

DATA ENGINEERING

CREDIT CARD SYSTEM

CASE STUDY

v. 1.0

Last Update:

09/17/2018

Table of Contents

I.	PROJECT SUMMARY		

PROJECT SUMMARY

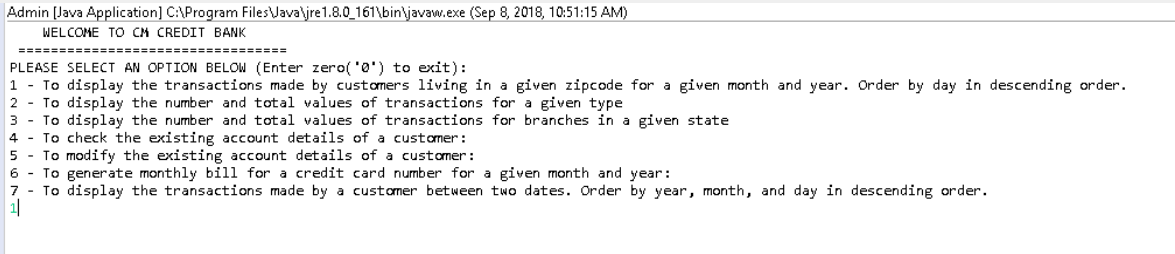
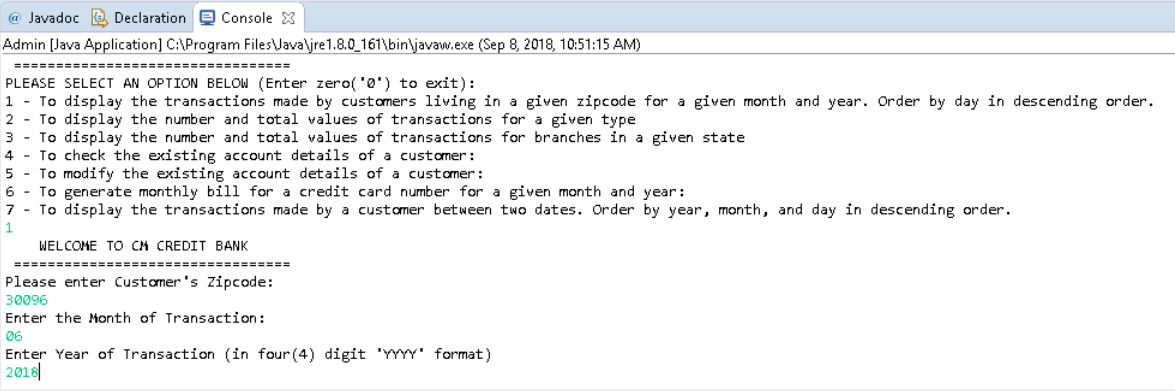
The purpose of this course project is to create a Credit Card System that can manage the customer and transactional details of a Bank. This includes several modules which control the flow of the actions as defined below in the “Project Flow”.

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 STRUCTURE

- 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).	
<ul style="list-style-type: none"> • 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	 <pre> Admin [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 8, 2018, 10:51:15 AM) ===== WELCOME TO CM 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 </pre>
2.1.1 Transactional Details Module	
<ul style="list-style-type: none"> • 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	 <pre> @ 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 CM CREDIT BANK ===== Please enter Customer's Zipcode: 30096 Enter the Month of Transaction: 06 Enter Year of Transaction (in four(4) digit 'YYYY' format) 2018 </pre>
2	To display the number and total values of transactions for branches in a given state (see Fig. 1C below)
<p>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.</p>	

MODULES

F
I
G
1
C

```
Admin [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 11, 2018, 4:46:59 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: 6861
TOTAL VALUE OF Bills TRANSACTIONS IS: 351405.0
=====
```

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_161\bin\javaw.exe (Sep 11, 2018, 5:32:38 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 |
| 138            | 445                  | 87.69              | Education        | 3   | 7      | 2018 | NY           | Example Bank | 46           |
| 371            | 434                  | 27.73              | Bills            | 19  | 1      | 2018 | NY           | Example Bank | 16           |
| 17             | 431                  | 27.04              | Test             | 28  | 9      | 2018 | NY           | Example Bank | 3            |
| 242            | 431                  | 45.27              | Entertainment    | 15  | 11     | 2018 | NY           | Example Bank | 8            |
| 5              | 416                  | 3.59               | Gas              | 10  | 10     | 2018 | NY           | Example Bank | 93           |
| 32             | 412                  | 13.34              | Education        | 12  | 12     | 2018 | NY           | Example Bank | 66           |
| 3              | 400                  | 56.7               | Grocery          | 8   | 7      | 2018 | NY           | Example Bank | 160          |
| 124            | 398                  | 95.3               | Gas              | 14  | 9      | 2018 | NY           | Example Bank | 178          |
| 87             | 393                  | 12.43              | Test             | 6   | 2      | 2018 | NY           | Example Bank | 175          |
| 18             | 383                  | 88.75              | 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)

```
Admin [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 12, 2018, 10:05:30 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.
4
TO CHECK THE EXISTING ACCOUNT DETAILS OF A CUSTOMER, ENTER THEIR CREDIT CARD NUMBER BELOW:
4210653382641045
=====
2 | FIRST NAME | MIDDLE NAME | LAST NAME | CREDITCARDNO | APTNo. | STREET NAME | CITY | STATE | ZIPCODE | COUNTRY | PHONE NO. | EMAIL ADDRESS |
A | Barnett | Ronny | Gunter | 4210653382641045 | 324 | Devon Road | // | GA | 30096 | United States | 1237641 | BGunter@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.

```
Admin [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 12, 2018, 10:09:25 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.
5
TO VIEW THE THE EXISTING ACCOUNT DETAILS OF A CUSTOMER'S ACCOUNT YOU WISH TO MODIFY, ENTER THEIR CREDIT CARD NUMBER BELOW:
4210653382641045
First Name: Barnett
Middle Name: Ronny
Last Name: Gunter
Street Name: Devon Road
Apartment Number: 324
City: //
State: GA
Zipcode: 30096
Country: United States
Phone Number: 1237641
Email Address: BGunter@example.com

TO MODIFY THE EXISTING ACCOUNT DETAILS OF A CUSTOMER, ENTER THE UPDATES NEXT TO FIELD YOU WISH TO UPDATE:
| First Name: Barnett 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..)

F
I
G
2
C

```
<terminated> Admin [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Sep 12, 2018, 11:49:19 AM)
=====
WELCOME TO CM 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.
6
=====
TO CHECK THE EXISTING ACCOUNT DETAILS OF A CUSTOMER, ENTER THEIR CREDIT CARD NUMBER BELOW:
4210653382641045
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      Ronny      Gunter      4210653382641045      324      Devan Road //      GA      30096      United States      1237641      BGunter@example.com
=====
See your billing details below for 6/2018 :
| YEAR | MONTH | DAY | BILL TOTAL |
| 2018 | 6      | 17   | 552.5 |
=====
< >
```

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_161\bin\javaw.exe (Sep 12, 2018, 12:28:03 PM)
=====
WELCOME TO CM 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.
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 |
| 336            | | 87.27             | 2018 | 2      | 1   |
| 409            | | 7.18              | 2018 | 2      | 1   |
| 523            | | 16.02             | 2018 | 2      | 1   |
| 1063           | | 47.87             | 2018 | 2      | 1   |
| 1200           | | 36.29             | 2018 | 2      | 1   |
| 1209           | | 69.25             | 2018 | 2      | 1   |
| 1230           | | 68.87             | 2018 | 2      | 1   |
| 1542           | | 83.29             | 2018 | 2      | 1   |
| 1608           | | 65.68             | 2018 | 2      | 1   |
| 1853           | | 5.09              | 2018 | 2      | 1   |
| 2120           | | 73.03             | 2018 | 2      | 1   |
| 2211           | | 55.19             | 2018 | 2      | 1   |
| 2666           | | 94.85             | 2018 | 2      | 1   |
| 2668           | | 56.25             | 2018 | 2      | 1   |
| 3078           | | 75.22             | 2018 | 2      | 1   |
| 3398           | | 1.93              | 2018 | 2      | 1   |
| 3702           | | 11.2              | 2018 | 2      | 1   |
| 3957           | | 73.7              | 2018 | 2      | 1   |
| 4225           | | 54.94             | 2018 | 2      | 1   |
| 4582           | | 13.03             | 2018 | 2      | 1   |
| 4958           | | 98.96             | 2018 | 2      | 1   |
| 5538           | | 43.95             | 2018 | 2      | 1   |
| 6122           | | 76.23             | 2018 | 2      | 1   |
| 6693           | | 80.89             | 2018 | 2      | 1   |
| 6699           | | 48.73             | 2018 | 2      | 1   |
| 6936           | | 20.24             | 2018 | 2      | 1   |
| 7361           | | 10.07             | 2018 | 2      | 1   |
| 7991           | | 70.39             | 2018 | 2      | 1   |
| 8504           | | 27.63             | 2018 | 2      | 1   |
| 8524           | | 82.92             | 2018 | 2      | 1   |
| 9347           | | 41.76             | 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:																																
<div>1. Branch data into CDW_SAPP_BRANCH</div> <div>2. Credit Card Data into CDW_SAPP_CREDITCARD</div> <div>3. Time data into CDW_SAPP_TIME</div> <div>4. Customer Data into CDW_SAPP_CUSTOMER</div>																																
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																															
F I G 3A	<div><div><div><div><div></div><div></div><div></div><div></div></div><div>/ > user > cm > cmdata</div></div><div>Total: 3 files or folders</div><div><div>Select All</div><div>New Folder</div><div>Upload</div><div></div></div></div><div><div>Search in current directory...</div><div></div></div><table><thead><tr><th>Name ></th><th>Size ></th><th>Last Modified ></th><th>Owner ></th><th>Group ></th><th>Permission</th></tr></thead><tbody><tr><td colspan="6"><div></div></td></tr><tr><td><div>CREDIT_CARD_SYSTEM</div></td><td>--</td><td>2018-09-06 17:24</td><td>hive</td><td>hdfs</td><td>drwxr-xr-x</td></tr><tr><td><div>SGOOP</div></td><td>--</td><td>2018-09-02 11:07</td><td>maria_dev</td><td>hdfs</td><td>drwxr-xr-x</td></tr><tr><td><div>cs</div></td><td>--</td><td>2018-09-07 10:59</td><td>maria_dev</td><td>hdfs</td><td>drwxr-xr-x</td></tr></tbody></table></div>		Name >	Size >	Last Modified >	Owner >	Group >	Permission	<div></div>						<div>CREDIT_CARD_SYSTEM</div>	--	2018-09-06 17:24	hive	hdfs	drwxr-xr-x	<div>SGOOP</div>	--	2018-09-02 11:07	maria_dev	hdfs	drwxr-xr-x	<div>cs</div>	--	2018-09-07 10:59	maria_dev	hdfs	drwxr-xr-x
	Name >	Size >	Last Modified >	Owner >	Group >	Permission																										
	<div></div>																															
	<div>CREDIT_CARD_SYSTEM</div>	--	2018-09-06 17:24	hive	hdfs	drwxr-xr-x																										
	<div>SGOOP</div>	--	2018-09-02 11:07	maria_dev	hdfs	drwxr-xr-x																										
<div>cs</div>	--	2018-09-07 10:59	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																															
F I G 3B	<div>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</div>																															
STEP 3	<div>CREATE DATABASE CALLED “cdw_sapp” using the command below:</div> <div>create database IF NOT EXISTS cdw_sapp;</div>																															
Step 4	Create EXTERNAL TABLE IN HIVE POINTING TO BRANCH directory in HDFS (will import data to the table from HDFS)																															

F I G 3C	<pre> 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"; </pre>
STEP 4	Create EXTERNAL PARTITION TABLE IN HIVE POINTING TO BRANCH directory in HDFS
	<pre> 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"; </pre>
STEP 5	Insert DATA FROM EXTERNAL branch TABLE TO EXTERNAL PARTITION TABLE

	<pre> 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; </pre>		
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
<p>1) Create an Oozie Workflow that will automate the processes of steps 2.2.1 and 2.2.2.</p> <ul style="list-style-type: none"> 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. <p>2) Incorporate that workflow into an Oozie Coordinator that will execute with the following conditions:</p> <ul style="list-style-type: none"> 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	<p>Create SQOOP JOBS TO RUN IN THE META STORE</p> <ul style="list-style-type: none"> see file 05Sept2018_OOZIEWORKFLOW-METASTOREJOBS_SCRIPT1
SAMPLE	<pre>sqoop job --meta-connect jdbc:hsqldb:hsqldb://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</pre>
STEP 2	<p>CREATE DATABASE CALLED “cdw_sapp” using the command below:</p> <p>create database IF NOT EXISTS cdw_sapp;</p>
Step 3	<p>CREATE HIVE QUERY FILE TO CREATE AND INSERT DATA FOR EACH TABLE</p> <ul style="list-style-type: none"> See files named: <ul style="list-style-type: none"> 02Sept2018_BRANCHDATA_HIVEQUERY.sql 02Sept2018_CREDITDATA_HIVEQUERY.sql 02Sept2018_TIMEIDCUSTOMERDATA-HIVESQUERY.sql <p>in 12Sep2018_DE03-NY_CaseStudy_AutomatingtheProccesWithOozie.zip & 12Sep2018_DE03-NY_CaseStudy_AutomatingtheProccesWithOozie_ManualRun.zip file</p>

```
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";
```

**S
A
M
P
L
E**

```
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	<p style="text-align: center;"><u>RUN THE WORKFLOW ONLY</u></p> <ul style="list-style-type: none"> • CREATE workflow 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 <ul style="list-style-type: none"> • oozie job -oozie http://localhost:11000/oozie -config Documents/oozie/cs/workflow/job.properties -run 		
Step 4	<p style="text-align: center;"><u>RUN COORDINATOR</u></p> <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • oozie job -oozie http://localhost:11000/oozie -config Documents/oozie/cs/coordinator/job.properties -run 		

2.2.4	Optimizing the Process MODULE
<p>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.</p> <ul style="list-style-type: none">• 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. <p>2) Modify the Oozie Coordinator to use this workflow rather than the original, unoptimized, workflow</p>	
Step 1	Create Updated SQOOP JOBS IN THE META STORE <ul style="list-style-type: none">• see file 05Sept2018_OOZIEWORKFLOW-METASTOREJOBS_SCRIPT2
SAMPLE	<pre>sqoop job --meta-connect jdbc:hsqldb:hsqldb://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</pre>
STEP 2	CREATE DATABASE CALLED “cdw_sapp” using the command below: create database IF NOT EXISTS cdw_sapp;
STEP 3	CREATE HIVE QUERY FILE TO CREATE AND INSERT DATA FOR EACH TABLE <ul style="list-style-type: none">• See files named:<ul style="list-style-type: none">• 02Sept2018_BRANCHDATA_HIVEQUERY.sql• 02Sept2018_CREDITDATA_HIVEQUERY.sql• 02Sept2018_TIMEIDCUSTOMERDATA-HIVESQUERY.sql <p>in 12Sep2018_DE03-NY_CaseStudy_OptimizingtheProcces_ManualRun.zip & 12Sep2018_DE03-NY_CaseStudy_OptimizingtheProcces.zip file</p>

```
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";
```

**S
A
M
P
L
E**

```
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;
```


2.2.5	Visualization of Dataset MODULE		
1) Use Hive Query and Hive Visualization tool			
<ul style="list-style-type: none">Find the top 20 zip codes(hint: branch_zip) by total transaction valueFind total transaction value for each transaction type by Quarter in 2018			
Step 1	RUN HIVE QUERY TO PULL “TOP 20 ZIPCODES BY TOTAL VALUE) <ul style="list-style-type: none">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		