DATA ENGINEERING

CREDIT CARD SYSTEM **CASE STUDY**

v. 1.0	Last Update:	09/12/2018

Table of Contents

PROJECT SUMMARY

The purpose of this course project is to create a Credit Card System that can manage the customer and transactional details of a Bank. This includes several modules which control the flow of the actions as defined below in the "Project Flow".

PROJECT FLOW

- The flow of action begins in the Admin Module, which presents the user with a menu of choices to select depending on which queries they wish to run.
- Once selected, a method is called, located in the TransactionApp.java file, which takes inputs from the user and then displays the results.
- Based on the user inputs, this data is placed in variables which are passed the TransactionDao.java file. The TransactionDAO makes a connection to the database, using prepared SQL queries called in a Java file called MySQLQueries.java, to pull the specific data that is needed. These values are then passed back to the Data Access Objects (one for transactions called TransactionDAO and one for Customers called CustomerDAO) and using Setters and Getters they are made available for use by the TransactionApp to display the data.

FILE STUCTURE

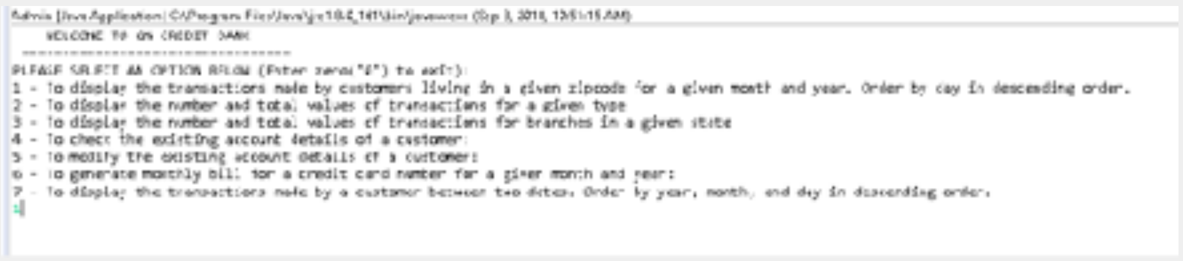
- ADMIN PACKAGE (/src/admin)
  - **admin.java** = main "runner" class where the user selects which queries they would like to perform
  - **TransactionApp.java** = where the various methods are called and data is printed according to user inputs.
- DAO PACKAGE (src/DAO)
  - **CustomerDAO.java** = sets the values for Customer Details, which get passed to the database to and from the Database using setters and getters and using SQL queries called as methods from the MYSQLQueries.java file.
  - **TransactionDAO.java** = sets the values for Transaction Details, which get passed to the database to and from the Database using setters and getters.
  - **JDBC_readerDAO.java** = the abstract class used to connect to the MySQL Database using prepared statements to ensure the database's integrity.
  - MODELS PACKAGE (src/MODELS)
    - **TransactionDetails.java** = variables, setters and getters used to store both Customer Detail and Transaction data from the MySQL database
  - SQLQUERIES (src/SQLQUERIES)
    - **MySQLQueries.java** = stores all the SQL queries used to retrieve and update data in the MySQL Database.
  - Referenced Libraries (/Referenced Libraries)
    - **mysql-connector_java-8.0.11.jar** = JDBC MySQL Driver
  - dbcredentials.properties (/dbcredentials.properties)
    - **dbcredentials.properties** = database connection properties (such as host name, user name and password), stored in plain-text, used to authenticate and connect to the database.

MODULES

Upon entering the Application, the user is presented with with an options menus (see fig.1A below).

- **Admin -** written in Java this module is the main entry point into the Credit Card System and allows the user to select from a menu different queries they wish to perform.
- Each selection to from the menu calls a method defined in the TransactionApp

FIG 1A



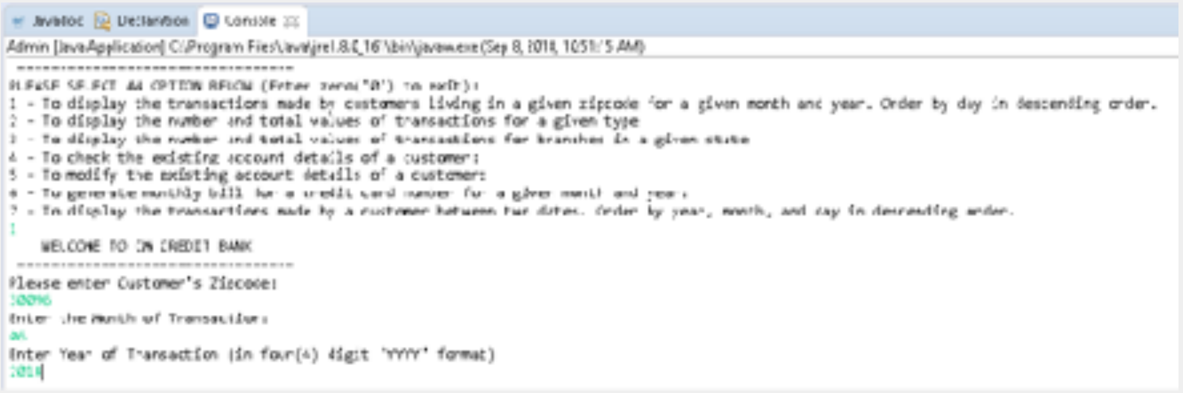
```
Admin [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 3, 2018, 12:51:15 AM)
    WELCOME TO CN CREDIT BANK
 ---------------------------------
PLEASE SELECT AN OPTION BELOW (Enter zero("0") to exit):
1 - To display the transactions made by customers living in a given zipcode for a given month and year. Order by day in descending order.
2 - To display the number and total values of transactions for a given type
3 - To display the number and total values of transactions for branches in a given state
4 - To check the existing account details of a customer:
5 - To modify the existing account details of a customer:
6 - To generate monthly bill for a credit card number for a given month and year:
7 - To display the transactions made by a customer between two dates. Order by year, month, and day in descending order.
1
```

2.1.1 | Transactional Details Module

- written in Java this module allows the user to performs queries against the SQL Database to retrieve transactional data. See each option below:

1 Enter selection one(1) and the Customer's Zipcode, transaction month and transaction year (**see Fig. 1B** below)

FIG 1B



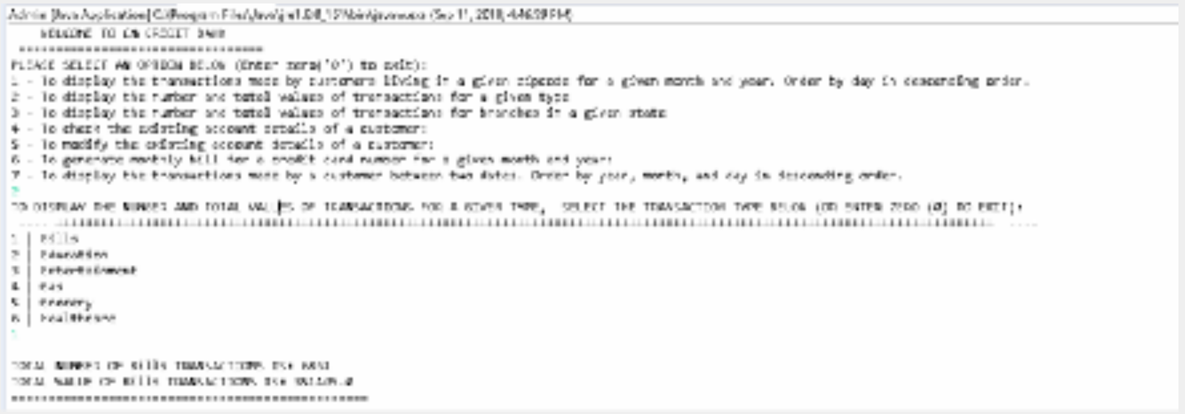
```
Javadoc  Declaration  Console
Admin [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 8, 2018, 10:51:15 AM)
 ---------------------------------
PLEASE SELECT AN OPTION BELOW (Enter zero("0") to exit):
1 - To display the transactions made by customers living in a given zipcode for a given month and year. Order by day in descending order.
2 - To display the number and total values of transactions for a given type
3 - To display the number and total values of transactions for branches in a given state
4 - To check the existing account details of a customer:
5 - To modify the existing account details of a customer:
6 - To generate monthly bill for a credit card number for a given month and year:
7 - To display the transactions made by a customer between two dates. Order by year, month, and day in descending order.
1
    WELCOME TO CN CREDIT BANK
 ---------------------------------
Please enter Customer's Zipcode:
10096
Enter the Month of Transaction:
06
Enter Year of Transaction (in four(4) digit 'YYYY' format)
2018
```

2 To display the number and total values of transactions for branches in a given state (**see Fig. 1C** below)

Once this option is selected, you are given an option to select, by number, the Transaction Type and this will run the program and output the results to the Console Screen. Upon completion, you are given the option to the run the same program again and choose another Transaction Type or exit.

MODULES

F I G 1 C

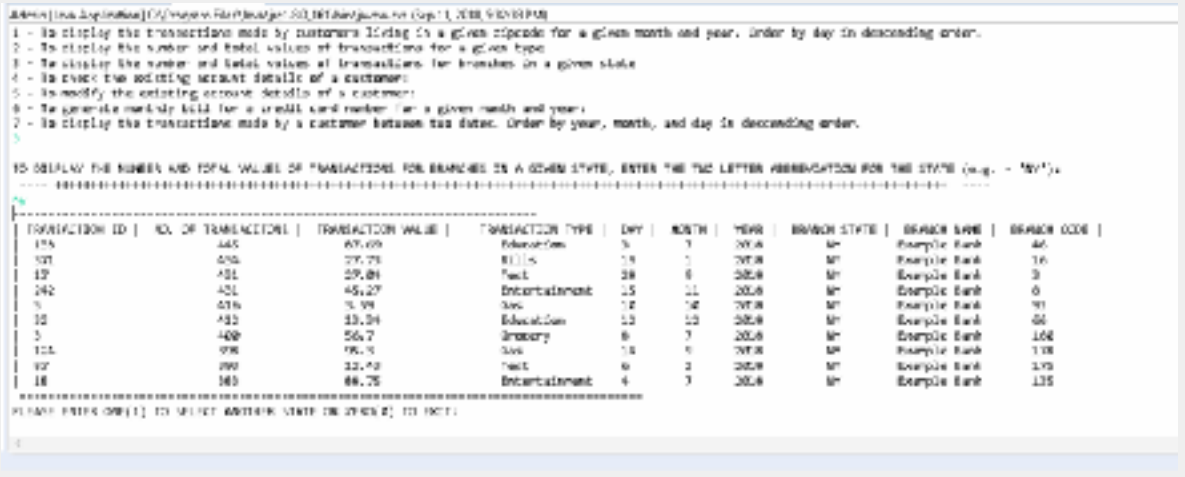


```
Admin [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Sep 11, 2018, 4:46:39 PM)
    WELCOME TO CN CREDIT BANK
 ********************************
PLEASE SELECT AN OPTION BELOW (Enter zero('0') to exit):
1 - To display the transactions made by customers living in a given zipcode for a given month and year. Order by day in descending order.
2 - To display the number and total values of transactions for a given type
3 - To display the number and total values of transactions for branches in a given state
4 - To check the existing account details of a customer:
5 - To modify the existing account details of a customer:
6 - To generate monthly bill for a credit card number for a given month and year:
7 - To display the transactions made by a customer between two dates. Order by year, month, and day in descending order.
2
TO DISPLAY THE NUMBER AND TOTAL VALUES OF TRANSACTIONS FOR A GIVEN TYPE,  SELECT THE TRANSACTION TYPE BELOW (OR ENTER ZERO (0) TO EXIT):
 ---- ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  ----
1 | Bills
2 | Education
3 | Entertainment
4 | Gas
5 | Grocery
6 | Healthcare
1

TOTAL NUMBER OF BILLS TRANSACTIONS IS: [illegible]
TOTAL VALUE OF BILLS TRANSACTIONS IS: [illegible]
************************************************
```

3 To display the number and total values of transactions for branches in a given state (**Fig .1D**)

Selecting this option, you are prompted to enter a State using the State's two-letter abbreviation. After displaying the results, you can choose to run the program again and enter another State.

F I G 1 D



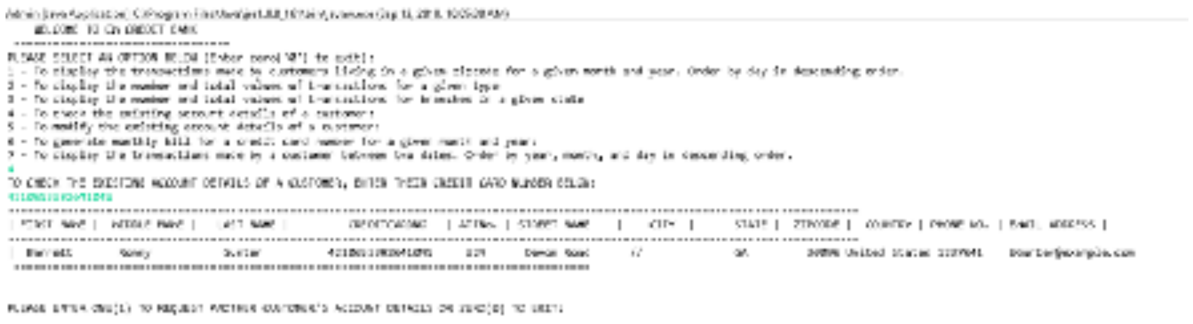
```
Admin [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Sep 11, 2018, [illegible] PM)
1 - To display the transactions made by customers living in a given zipcode for a given month and year. Order by day in descending order.
2 - To display the number and total values of transactions for a given type
3 - To display the number and total values of transactions for branches in a given state
4 - To check the existing account details of a customer:
5 - To modify the existing account details of a customer:
6 - To generate monthly bill for a credit card number for a given month and year:
7 - To display the transactions made by a customer between two dates. Order by year, month, and day in descending order.
3

TO DISPLAY THE NUMBER AND TOTAL VALUES OF TRANSACTIONS FOR BRANCHES IN A GIVEN STATE, ENTER THE TWO LETTER ABBREVIATION FOR THE STATE (e.g. - 'NY'):
 ---- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  ----
ny
|--------------------------------------------------------------------------
| TRANSACTION ID |  NO. OF TRANSACTIONS |  TRANSACTION VALUE |    TRANSACTION TYPE |  DAY |  MONTH |  YEAR |  BRANCH STATE |  BRANCH NAME |  BRANCH CODE |
|  175               445              67.69               Education        5       7      2018      NY        Example Bank     46
|  531               456              77.79               Bills            19      1      2018      NY        Example Bank     16
|  17                451              59.04               Test             28      6      2018      NY        Example Bank     3
|  242               451              45.27               Entertainment    15      11     2018      NY        Example Bank     6
|  5                 476              5.99                Gas              12      12     2018      NY        Example Bank     91
|  33                453              13.34               Education        13      13     2018      NY        Example Bank     66
|  3                 400              56.7                Grocery          6       7      2018      NY        Example Bank     168
|  111               398              95.5                Gas              16      5      2018      NY        Example Bank     118
|  97                399              33.43               Test             6       3      2018      NY        Example Bank     175
|  18                385              66.75               Entertainment    4       7      2018      NY        Example Bank     135
 ************************************************************************************
PLEASE ENTER ONE(1) TO SELECT ANOTHER STATE OR ZERO(0) TO EXIT:
```

2.1.2 | CUSTOMER DETAILS MODULE

Using Java Object-Orientated Concepts, a JDBC driver to connect to the MySQL Database and MySQL Queries the user is prompted with options to perform queries to obtain specific Customer details.

4 To check the existing account details of a customer (see Fig 2A)

FIG 2A

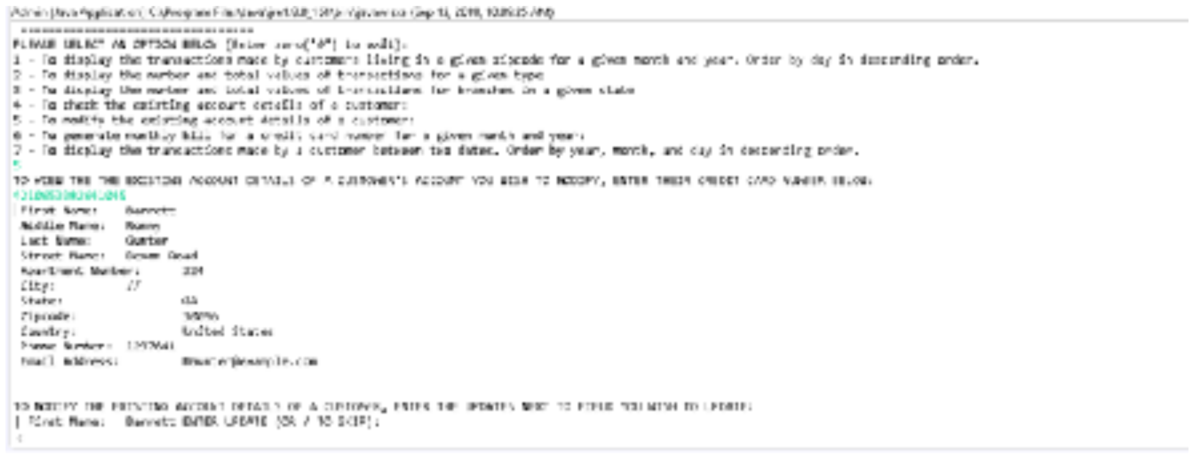


```
[illegible]
    WELCOME TO CN CREDIT CARD
 ---------------------------------
PLEASE SELECT AN OPTION BELOW [Enter zero('0') to exit]:
1 - To display the transactions made by customers living in a given zipcode for a given month and year. Order by day in descending order.
2 - To display the number and total values of transactions for a given type
3 - To display the number and total values of transactions for branches in a given state
4 - To check the existing account details of a customer:
5 - To modify the existing account details of a customer:
6 - To generate monthly bill for a credit card number for a given month and year:
7 - To display the transactions made by a customer between two dates. Order by year, month, and day in descending order.
4
TO CHECK THE EXISTING ACCOUNT DETAILS OF A CUSTOMER, ENTER THEIR CREDIT CARD NUMBER BELOW:
[illegible]
-----------------------------------------------------------------------------------------------------------
| FIRST NAME |  MIDDLE NAME |   LAST NAME |        CREDITCARDNO    | APTNo. | STREET NAME    |    CITY  |     STATE |  ZIPCODE |  COUNTRY | PHONE NO. | EMAIL ADDRESS |
-----------------------------------------------------------------------------------------------------------
| Barrett      Sonny          Suter          [illegible]     [illegible]      Devon Road       //              GA        [illegible] United States [illegible]    [illegible]@example.com
 ***************************************************************************


PLEASE ENTER ONE(1) TO REQUEST ANOTHER CUSTOMER'S ACCOUNT DETAILS OR ZERO(0) TO EXIT:
```

5 To modify the existing account details of a customer (See **FIG. 2B** below..)

Using the customer's credit card number, the account details are retrieved and the user is given an option of updating or skipping (leaving the same) each field.

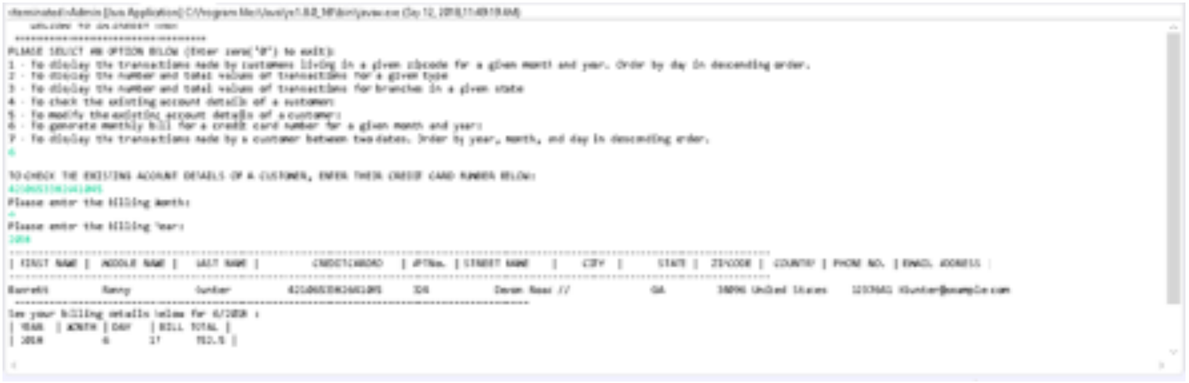
FIG 2B



```
[illegible]
 *********************************
PLEASE SELECT AN OPTION BELOW [Enter zero('0') to exit]:
1 - To display the transactions made by customers living in a given zipcode for a given month and year. Order by day in descending order.
2 - To display the number and total values of transactions for a given type
3 - To display the number and total values of transactions for branches in a given state
4 - To check the existing account details of a customer:
5 - To modify the existing account details of a customer:
6 - To generate monthly bill for a credit card number for a given month and year:
7 - To display the transactions made by a customer between two dates. Order by year, month, and day in descending order.
5
TO VIEW THE THE EXISTING ACCOUNT DETAILS OF A CUSTOMER'S ACCOUNT YOU WISH TO MODIFY, ENTER THEIR CREDIT CARD NUMBER BELOW:
[illegible]
| First Name:    Barrett
 Middle Name:    Sonny
 Last Name:      Suter
 Street Name:    Devon Road
 Apartment Number:       [illegible]
 City:           //
 State:                  GA
 Zipcode:                [illegible]
 Country:                United States
 Phone Number:  [illegible]
 Email Address:          [illegible]@example.com


TO MODIFY THE EXISTING ACCOUNT DETAILS OF A CUSTOMER, ENTER THE UPDATES NEXT TO FIELD YOU WISH TO UPDATE:
| First Name:    Barrett ENTER UPDATE (OR / TO SKIP):
```

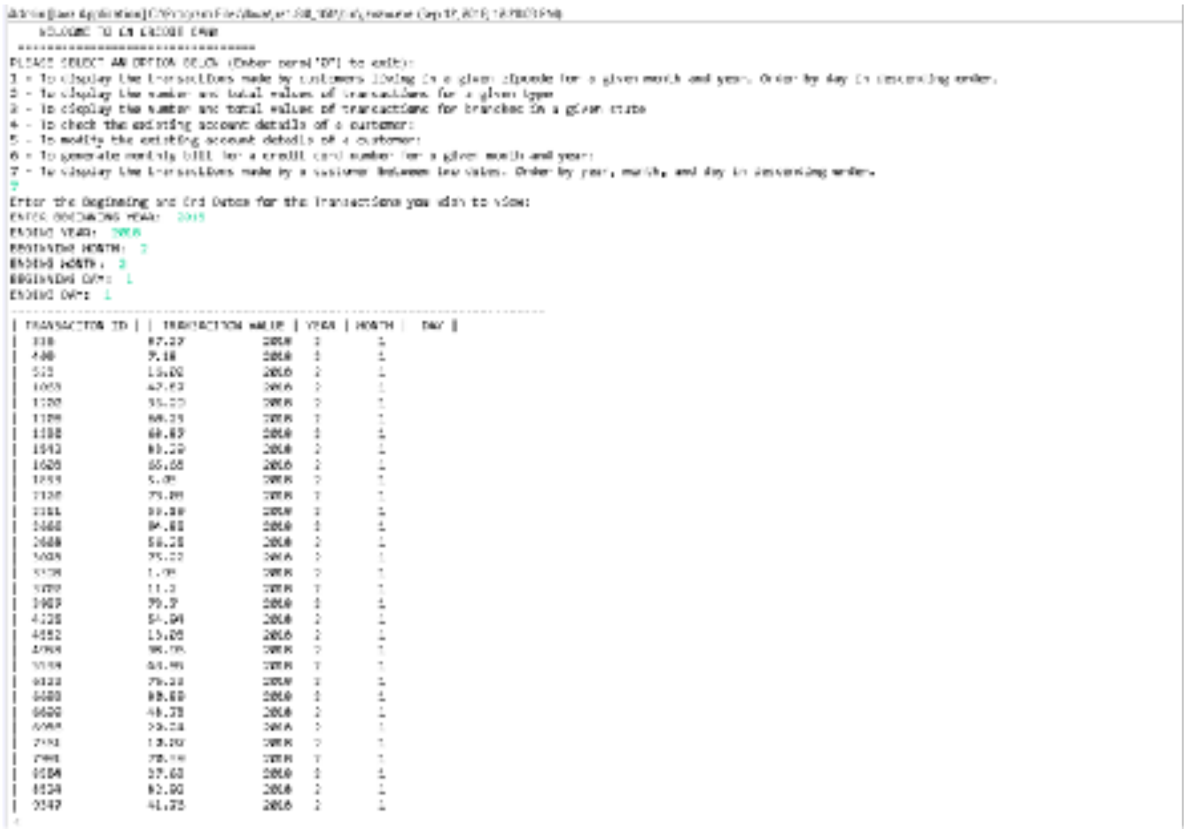
6 To generate monthly bill for a credit card number for a given month and year. (see **Fig. 2C** below..)

FIG 2C



```
<terminated> Admin [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Sep 12, 2018, 11:49:19 AM)
    WELCOME TO CDW CREDIT CARD
 *********************************
PLEASE SELECT AN OPTION BELOW (Enter zero('0') to exit):
1 - To display the transactions made by customers living in a given zipcode for a given month and year. Order by day in descending order.
2 - To display the number and total values of transactions for a given type
3 - To display the number and total values of transactions for branches in a given state
4 - To check the existing account details of a customer:
5 - To modify the existing account details of a customer:
6 - To generate monthly bill for a credit card number for a given month and year:
7 - To display the transactions made by a customer between two dates. Order by year, month, and day in descending order.
6

TO CHECK THE EXISTING ACCOUNT DETAILS OF A CUSTOMER, ENTER THEIR CREDIT CARD NUMBER BELOW:
4210653310341085
Please enter the billing month:
6
Please enter the billing Year:
2018
--------------------------------------------------------------------------------------------------------------------------------
| FIRST NAME |  MIDDLE NAME |   LAST NAME |         CREDITCARDNO    | APTNo. | STREET NAME    |    CITY  |     STATE |  ZIPCODE |  COUNTRY | PHONE NO. | EMAIL ADDRESS |
--------------------------------------------------------------------------------------------------------------------------------
Barrett        Renny          Gunter         4210653310341085     326          Devon Road //           GA         30096 United States    1237661 RGunter@example.com
 -------------------------------------------------------------------------------------
See your billing details below for 6/2018 :
| YEAR   | MONTH | DAY    | BILL TOTAL |
| 2018          6        17       792.5 |
```

7 To display the transactions made by a customer between two dates. Order by year, month, and day in descending order. (See **Fig. 2D** below ..)



```
Admin [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Sep 17, 2018, 12:35:03 PM)
    WELCOME TO CDW CREDIT CARD
 *********************************
PLEASE SELECT AN OPTION BELOW (Enter zero('0') to exit):
1 - To display the transactions made by customers living in a given zipcode for a given month and year. Order by day in descending order.
2 - To display the number and total values of transactions for a given type
3 - To display the number and total values of transactions for branches in a given state
4 - To check the existing account details of a customer:
5 - To modify the existing account details of a customer:
6 - To generate monthly bill for a credit card number for a given month and year:
7 - To display the transactions made by a customer between two dates. Order by year, month, and day in descending order.
7
Enter the Beginning and End Dates for the Transactions you wish to view:
ENTER BEGINNING YEAR:  2018
ENDING YEAR:  2018
BEGINNING MONTH:  2
ENDING MONTH :  2
BEGINNING DAY:  1
ENDING DAY:  1
-----------------------------------------------------------------------------
| TRANSACTION ID | | TRANSACTION VALUE | YEAR | MONTH |   DAY |
|  338             87.27            2018    2         1
|  400             7.18             2018    2         1
|  523             15.02            2018    2         1
|  1053            47.53            2018    2         1
|  1100            55.22            2018    2         1
|  1108            66.23            2018    2         1
|  1188            60.83            2018    2         1
|  1542            80.22            2018    2         1
|  1626            65.68            2018    2         1
|  1853            5.09             2018    2         1
|  2126            73.88            2018    2         1
|  2161            99.30            2018    2         1
|  2466            94.88            2018    2         1
|  2688            56.28            2018    2         1
|  3028            25.22            2018    2         1
|  3128            1.09             2018    2         1
|  3272            11.3             2018    2         1
|  3987            79.7             2018    2         1
|  4228            54.94            2018    2         1
|  4552            15.26            2018    2         1
|  4798            98.75            2018    2         1
|  5158            64.95            2018    2         1
|  6122            79.22            2018    2         1
|  6480            88.60            2018    2         1
|  6620            48.78            2018    2         1
|  6968            55.24            2018    2         1
|  7151            18.02            2018    2         1
|  7991            78.14            2018    2         1
|  8584            57.63            2018    2         1
|  8624            62.02            2018    2         1
|  9347            41.75            2018    2         1
```

2.2.1 Data Extraction and Transportation with Sqoop MODULE

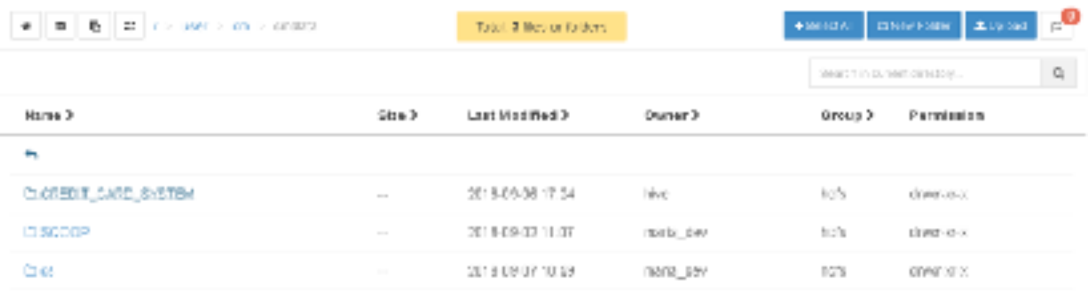
Utilize Sqoop to extract the following data according to the specifications found in the mapping document:

1. Branch data into CDW_SAPP_BRANCH
2. Credit Card Data into CDW_SAPP_CREDITCARD
3. Time data into CDW_SAPP_TIME
4. Customer Data into CDW_SAPP_CUSTOMER

Using Sqoop commands (see below) data can be pulled from the MYSQL database and transformed using a Select query and placed into HDFS in the above directories

Step 1 Create main folder in HDFS called “/user/cm/cmdata/CREDIT_CARD_SYSTEM

FIG 3A



Name	Size	Last Modified	Owner	Group	Permission
CREDIT_CARD_SYSTEM	--	2018-09-08 17:54	hive	hdfs	drwxr-xr-x
SQOOP	--	2018-09-[illegible] 11:0[illegible]	maria_dev	hdfs	drwxr-xr-x
[illegible]	--	2018-09-07 10:[illegible]	maria_dev	hdfs	drwxr-xr-x

Step 2 Use Sqoop command on CLI to Extract, Transform and Load the data into CDW_SAPP_BRANCH Directory

FIG 3B

```
sqoop import --connect jdbc:mysql://localhost/CDW_SAPP --driver com.mysql.jdbc.Driver -
m1 --query 'select BRANCH_CODE, BRANCH_NAME, BRANCH_STREET, BRANCH_CITY,
BRANCH_STATE,
 IFNULL(BRANCH_ZIP, "999999") as BRANCH_ZIP, concat("(",substr(BRANCH_PHONE,
1,3),") ",substr(BRANCH_PHONE,4,3),"-",substr(BRANCH_PHONE,7)) AS BRANCH_PHONE,
LAST_UPDATED FROM CDW_SAPP_BRANCH WHERE $CONDITIONS' --target-dir /user/
cm/cmdata/CREDIT_CARD_SYSTEM/CDW_SAPP_BRANCH
```

Step 3 Create EXTERNAL TABLE IN HIVE POINTING TO BRANCH directory in HDFS (will import data to the table from HDFS)

FIG 3C

```
create database cdw_sapp;

use cdw_sapp;
CREATE EXTERNAL TABLE branch (
BRANCH_CODE INT,
BRANCH_NAME STRING,
BRANCH_STREET STRING,
BRANCH_CITY STRING,
BRANCH_STATE STRING,
BRANCH_ZIP INT,
BRANCH_PHONE STRING,
LAST_UPDATED TIMESTAMP
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
location "/user/cm/cmdata/CREDIT_CARD_SYSTEM/CDW_SAPP_BRANCH";
```

STEP 4 Create EXTERNAL PARTITION TABLE IN HIVE POINTING TO BRANCH directory in HDFS

```
use cdw_sapp;

SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;

CREATE EXTERNAL TABLE CDW_SAPP_D_BRANCH (
BRANCH_CODE INT,
BRANCH_NAME STRING,
BRANCH_STREET STRING,
BRANCH_CITY STRING,
BRANCH_ZIP INT,
BRANCH_PHONE STRING,
LAST_UPDATED TIMESTAMP
)
PARTITIONED BY (BRANCH_STATE STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
location "/user/cm/cmdata/CREDIT_CARD_SYSTEM/CDW_SAPP_BRANCH";
```

STEP 5 Insert DATA FROM EXTERNAL branch TABLE TO EXTERNAL PARTITION TABLE

	use cdw_sapp; SET hive.exec.dynamic.partition=true; SET hive.exec.dynamic.partition.mode=nonstrict; INSERT OVERWRITE TABLE CDW_SAPP_D_BRANCH PARTITION (BRANCH_STATE) SELECT BRANCH_CODE, BRANCH_NAME, BRANCH_STREET, BRANCH_CITY, BRANCH_ZIP, BRANCH_PHONE, LAST_UPDATED, BRANCH_STATE FROM branch;		
STEP 6	REPEAT EACH OF THE ABOVE PROCESSES BUT FOR CREDITCARD, CUSTOMER AND TIMEID TABLES/DIRECTORIES. (see files in 12Sep2018_DE03-NY_CaseStudy_DataLoadingwithHive.zip file)		

2.2.3	**Automating the Process with Oozie MODULE**
1) **Create an Oozie Workflow that will automate the processes of steps 2.2.1 and 2.2.2.** • Each of the files in step 2.2.1 should be deleted before the workflow is executed in order to prevent storage of redundant data • The tables created in step 2.2.2 should be dropped before executing the hive workflow in order to prevent redundancy. 2) **Incorporate that workflow into an Oozie Coordinator that will execute with the following conditions:** • Every weekday between 08:00 and 18:00 EST • Executes once every 20 minutes • Starts on April 2nd 2018 at 08:00 EST • Ends on March 29th 2025 at 18:00 EST	
Step 1	Create SQOOP JOBS TO RUN IN THE META STORE • see file 05Sept2018_OOZIEWORKFLOW-METASTOREJOBS_SCRIPT1
S A M P L E	sqoop job --meta-connect jdbc:hsqldb:hsql://localhost:16000/sqoop --create etlBranchData -- import --connect jdbc:mysql://localhost/CDW_SAPP --driver com.mysql.jdbc.Driver --query 'SELECT BRANCH_CODE, BRANCH_NAME, BRANCH_STREET, BRANCH_CITY, BRANCH_STATE, IFNULL(BRANCH_ZIP, "999999") as BRANCH_ZIP, concat("(",substr(BRANCH_PHONE, 1,3),")",substr(BRANCH_PHONE,4,3),"-",substr(BRANCH_PHONE,7)) AS BRANCH_PHONE, LAST_UPDATED FROM CDW_SAPP_BRANCH WHERE $CONDITIONS' --fields-terminated-by ',' --delete-target-dir --target-dir /user/cm/cmdata/CREDIT_CARD_SYSTEM/CDW_SAPP_BRANCH -m1
Step 2	CREATE HIVE QUERY FILE TO CREATE AND INSERT DATA FOR EACH TABLE • See files named: • 02Sept2018_BRANCHDATA_HIVEQUERY.sql • 02Sept2018_CREDITDATA_HIVEQUERY.sql • 02Sept2018_TIMEIDCUSTOMERDATA-HIVESQUERY.sql in 12Sep2018_DE03-NY_CaseStudy_AutomatingtheProccesWithOozie.zip file

S A M P L E	see code below

```
create database if not exists cdw_sapp;

use cdw_sapp;

SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;

CREATE EXTERNAL TABLE branch (
BRANCH_CODE INT,
BRANCH_NAME STRING,
BRANCH_STREET STRING,
BRANCH_CITY STRING,
BRANCH_STATE STRING,
BRANCH_ZIP INT,
BRANCH_PHONE STRING,
LAST_UPDATED TIMESTAMP
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
location "/user/cm/cmdata/CREDIT_CARD_SYSTEM/CDW_SAPP_BRANCH";

CREATE EXTERNAL TABLE CDW_SAPP_D_BRANCH (
BRANCH_CODE INT,
BRANCH_NAME STRING,
BRANCH_STREET STRING,
BRANCH_CITY STRING,
BRANCH_ZIP INT,
BRANCH_PHONE STRING,
LAST_UPDATED TIMESTAMP
)
PARTITIONED BY (BRANCH_STATE STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
location "/user/cm/cmdata/CREDIT_CARD_SYSTEM/CDW_SAPP_BRANCH";

INSERT OVERWRITE TABLE CDW_SAPP_D_BRANCH
PARTITION (BRANCH_STATE)
SELECT
BRANCH_CODE,
BRANCH_NAME,
BRANCH_STREET,
BRANCH_CITY,
BRANCH_ZIP,
BRANCH_PHONE,
LAST_UPDATED,
BRANCH_STATE

FROM branch;
```

Step 3	• CREATE workflow, coordinator and job.properties files and download java-json.jar file • Place job.properties file in a directory in Linux (see sample path below) • Run script from Linux CLI pointing to job.properties file • oozie job -oozie http://localhost:11000/oozie -config Documents/oozie/cs/coordinator/job.properties -run		

2.2.4	**Optimizing the Process MODULE**
1) Create a new Oozie workflow similar to the process of 2.2.3. This time, however, Sqoop should only import new data. Hive should then import only that new data. Original data should not be deleted in this process. • Each of the files in step 2.2.1 should be deleted before the workflow is executed in order to prevent storage of redundant data • The tables created in step 2.2.2 should be dropped before executing the hive workflow in order to prevent redundancy. **2) Modify the Oozie Coordinator to use this workflow rather than the original, unoptimized, workflow**	
Step 1	Create Updated SQOOP JOBS IN THE META STORE • see file 05Sept2018_OOZIEWORKFLOW-METASTOREJOBS_SCRIPT2
S A M P L E	sqoop job --meta-connect jdbc:hsqldb:hsql://localhost:16000/sqoop --create etlBranchData_Mod -- import --connect jdbc:mysql://localhost/CDW_SAPP --driver com.mysql.jdbc.Driver --query 'SELECT BRANCH_CODE, BRANCH_NAME, BRANCH_STREET, BRANCH_CITY, BRANCH_STATE, IFNULL(BRANCH_ZIP, "999999") as BRANCH_ZIP, concat("(",substr(BRANCH_PHONE, 1,3),")",substr(BRANCH_PHONE,4,3),"-",substr(BRANCH_PHONE,7)) AS BRANCH_PHONE, LAST_UPDATED FROM CDW_SAPP_BRANCH WHERE $CONDITIONS' --incremental append --check-column LAST_UPDATED --last-value '2018-04-18 20:51:47.0' --fields-terminated-by ',' --target-dir /user/cm/cmdata/CREDIT_CARD_SYSTEM/ CDW_SAPP_BRANCH -m1
Step 2	CREATE HIVE QUERY FILE TO CREATE AND INSERT DATA FOR EACH TABLE • See files named: • 02Sept2018_BRANCHDATA_HIVEQUERY.sql • 02Sept2018_CREDITDATA_HIVEQUERY.sql • 02Sept2018_TIMEIDCUSTOMERDATA-HIVESQUERY.sql in 12Sep2018_DE03-NY_CaseStudy_OptimizingingtheProcces.zip file

S A M P L E	See code below

S
A
M
P
L
E

```
create database if not exists cdw_sapp;

use cdw_sapp;

SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;

CREATE EXTERNAL TABLE branch (
BRANCH_CODE INT,
BRANCH_NAME STRING,
BRANCH_STREET STRING,
BRANCH_CITY STRING,
BRANCH_STATE STRING,
BRANCH_ZIP INT,
BRANCH_PHONE STRING,
LAST_UPDATED TIMESTAMP
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
location "/user/cm/cmdata/CREDIT_CARD_SYSTEM/CDW_SAPP_BRANCH";

CREATE EXTERNAL TABLE CDW_SAPP_D_BRANCH (
BRANCH_CODE INT,
BRANCH_NAME STRING,
BRANCH_STREET STRING,
BRANCH_CITY STRING,
BRANCH_ZIP INT,
BRANCH_PHONE STRING,
LAST_UPDATED TIMESTAMP
)
PARTITIONED BY (BRANCH_STATE STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
location "/user/cm/cmdata/CREDIT_CARD_SYSTEM/CDW_SAPP_BRANCH";

INSERT OVERWRITE TABLE CDW_SAPP_D_BRANCH
PARTITION (BRANCH_STATE)
SELECT
BRANCH_CODE,
BRANCH_NAME,
BRANCH_STREET,
BRANCH_CITY,
BRANCH_ZIP,
BRANCH_PHONE,
LAST_UPDATED,
BRANCH_STATE

FROM branch;
```

Step 3	• CREATE workflow, coordinator and job.properties files and download java-json.jar file • Place job.properties file in a directory in Linux (see sample path below) • Run script from Linux CLI pointing to job.properties file • oozie job -oozie http://localhost:11000/oozie -config Documents/oozie/cs/coordinatormod/job.properties -run		

2.2.5	**Visualization of Dataset MODULE**		
1) Use Hive Query and Hive Visualization tool • Find the top 20 zip codes(hint: branch_zip) by total transaction value • Find total transaction value for each transaction type by Quarter in 2018			
Step 1	RUN HIVE QUERY TO PULL “TOP 20 ZIPCODES BY TOTAL VALUE) • see script below		

	use cdw_sapp;
	
	SELECT SUM(cc.TRANSACTION_VALUE), br.BRANCH_ZIP
	  from cdw_sapp_f_credit_card cc
	  join cdw_sapp_d_branch br
	  on br.BRANCH_CODE = cc.BRANCH_CODE GROUP BY br.branch_zip order by 1 desc limit 20;		

Step 2	ONCE THE RESULTS ARE DISPLAYED CLICK ON THE VISUALIZATION TOOL TO SEE THE DATA IN A VISUAL FORMAT		
Step 3	RUN HIVE QUERY TO FIND TRANSACTION VALUE OF EACH TRANSACTION TYPE BY QUARTER		

	use cdw_sapp;
	
	SELECT SUM(transaction_value), transaction_type, CASE WHEN cast(substr(timeid,5,2) as INT) <= 3 THEN 1 WHEN cast(substr(timeid,5,2) as INT) BETWEEN 3 AND 6 THEN 2 WHEN cast(substr(timeid,5,2) as INT) BETWEEN 7 AND 9 THEN 3 ELSE 4 END AS QUARTER
	from cdw_sapp_f_credit_card group by transaction_type, CASE WHEN cast(substr(timeid,5,2) as INT) <= 3 THEN 1 WHEN cast(substr(timeid,5,2) as INT) BETWEEN 3 AND 6 THEN 2 WHEN cast(substr(timeid,5,2) as INT) BETWEEN 7 AND 9 THEN 3 ELSE 4 END;		

Step 4	ONCE THE RESULTS ARE DISPLAYED CLICK ON THE VISUALIZATION TOOL TO SEE THE DATA IN A VISUAL FORMAT		