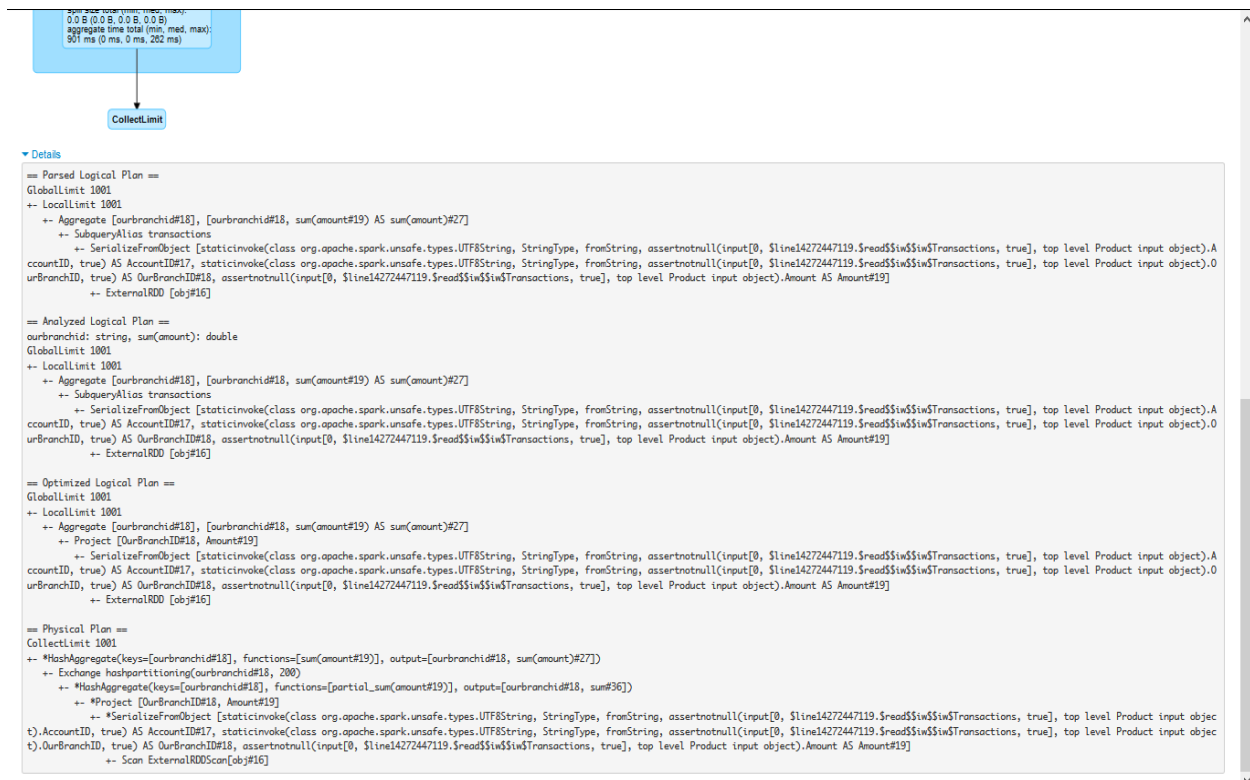


Figure 23 Zeppelin Analysis Details

The analysis was repeated and recorded with different dimensions and recorded as shown on the table 13 which were labeled as query 1 all through to 3.

Process/Activity	Native DBMS Time (Seconds)	Apache Spark Time (Seconds)	Ratio
Data Loading	274	98	2.795918
Q1 - Data Analysis – Measuring Value of transactions by account dimension	218	63	3.460317
Q2 - Data Analysis – Measuring the number of transactions by account dimension	234	57	4.105263
Q3 - Data Analysis – Measuring the average value of transactions			4.811321

using the account dimension	255	53	
-----------------------------	-----	----	--

Table 13 Performance Comparison

Some analysis of the performance in form of a chart is as shown in figure 36 that plots a line graph comparing the time in seconds of a traditional tool (DBMS) against Spark.

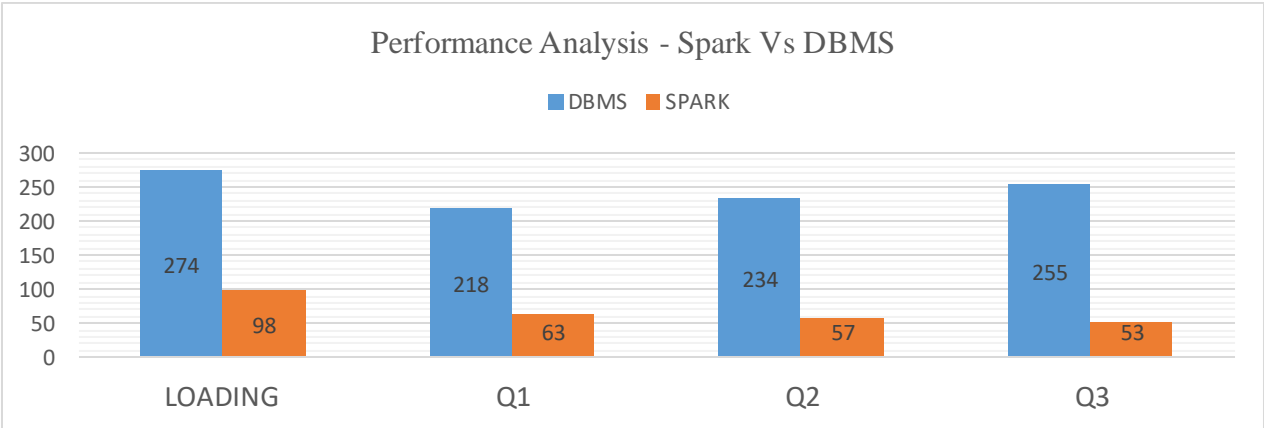


Figure 24 Performance Comparison Chart

From the literature that was reviewed/synthesized, it showed possibilities of carrying out the experiment to test if banking analytics could be improved using an in-memory data warehouse prototype. The project was therefore recommended to test this hypothesis and show the results which were to either agree or disagree with the literature.

CHAPTER 5 CONCLUSIONS

From the evaluation results of the prototype, it was clear that there were improvements in data analytics if Apache Spark and Zeppelin were to be decided on as tools to aid in analytics. The processes were improved from three stages to one stage. Visualization tools that could be accessed by multiple users on the web with an ability to allow authentication gave a plus to Apache tools compared to excel as an analysis tool. Costs that would be associated with an Apache implementation would present a cheaper alternative as compared to other commercial in-memory alternatives since Apache is open source.

The project has achieved its objectives of developing a prototype that can reduce the latency of handling analytics data in a banking setup. The prototype was evaluated and the results compared to a traditional tool which showed improvements in latency reduction when using an in-memory system as compared to a traditional tool. The results were tried in another fields that was not necessarily banking to show that these results could be generalized to other industries. These results therefore show that in-memory tools can be used to improve performance of data analytics tools.

5.1 Latency Reduction

A major improvement which was the main purpose of this research was on performance improvement or latency reduction. The details of these improvements are shown in the figures under the performance evaluations section. From the experiment, therefore, it can be observed that spark was 3 times faster in data loading and Q1, 4 times faster in Q2 and 5 times faster in Q3 as compared to DBMS.

5.2 Generalization to other Industries

These results can be generalized to other industries if the amount of data that would be loaded is less than the amount of memory allocated to Apache Spark for analysis.

CHAPTER 6: RECOMMENDATIONS

This chapter describes the achievements that the project made in relation to what other researchers have done in this field of study. Some recommendations as also discussed to show how other researchers can improve on the results of this study by giving out a list of limitations that were identified and that can be considered by other researchers.

6.1 Recommendations

From the study, it can be recommended that the in-memory tools can be used to improve on the latency issues that are common with use of the traditional analytics tools that store data in disk.

6.2 Limitations of the Study

The study was limited to only performance of the tools but other areas were not compared such as drill down which Apache Zeppelin is yet to avail.

6.3 Further Study

More studies need to be in this area to show how in-memory tools are better than traditional tools other than in performance or speed of execution. For example, it needs to be clear if Spark can be used to replace a normal database for a transaction processing system. There is also a need to evaluate if addition of more clusters to the Apache spark system would lead to direct performance improvements since this report was based on a single cluster setup.

REFERENCES

1. Xinhui T., Rui H., Lei W., Gang L., Jianfeng Z. (2015). Latency Critical Big Data Computing in Finance. *The Journal of Finance and Data Science*. Vol 1, Issue 1, pp. 33-41.
2. Utkarsh S., Santosh G. (2015). Impact of Big Data Analytics on Banking Sector: Learning for Indian Banks. *Procedia Computer Science*. Vol 50. pp. 643-652.
3. Orestis p., Arun R., Kenneth A., (2015). Rethinking SIMD Vectorization for In-Memory Databases. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. pp. 1493-1508.
4. Matei Z., Mosharaf C., Et al (2012). Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. *Association of Computing and Machinery Digital Library*. Pp. 2-2
5. Santhosh B., Renjith P. (2013). Next Generation Data Warehouse Design with Big Data for Big Analytics and Better Insights. *Global Journal of Computer Science and Technology Software & Data Engineering*. Vol 12, No. 7
6. Dale, R., Tara, N., Nayem R. (2012). Practical Implications of Real Time Business Intelligence. *Journal of Computing and Information Technology*. Vol 20, No. 4, pp. 257–264.
7. Nayem R., Dale R., Shameem A., Fahad A. (2014). Emerging Technologies in Business Intelligence and Advanced Analytics. *ULAB Journal of Science and Engineering*. Vol 5 No. 1.
8. Depali D. (2016). Apache Spark and Big Data Analytics for Solving Real World Problems. *International Journal of Computer Science Trends and technology*. Vol. 4 No. 2.
9. Hsinchun C., Roger H., Veda C. (2012). *Business Intelligence and Analytics: From Big Data to Big Impact*. MIS Quarterly. Vol. 36 No. 4, pp. 1165-1188.
10. Janina P., Ion L. (2012). Real-Time Business Intelligence for The Utilities Industry. *Database Systems Journal*. Vol. 3, No. 4.
11. Adrian M., Ovidiu R. (2012). Banking Intelligence Accelerator - Decision Support. *Database Systems Journal*. vol. 3, No. 2.

12. Haug, A., Zachariassen, F., Liempd, D. (2011). The cost of poor data quality. *Journal of Industrial Engineering and Management*, Vol. 4, No. 2, pp 168-193
13. Hossein M., Ardeshir B. (2010). A New Framework for Evaluating the Functional Capabilities of Intra-EAI Technologies. *Journal of Systems Integration*. Vol. 1, No. 4, pp. 50-62.
14. James E., Jennifer A., (2010). MS in Business Intelligence. *Business Intelligence Journal*. Vol. 15, No. 1.
15. Nikhil D., Gautam N., Hillol D. (2013). Analysis of Data Quality and Performance Issues in Data Warehousing and Business Intelligence. *International Journal of Computer Applications*. Vol. 79, No. 15.
16. Berendt, B. (2007). Intelligent business intelligence and privacy: More knowledge through less data? *Business Intelligence: Methods and Applications* (pp. 63-79).
17. Bernhard, W. & Maria O. (2015). The Impact of Business Intelligence on the Quality of Decision Making – A Mediation Model. *Paper presented at the Conference on enterprise Information Systems / Centeris*. Vol 64, No. 1 2015, pp 20-35.
18. Bogdan N., (2013). Business Intelligence Systems. University of Economic Studies, Bucharest, Romania. *Database Systems Journal*. Vol. 4, No. 4, pp 40-70.
19. Mayank G., Rana M., Et al (2013). Turn Hours into Seconds: BI Paradigm with In-Memory Analytics. *International Journal of Software and Web Sciences (IJSWS)*. Vol 1. No. 6. Pp 24 – 27.
20. Nur Z., Suraya M., Haslina H., Et al (2015). Conceptual Framework of Business Intelligence Analysis in Academic Environment Using Birt. *ARPJN Journal of Engineering and Applied Sciences*. Vol. 10, No. 23.
21. Muriithi, G. M., Kotzé, J. E. (2013). A conceptual framework for delivering cost effective business intelligence solutions as a service. *In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*. Vol. 45 No. 1, pp. 96-100.
22. Azvin, B., Cui, Z., Nauck D. (2005). Towards real-time business intelligence. *BT Technology Journal*. Vol. 23, No. 3, pp 214–225.

23. Celina, M., Ewa, Z., etal (2002). Approach to Building and Implementing Business Intelligence Systems. *Interdisciplinary Journal of Information, Knowledge, and Management*. Vol. 2, pp 50–85.
24. Sahay, B., Jayanthi, R. (2008). Real time business intelligence in supply chain analytics. *Information Management & Computer Security*. Vol. 16 No. 1, pp. 28-48.
25. Zeljko, P. (2007). Just-in-Time Business Intelligence and Real-Time Decisioning. *International Journal of Applied Mathematics and Informatics*. Vol. 1, No. 1, pp 106-111.
26. Mansmann S., Neumuth M., (2007). An OLAP Technology for Business Process Intelligence: Challenges and Solutions. *Data Warehousing and Knowledge Discovery*, Vol. 4654, pp 111-122.

APPENDICES

7.1 Appendix I: User Manual

The system can be hosted on a server running either on windows or Linux operating system which makes it accessible from a browser or over the internet. Sample figures 19 to 29 represents the combination of several measures and dimensions whose output simulate some few analyses on the loan fact.

They capture the various measures (outstanding loan balances, Arrears amount, arrears days, non-performing loans, number of loans) against dimensions (branch, Product type, currency, loan classification and date). These were identified as key reports by the business users that were identified.

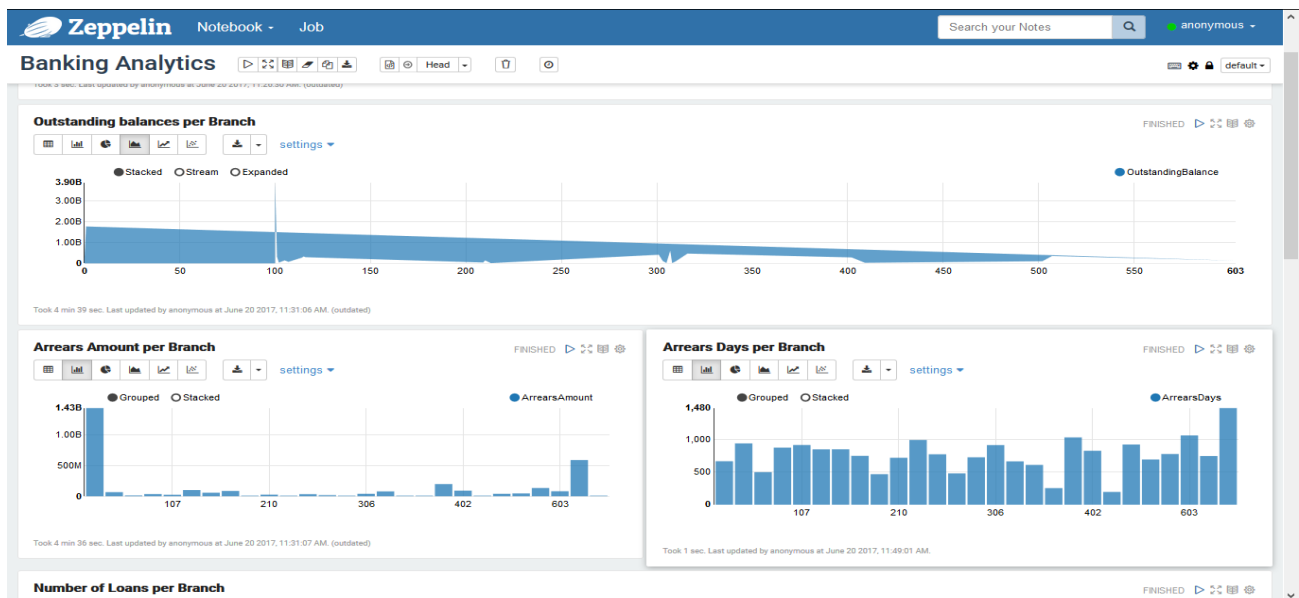


Figure 25 Zeppelin Banking Analytics Notebook – Loan fact per branch dimension analysis

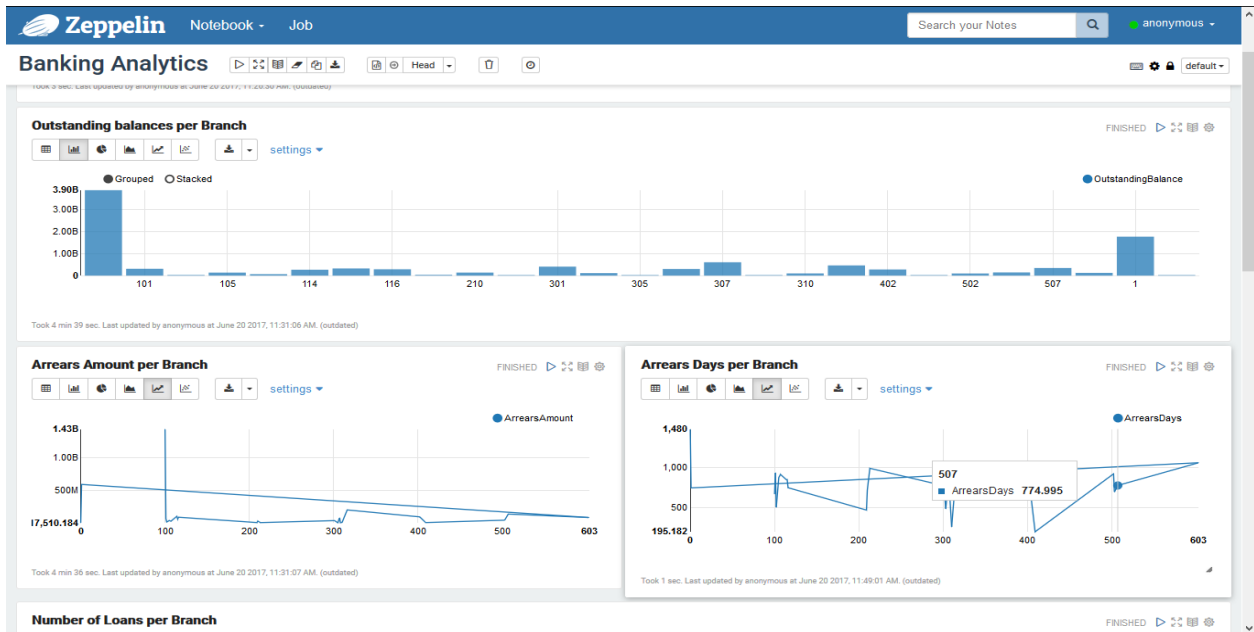


Figure 26 Zeppelin Banking Analytics Notebook – Loan fact per branch dimension analysis

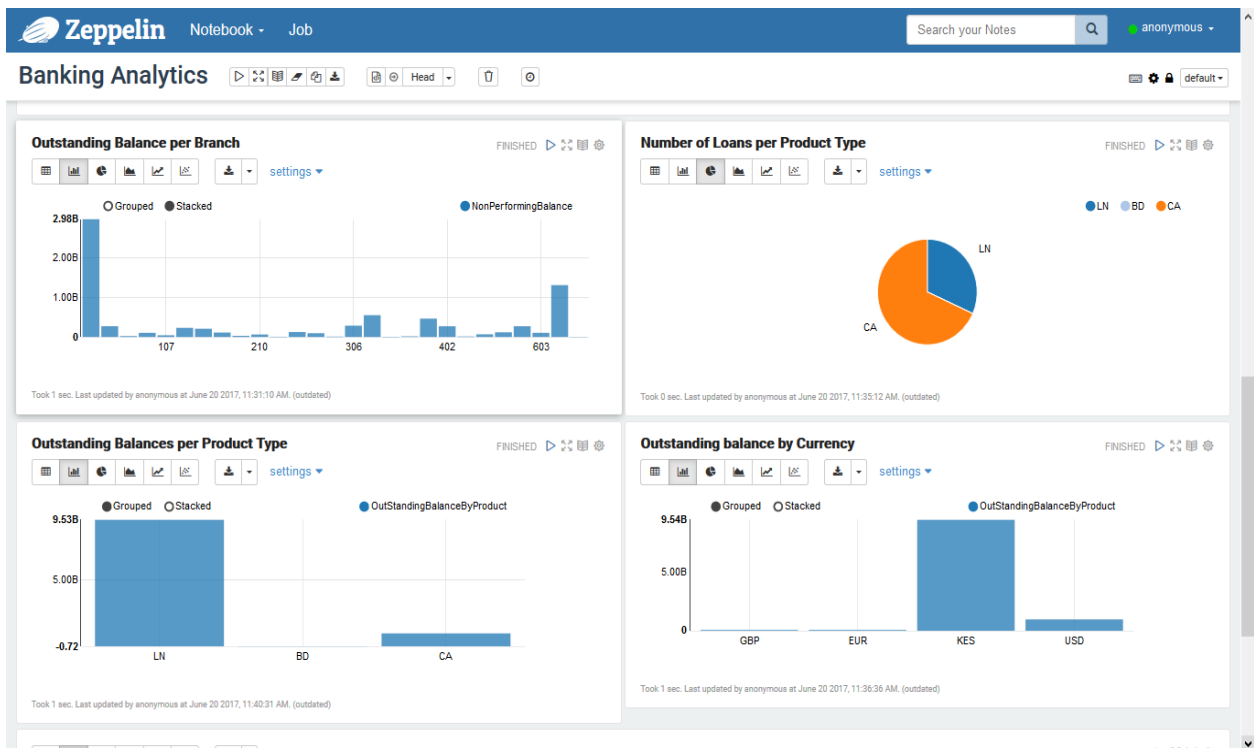


Figure 27 Zeppelin Banking Analytics Notebook – Loan fact per branch, productype, and currency dimensions analysis.

Figure 22 shows a graph that visualizes channel transactions by branch code dimension for each of the channels and the code that was used to load the data into Zeppelin.

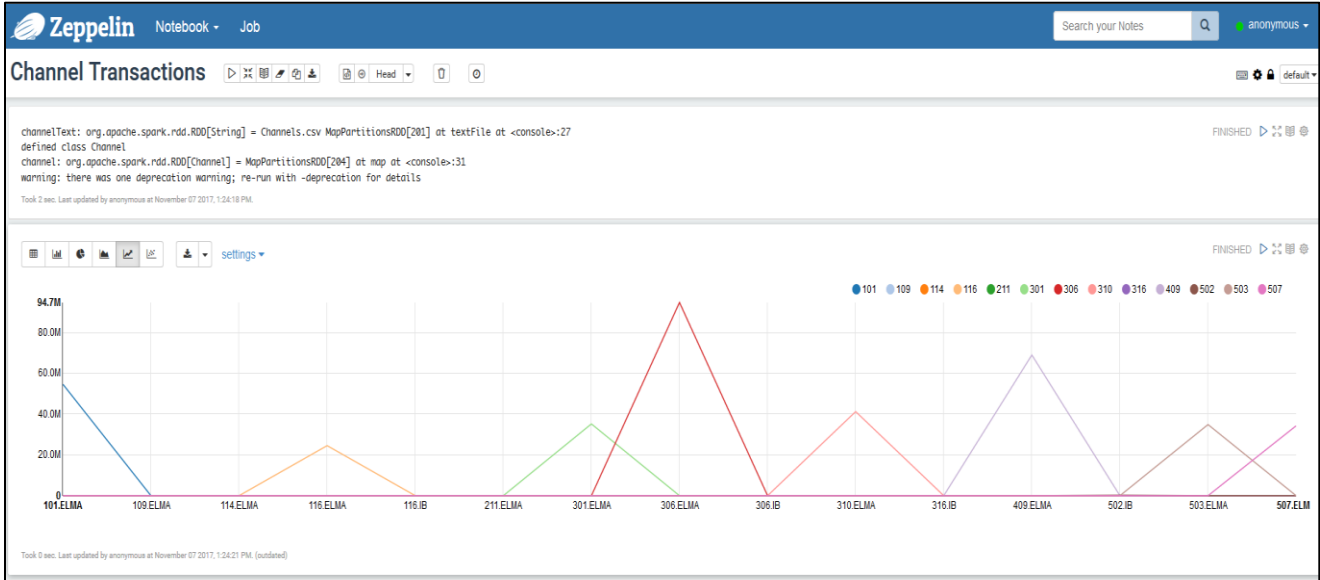


Figure 28 Zeppelin Banking Analytics Notebook – Measuring transactions fact by channel type dimension.

Figure 23 measures the average pay per department dimension from the human resource data.

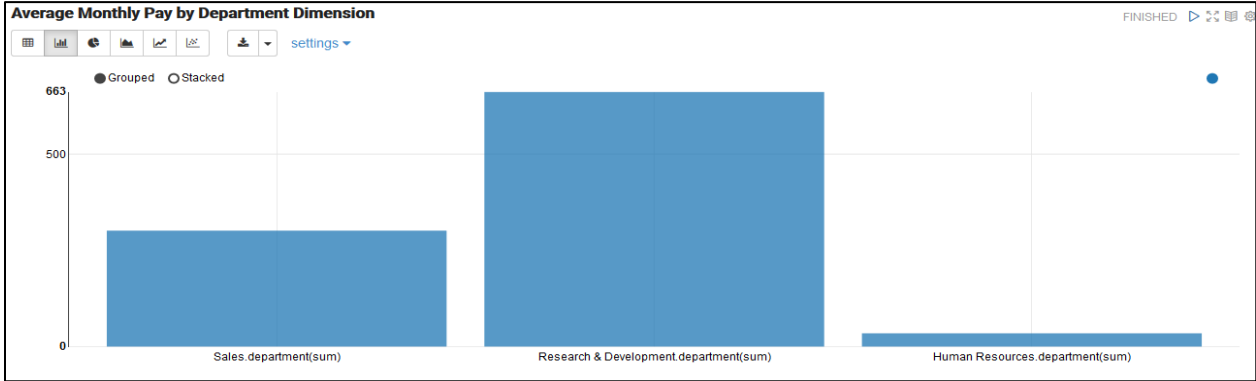


Figure 29 Zeppelin Banking Analytics Notebook – HR

Figure 24 shows the relationship between gender, age and employee attrition which means that one measure(attrition) was analyzed by looking at the gender and age dimensions.

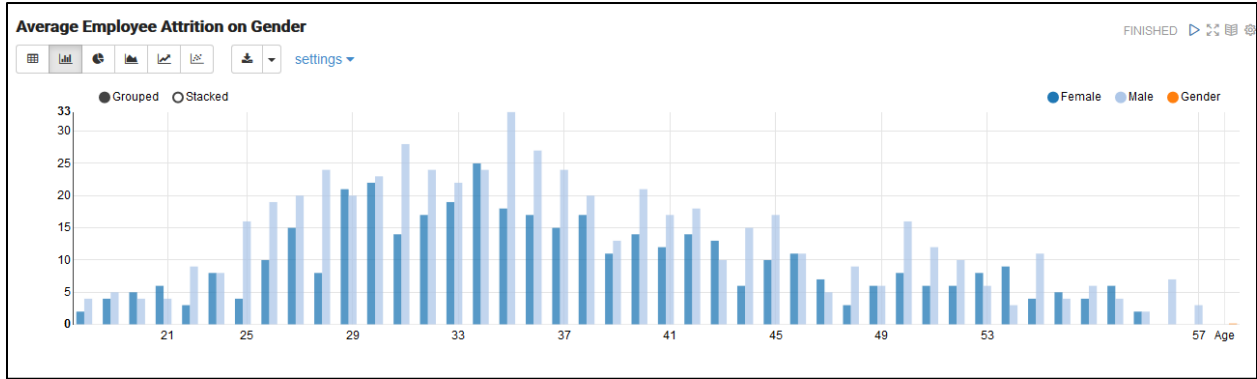


Figure 30 Zeppelin banking Analytics Notebook – HR

Figure 25 looked at the average pay per employee by their education level and job level at the company.

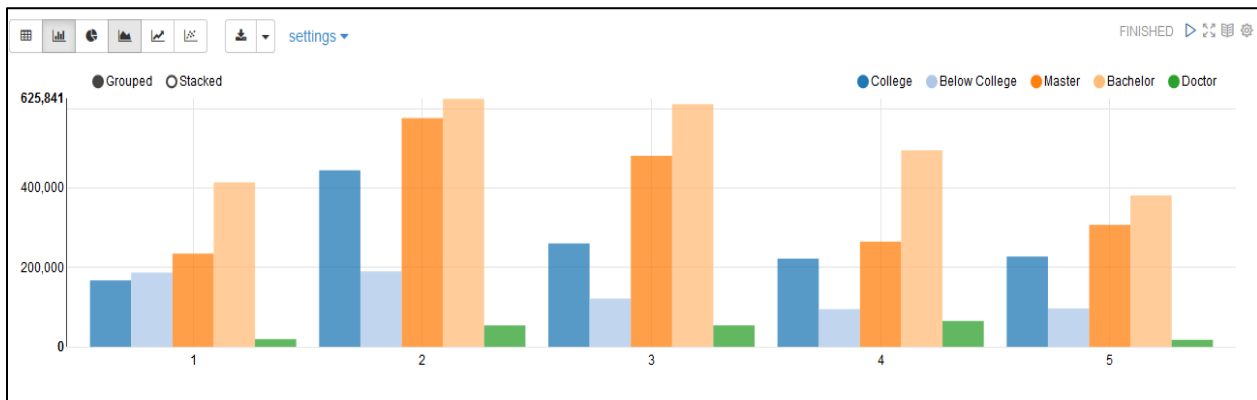



Figure 31 Zeppelin Banking Analytics Notebook - HR

Databricks Spark 2.1.0													Zeppelin application UI		
Executors															
Summary															
	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write			
Active(1)	5	89.9 KB / 428.8 MB	0.0 B	4	4	0	275	279	12 min (14 s)	2.3 GB	0.0 B	13.5 MB			
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B			
Total(1)	5	89.9 KB / 428.8 MB	0.0 B	4	4	0	275	279	12 min (14 s)	2.3 GB	0.0 B	13.5 MB			
Executors															
Show 20 entries															
Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump
driver	10.0.1.97:17556	Active	5	89.9 KB / 428.8 MB	0.0 B	4	4	0	275	279	12 min (14 s)	2.3 GB	0.0 B	13.5 MB	Thread Dump
Showing 1 to 1 of 1 entries															

Figure 32 Spark Executors

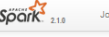
 2.1.0 Jobs Stages Storage Environment Executors **SQL** Zeppelin application UI

SQL

Completed Queries

ID	Description	Submitted	Duration	Jobs
4	take at NativeMethodAccessorImpl.java:0	+details 2017/06/23 12:51:34	1.9 min	8
3	take at NativeMethodAccessorImpl.java:0	+details 2017/06/23 11:33:45	52 s	7
2	take at NativeMethodAccessorImpl.java:0	+details 2017/06/23 11:32:01	57 s	6
1	take at NativeMethodAccessorImpl.java:0	+details 2017/06/23 11:30:33	40 s	5
0	take at NativeMethodAccessorImpl.java:0	+details 2017/06/23 11:25:42	1.0 min	0 1 2 3 4

Figure 33 Spark SQL Completed Queries

 2.1.0 Jobs Stages Storage Environment Executors **SQL** Zeppelin application UI

Stages for All Jobs

Completed Stages: 14
2 Fair Scheduler Pools

Pool Name	Minimum Share	Pool Weight	Active Stages	Running Tasks	SchedulingMode
default	0	1	0	0	FIFO
fair	0	1	0	0	FAIR

Completed Stages (14)

Stage Id	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
17	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 12:53:28	0.3 s	1/1			26.5 KB	
16	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 12:51:38	1.8 min	18/18	549.9 MB			4.9 MB
15	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:34:38	74 ms	1/1			26.5 KB	
14	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:33:46	52 s	18/18	549.9 MB			4.9 MB
13	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:32:57	66 ms	1/1			20.1 KB	
12	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:32:02	55 s	18/18	549.9 MB			3.7 MB
11	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:31:12	75 ms	1/1			22.8 KB	
10	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:30:33	39 s	18/18	549.9 MB			4.2 MB
9	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:28:40	0.7 s	75/75			8.4 KB	
7	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:26:39	1 s	100/100			6.8 KB	
5	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:26:39	0.3 s	20/20			2.1 KB	
3	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:26:38	0.5 s	4/4			228.0 B	
1	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:26:33	3 s	1/1				
0	default	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/06/23 11:25:45	46 s	18/18	549.9 MB			17.5 KB

Figure 34 Spark - Zeppelin Stages

Spark 2.10 Jobs Stages Storage Environment Executors SQL Zeppelin application UI

Spark Jobs (?)

User: John.Mburu
 Total Uptime: 1.7 h
 Scheduling Mode: FAIR
 Completed Jobs: 9
[Event Timeline](#)

Completed Jobs (9)

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
8 (zeppelin-20170821-203128_1526260064)	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/08/23 12:51:36	1.8 min	2/2	19/19
7 (zeppelin-20170821-203128_1526260064)	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/08/23 11:33:46	52 s	2/2	19/19
6 (zeppelin-20170821-203128_1526260064)	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/08/23 11:32:02	55 s	2/2	19/19
5 (zeppelin-20170821-203128_1526260064)	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/08/23 11:30:33	40 s	2/2	19/19
4 (zeppelin-20170821-203128_1526260064)	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/08/23 11:28:40	0.7 s	1/1 (1 skipped)	75/75 (18 skipped)
3 (zeppelin-20170821-203128_1526260064)	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/08/23 11:28:39	1 s	1/1 (1 skipped)	100/100 (18 skipped)
2 (zeppelin-20170821-203128_1526260064)	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/08/23 11:28:39	0.3 s	1/1 (1 skipped)	20/20 (18 skipped)
1 (zeppelin-20170821-203128_1526260064)	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/08/23 11:28:38	1.0 s	1/1 (1 skipped)	4/4 (18 skipped)
0 (zeppelin-20170821-203128_1526260064)	Zeppelin take at NativeMethodAccessorImpl.java:0	2017/08/23 11:25:45	51 s	2/2	19/19

Figure 35 Spark - Zeppelin Jobs

7.2 Appendix II: Prototype Code

```
== Parsed Logical Plan ==
GlobalLimit 1001
+- LocalLimit 1001
  +- Aggregate [OurBranchID#19], [OurBranchID#19,
sum((OutstandingBalance_Local#29 * cast(-1 as double))) AS
OutstandingBalance#56]
    +- SubqueryAlias bank
      +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertNotNull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertNotNull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
```

```

OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- ExternalRDD [obj#18]

```

== Analyzed Logical Plan ==

```

OurBranchID: string, OutstandingBalance: double
GlobalLimit 1001
+- LocalLimit 1001
  +- Aggregate [OurBranchID#19], [OurBranchID#19,
sum((OutstandingBalance_Local#29 * cast(-1 as double))) AS
OutstandingBalance#56]
    +- SubqueryAlias bank
      +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,

```

```

staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- ExternalRDD [obj#18]

```

```

== Optimized Logical Plan ==
GlobalLimit 1001
+- LocalLimit 1001

```

```

+- Aggregate [OurBranchID#19], [OurBranchID#19,
sum((OutstandingBalance_Local#29 * -1.0)) AS
OutstandingBalance#56]
+- Project [OurBranchID#19, OutstandingBalance_Local#29]
+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,

```

```

true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- ExternalRDD [obj#18]

```

```

== Physical Plan ==

```

```

CollectLimit 1001
+- *HashAggregate(keys=[OurBranchID#19],
functions=[sum((OutstandingBalance_Local#29 * -1.0))],
output=[OurBranchID#19, OutstandingBalance#56])
+- Exchange hashpartitioning(OurBranchID#19, 200)
+- *HashAggregate(keys=[OurBranchID#19],
functions=[partial_sum((OutstandingBalance_Local#29 * -1.0))],
output=[OurBranchID#19, sum#66])
+- *Project [OurBranchID#19,
OutstandingBalance_Local#29]
+- *SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,

```

```

StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertNotNull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertNotNull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertNotNull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- Scan ExternalRDDScan[obj#18]

```

```

== Parsed Logical Plan ==
GlobalLimit 1001
+- LocalLimit 1001
  +- Project [Age#105, Attrition#106, BusinessTravel#107,
DailyRate#108, Department#109, DistanceFromHome#110,
Education#111, EducationField#112, Gender#113, JobLevel#114,
JobSatisfaction#115, MaritalStatus#116, MonthlyIncome#117,
MonthlyRate#118, EmployeeNumber#119]
    +- Filter NOT education#111 LIKE Education
      +- SubqueryAlias hr
        +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class

```



```

org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

```

== Analyzed Logical Plan ==

```

Age: string, Attrition: string, BusinessTravel: string,
DailyRate: string, Department: string, DistanceFromHome: string,
Education: string, EducationField: string, Gender: string,
JobLevel: string, JobSatisfaction: string, MaritalStatus:
string, MonthlyIncome: string, MonthlyRate: string,
EmployeeNumber: string
GlobalLimit 1001
+- LocalLimit 1001
+- Project [Age#105, Attrition#106, BusinessTravel#107,
DailyRate#108, Department#109, DistanceFromHome#110,
Education#111, EducationField#112, Gender#113, JobLevel#114,
JobSatisfaction#115, MaritalStatus#116, MonthlyIncome#117,
MonthlyRate#118, EmployeeNumber#119]
+- Filter NOT education#111 LIKE Education
+- SubqueryAlias hr
+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class

```

```
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
```

```

input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

```

== Optimized Logical Plan ==

```

GlobalLimit 1001
+- LocalLimit 1001
+- Filter (isnotnull(education#111) && NOT (education#111 =
Education))
+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,

```

```

$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]
== Physical Plan ==
CollectLimit 1001
+- *Filter (isnotnull(education#111) && NOT (education#111 =
Education))
+- *SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product

```

```
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
```

```

$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
    +- Scan ExternalRDDScan[obj#104]
== Parsed Logical Plan ==
GlobalLimit 1001
+- LocalLimit 1001
    +- Project [Age#105, Attrition#106, BusinessTravel#107,
DailyRate#108, Department#109, DistanceFromHome#110,
Education#111, EducationField#112, Gender#113, JobLevel#114,
JobSatisfaction#115, MaritalStatus#116, MonthlyIncome#117,
MonthlyRate#118, EmployeeNumber#119]
    +- Filter NOT education#111 LIKE Education
        +- SubqueryAlias hr
            +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,

```

```

StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

```

== Analyzed Logical Plan ==

```

Age: string, Attrition: string, BusinessTravel: string,
DailyRate: string, Department: string, DistanceFromHome: string,
Education: string, EducationField: string, Gender: string,
JobLevel: string, JobSatisfaction: string, MaritalStatus:
string, MonthlyIncome: string, MonthlyRate: string,
EmployeeNumber: string
GlobalLimit 1001
+- LocalLimit 1001
+- Project [Age#105, Attrition#106, BusinessTravel#107,
DailyRate#108, Department#109, DistanceFromHome#110,
Education#111, EducationField#112, Gender#113, JobLevel#114,
JobSatisfaction#115, MaritalStatus#116, MonthlyIncome#117,
MonthlyRate#118, EmployeeNumber#119]
+- Filter NOT education#111 LIKE Education
+- SubqueryAlias hr
+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,

```

```

fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticInvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticInvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,

```



```

staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

== Optimized Logical Plan ==
GlobalLimit 1001
+- LocalLimit 1001
+- Filter (isnotnull(education#111) && NOT (education#111 =
Education))
+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnonnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product

```

```

input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]
== Physical Plan ==
CollectLimit 1001
+- *Filter (isnotnull(education#111) && NOT (education#111 =
Education))
+- *SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,

```

```
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
```

```

input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- Scan ExternalRDDScan[obj#104]
== Parsed Logical Plan ==
GlobalLimit 1001
+- LocalLimit 1001
+- Project [Age#105, Attrition#106, BusinessTravel#107,
DailyRate#108, Department#109, DistanceFromHome#110,
Education#111, EducationField#112, Gender#113, JobLevel#114,
JobSatisfaction#115, MaritalStatus#116, MonthlyIncome#117,
MonthlyRate#118, EmployeeNumber#119]
+- SubqueryAlias where
+- SubqueryAlias hr
+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,

```

```

$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

```

== Analyzed Logical Plan ==

```

Age: string, Attrition: string, BusinessTravel: string,
DailyRate: string, Department: string, DistanceFromHome: string,
Education: string, EducationField: string, Gender: string,
JobLevel: string, JobSatisfaction: string, MaritalStatus:
string, MonthlyIncome: string, MonthlyRate: string,
EmployeeNumber: string
GlobalLimit 1001
+- LocalLimit 1001
+- Project [Age#105, Attrition#106, BusinessTravel#107,
DailyRate#108, Department#109, DistanceFromHome#110,
Education#111, EducationField#112, Gender#113, JobLevel#114,
JobSatisfaction#115, MaritalStatus#116, MonthlyIncome#117,
MonthlyRate#118, EmployeeNumber#119]
+- SubqueryAlias where

```

```

+- SubqueryAlias hr
  +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,

```

```

StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

```

== Optimized Logical Plan ==

GlobalLimit 1001

+- LocalLimit 1001

```

+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,

```

```

$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

```

```

== Physical Plan ==

```

```

CollectLimit 1001

```

```

+- *SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,

```



```
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
```

```

input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- Scan ExternalRDDScan[obj#104]

```

```

== Parsed Logical Plan ==

```

```

GlobalLimit 1001

```

```

+- LocalLimit 1001

```

```

+- Project [Age#105, Attrition#106, BusinessTravel#107,
DailyRate#108, Department#109, DistanceFromHome#110,
Education#111, EducationField#112, Gender#113, JobLevel#114,
JobSatisfaction#115, MaritalStatus#116, MonthlyIncome#117,
MonthlyRate#118, EmployeeNumber#119]

```

```

+- SubqueryAlias where

```

```

+- SubqueryAlias hr

```

```

+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,

```

```

staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

```

== Analyzed Logical Plan ==

```

Age: string, Attrition: string, BusinessTravel: string,
DailyRate: string, Department: string, DistanceFromHome: string,
Education: string, EducationField: string, Gender: string,
JobLevel: string, JobSatisfaction: string, MaritalStatus:
string, MonthlyIncome: string, MonthlyRate: string,
EmployeeNumber: string
GlobalLimit 1001
+- LocalLimit 1001
+- Project [Age#105, Attrition#106, BusinessTravel#107,
DailyRate#108, Department#109, DistanceFromHome#110,
Education#111, EducationField#112, Gender#113, JobLevel#114,

```

```

JobSatisfaction#115, MaritalStatus#116, MonthlyIncome#117,
MonthlyRate#118, EmployeeNumber#119]
+- SubqueryAlias where
+- SubqueryAlias hr
+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,

```

```

$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

```

== Optimized Logical Plan ==

GlobalLimit 1001

+- LocalLimit 1001

```

+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product

```

```

input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- ExternalRDD [obj#104]

```

```

== Physical Plan ==

```

```

CollectLimit 1001

```

```

+- *SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Age, true) AS Age#105, staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,

```

```
fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Attrition, true) AS Attrition#106,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).BusinessTravel, true) AS BusinessTravel#107,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DailyRate, true) AS DailyRate#108,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Department, true) AS Department#109,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).DistanceFromHome, true) AS DistanceFromHome#110,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Education, true) AS Education#111,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EducationField, true) AS EducationField#112,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).Gender, true) AS Gender#113, staticInvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobLevel, true) AS JobLevel#114,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).JobSatisfaction, true) AS JobSatisfaction#115,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MaritalStatus, true) AS MaritalStatus#116,
staticInvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertNotNull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyIncome, true) AS MonthlyIncome#117,
```

```

staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).MonthlyRate, true) AS MonthlyRate#118,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925623.$read$$iw$$iw$HR, true], top level Product
input object).EmployeeNumber, true) AS EmployeeNumber#119]
+- Scan ExternalRDDScan[obj#104]

```

== Parsed Logical Plan ==

```

GlobalLimit 1001
+- LocalLimit 1001
  +- Aggregate [Classification#24], [Classification#24,
sum((OutstandingBalance_Local#29 * cast(-1 as double))) AS
OutStandingBalanceByProduct#289]
    +- SubqueryAlias bank
      +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product

```



```

input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- ExternalRDD [obj#18]

```

```

== Analyzed Logical Plan ==

```

```

Classification: string, OutstandingBalanceByProduct: double

```

```

GlobalLimit 1001

```

```

+- LocalLimit 1001

```

```

+- Aggregate [Classification#24], [Classification#24,
sum((OutstandingBalance_Local#29 * cast(-1 as double))) AS
OutstandingBalanceByProduct#289]

```

```

+- SubqueryAlias bank

```

```

+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,

```

```
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
```

```

input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- ExternalRDD [obj#18]

```

```

== Optimized Logical Plan ==

```

```

GlobalLimit 1001

```

```

+- LocalLimit 1001

```

```

+- Aggregate [Classification#24], [Classification#24,
sum((OutstandingBalance_Local#29 * -1.0)) AS

```

```

OutStandingBalanceByProduct#289]

```

```

+- Project [Classification#24,
OutstandingBalance_Local#29]

```

```

+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,

```

```

$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- ExternalRDD [obj#18]

```

```

== Physical Plan ==

```

```

CollectLimit 1001
+- *HashAggregate(keys=[Classification#24],
functions=[sum((OutstandingBalance_Local#29 * -1.0))],
output=[Classification#24, OutstandingBalanceByProduct#289])
+- Exchange hashpartitioning(Classification#24, 200)
+- *HashAggregate(keys=[Classification#24],
functions=[partial_sum((OutstandingBalance_Local#29 * -1.0))],
output=[Classification#24, sum#299])
+- *Project [Classification#24,
OutstandingBalance_Local#29]
+- *SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,

```

```

$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product

```

```
input object).ProductTypeID, true) AS ProductTypeID#34,  
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,  
StringType, fromString, assertnotnull(input[0,  
$line66882925615.$read$$iw$$iw$Bank, true], top level Product  
input object).LastCreditDate, true) AS LastCreditDate#35]  
+- Scan ExternalRDDScan[obj#18]
```

```

== Parsed Logical Plan ==
GlobalLimit 1001
+- LocalLimit 1001
  +- Aggregate [CurrencyID#22], [CurrencyID#22,
sum((OutstandingBalance_Local#29 * cast(-1 as double))) AS
OutStandingBalanceByProduct#300]
    +- SubqueryAlias bank
      +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,

```

```

$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- ExternalRDD [obj#18]

```

== Analyzed Logical Plan ==

```

CurrencyID: string, OutstandingBalanceByProduct: double
GlobalLimit 1001
+- LocalLimit 1001
  +- Aggregate [CurrencyID#22], [CurrencyID#22,
sum((OutstandingBalance_Local#29 * cast(-1 as double))) AS
OutstandingBalanceByProduct#300]
    +- SubqueryAlias bank
      +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,

```



```

StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- ExternalRDD [obj#18]

```

```

== Optimized Logical Plan ==
GlobalLimit 1001
+- LocalLimit 1001

```

```

+- Aggregate [CurrencyID#22], [CurrencyID#22,
sum((OutstandingBalance_Local#29 * -1.0)) AS
OutStandingBalanceByProduct#300]
+- Project [CurrencyID#22, OutstandingBalance_Local#29]
+- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,

```

```

true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- ExternalRDD [obj#18]

```

```

== Physical Plan ==

```

```

CollectLimit 1001
+- *HashAggregate(keys=[CurrencyID#22],
functions=[sum((OutstandingBalance_Local#29 * -1.0))],
output=[CurrencyID#22, OutstandingBalanceByProduct#300])
+- Exchange hashpartitioning(CurrencyID#22, 200)
+- *HashAggregate(keys=[CurrencyID#22],
functions=[partial_sum((OutstandingBalance_Local#29 * -1.0))],
output=[CurrencyID#22, sum#310])
+- *Project [CurrencyID#22,
OutstandingBalance_Local#29]
+- *SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType,
fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OurBranchID, true) AS OurBranchID#19,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).AccountID, true) AS AccountID#20,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductID, true) AS ProductID#21,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).CurrencyID, true) AS CurrencyID#22,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,

```

```

StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ExchangeRate, true) AS ExchangeRate#23,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Classification, true) AS Classification#24,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).AmountFinanced AS
AmountFinanced#25, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).InstallmentAmount AS InstallmentAmount#26,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).Frequency, true) AS Frequency#27,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).OutstandingBalance AS
OutstandingBalance#28, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OutstandingBalance_Local AS
OutstandingBalance_Local#29, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsAmount AS ArrearsAmount#30,
assertnotnull(input[0, $line66882925615.$read$$iw$$iw$Bank,
true], top level Product input object).ArrearsAmount_Local AS
ArrearsAmount_Local#31, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ArrearsDays.intValue AS ArrearsDays#32,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).OfficerName, true) AS OfficerName#33,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).ProductTypeID, true) AS ProductTypeID#34,
staticinvoke(class org.apache.spark.unsafe.types.UTF8String,
StringType, fromString, assertnotnull(input[0,
$line66882925615.$read$$iw$$iw$Bank, true], top level Product
input object).LastCreditDate, true) AS LastCreditDate#35]
+- Scan ExternalRDDScan[obj#18]

```

```

val bankText = sc.textFile("loans.csv")

case class Bank(OurBranchID:String, AccountID: String, ProductID:String, CurrencyID:String,
ExchangeRate:String, Classification:String, AmountFinanced:Double,
InstallmentAmount:Double, Frequency:String, OutstandingBalance:Double,
OutstandingBalance_Local:Double, ArrearsAmount:Double, ArrearsAmount_Local:Double,
ArrearsDays:Integer, OfficerName:String, ProductTypeID:String, LastCreditDate:String
)

val bank = bankText.map(s=>s.split(",")).filter(s=>s(0)!="\OurBranchID").map(
  s=>Bank(s(0).replaceAll("\", ""),
    s(1).replaceAll("\", ""),
    s(2).replaceAll("\", ""),
    s(3).replaceAll("\", ""),
    s(4).replaceAll("\", ""),
    s(5).replaceAll("\", ""),
    s(6).toString.toDouble,
    s(7).toString.toDouble,
    s(8).replaceAll("\", ""),
    s(9).toString.toDouble,
    s(10).toString.toDouble,
    s(11).toString.toDouble,
    s(12).toString.toDouble,
    s(13).toInt,
    s(14).replaceAll("\", ""),
    s(15).replaceAll("\", ""),
    s(16).replaceAll("\", "")
  )
)

```

```
// convert to DataFrame and create temporal table  
bank.toDF().registerTempTable("bank")
```

```
val channelText = sc.textFile("Channels.csv")
case class Channel(Channel:String, OurBranchID: String, Value:String)

val channel = channelText.map(s=>s.split(",")).filter(s=>s(0)!="\Channel\").map(
  s=>Channel(s(0).replaceAll("\\"", ""),
    s(1).replaceAll("\\"", ""),
    s(2).replaceAll("\\"", "")
  )
)

// convert to DataFrame and create temporal table
channel.toDF().registerTempTable("channel")
```

```

val HRText = sc.textFile("HR.csv")

case class HR(Age:String, Attrition: String, BusinessTravel:String, DailyRate:String,
Department:String, DistanceFromHome:String, Education:String, EducationField:String,
Gender:String, JobLevel:String, JobSatisfaction:String, MaritalStatus:String,
MonthlyIncome:String, MonthlyRate:String, EmployeeNumber:String)

val hr = HRText.map(s=>s.split(",")).filter(s=>s(0)!="Age\").map(
  s=>HR(s(0).replaceAll("\", ""),
    s(1).replaceAll("\", ""),
    s(2).replaceAll("\", ""),
    s(3).replaceAll("\", ""),
    s(4).replaceAll("\", ""),
    s(5).replaceAll("\", ""),
    s(6).replaceAll("\", ""),
    s(7).replaceAll("\", ""),
    s(8).replaceAll("\", ""),
    s(9).replaceAll("\", ""),
    s(10).replaceAll("\", ""),
    s(11).replaceAll("\", ""),
    s(12).replaceAll("\", ""),
    s(13).replaceAll("\", ""),
    s(13).replaceAll("\", ""))
)

// convert to DataFrame and create temporal table
hr.toDF().registerTempTable("hr")

```


7.3 Appendix III: Observation Sheet

This section covers that data collection tools that were used to collect the data that informed this research and prototype development.

7.3.1 General Information

Name: _____ Date: _____

Job Role: _____ Time: Start _____ Stop _____

Years in Analytics: _____

Gender: Male Female

7.3.2 Data Sources

Data Sources Identified: 1. _____ 2. _____ 3.

4. _____ 5. _____ 6. _____

Brief explanation of the data extraction process _____

7.3.3 Data Cleanup

Data cleanup Tools used: 1. _____ 2. _____

3. _____ 5. _____ 6. _____

Brief explanation of the data cleanup steps _____

7.3.4 Data Analysis

Time to analyze one report: _____ Minutes.

of reports analyzed: _____

Data analysis Tools used: 1. _____ 2. _____

3. _____ 5. _____ 6. _____

Brief explanation of the data analysis steps _____

7.3.5 Data Visualization

Data visualization Tools used: 1. _____ 2. _____

3. _____ 5. _____ 6. _____

Brief explanation of the visualization Details _____

7.3.6 Dashboards produced

1. _____

Details

2. _____

Details

3. _____

Details

5. _____

Details
