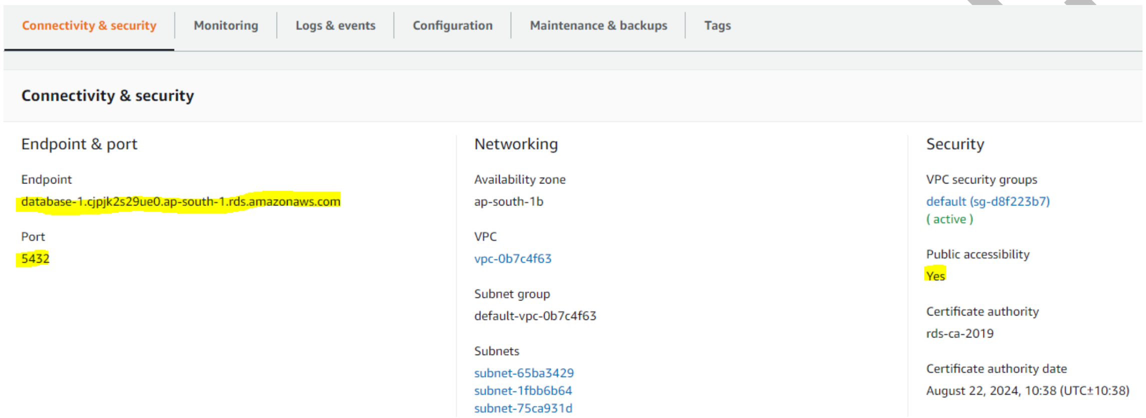
CCDA : Credit Card Data Analysis

1.Database Design

01.ccda_final_design.pdf

2. Create AWS RDS instance of Postgre (PostgreSQL 12.5-R1) type

Note : Make DB public , remember Port,Username,Password and Endpoint



Connectivity & security

Endpoint & port

Endpoint
database-1.cjpjk2s29ue0.ap-south-1.rds.amazonaws.com

Port
5432

Networking

Availability zone
ap-south-1b

VPC
vpc-0b7c4f63

Subnet group
default-vpc-0b7c4f63

Subnets
subnet-65ba3429
subnet-1fbb6b64
subnet-75ca931d

Security

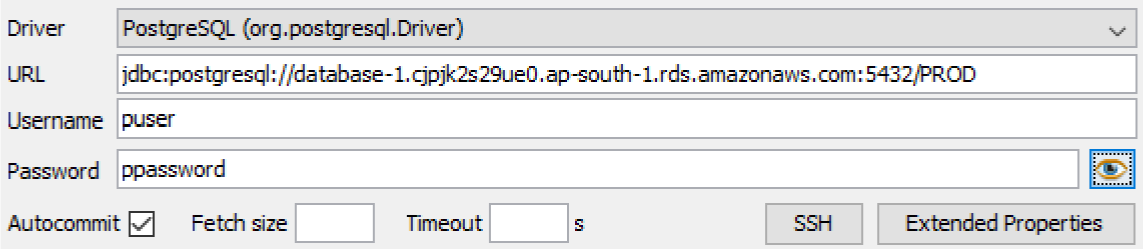
VPC security groups
default (sg-d8f223b7)
(active)

Public accessibility
Yes

Certificate authority
rds-ca-2019

Certificate authority date
August 22, 2024, 10:38 (UTC±10:38)

3. Connect That Created Postgre instance using workbench



Driver: PostgreSQL (org.postgresql.Driver)

URL: jdbc:postgresql://database-1.cjpjk2s29ue0.ap-south-1.rds.amazonaws.com:5432/PROD

Username: puser

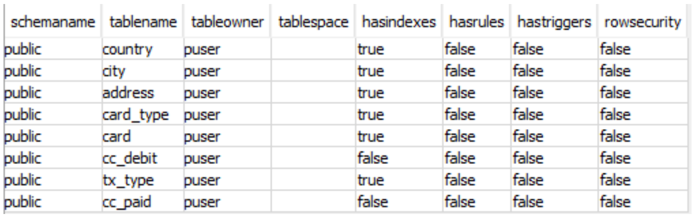
Password: ppassword

Autocommit ☑ Fetch size Timeout s

Connect

4. Load Data to create all tables in Postgre

02.Postgre_DDL.sql



schemaname	tablename	tableowner	tablespace	hasindexes	hasrules	hastriggers	rowsecurity
public	country	puser		true	false	false	false
public	city	puser		true	false	false	false
public	address	puser		true	false	false	false
public	card_type	puser		true	false	false	false
public	card	puser		true	false	false	false
public	cc_debit	puser		false	false	false	false
public	tx_type	puser		true	false	false	false
public	cc_paid	puser		false	false	false	false

5.Initial Data load using

03.Initial_Dataload.sql

Check table structure and sample data

6. Create EMR (5.33) with Below Software's

- Hadoop
- Spark
- Hbase
- Zookeeper
- Hive
- Hue
- Phoenix

7. Copy All dependencies on EMR

/home/hadoop/dep/*

8. Create Hbase table using phoenix Script

04.Pheonix_Hbase_DDL_CCDA.txt

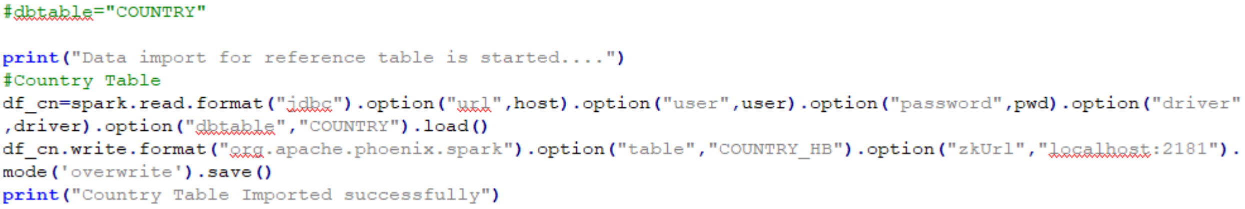
```
CREATE TABLE COUNTRY_HB (cn_id VARCHAR(10) PRIMARY KEY,cn_name VARCHAR(45));
CREATE TABLE CITY_HB (ct_id VARCHAR(10) PRIMARY KEY,ct_name VARCHAR(45),cn_id VARCHAR(10));
CREATE TABLE CARD_TYPE_HB (c_type VARCHAR(10) PRIMARY KEY,max_limit INTEGER);
CREATE TABLE TX_TYPE_HB (tx_type_id VARCHAR(10) PRIMARY KEY,tx_type_desc VARCHAR(45));
```

Note : To run pyspark job on EMR cluster you can use below command :

/usr/bin/spark-submit --jars /home/hadoop/dep/postgresql-42.2.14.jar,/home/hadoop/dep/phoenix-4.14.3-HBase-1.4-client.jar,/home/hadoop/dep/phoenix-spark-4.14.3-HBase-1.4.jar --master yarn --deploy-mode client --driver-memory 3g --executor-memory 2g --num-executors 1 --executor-cores 1 /home/hadoop/ 05.CCDA_RDS_to_HBASE_Ref_import.py

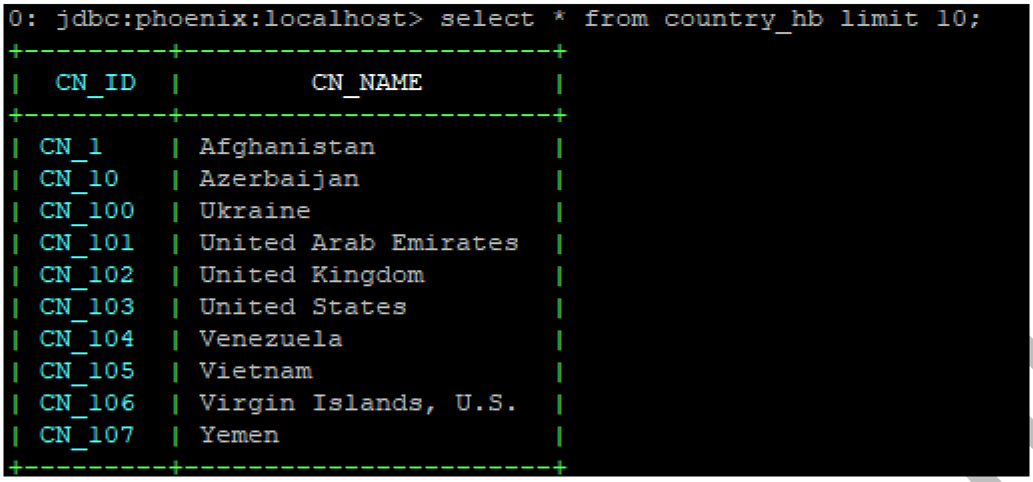
9. Import Ref data from RDBMS to Hbase using (Change the DB details in Script)

05.CCDA_RDS_to_HBASE_Ref_import.py



```
#dbtable="COUNTRY"

print("Data import for reference table is started....")
#Country Table
df_cn=spark.read.format("jdbc").option("url",host).option("user",user).option("password",pwd).option("driver"
,driver).option("dbtable","COUNTRY").load()
df_cn.write.format("org.apache.phoenix.spark").option("table","COUNTRY_HB").option("zkUrl","localhost:2181").
mode('overwrite').save()
print("Country Table Imported successfully")
```

Sample Data :



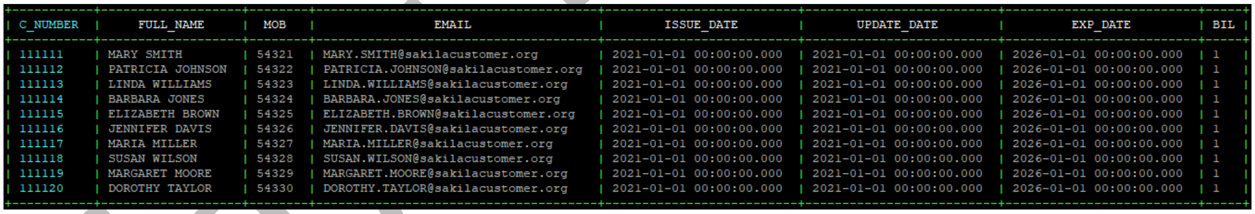
```
0: jdbc:phoenix:localhost> select * from country_hb limit 10;
+----------+------------------------+
|  CN_ID   |        CN_NAME         |
+----------+------------------------+
| CN_1     | Afghanistan            |
| CN_10    | Azerbaijan             |
| CN_100   | Ukraine                |
| CN_101   | United Arab Emirates   |
| CN_102   | United Kingdom         |
| CN_103   | United States          |
| CN_104   | Venezuela              |
| CN_105   | Vietnam                |
| CN_106   | Virgin Islands, U.S.   |
| CN_107   | Yemen                  |
+----------+------------------------+
```

10. Import Trans data from RDBMS to Hbase using (Change the DB details in Script)

06.RDS_to_HBASE_Hist_Trans_import.py

```
print("Historical Data import for transactional table is started....")

#CARD Table
df_sb=spark.read.format("jdbc").option("url",host).option("user",user).option("password",pwd).option("driver"
,driver).option("dbtable","CARD").load()
df_sb.write.format("org.apache.phoenix.spark").option("table","CARD_HB").option("zkUrl","localhost:2181").
mode('overwrite').save()
print("CARD Table Imported successfully")
```

Sample Data :



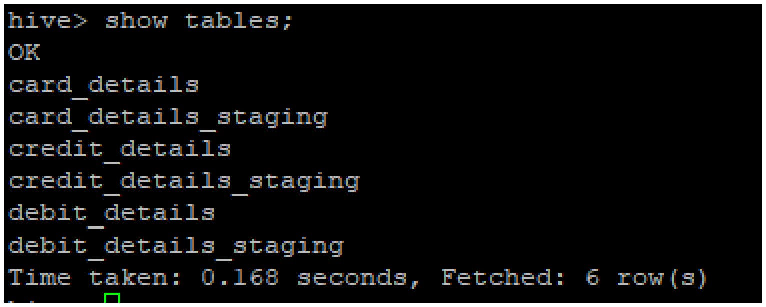
```
+-----------+-------------------+--------+-----------------------------------------+--------------------------+--------------------------+--------------------------+------+
| C_NUMBER  |     FULL_NAME     |  MOB   |                  EMAIL                  |        ISSUE_DATE        |       UPDATE_DATE        |         EXP_DATE         | BIL  |
+-----------+-------------------+--------+-----------------------------------------+--------------------------+--------------------------+--------------------------+------+
| 111111    | MARY SMITH        | 54321  | MARY.SMITH@sakilacustomer.org           | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
| 111112    | PATRICIA JOHNSON  | 54322  | PATRICIA.JOHNSON@sakilacustomer.org     | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
| 111113    | LINDA WILLIAMS    | 54323  | LINDA.WILLIAMS@sakilacustomer.org       | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
| 111114    | BARBARA JONES     | 54324  | BARBARA.JONES@sakilacustomer.org        | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
| 111115    | ELIZABETH BROWN   | 54325  | ELIZABETH.BROWN@sakilacustomer.org      | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
| 111116    | JENNIFER DAVIS    | 54326  | JENNIFER.DAVIS@sakilacustomer.org       | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
| 111117    | MARIA MILLER      | 54327  | MARIA.MILLER@sakilacustomer.org         | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
| 111118    | SUSAN WILSON      | 54328  | SUSAN.WILSON@sakilacustomer.org         | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
| 111119    | MARGARET MOORE    | 54329  | MARGARET.MOORE@sakilacustomer.org       | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
| 111120    | DOROTHY TAYLOR    | 54330  | DOROTHY.TAYLOR@sakilacustomer.org       | 2021-01-01 00:00:00.000  | 2021-01-01 00:00:00.000  | 2026-01-01 00:00:00.000  | 1    |
+-----------+-------------------+--------+-----------------------------------------+--------------------------+--------------------------+--------------------------+------+
```

11. Create Hive tables deployment to transformed data using

07.CCDA_hive_ddl.hql

```
create table card_details_staging (CardNumber integer,CardType string,full_name string,contactnumber
integer,emailId string,address string,city string,country string,issuedate timestamp,update_date
timestamp,BillingDate integer,CardLimit integer,Active_flag Char(1))
Row format delimited
fields terminated by ',';

create table card_details (CardNumber integer,CardType string,full_name string,contactnumber integer,emailId
string,address string,city string,country string,issuedate timestamp,update_date timestamp,BillingDate
integer,CardLimit integer,Active_flag Char(1))
Row format delimited
fields terminated by ',';
```

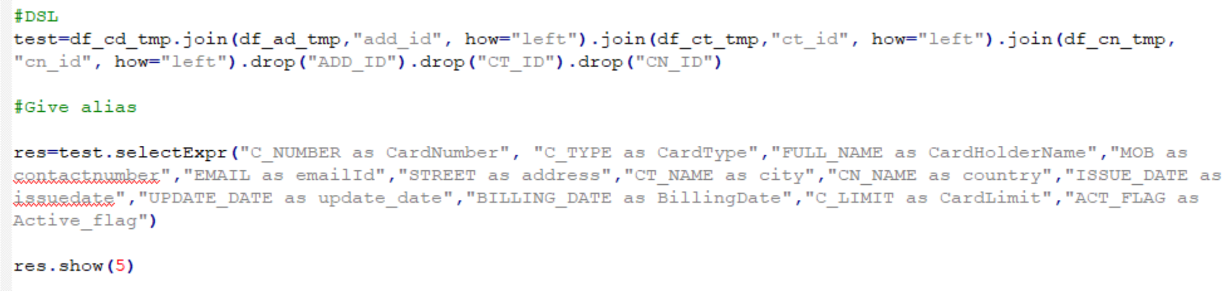
Tables in Hive :



```
hive> show tables;
OK
card_details
card_details_staging
credit_details
credit_details_staging
debit_details
debit_details_staging
Time taken: 0.168 seconds, Fetched: 6 row(s)
```

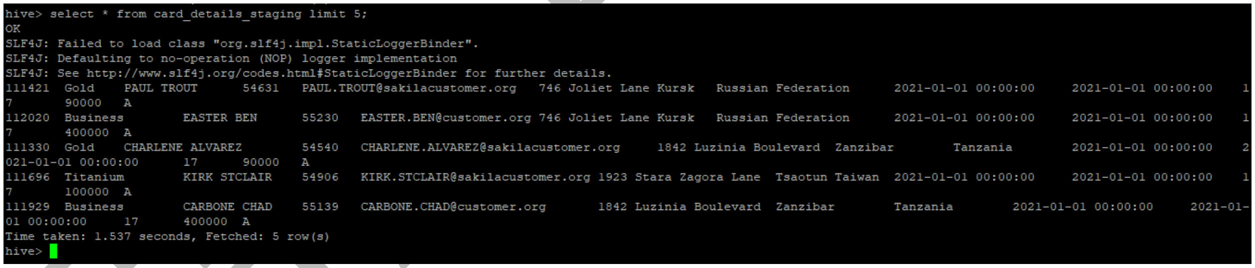
12. To process Card details context using below script

08.CCDA_Hb_to_hive_card_details.py



```
#DSL
test=df_cd_tmp.join(df_ad_tmp,"add_id", how="left").join(df_ct_tmp,"ct_id", how="left").join(df_cn_tmp,
"cn_id", how="left").drop("ADD_ID").drop("CT_ID").drop("CN_ID")

#Give alias

res=test.selectExpr("C_NUMBER as CardNumber", "C_TYPE as CardType","FULL_NAME as CardHolderName","MOB as
contactnumber","EMAIL as emailId","STREET as address","CT_NAME as city","CN_NAME as country","ISSUE_DATE as
issuedate","UPDATE_DATE as update_date","BILLING_DATE as BillingDate","C_LIMIT as CardLimit","ACT_FLAG as
Active_flag")

res.show(5)
```

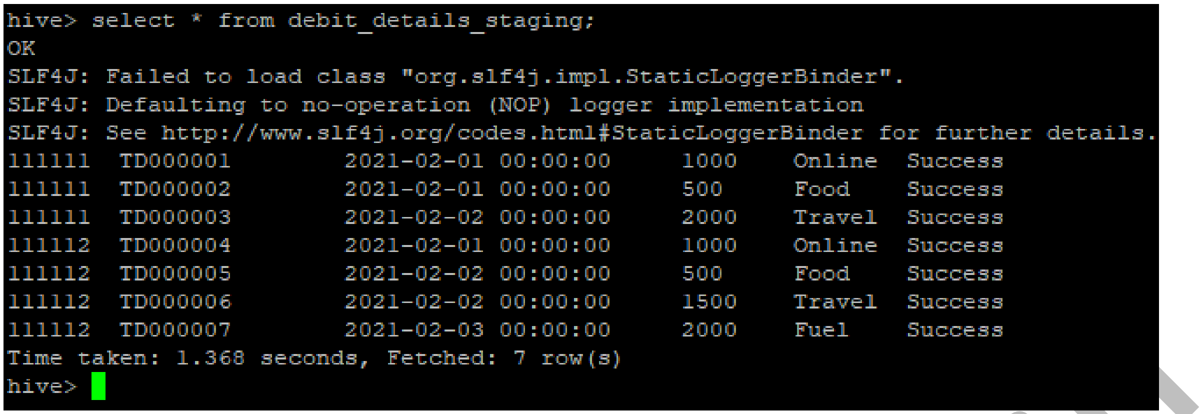
Sample Data :



```
hive> select * from card_details_staging limit 5;
OK
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
111421  Gold    PAUL TROUT      54631   PAUL.TROUT@sakilacustomer.org   746 Joliet Lane Kursk   Russian Federation      2021-01-01 00:00:00     2021-01-01 00:00:00     1
7       90000   A
112020  Business        EASTER BEN      55230   EASTER.BEN@customer.org 746 Joliet Lane Kursk   Russian Federation      2021-01-01 00:00:00     2021-01-01 00:00:00     1
7       400000  A
111330  Gold    CHARLENE ALVAREZ        54540   CHARLENE.ALVAREZ@sakilacustomer.org     1842 Luzinia Boulevard  Zanzibar        Tanzania        2021-01-01 00:00:00     2
021-01-01 00:00:00      17      90000   A
111696  Titanium        KIRK STCLAIR    54906   KIRK.STCLAIR@sakilacustomer.org 1923 Stara Zagora Lane  Tsaotun Taiwan  2021-01-01 00:00:00     2021-01-01 00:00:00     1
7       100000  A
111929  Business        CARBONE CHAD    55139   CARBONE.CHAD@customer.org       1842 Luzinia Boulevard  Zanzibar        Tanzania        2021-01-01 00:00:00     2021-01-
01 00:00:00     17      400000  A
Time taken: 1.537 seconds, Fetched: 5 row(s)
hive>
```

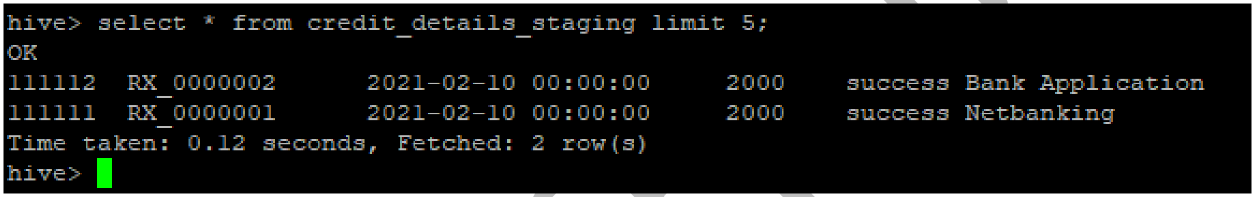
13. To process debit and Credit details context use below script

09.CCDA_Hb_to_hive_debit_details.py



```
hive> select * from debit_details_staging;
OK
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
111111  TD000001        2021-02-01 00:00:00     1000    Online  Success
111111  TD000002        2021-02-01 00:00:00     500     Food    Success
111111  TD000003        2021-02-02 00:00:00     2000    Travel  Success
111112  TD000004        2021-02-01 00:00:00     1000    Online  Success
111112  TD000005        2021-02-02 00:00:00     500     Food    Success
111112  TD000006        2021-02-02 00:00:00     1500    Travel  Success
111112  TD000007        2021-02-03 00:00:00     2000    Fuel    Success
Time taken: 1.368 seconds, Fetched: 7 row(s)
hive>
```

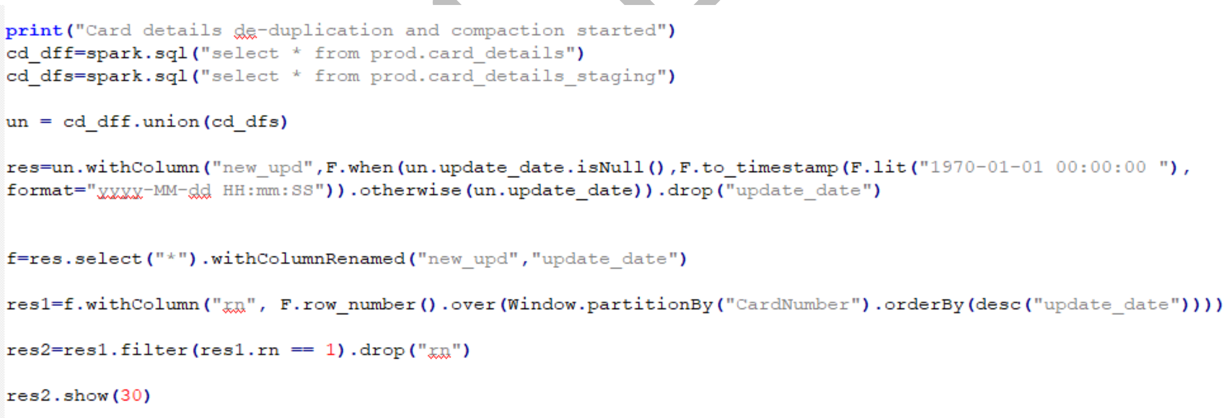
10.CCDA_Hb_to_hive_credit_details.py



```
hive> select * from credit_details_staging limit 5;
OK
111112  RX_0000002      2021-02-10 00:00:00     2000    success Bank Application
111111  RX_0000001      2021-02-10 00:00:00     2000    success Netbanking
Time taken: 0.12 seconds, Fetched: 2 row(s)
hive>
```

14. Remove duplicate data using

10.dedup_Compaction.py



```
print("Card details de-duplication and compaction started")
cd_dff=spark.sql("select * from prod.card_details")
cd_dfs=spark.sql("select * from prod.card_details_staging")

un = cd_dff.union(cd_dfs)

res=un.withColumn("new_upd",F.when(un.update_date.isNull(),F.to_timestamp(F.lit("1970-01-01 00:00:00 "),
format="yyyy-MM-dd HH:mm:SS")).otherwise(un.update_date)).drop("update_date")


f=res.select("*").withColumnRenamed("new_upd","update_date")

res1=f.withColumn("rn", F.row_number().over(Window.partitionBy("CardNumber").orderBy(desc("update_date"))))

res2=res1.filter(res1.rn == 1).drop("rn")

res2.show(30)
```

15. To Process delta use below script

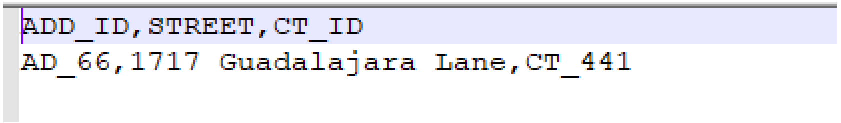
Copy data into Hbase at /data location

Run this script to process new delta

11.Dedup_CCDA.py

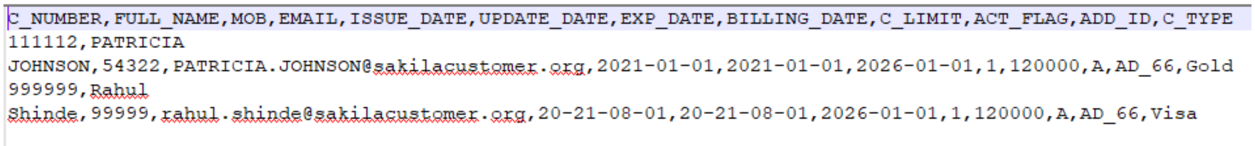
Delta :

Address :



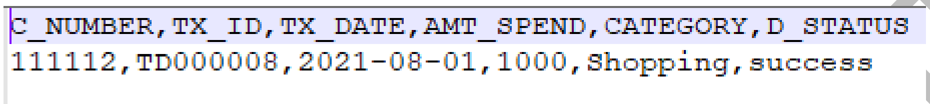
```
ADD_ID,STREET,CT_ID
AD_66,1717 Guadalajara Lane,CT_441
```

Card :



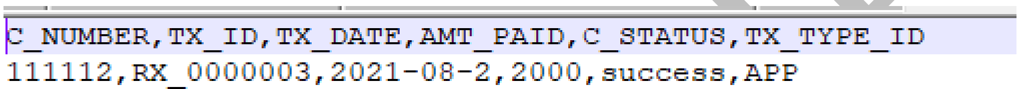
```
C_NUMBER,FULL_NAME,MOB,EMAIL,ISSUE_DATE,UPDATE_DATE,EXP_DATE,BILLING_DATE,C_LIMIT,ACT_FLAG,ADD_ID,C_TYPE
111112,PATRICIA
JOHNSON,54322,PATRICIA.JOHNSON@sakilacustomer.org,2021-01-01,2021-01-01,2026-01-01,1,120000,A,AD_66,Gold
999999,Rahul
Shinde,99999,rahul.shinde@sakilacustomer.org,20-21-08-01,20-21-08-01,2026-01-01,1,120000,A,AD_66,Visa
```

Debit :



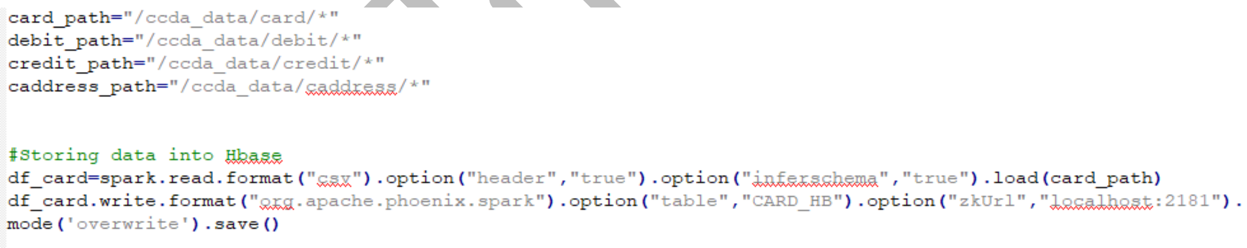
```
C_NUMBER,TX_ID,TX_DATE,AMT_SPEND,CATEGORY,D_STATUS
111112,TD000008,2021-08-01,1000,Shopping,success
```

Credit:

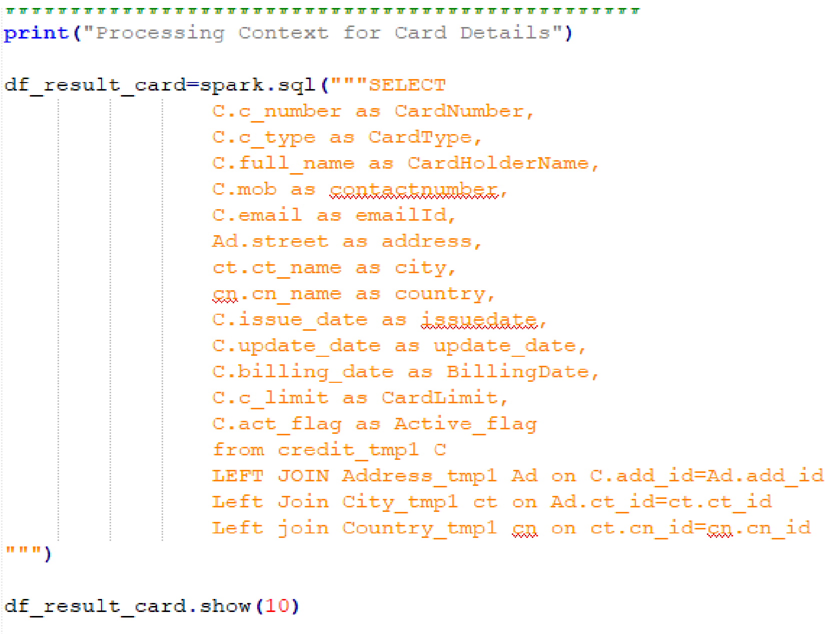


```
C_NUMBER,TX_ID,TX_DATE,AMT_PAID,C_STATUS,TX_TYPE_ID
111112,RX_0000003,2021-08-2,2000,success,APP
```

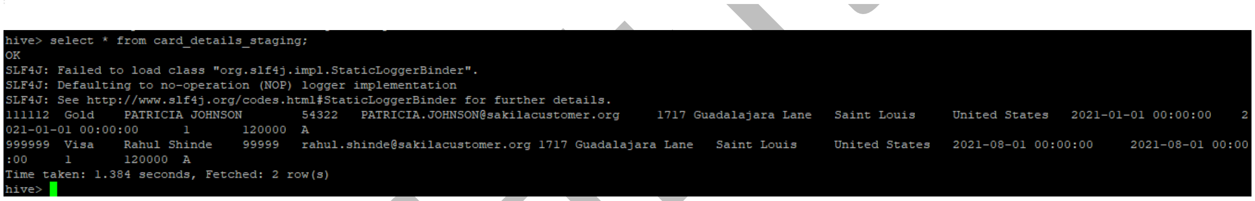
Script to process delta :



```
card_path="/ccda_data/card/*"
debit_path="/ccda_data/debit/*"
credit_path="/ccda_data/credit/*"
caddress_path="/ccda_data/caddress/*"


#Storing data into Hbase
df_card=spark.read.format("csv").option("header","true").option("inferschema","true").load(card_path)
df_card.write.format("org.apache.phoenix.spark").option("table","CARD_HB").option("zkUrl","localhost:2181").
mode('overwrite').save()
```



```
#################################################
print("Processing Context for Card Details")

df_result_card=spark.sql("""SELECT
                C.c_number as CardNumber,
                C.c_type as CardType,
                C.full_name as CardHolderName,
                C.mob as contactnumber,
                C.email as emailId,
                Ad.street as address,
                ct.ct_name as city,
                cn.cn_name as country,
                C.issue_date as issuedate,
                C.update_date as update_date,
                C.billing_date as BillingDate,
                C.c_limit as CardLimit,
                C.act_flag as Active_flag
                from credit_tmp1 C
                LEFT JOIN Address_tmp1 Ad on C.add_id=Ad.add_id
                Left Join City_tmp1 ct on Ad.ct_id=ct.ct_id
                Left join Country_tmp1 cn on ct.cn_id=cn.cn_id
""")

df_result_card.show(10)
```

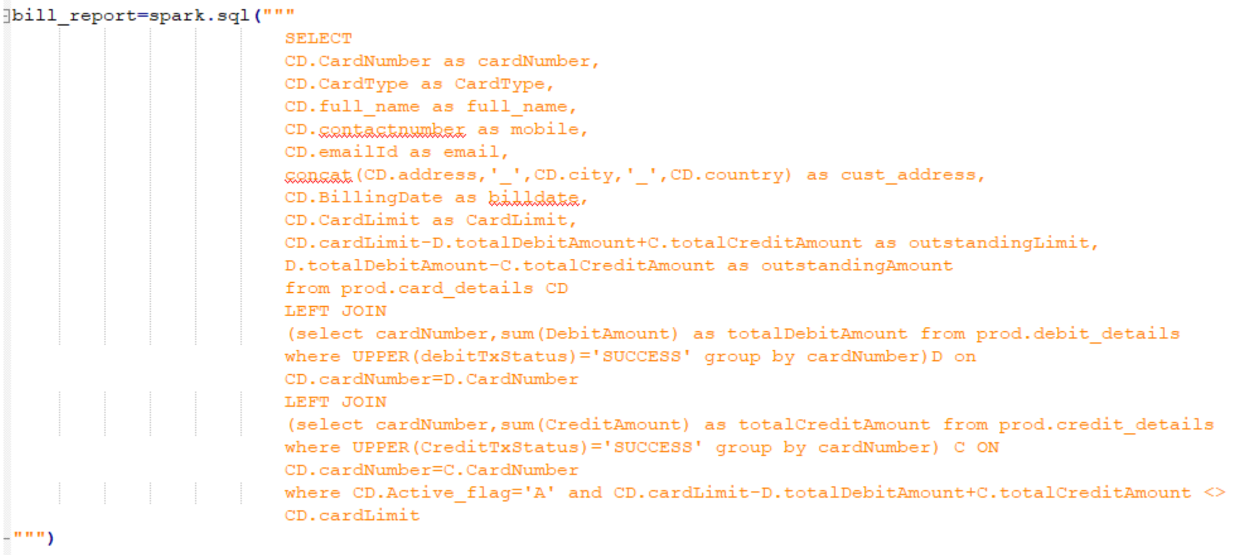


```
hive> select * from card_details_staging;
OK
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
111112  Gold    PATRICIA JOHNSON        54322   PATRICIA.JOHNSON@sakilacustomer.org     1717 Guadalajara Lane   Saint Louis     United States   2021-01-01 00:00:00     2
021-01-01 00:00:00      1       120000  A
999999  Visa    Rahul Shinde    99999   rahul.shinde@sakilacustomer.org 1717 Guadalajara Lane   Saint Louis     United States   2021-08-01 00:00:00     2021-08-01 00:00
:00     1       120000  A
Time taken: 1.384 seconds, Fetched: 2 row(s)
hive>
```

Note : You have successfully Processed Historical and Delta Data

Phase 2 :

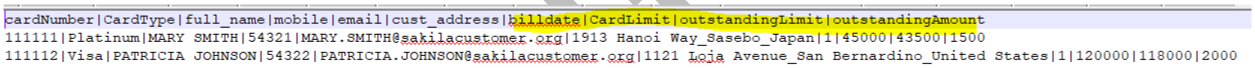
Extraction :



```
bill_report=spark.sql("""
                        SELECT
                        CD.CardNumber as cardNumber,
                        CD.CardType as CardType,
                        CD.full_name as full_name,
                        CD.contactnumber as mobile,
                        CD.emailId as email,
                        concat(CD.address,'_',CD.city,'_',CD.country) as cust_address,
                        CD.BillingDate as billdate,
                        CD.CardLimit as CardLimit,
                        CD.cardLimit-D.totalDebitAmount+C.totalCreditAmount as outstandingLimit,
                        D.totalDebitAmount-C.totalCreditAmount as outstandingAmount
                        from prod.card_details CD
                        LEFT JOIN
                        (select cardNumber,sum(DebitAmount) as totalDebitAmount from prod.debit_details
                        where UPPER(debitTxStatus)='SUCCESS' group by cardNumber)D on
                        CD.cardNumber=D.CardNumber
                        LEFT JOIN
                        (select cardNumber,sum(CreditAmount) as totalCreditAmount from prod.credit_details
                        where UPPER(CreditTxStatus)='SUCCESS' group by cardNumber) C ON
                        CD.cardNumber=C.CardNumber
                        where CD.Active_flag='A' and CD.cardLimit-D.totalDebitAmount+C.totalCreditAmount <>
                        CD.cardLimit
""")
```

Sample Data :

Now customer can extract report from S3 and can use for their further use like visualization etc.

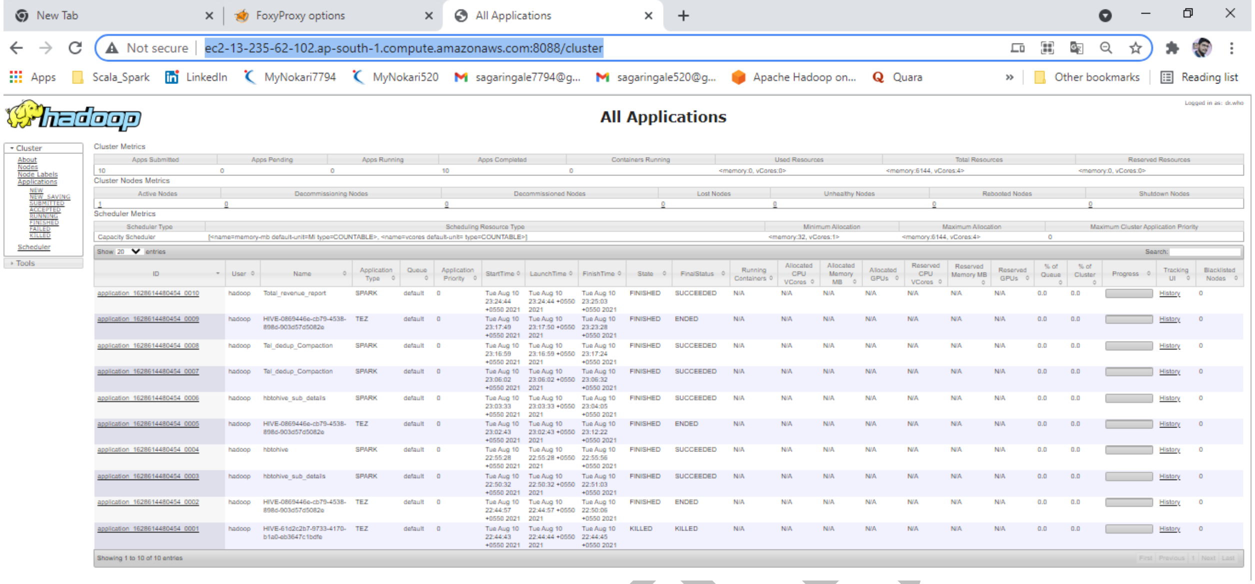


```
cardNumber|CardType|full_name|mobile|email|cust_address|billdate|CardLimit|outstandingLimit|outstandingAmount
111111|Platinum|MARY SMITH|54321|MARY.SMITH@sakilacustomer.org|1913 Hanoi Way_Sasebo_Japan|1|45000|43500|1500
111112|Visa|PATRICIA JOHNSON|54322|PATRICIA.JOHNSON@sakilacustomer.org|1121 Loja Avenue_San Bernardino_United States|1|120000|118000|2000
```

Job Monitoring on Resource Manager :

http://ec2-13-235-62-102.ap-south-1.compute.amazonaws.com:8088/cluster

All Applications :



ec2-13-235-62-102.ap-south-1.compute.amazonaws.com:8088/cluster

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources
10	0	0	10	0	<memory:0, vCores:0>	<memory:6144, vCores:4>	<memory:0, vCores:0>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

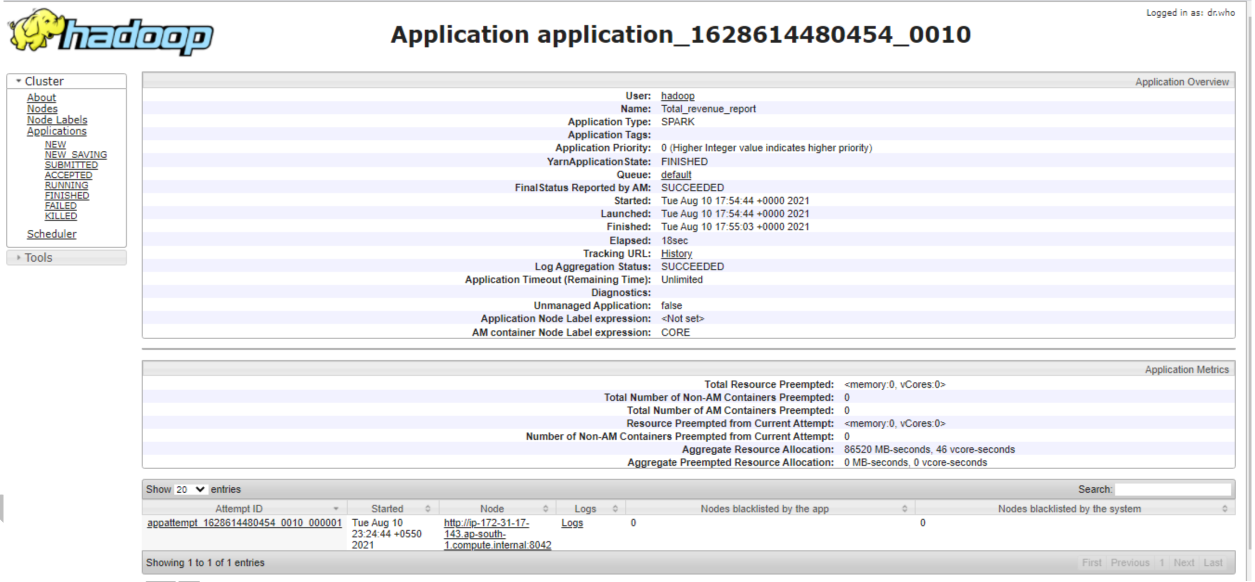
Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[<name=memory-mb default-unit=Mi type=COUNTABLE>, <name=vcores default-unit= type=COUNTABLE>]	<memory:32, vCores:1>	<memory:6144, vCores:4>	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1628614480454_0010	hadoop	Total_revenue_report	SPARK	default	0	Tue Aug 10 23:24:44 +0550 2021	Tue Aug 10 23:24:44 +0550 2021	Tue Aug 10 23:25:03 +0550 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1628614480454_0009	hadoop	HIVE-0869446e-cb79-4538-898d-903d57d5082e	TEZ	default	0	Tue Aug 10 23:17:49 +0550 2021	Tue Aug 10 23:17:50 +0550 2021	Tue Aug 10 23:23:28 +0550 2021	FINISHED	ENDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1628614480454_0008	hadoop	Tel_dedup_Compaction	SPARK	default	0	Tue Aug 10 23:16:59 +0550 2021	Tue Aug 10 23:16:59 +0550 2021	Tue Aug 10 23:17:24 +0550 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1628614480454_0007	hadoop	Tel_dedup_Compaction	SPARK	default	0	Tue Aug 10 23:06:02 +0550 2021	Tue Aug 10 23:06:02 +0550 2021	Tue Aug 10 23:06:32 +0550 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1628614480454_0006	hadoop	hbtohive_sub_details	SPARK	default	0	Tue Aug 10 23:03:33 +0550 2021	Tue Aug 10 23:03:33 +0550 2021	Tue Aug 10 23:04:05 +0550 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1628614480454_0005	hadoop	HIVE-0869446e-cb79-4538-898d-903d57d5082e	TEZ	default	0	Tue Aug 10 23:02:43 +0550 2021	Tue Aug 10 23:02:43 +0550 2021	Tue Aug 10 23:12:22 +0550 2021	FINISHED	ENDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1628614480454_0004	hadoop	hbtohive	SPARK	default	0	Tue Aug 10 22:55:28 +0550 2021	Tue Aug 10 22:55:28 +0550 2021	Tue Aug 10 22:55:56 +0550 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1628614480454_0003	hadoop	hbtohive_sub_details	SPARK	default	0	Tue Aug 10 22:50:32 +0550 2021	Tue Aug 10 22:50:32 +0550 2021	Tue Aug 10 22:51:03 +0550 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1628614480454_0002	hadoop	HIVE-0869446e-cb79-4538-898d-903d57d5082e	TEZ	default	0	Tue Aug 10 22:44:57 +0550 2021	Tue Aug 10 22:44:57 +0550 2021	Tue Aug 10 22:50:06 +0550 2021	FINISHED	ENDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1628614480454_0001	hadoop	HIVE-61d2c2b7-9733-4170-b1a0-eb3647c1bdfe	TEZ	default	0	Tue Aug 10 22:44:43 +0550 2021	Tue Aug 10 22:44:44 +0550 2021	Tue Aug 10 22:44:45 +0550 2021	KILLED	KILLED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0

Showing 1 to 10 of 10 entries

Application Logs :



Application application_1628614480454_0010

Application Overview

User:	hadoop
Name:	Total_revenue_report
Application Type:	SPARK
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Tue Aug 10 17:54:44 +0000 2021
Launched:	Tue Aug 10 17:54:44 +0000 2021
Finished:	Tue Aug 10 17:55:03 +0000 2021
Elapsed:	18sec
Tracking URL:	History
Log Aggregation Status:	SUCCEEDED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	CORE

Application Metrics

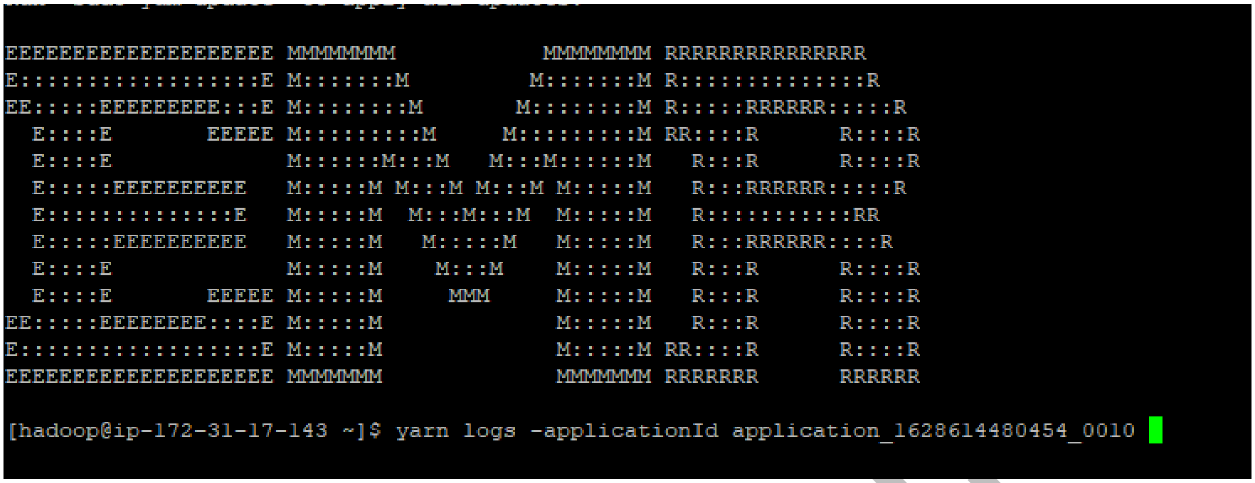
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	86520 MB-seconds, 46 vcore-seconds
Aggregate Preempted Resource Allocation:	0 MB-seconds, 0 vcore-seconds

Show 20 entries Search:

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1628614480454_0010_000001	Tue Aug 10 23:24:44 +0550 2021	http://ip-172-31-17-143.ap-south-1.compute.internal:8042	Logs	0	0

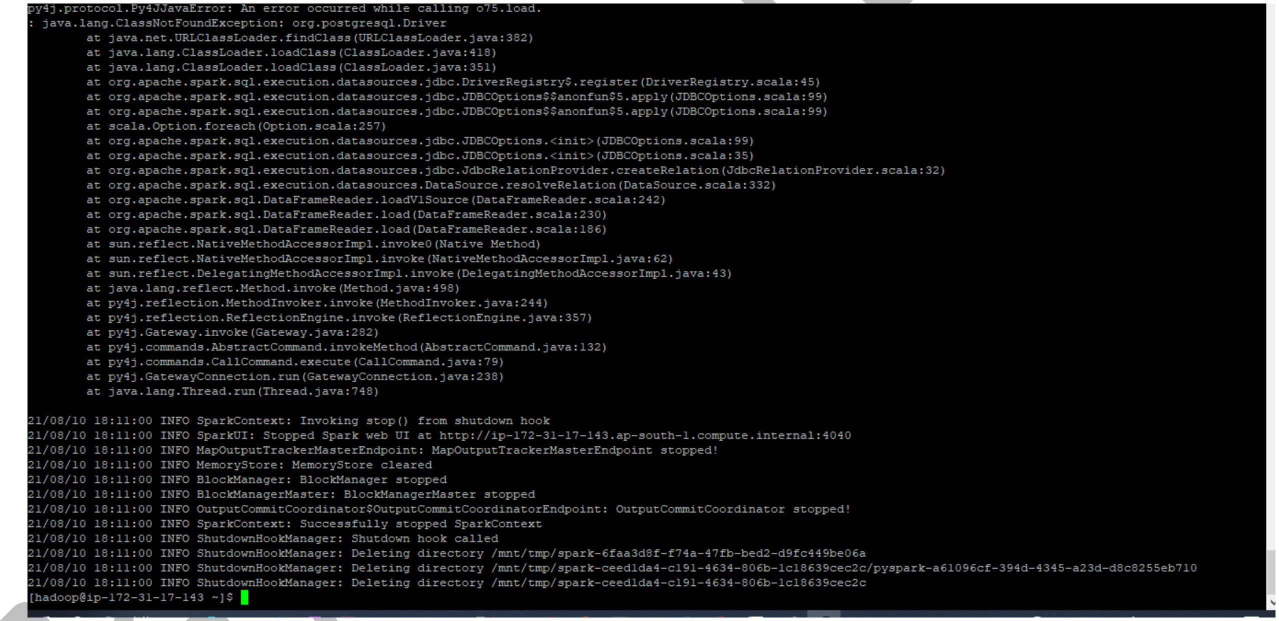
Showing 1 to 1 of 1 entries

Check Log on Terminal :



```
EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-17-143 ~]$ yarn logs -applicationId application_1628614480454_0010
```

Error Occurred :



```
py4j.protocol.Py4JJavaError: An error occurred while calling o75.load.
: java.lang.ClassNotFoundException: org.postgresql.Driver
        at java.net.URLClassLoader.findClass(URLClassLoader.java:382)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
        at org.apache.spark.sql.execution.datasources.jdbc.DriverRegistry$.register(DriverRegistry.scala:45)
        at org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions$$anonfun$5.apply(JDBCOptions.scala:99)
        at org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions$$anonfun$5.apply(JDBCOptions.scala:99)
        at scala.Option.foreach(Option.scala:257)
        at org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions.<init>(JDBCOptions.scala:99)
        at org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions.<init>(JDBCOptions.scala:35)
        at org.apache.spark.sql.execution.datasources.jdbc.JdbcRelationProvider.createRelation(JdbcRelationProvider.scala:32)
        at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:332)
        at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:242)
        at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:230)
        at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:186)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
        at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
        at py4j.Gateway.invoke(Gateway.java:282)
        at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
        at py4j.commands.CallCommand.execute(CallCommand.java:79)
        at py4j.GatewayConnection.run(GatewayConnection.java:238)
        at java.lang.Thread.run(Thread.java:748)

21/08/10 18:11:00 INFO SparkContext: Invoking stop() from shutdown hook
21/08/10 18:11:00 INFO SparkUI: Stopped Spark web UI at http://ip-172-31-17-143.ap-south-1.compute.internal:4040
21/08/10 18:11:00 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/08/10 18:11:00 INFO MemoryStore: MemoryStore cleared
21/08/10 18:11:00 INFO BlockManager: BlockManager stopped
21/08/10 18:11:00 INFO BlockManagerMaster: BlockManagerMaster stopped
21/08/10 18:11:00 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/08/10 18:11:00 INFO SparkContext: Successfully stopped SparkContext
21/08/10 18:11:00 INFO ShutdownHookManager: Shutdown hook called
21/08/10 18:11:00 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-6faa3d8f-f74a-47fb-bed2-d9fc449be06a
21/08/10 18:11:00 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-ceed1da4-c191-4634-806b-1c18639cec2c/pyspark-a61096cf-394d-4345-a23d-d8c8255eb710
21/08/10 18:11:00 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-ceed1da4-c191-4634-806b-1c18639cec2c
[hadoop@ip-172-31-17-143 ~]$
```

UI Port number :

https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-web-interfaces.html

Name of interface	URI
Flink history server (EMR version 5.33 and later)	http://*master-public-dns-name*:8082/
Ganglia	http://*master-public-dns-name*/ganglia/

Name of interface	URI
Hadoop HDFS NameNode (EMR version pre-6.x)	https://*master-public-dns-name*:50470/
Hadoop HDFS NameNode	http://*master-public-dns-name*:50070/
Hadoop HDFS DataNode	http://*coretask-public-dns-name*:50075/
Hadoop HDFS NameNode (EMR version 6.x)	https://*master-public-dns-name*:9871/
Hadoop HDFS DataNode (EMR version pre-6.x)	https://*coretask-public-dns-name*:50475/
Hadoop HDFS DataNode (EMR version 6.x)	https://*coretask-public-dns-name*:9865/
HBase	http://*master-public-dns-name*:16010/
Hue	http://*master-public-dns-name*:8888/
JupyterHub	https://*master-public-dns-name*:9443/
Livy	http://*master-public-dns-name*:8998/
Spark HistoryServer	http://*master-public-dns-name*:18080/
Tez	http://*master-public-dns-name*:8080/tez-ui
YARN NodeManager	http://*coretask-public-dns-name*:8042/
YARN ResourceManager	http://*master-public-dns-name*:8088/
Zeppelin	http://*master-public-dns-name*:8890/