# Software Project Report

**for**

# SmartSales Project

**Version 1.0 approved**

**Prepared by Group F-22**

**San Jose State University**

**Dec 4th, 2016**

# Table of Contents

## Table of Contents

# 1. Introduction

## 1.1 Purpose

This document is a report for SmartSales project. The project was built based on requirement of course CMPE272-Enterprise Software System for final project. SmartSales is stand-alone application that utilize Tableau to visualize data and provide useful sale reports.

## 1.2 Document Conventions

This document follows IEEE standard template, **Copyright © 2005 by Karl E. Wiegers, IEEE.**

## 1.3 Intended Audience and Reading Suggestions

- End user, business user with or without programming knowledge
- Instructor Andrew and Teaching Assistant

The first part of the document will focus on the high level design of SmartSales. The later section describe the algorithm & technologies use in this project and finally describe the implementation process. Viewer are advised to read this document from beginning till the end in that order, as it is not dictionary type.

## 1.4 Product Scope

Project 'SMART SALES' leverages the concepts of Data Warehousing and Business Intelligence to achieve insights from those raw data. It visualizes data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns

In the context of this project, we assume that the business have chain stores. Sales data is scattered across retail stores, wholesale stores and online store for Internet customers. Data of retail and wholesale stores is managed by a legacy system, while online store system is just built recently. Therefore we have at least two different data formats.

## 1.5 References

- https://community.tableau.com/community/developers
- http://developer.okta.com
- https://www.talend.com/solutions/for-developers
- http://onlinehelp.tableau.com/samples/en-us/js_api/tutorial.htm

# 2. Overall Description

## 2.1 Product Perspective

The survival of a business lies in its ability to understand the market and adjust its marketing and sales strategies appropriately. Knowledge about a particular market is learnt from reliable and updated data sources. Sales data is one of the great sources of data to diagnose the strength and weakness of marketing & sales strategy as they reflect the taste of the market. Unfortunately, the amount of sale data is huge, unclassified, and normally distributed in many system databases. It's a very challenging, costly for traditional tool like Excel to centralize all data; it can be thousands of spreadsheets. Next thing, after data is put in the same place, it's a difficult task to dive into those huge data and extract knowledge from it.

Project 'SMART SALES' leverages the concepts of Data Warehousing and Business Intelligence to achieve insights from those raw data. It visualizes data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns

In the context of this project, we assume that the business have chain stores. Sales data is scattered across retail stores, wholesale stores and online store for Internet customers. Data of retail and wholesale stores is managed by a legacy system, while online store system is just built recently. Therefore we have at least two different data formats.

## 2.2 Product Functions

- Centralize different data sources by using ETL tool (Talend)
- Single Sign On
- Data warehouse and visualize data
- Responsive Web UI, that can be used with different size of screen (big screen or mobile phone)

## 2.3 User Classes and Characteristics

- End user: business users, like employee of sale department
- Administrator: who have some decent programming knowledge to customize the report when needed

## 2.4 Operating Environment

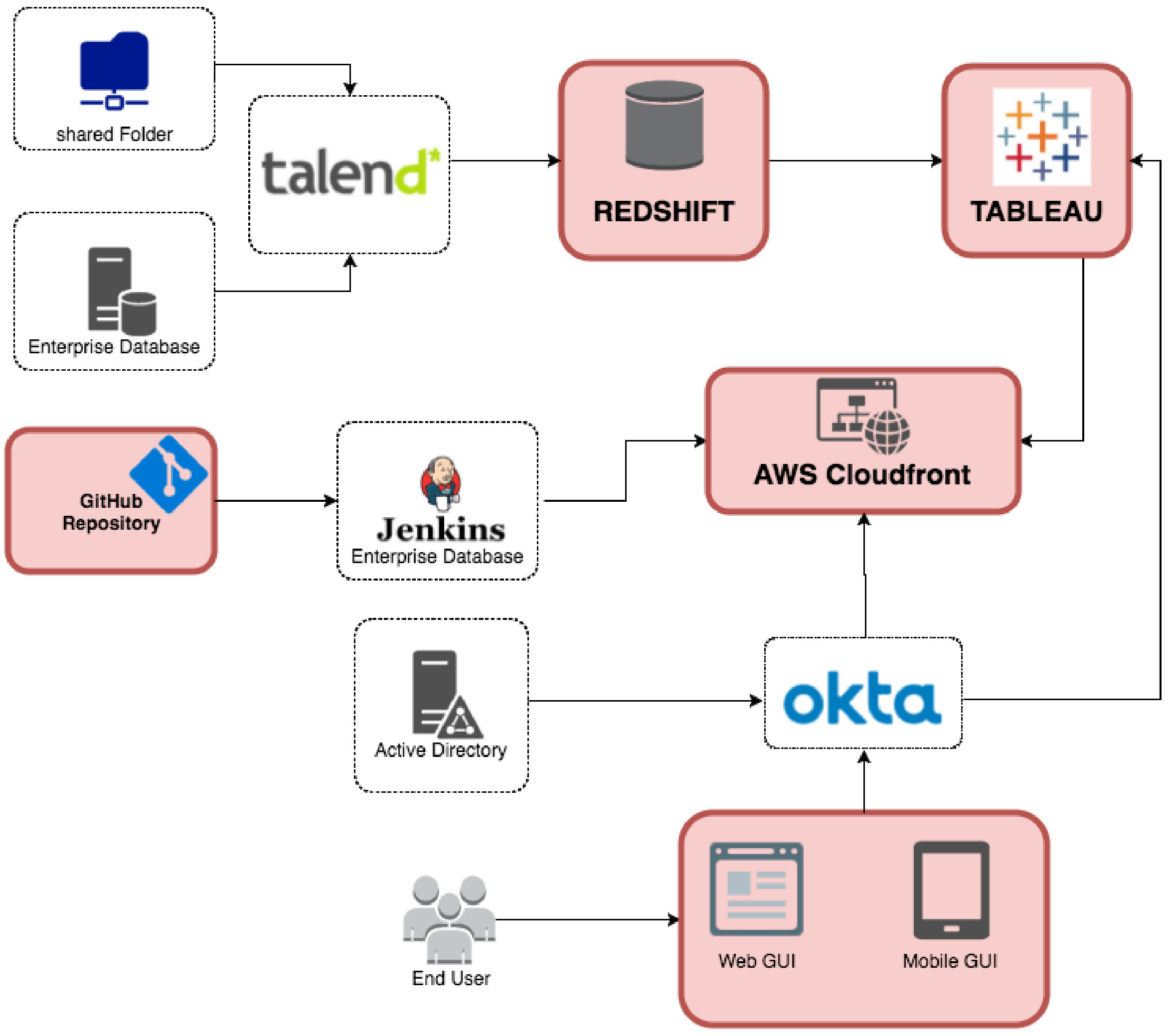


Figure 1 - Infrastructure design

## 2.5 Design and Implementation Constraints

Technologies constraint:

- Tableau: limited free account which only allow 14 days of trial use.
- Okta: limited free account which only allow 3 application
- Talend: Talend Open Studios is a free ETL/ELT desktop tool. There is no limitation over it. Talend Server comes with a trail period of 14 days.

Hardware constraint:

- AWS Elastic Beanstalk: to host web application. The student account have limited balance to 100$

## 2.6 User Documentation

(no documentation)

## 2.7 Assumptions and Dependencies

- Okta is set up and connected to Tableau and Github and SmartSales
- Tableau account is still active and connect to Redshift datasource
- Jenkin server is running and connect to github repo, where the code is stored

# 3. External Interface Requirements

## 3.1 User Interfaces

### 3.1.1 Login Screen

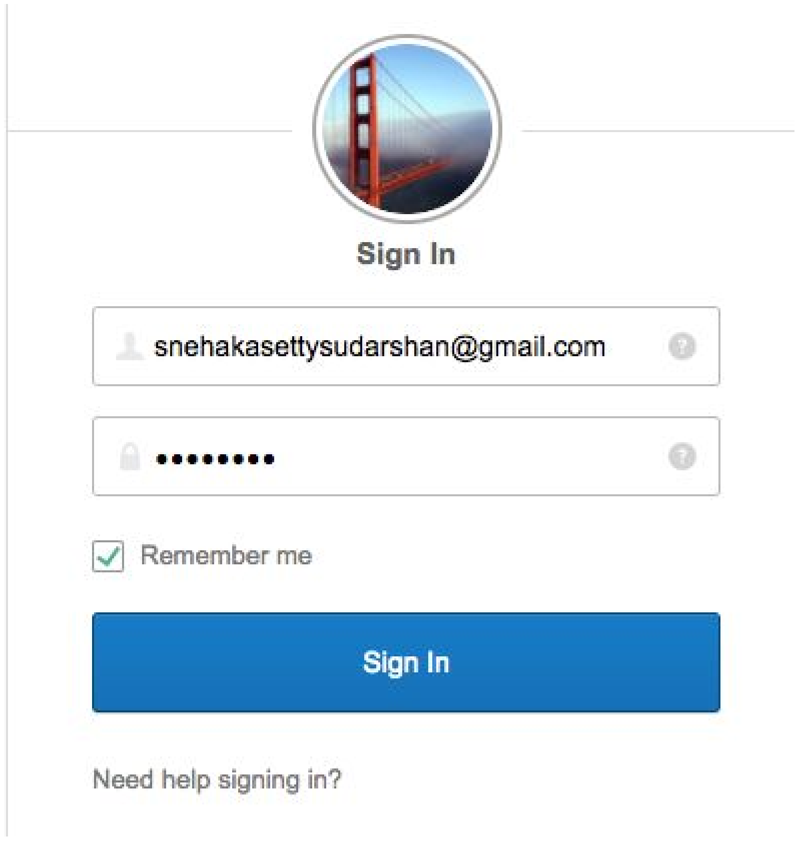


Figure 2 - Login Screen

### 3.1.3 Application Dashboard

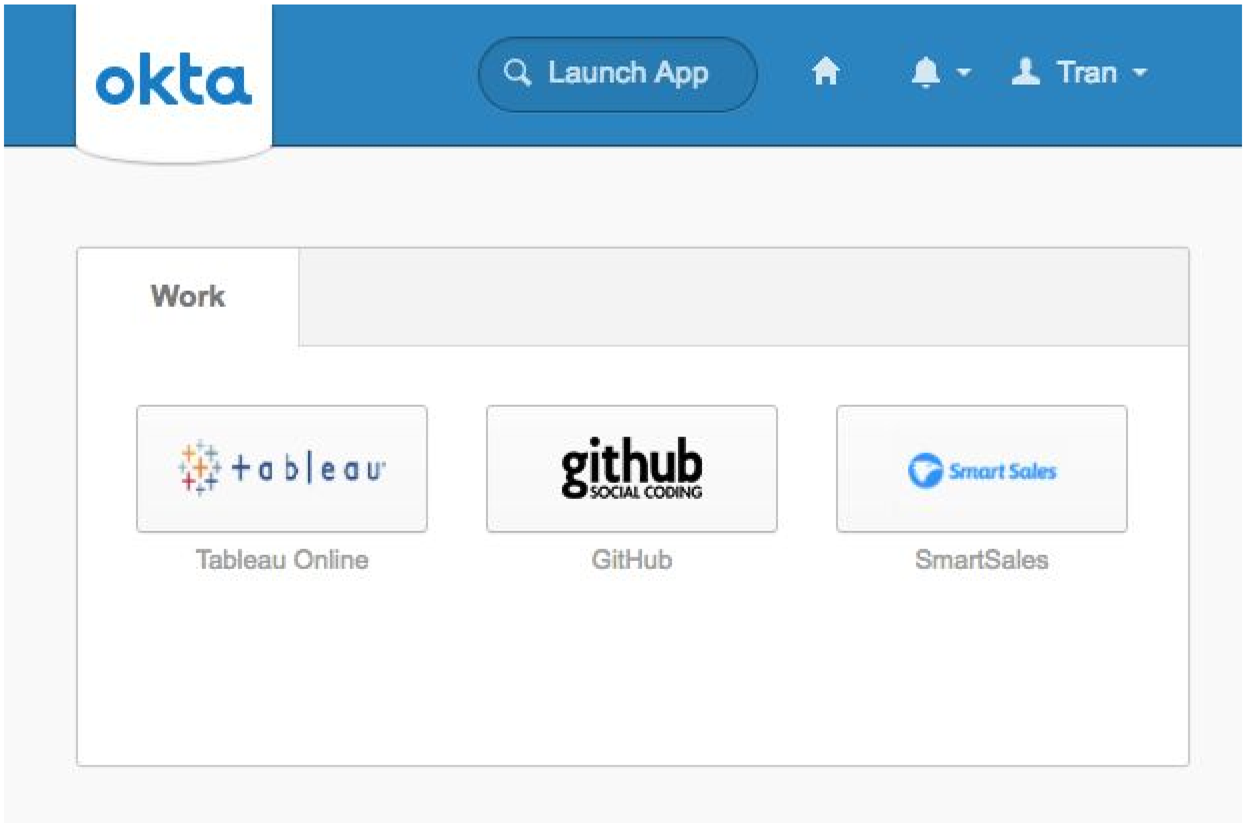


Figure 3 - Okta Application Dashboard

### 3.1.2 Smart Sales Application Dashboard

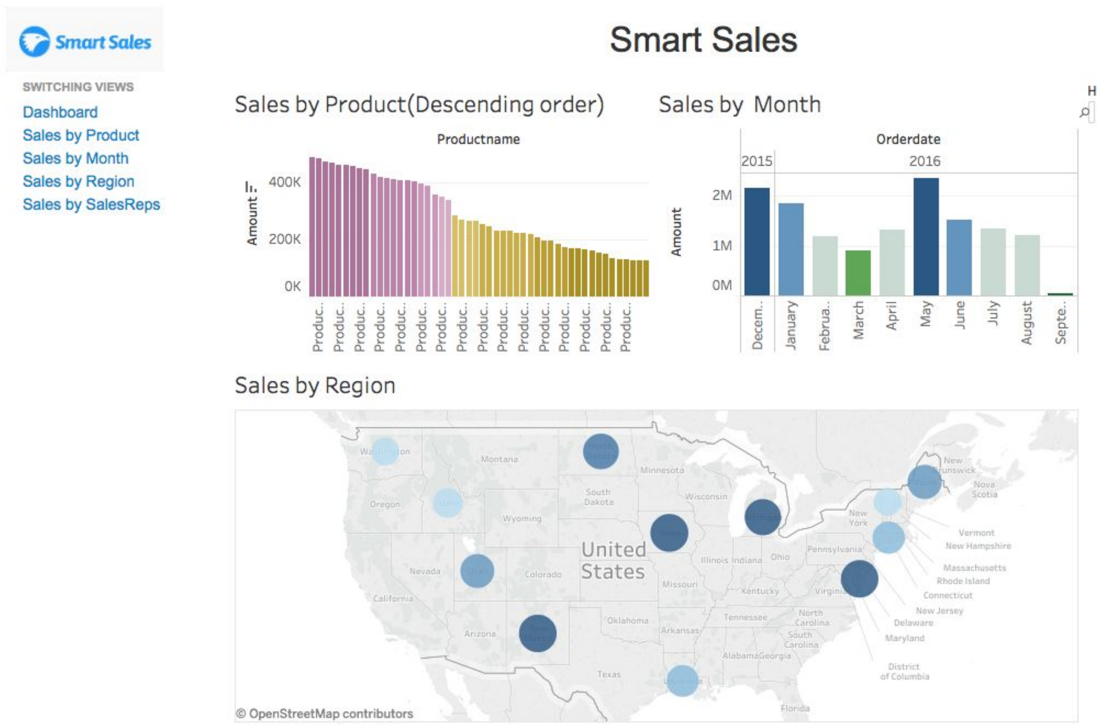


Figure 4 - SmartSales Dashboard

## 3.2 Hardware Interfaces

Not required. SmartSales is deployed to AWS

## 3.3 Software Interfaces

- Talend
- Okta API
- Tableau API

## 3.4 Communications Interfaces

- **HTTP:**
    - Access SmartSales from Web GUI or Mobile GUI
    - Access Okta API interface from SmartSales (web server)
    - Access Tableau Javascript API from SmartSales

# 4. System Features

## 4.1 Data Sources Centralization

### 4.1.1 Description and Priority

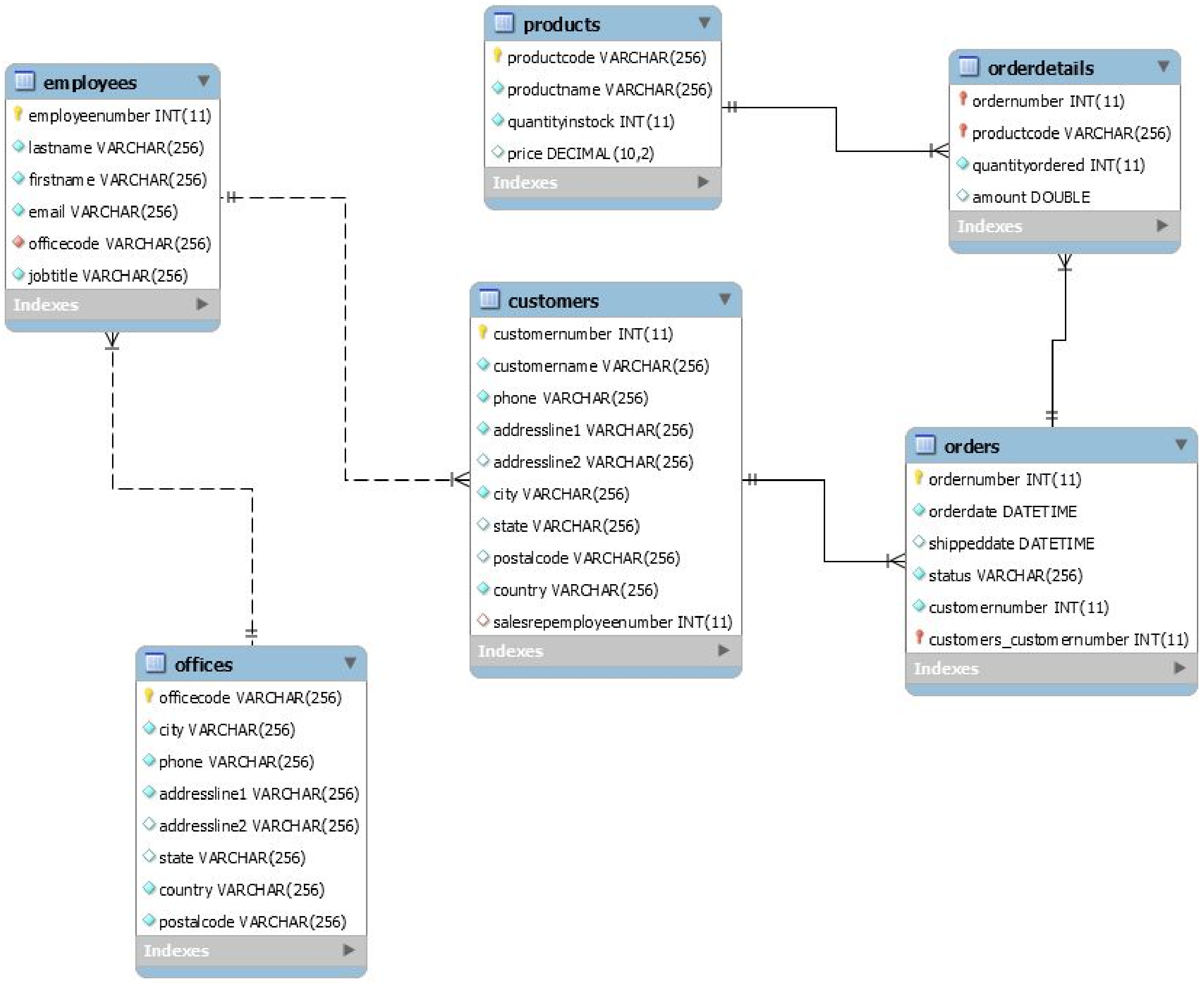
Talend Open Studios is an ETL/ELT tool. It is used to extract data from multiple sources like CSV, Json, XML etc.. Data is extracted using talend input components like, tfileInputDelimited, tRedshiftInput etc. The data then could be transformed based on business logic using the transformation components like tMap, tJoin etc. The data then is dumped into the target. The target could be a file or a database. Talend uses components like tfileOuputDelimited, tRedshiftOuput etc. to load the data.
In this project, we are generating data using talend. The tRowGenerator component of talend generated data and

### 4.1.2 Stimulus/Response Sequences

In this project, we are generating data using talend. The tRowGenerator component of talend generated data and puts it in the CSV file. The CSV file is then extracted but the talend input component and is loaded into tMap component. The data is then dumped into AWS Redshift. Tableau then target to the AWS Redshift and creates visualization.

### 4.1.3 Functional Requirements

<u>REQ-1: Data Integration</u>
Retail stores are present all over USA. Customer from different parts of the country buy products from the store. Now initially the data is present in the local servers of the stores. It has to be integrated and centralized to the data warehouse. In order to do that we write ETL jobs. Below is the data model for the database.



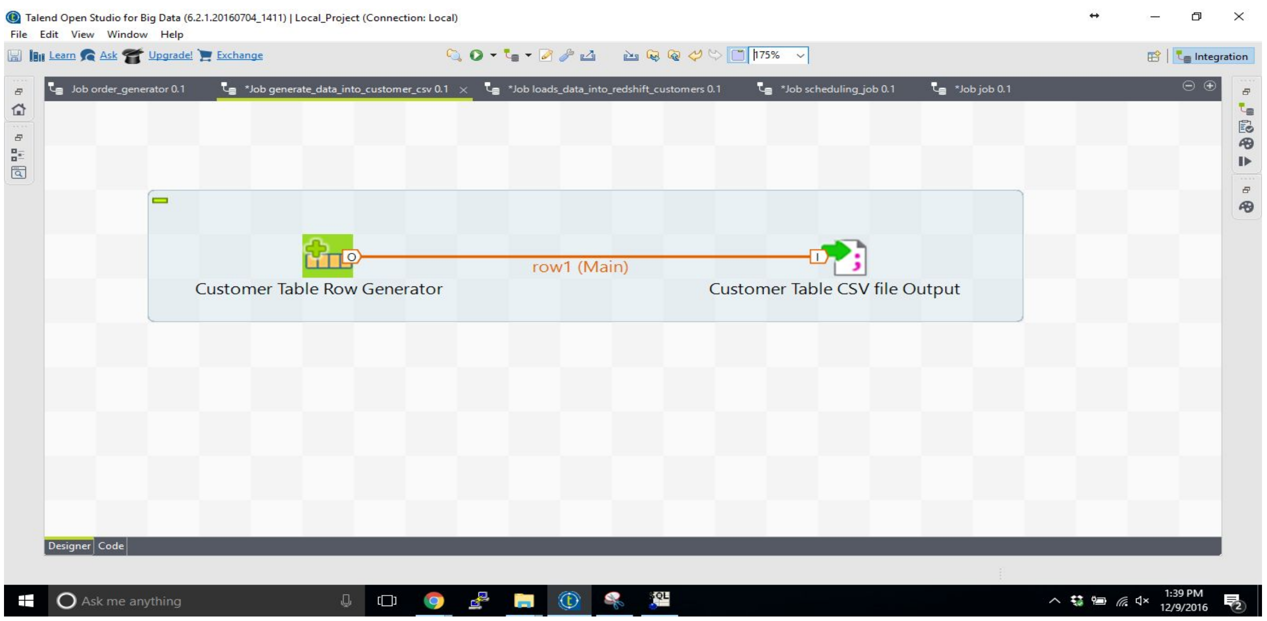
<u>REQ-2: Data Warehouse</u>

We need to have a centralized data warehouse to integrate the data from all sources. We use Amazon redshift as our data warehouse. In order to create a data warehouse create a free account in the AWS Redshift. Give the details of the database like domain, database, username and password. Once the configuration is done, download SQL Workbench/j to performs queries on the database. Using above data model, create ddl statements and execute in the the AWS Redshift.

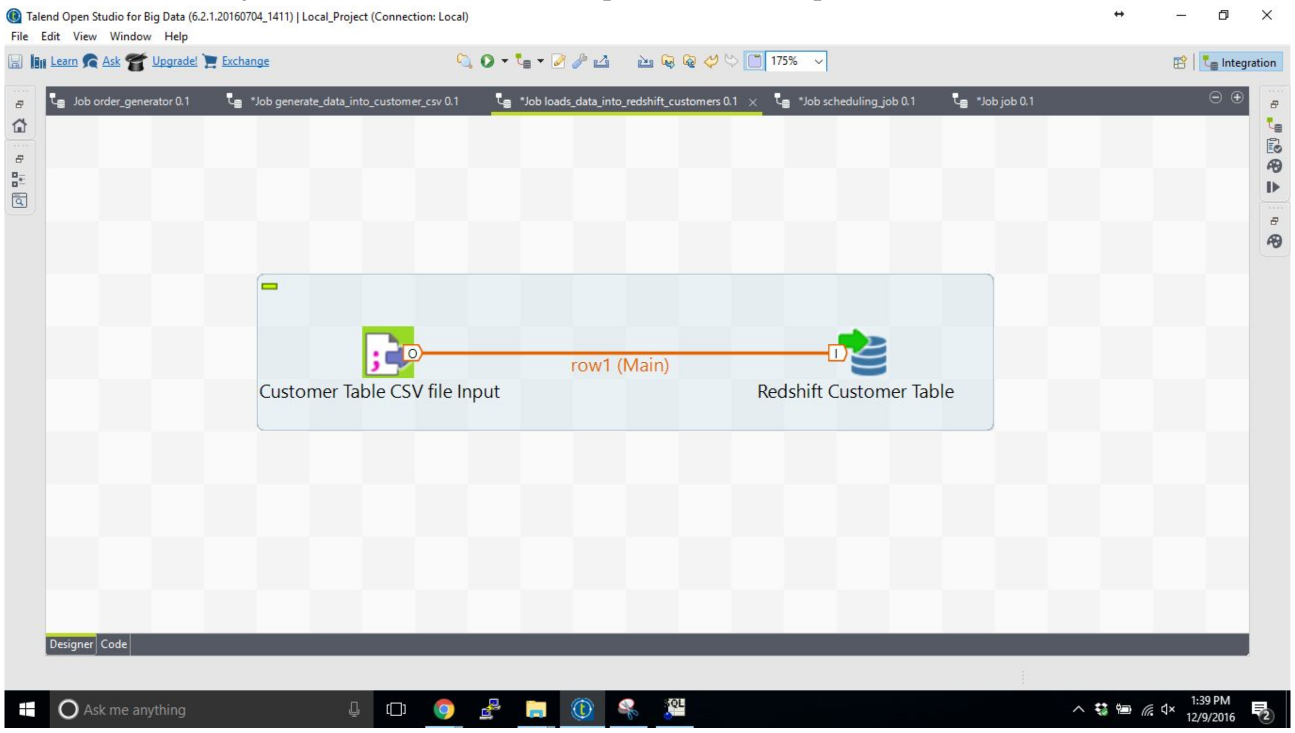
<u>REQ-3: Talend Jobs</u>

In order to populate each table in the database, we need to generate data in files first, Then we extract data from the files and dump in into the tables.

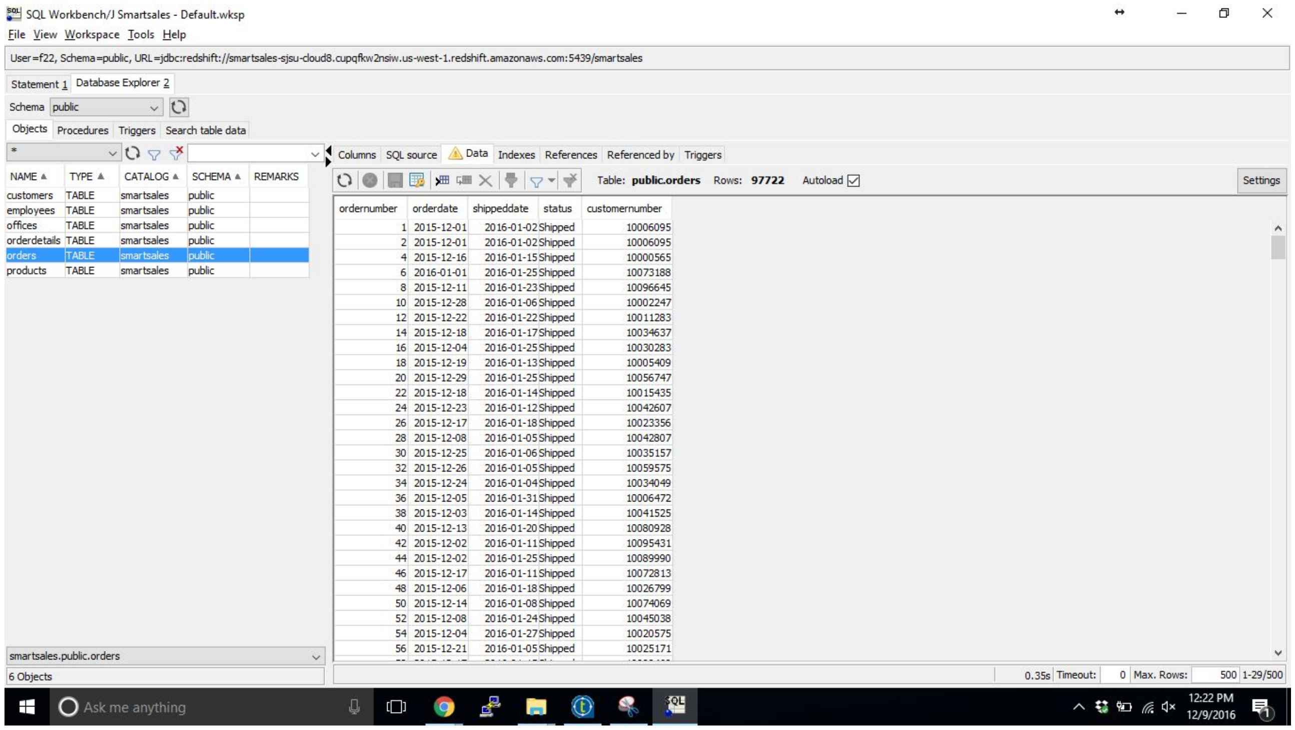
Each Job is divided into two subjobs, First Job(below diagram) generated data from tRowGenerator and dumps it in the CSV file.


Talend Open Studio for Big Data (6.2.1.20160704_1411) | Local_Project (Connection: Local)
Job order_generator 0.1
*Job generate_data_into_customer_csv 0.1
*Job loads_data_into_redshift_customers 0.1
*Job scheduling_job 0.1
*Job job 0.1
Customer Table Row Generator
row1 (Main)
Customer Table CSV file Output
Designer
Code

Second subjob extracts data from the input file and dumps it into AWS Redshift table.


Talend Open Studio for Big Data (6.2.1.20160704_1411) | Local_Project (Connection: Local)
Job order_generator 0.1
*Job generate_data_into_customer_csv 0.1
*Job loads_data_into_redshift_customers 0.1
*Job scheduling_job 0.1
*Job job 0.1
Customer Table CSV file Input
row1 (Main)
Redshift Customer Table
Designer
Code

The data then gets populated into AWS Redshift database.


SQL Workbench/J Smartsales - Default.wksp
File View Workspace Tools Help
Statement 1 Database Explorer 2
Objects Procedures Triggers Search table data
Columns SQL source Data Indexes References Referenced by Triggers
Settings
Ask me anything

User=f22, Schema=public, URL=jdbc:redshift://smartsales-sjsu-cloud8.cupqfkw2nsiw.us-west-1.redshift.amazonaws.com:5439/smartsales

Schema: public

| NAME | TYPE | CATALOG | SCHEMA | REMARKS |
|---|---|---|---|---|
| customers | TABLE | smartsales | public | |
| employees | TABLE | smartsales | public | |
| offices | TABLE | smartsales | public | |
| orderdetails | TABLE | smartsales | public | |
| orders | TABLE | smartsales | public | |
| products | TABLE | smartsales | public | |

smartsales.public.orders
6 Objects

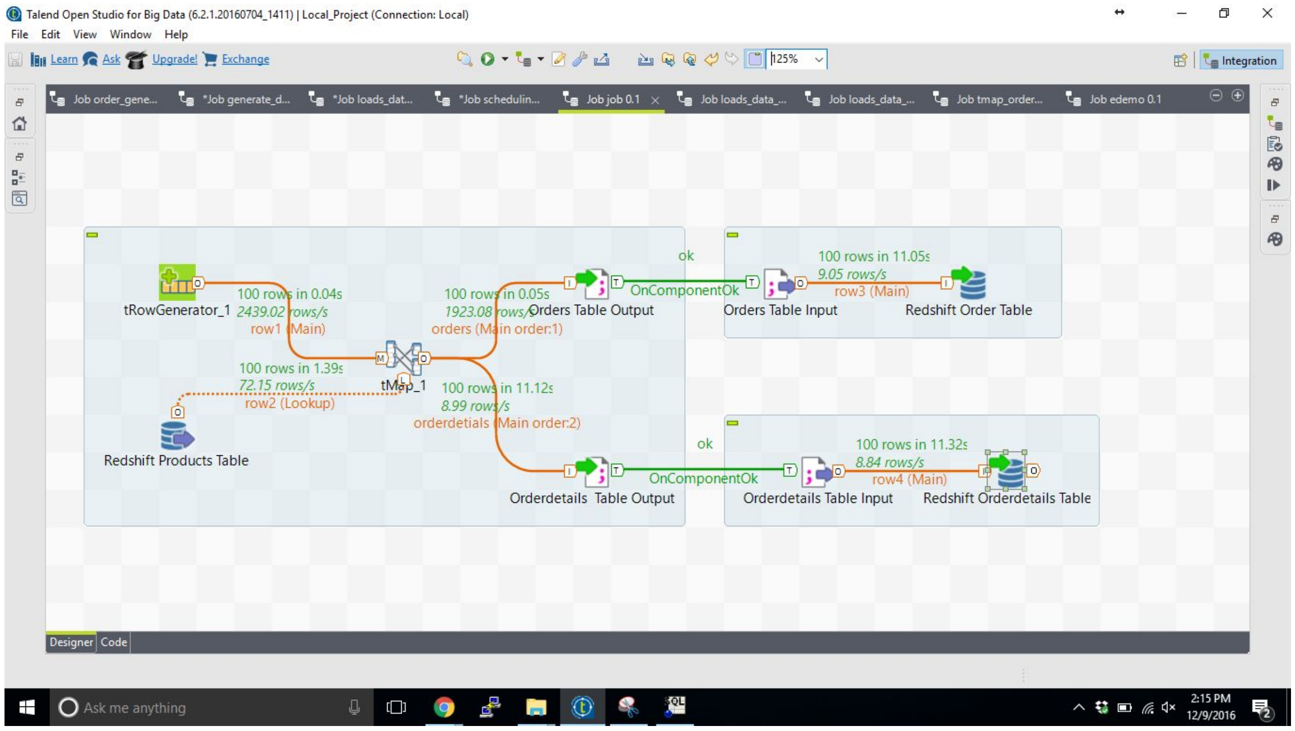
Table: public.orders Rows: 97722 Autoload

| ordernumber | orderdate | shippeddate | status | customernumber |
|---|---|---|---|---|
| 1 | 2015-12-01 | 2016-01-02 | Shipped | 10006095 |
| 2 | 2015-12-01 | 2016-01-02 | Shipped | 10006095 |
| 4 | 2015-12-16 | 2016-01-15 | Shipped | 10000565 |
| 6 | 2016-01-01 | 2016-01-25 | Shipped | 10073188 |
| 8 | 2015-12-11 | 2016-01-23 | Shipped | 10096645 |
| 10 | 2015-12-28 | 2016-01-06 | Shipped | 10002247 |
| 12 | 2015-12-22 | 2016-01-22 | Shipped | 10011283 |
| 14 | 2015-12-18 | 2016-01-17 | Shipped | 10034637 |
| 16 | 2015-12-04 | 2016-01-25 | Shipped | 10030283 |
| 18 | 2015-12-19 | 2016-01-13 | Shipped | 10005409 |
| 20 | 2015-12-29 | 2016-01-25 | Shipped | 10056747 |
| 22 | 2015-12-18 | 2016-01-14 | Shipped | 10015435 |
| 24 | 2015-12-23 | 2016-01-12 | Shipped | 10042607 |
| 26 | 2015-12-17 | 2016-01-18 | Shipped | 10023356 |
| 28 | 2015-12-08 | 2016-01-05 | Shipped | 10042807 |
| 30 | 2015-12-25 | 2016-01-06 | Shipped | 10035157 |
| 32 | 2015-12-26 | 2016-01-05 | Shipped | 10059575 |
| 34 | 2015-12-24 | 2016-01-04 | Shipped | 10034049 |
| 36 | 2015-12-05 | 2016-01-31 | Shipped | 10006472 |
| 38 | 2015-12-03 | 2016-01-14 | Shipped | 10041525 |
| 40 | 2015-12-13 | 2016-01-20 | Shipped | 10080928 |
| 42 | 2015-12-02 | 2016-01-11 | Shipped | 10095431 |
| 44 | 2015-12-02 | 2016-01-25 | Shipped | 10089990 |
| 46 | 2015-12-17 | 2016-01-11 | Shipped | 10072813 |
| 48 | 2015-12-06 | 2016-01-18 | Shipped | 10026799 |
| 50 | 2015-12-14 | 2016-01-08 | Shipped | 10074069 |
| 52 | 2015-12-08 | 2016-01-24 | Shipped | 10045038 |
| 54 | 2015-12-04 | 2016-01-27 | Shipped | 10020575 |
| 56 | 2015-12-21 | 2016-01-05 | Shipped | 10025171 |

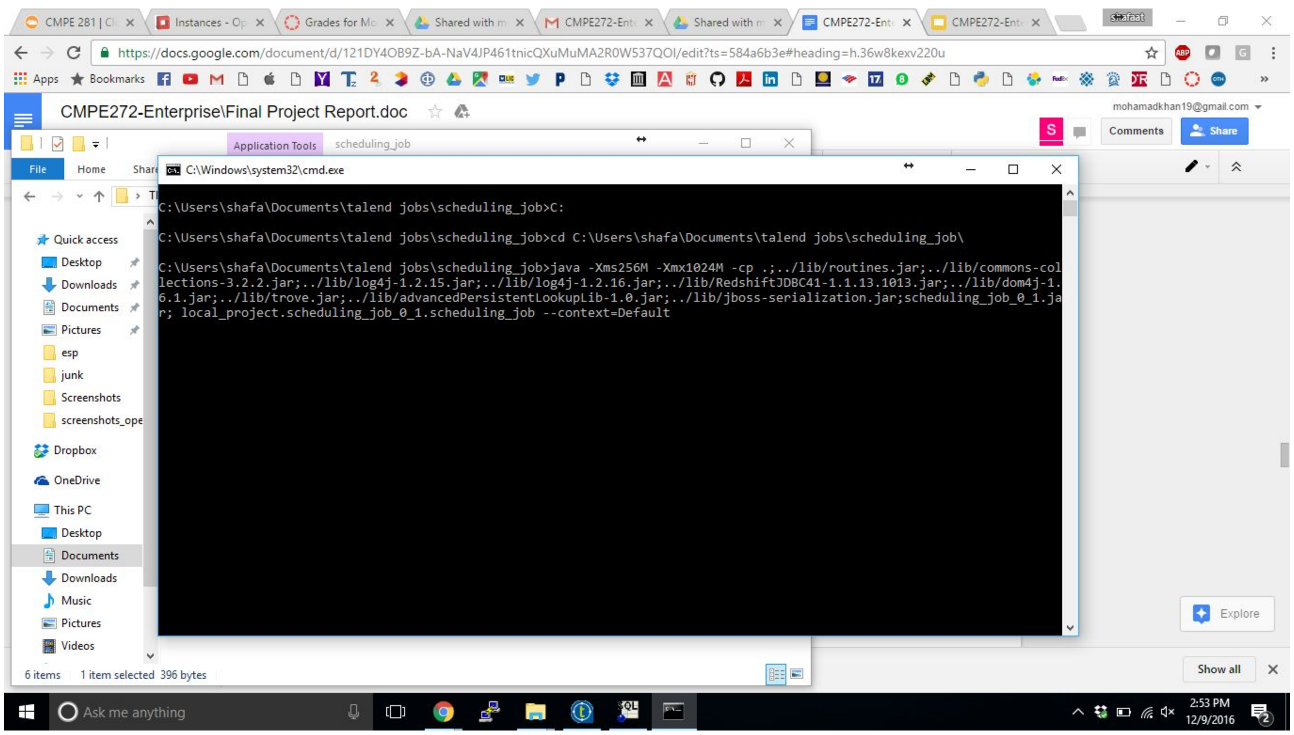
0.35s Timeout: 0 Max. Rows: 500 1-29/500

12:22 PM 12/9/2016

A scheduling job is created in talend which is a batch process, it generated 10 rows in every 5 mins and populated the table.



The job is scheduled using windows task manager.



```
C:\Users\shafa\Documents\talend jobs\scheduling_job>C:

C:\Users\shafa\Documents\talend jobs\scheduling_job>cd C:\Users\shafa\Documents\talend jobs\scheduling_job\

C:\Users\shafa\Documents\talend jobs\scheduling_job>java -Xms256M -Xmx1024M -cp .;../lib/routines.jar;../lib/commons-col
lections-3.2.2.jar;../lib/log4j-1.2.15.jar;../lib/log4j-1.2.16.jar;../lib/RedshiftJDBC41-1.1.13.1013.jar;../lib/dom4j-1.
6.1.jar;../lib/trove.jar;../lib/advancedPersistentLookupLib-1.0.jar;../lib/jboss-serialization.jar;scheduling_job_0_1.ja
r; local_project.scheduling_job_0_1.scheduling_job --context=Default
```

## 4.2 Single Sign On

### 4.2.1 Description and Priority

Using several external component: Tableau, Github, corporate account (in Active Directory) lead to annoyance to user, as they have to remember several account, and do multiple log in for each component.

Single Sign On make it easy and user friendly. User just need to log in once.

### 4.2.2 Stimulus/Response Sequences

A. User first login to okta domain
B. Dashboard page will display list of applications supported with SSO
C. User choose applications, and will not be asked for credential

### 4.2.3 Functional Requirements

Log in once. No more account authentication box pop up anymore.

## 4.3 Visualization Using Tableau

### 4.3.1 Description and Priority

Tableau is a Data Visualization and Data Analyzing tool that helps us create visually-appealing reports, charts, graphs and dashboards using the data available. Visual representation of data brings out the insights hidden. Tableau helps us in finding solutions to lot of problems. Reports generated using Tableau are interactive and can be easily shared with others in a secure way.Tableau connects easily to nearly any data source, be it MySQL, Oracle, MS Excel, Amazon Redshift and so on.

### 4.3.2 Stimulus/Response Sequences

All the development work like creating reports, charts, formatting them and putting together as a dashboard is done in Tableau Desktop. The charts are interrelated in the dashboard and applying filters on one of the values will apply the same filters for the other charts as well in the same dashboard. Later, the dashboards, reports can be published to Tableau Online, which is hosted in cloud to share with other users. The users can view reports and dashboards by logging into Tableau Online. Any changes made to the data source will reflect in the reports and dashboards.

### 4.3.3 Functional Requirements

REQ-1: Data Source
In order to generate reports, charts and dashboards, Tableau should be connected to at least one data source. In our project, Tableau is connected to Amazon Redshift.

REQ-2: Tableau Desktop
All the development work related to reports and dashboards is done in Tableau Desktop.

REQ-3: Tableau Online
Tableau Online is used to share the reports and dashboards with other users across geographic locations. When the reports and dashboards are ready, it will be published to Tableau Online, which is hosted in cloud.

In our Project, Total Sales is broken down by Region, Time, Sales Representatives and Product. A dashboard is prepared by bringing together the individual reports. The same is shown in below screenshots.

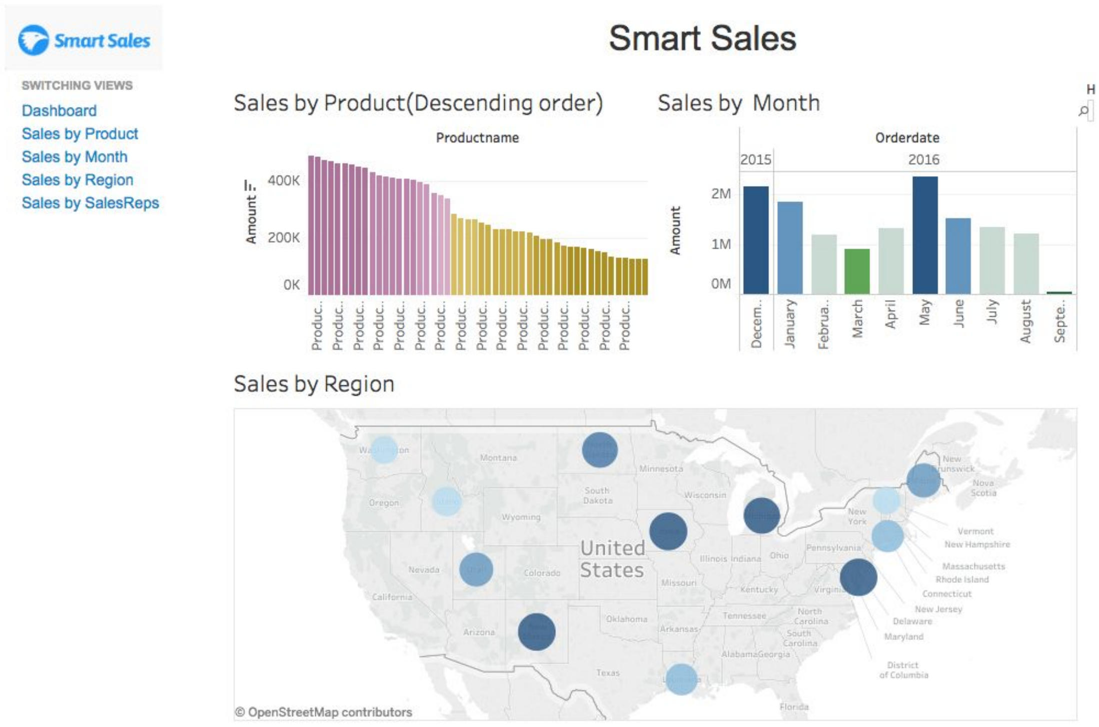


Figure 4 - SmartSales Dashboard

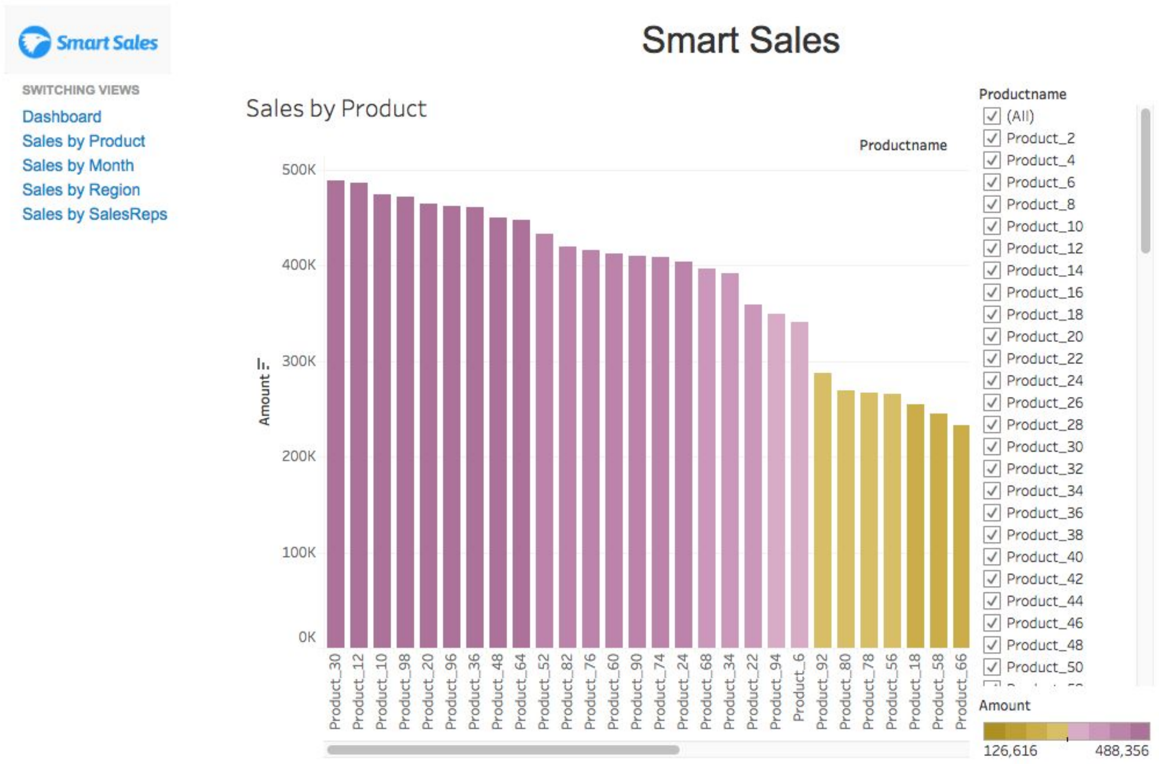


Figure 5 - Sales by Product

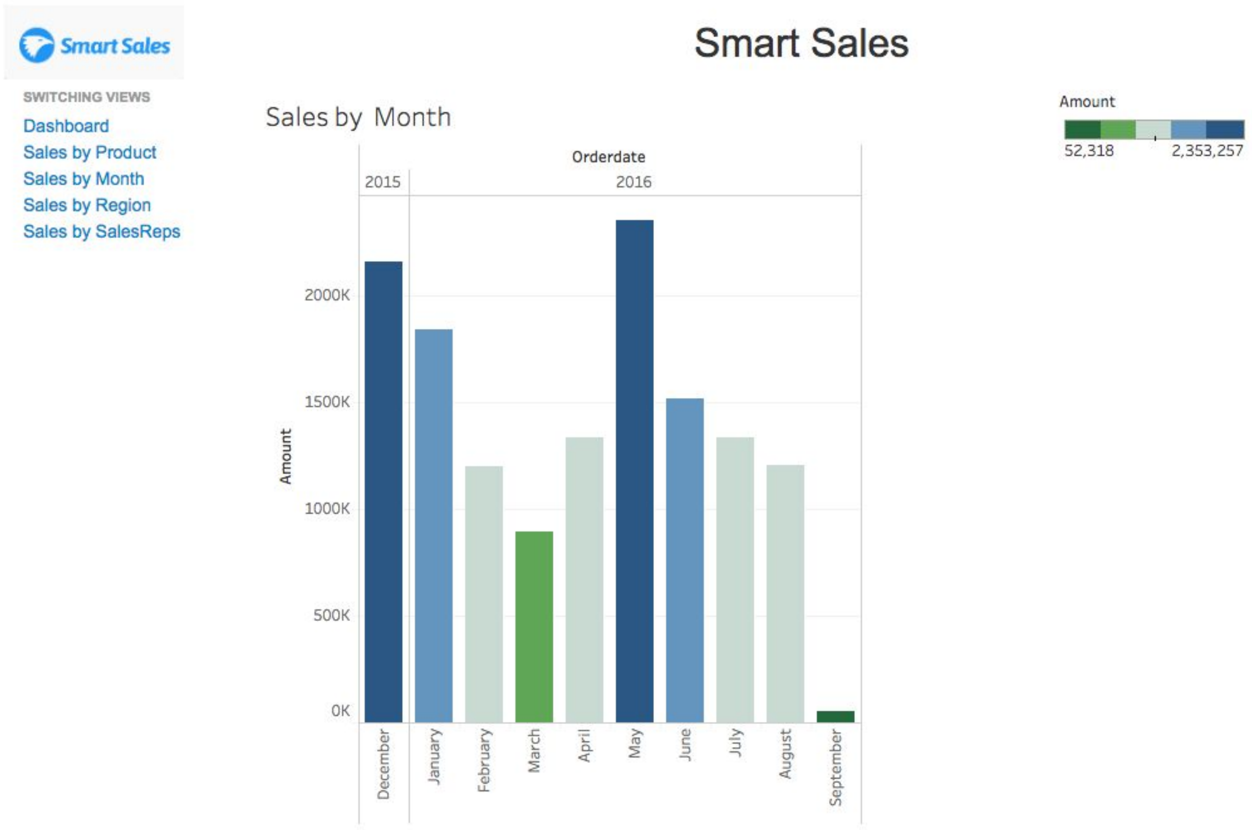


Figure 5 - Sales by Month

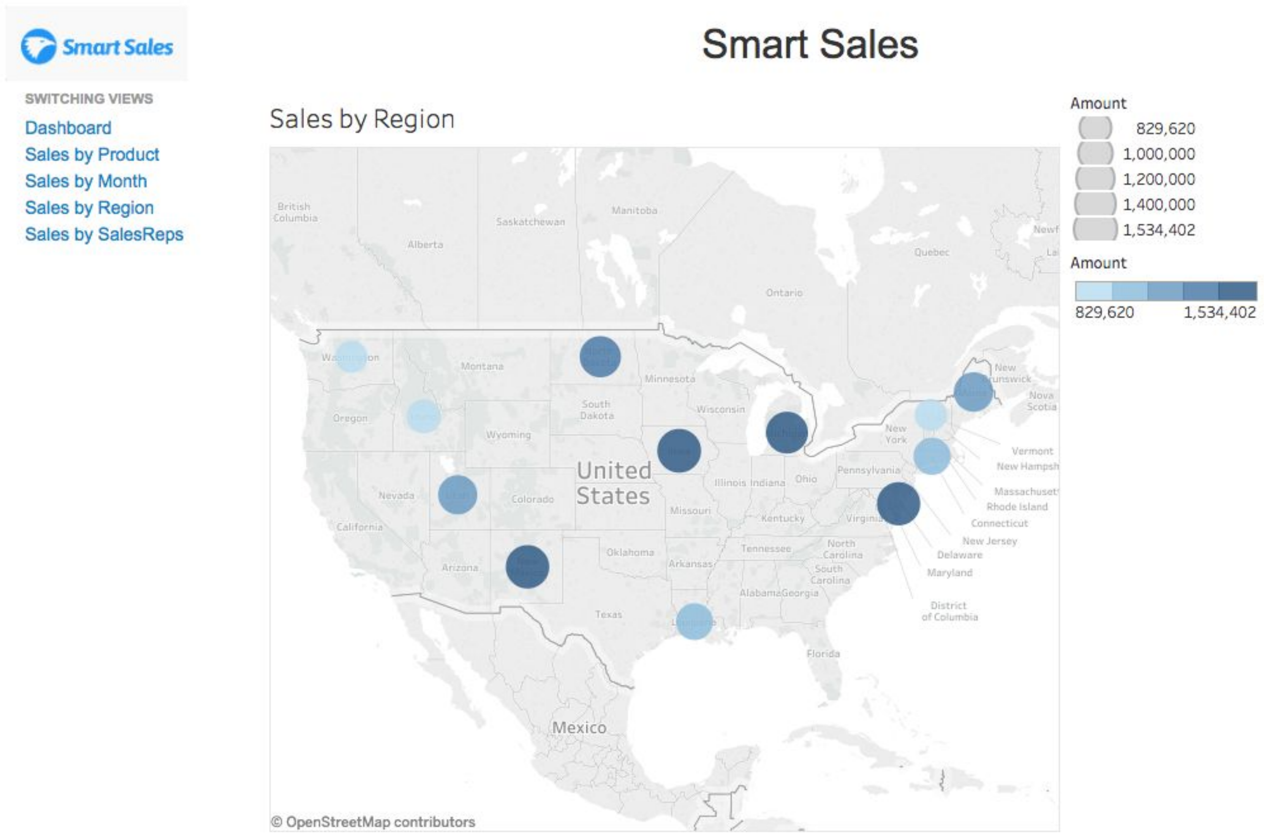


Figure 6 - Sales by Region

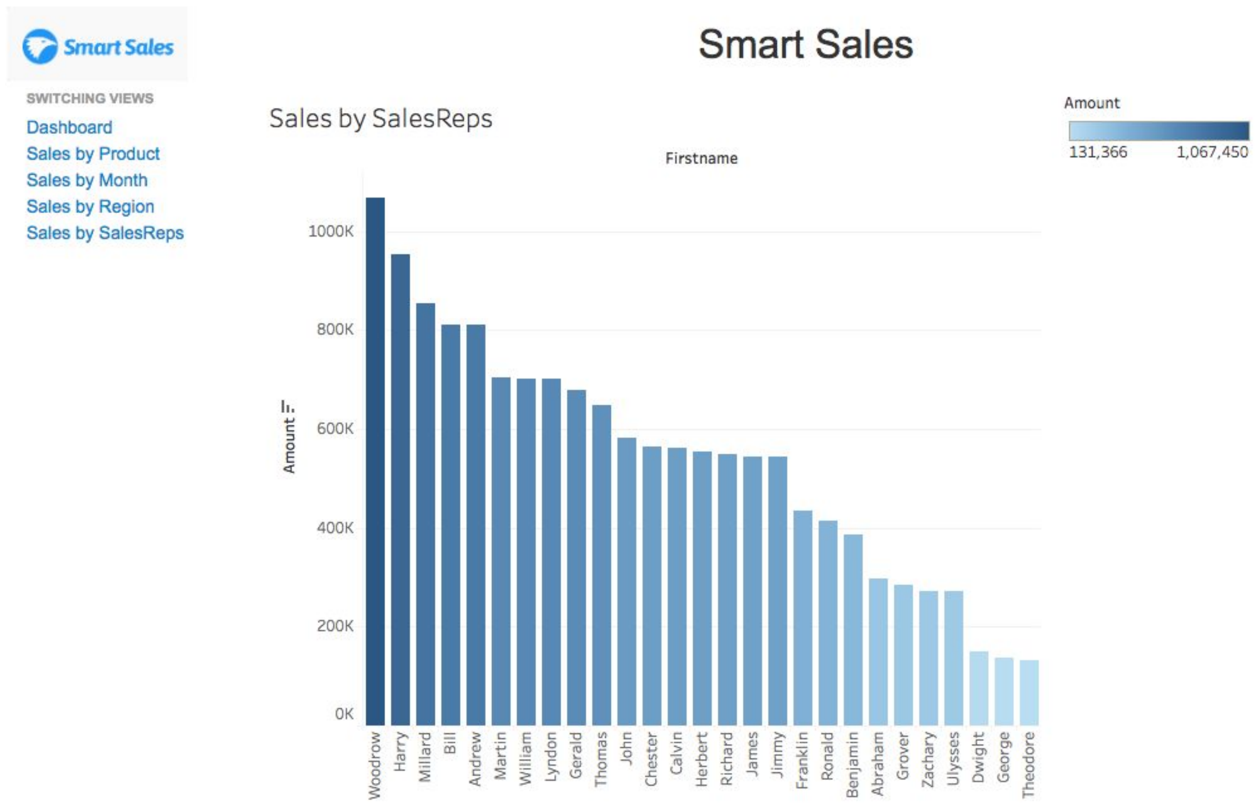


Figure 5 - Sales by Employee

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

No performance requirement but extensive error handling is required as OpenStack can produce wide range of issues.

## 5.2 Safety Requirements

Tableau account, Okta account and Talend account used in this project are trial account and they will expired soon. Unless those account are refreshed, user may not be able to use the app properly

## 5.3 Security Requirements

This application is run in lab environment and doesn't spend any serious effort on security issues for now

### 5.4 Software Quality Attributes

Good error handling and don’t crash too often.

### 5.5 Business Rules

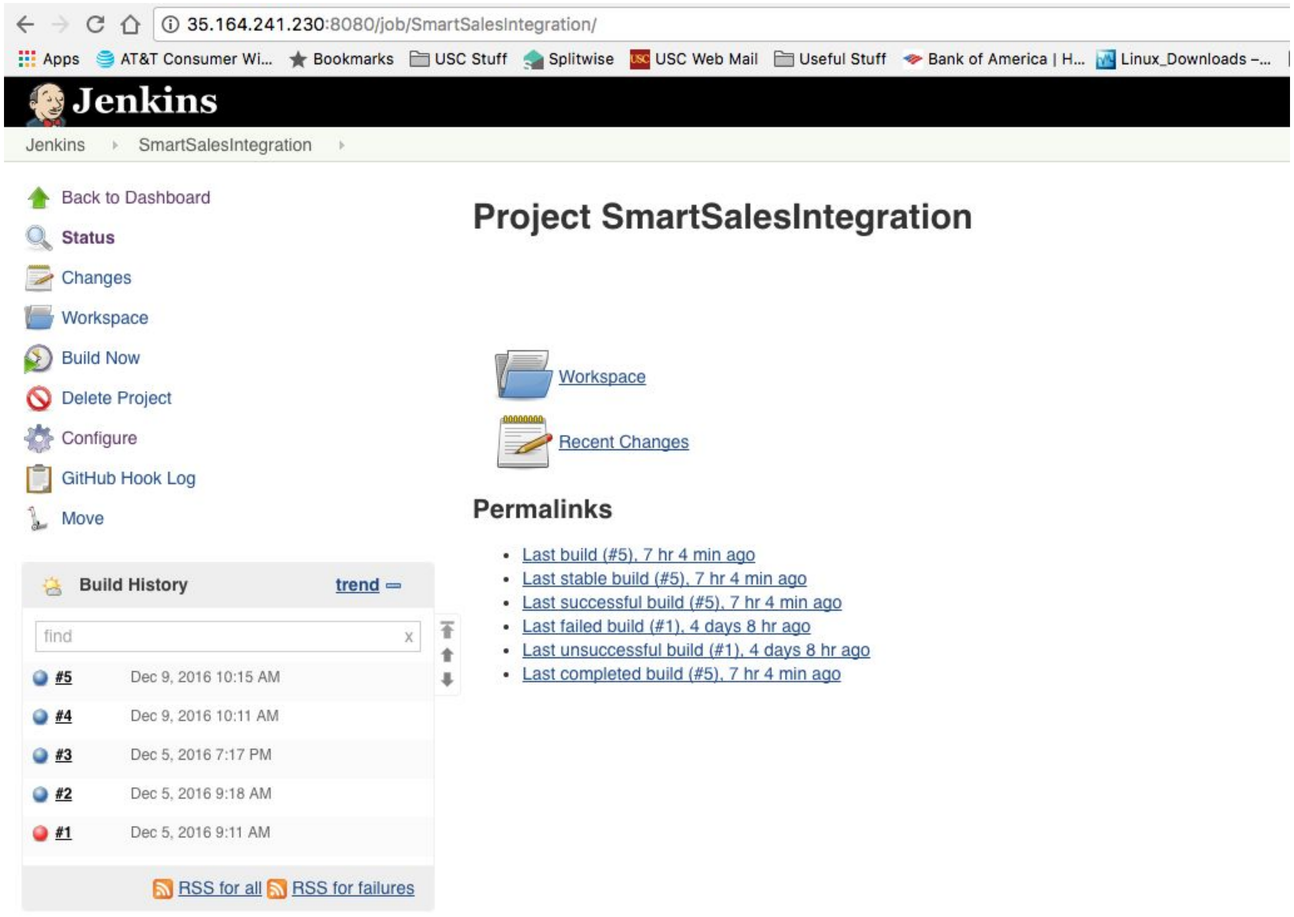
This application is run on lab environment for experiment purpose, therefore, not comply to any specific business rule

## 6. Other Requirements

### 6. 1 Continuous Integration with Jenkins

Jenkins is a continuous integration and continuous delivery application tool. In our project we have used Jenkins for deploying our application to AWS S3 everytime we make changes to the code and commit it to github.
Installed Jenkins on an EC2 instance and installed git plugin and S3 plugin
Created a new job on Jenkins to build and deploy to S3 when a change is pushed into github
On github repository give the jenkins url in the webhooks payload URL

Screen shot of automatic Jenkins Smart integration build triggered when code change in github has been pushed



35.164.241.230:8080/job/SmartSalesIntegration/

Apps AT&T Consumer Wi... Bookmarks USC Stuff Splitwise USC Web Mail Useful Stuff Bank of America | H... Linux_Downloads –...

Jenkins

Jenkins › SmartSalesIntegration ›

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure
- GitHub Hook Log
- Move

Build History trend

find

| | |
|---|---|
| #5 | Dec 9, 2016 10:15 AM |
| #4 | Dec 9, 2016 10:11 AM |
| #3 | Dec 5, 2016 7:17 PM |
| #2 | Dec 5, 2016 9:18 AM |
| #1 | Dec 5, 2016 9:11 AM |

RSS for all RSS for failures

Project SmartSalesIntegration

Workspace

Recent Changes

Permalinks

- Last build (#5), 7 hr 4 min ago
- Last stable build (#5), 7 hr 4 min ago
- Last successful build (#5), 7 hr 4 min ago
- Last failed build (#1), 4 days 8 hr ago
- Last unsuccessful build (#1), 4 days 8 hr ago
- Last completed build (#5), 7 hr 4 min ago