

# EDWA : Enterprise Data warehouse Analysis

## 1.Database Design



01.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

The screenshot shows the AWS RDS instance configuration page. The 'Connectivity & security' tab is active. It displays the following information:

- Endpoint & port:** Endpoint is `database-1.cjpk2s29ue0.ap-south-1.rds.amazonaws.com` and Port is `5432`.
- Networking:** Availability zone is `ap-south-1b`, VPC is `vpc-0b7c4f63`, Subnet group is `default-vpc-0b7c4f63`, and Subnets are `subnet-65ba3429`, `subnet-1fbb6b64`, and `subnet-75ca931d`.
- Security:** VPC security groups is `default (sg-d8f223b7) (active)`, Public accessibility is `Yes`, Certificate authority is `rds-ca-2019`, and Certificate authority date is `August 22, 2024, 10:38 (UTC+10:38)`.

## 3. Connect That Created Postgre instance using workbench

The screenshot shows the DBeaver connection configuration dialog for PostgreSQL. The fields are filled as follows:

- Driver:** PostgreSQL (org.postgresql.Driver)
- URL:** `jdbc:postgresql://database-1.cjpk2s29ue0.ap-south-1.rds.amazonaws.com:5432/PROD`
- Username:** `puser`
- Password:** `ppassword`
- Autocommit:** ☒
- Fetch size:**
- Timeout:**  s
- SSH:** ☐
- Extended Properties:**

Connect

## 4. Load Data to create all tables in Postgre

02.Postgre\_DDL.sql

```
1 SELECT * FROM pg_catalog.pg_tables where schemaname='public';
```

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	plan_prepaid	puser		true	false	false	false
public	plan_postpaid	puser		true	false	false	false
public	staff	puser		true	false	false	false
public	complaint	puser		true	false	false	false
public	subscriber	puser		true	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.Phoenix\_DDL\_Hbase\_Table\_Cre.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 ADDRESS_HB (add_id VARCHAR(10) PRIMARY KEY,Street VARCHAR(45),ct_id VARCHAR(10));
```

**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.RDS_to_HBASE_Ref_import.py
```

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

05.RDS\_to\_HBASE\_Ref\_import.py

```

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("RDStoHBASE_Ref_Table").master("local").getOrCreate()
host="jdbc:postgresql://database-1.cjpk2s29ue0.ap-south-1.rds.amazonaws.com:5432/PROD"
user="user"
pwd="password"
driver="org.postgresql.Driver"
#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 :

CN_ID	CN_NAME
CN_90	Sweden
CN_91	Switzerland
CN_92	Taiwan
CN_93	Tanzania
CN_94	Thailand
CN_95	Tonga
CN_96	Tunisia
CN_97	Turkey
CN_98	Turkmenistan
CN_99	Tuvalu

109 rows selected (0.042 seconds)

0: jdbc:phoenix:localhost> █

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

06.RDS\_to\_HBASE\_Hist\_Trans\_import.py

```

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("RDStoHDFS_Trans_Table").master("local").getOrCreate()
host="jdbc:postgresql://database-1.cjpk2s29ue0.ap-south-1.rds.amazonaws.com:5432/PROD"
user="user"
pwd="password"
driver="org.postgresql.Driver"
dbtable="COUNTRY"

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

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

```

Sample Data :

```
0: jdbc:phoenix:localhost> select * from staff_hb;
```

STAFF_ID	NAME	MOB	EMAIL	SYS_CRE_DATE	SYS_UPD_DATE	ACTIVE_FLAG	ADD_ID
10001	A	12345	A@A.com	2021-01-01 00:00:00.000	2021-01-01 00:00:00.000	A	AD_101
10002	B	12346	B@B.com	2021-01-01 00:00:00.000	2021-01-01 00:00:00.000	A	AD_102
10003	C	12347	C@C.com	2021-01-01 00:00:00.000	2021-01-01 00:00:00.000	A	AD_103

## 11. Create Hive tables deployment to transformed data using

07.hive\_ddl.hql

```
create database prod;

use prod;

create table subscriber_details_staging (subscriberId integer,subscribername string,contactnumber
integer,emailId string,address string,city string,country string,create_date timestamp,update_date
timestamp,Active_flag char(1),prepaid_desc string,pre_amount integer,postpaid_desc string,pos_amount integer)
Row format delimited
fields terminated by ',';

create table subscriber_details (subscriberId integer,subscribername string,contactnumber integer,emailId
string,address string,city string,country string,create_date timestamp,update_date timestamp,Active_flag
char(1),prepaid_desc string,pre_amount integer,postpaid_desc string,pos_amount integer)
Row format delimited
fields terminated by ',';
```

Tables in Hive :

```
hive> show tables;
OK
complaint_details
complaint_details_staging
subscriber_details
subscriber_details_staging
Time taken: 0.026 seconds, Fetched: 4 row(s)
```

## 12. To process Subscriber details context using below script

08.Hb\_to\_hive\_sub\_details.py

```

#DSL
test=df_sb.join(df_ad,"add_id", how="left").join(df_ct,"ct_id", how="left").join(df_cn, "cn_id", how="left")
test1=test.join(df_ppr, df_ppr.PLAN_ID == test.PREPAID_PLAN_ID, how= "left").drop("ADD_ID").drop("CT_ID").
drop("CN_ID").drop("PLAN_ID").withColumnRenamed("PLAN_DESC","PRE_PLAN_DESC").withColumnRenamed("AMOUNT",
"PRE_AMOUNT")

test2=test1.join(df_ppo, df_ppo.PLAN_ID == test1.POSTPAID_PLAN_ID, how= "left").drop("PLAN_ID").
withColumnRenamed("PLAN_DESC","POS_PLAN_DESC").withColumnRenamed("AMOUNT","POS_AMOUNT")

#Give alias

res=test2.selectExpr("SID as subscriberId", "NAME as subscribername", "MOB as contactnumber", "EMAIL as
emailId", "STREET as address", "CT_NAME as city", "CN_NAME as country", "SYS_CRE_DATE as create_date",
"SYS_UPD_DATE as update_date", "ACTIVE_FLAG as Active_flag", "PRE_PLAN_DESC as prepaid_desc", "PRE_AMOUNT as
pre_amount", "POS_PLAN_DESC as postpaid_desc", "POS_AMOUNT as pos_amount")

res.show(5)

#Write Data in Hive
res.write.mode('overwrite').saveAsTable("prod.subscriber_details_staging")

print("subscriber details data written successfully in hive staging")

```

### Sample Data :

```

hive> select * from subscriber_details_staging limit 5;
OK
311    PAUL TROUT      54631    PAUL.TROUT@sakilacustomer.org    746 Joliet Lane Kursk    Russian Federation    2021-01-01
ULL
910    EASTER BEN      55230    EASTER.BEN@customer.org    746 Joliet Lane Kursk    Russian Federation    2021-01-01 00:00:0
49     NULL      NULL
220    CHARLENE ALVAREZ      54540    CHARLENE.ALVAREZ@sakilacustomer.org    1842 Luzinia Boulevard    Zanzibar    Ta
ULL    NULL    unlimited call    150 GB    1000 SMS    499
586    KIRK STCLAIR      54906    KIRK.STCLAIR@sakilacustomer.org    1923 Stara Zagora Lane    Tsaotun Taiwan    2021-01-01 00:00:0
49     NULL      NULL
819    CARBONE CHAD      55139    CARBONE.CHAD@customer.org    1842 Luzinia Boulevard    Zanzibar    Tanzania    20
nlimited call 100GB    399
Time taken: 0.122 seconds, Fetched: 5 row(s)

```

13. To process Complaint details context use below script

09.Hb\_to\_hive\_cmp\_details.py

14. Remove duplicate data using

10.dedup\_Compaction.py

```
#####Dedup Suncruber details#####

print("Suncruber details de-duplication and compaction started")
sd_dff=spark.sql("select * from prod.subscriber_details")
sd_dfs=spark.sql("select * from prod.subscriber_details_staging")

un = sd_dff.union(sd_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("subscriberId").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.delta\_processing\_sb\_cmp.py

Delta :

Address :

```
ADD_ID,STREET,CT_ID
AD_139,SB road Parbhani,CT_398
AD_1111,ABC chowk,CT_417
AD_1112,Shivaji nagar,CT_417
AD_1113,Kharadi,CT_417
```

Complaint :

```
CMP_ID,SID,REGARDING,DESCR,STATUS,STAFF_ID,SYS_CRE_DATE,SYS_UPD_DATE
11118,47,Recharge,Recharge Related Query,Closed,10004,2021-01-01,2021-01-01
11119,21,Recharge,Recharge Related Query,Closed,10005,2021-01-01,2021-01-01
11120,300,Balance,Balance related Query,Closed,10005,2021-01-01,2021-01-01
222222,111,Billing,Billing related query,Open,10001,2021-06-01,2021-06-01
222222,112,Billing,Billing related query,Open,10001,2021-06-01,2021-06-01
222222,112,Port,I want to port,Open,10001,2021-06-01,2021-06-01
```

Staff :

```
STAFF_ID,NAME,MOB,EMAIL,SYS_CRE_DATE,SYS_UPD_DATE,ACTIVE_FLAG,ADD_ID,PREPAID_PLAN_ID,POSTPAID_PLAN_ID
10001,A,12345,A@A.com,2021-01-01,2021-01-01,A,AD_101,PR_1,PO_1
```

Subscriber:

SID	NAME	MOB	EMAIL	ADD_ID	SYS_CRE_DATE	SYS_UPD_DATE	ACTIVE_FLAG	PREPAID_PLAN_ID	POSTPAID_PLAN_ID
135	JUANITA MASON	54455	JUANITA.MASON@sakilacustomer.org	AD_139	2021-01-01	2021-06-01	A	null	PO_3
734	MCWHORTER RAYMOND	55054	MCWHORTER.RAYMOND@customer.org	AD_139	2021-01-01	2021-06-01	A	null	PO_2
1111	Rahul Gupta	11111	rahul.gupta@customer.org	AD_1111	2021-06-01	2021-06-01	A	null	PO_5
1112	Rahul Sharma	11111	rahul.sharma@customer.org	AD_1112	2021-06-01	2021-06-01	A	null	PO_5
1113	Rahul Verma	11111	rahul.verma@customer.org	AD_1113	2021-06-01	2021-06-01	A	null	PO_5

Script to process delta :

```
sub_path="/data/subscriber/*"
add_path="/data/address/*"
cmp_path="/data/complaint/*"
staff_path="/data/staff/*"

#Storing data into Hbase
df_sb=spark.read.format("cav").option("header","true").option("inferSchema","true").load(sub_path)
df_sb.write.format("org.apache.phoenix.spark").option("table","SUBSCRIBER_HB").option("zookeeper",
"localhost:2181").mode('overwrite').save()

df_result_sd=spark.sql("""SELECT
    S.sid as subscriberId,
    S.name as subscribername,
    S.mob as contactnumber,
    S.email as emailId,
    Ad.street as address,
    ct.ct_name as city,
    cn.cn_name as country,
    S.sys_cre_date as create_date,
    S.sys_upd_date as update_date,
    S.Active_flag as Active_flag,
    ppr.plan_desc as prepaid_desc,
    ppr.amount as pre_amount,
    ppo.plan_desc as postpaid_desc,
    ppo.amount as pos_amount
from Subscriber S
LEFT JOIN Address Ad on S.add_id=Ad.add_id
Left Join City ct on Ad.ct_id=ct.ct_id
Left join Country cn on ct.cn_id=cn.cn_id
Left join Plan_Prepaid ppr on S.prepaid_plan_id=ppr.plan_id
Left join Plan_Postpaid ppo on S.postpaid_plan_id=ppo.plan_id
""")

df_result_sd.show(10)

df_result_sd.write.mode('overwrite').saveAsTable("prod.subscriber_details_staging")
```

Note : You have successfully Processed Historical and Delta Data

Phase 2 :

Extraction :

```

revenue_report=spark.sql("""
    SELECT
    SD.country as Country,
    count(SD.subscriberId) as Total_Subscriber,
    sum(coalesce(pre_amount,pos_amount,0)) as total_revenue
    FROM prod.subscriber_details SD
    group by SD.country,Active_flag having SD.Active_flag='A'
    """)

revenue_report.show()

revenue_report.coalesce(1).write.mode("append").format("CSV").option("header","true").option("delimiter","|").
save("s3://sagarinfo/revenue_report_"+currentdate)

print("Subscriber revenue extraction job finished successfully and data stored in s3")

```

Sample Data :

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

Country	Total_Subscriber	total_revenue
Israel	7	3643
South Africa	18	4189
Cameroon	4	1097
Brazil	51	17206
Tuvalu	2	898
Finland	2	1248
Czech Republic	2	199
Ukraine	10	4041
Canada	6	2394
Liechtenstein	2	898
Senegal	1	0
Vietnam	8	3192
South Korea	9	4141
Saint Vincent and the Grenadines	2	498
Russian Federation	49	18208
Kenya	4	897
American Samoa	2	698
Germany	13	5439
Spain	8	4092
Iraq	1	399
Chad	1	299

Job Monitoring on Resource Manager :

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



## All Applications :

New Tab x FoxyProxy options x All Applications x +

Not secure | ec2-13-235-62-102.ap-south-1.compute.amazonaws.com:8088/cluster

Apps | Scale\_Spark | LinkedIn | MyNokari7794 | MyNokari520 | sagaringale7794@g... | sagaringale520@g... | Apache Hadoop on... | Quara | Other bookmarks | Reading list

### 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
4	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	<memory:mb default:unim type:COUNTABLE>, <memory:vcores default:unim type:COUNTABLE>	<memory:32, vCores:1>	<memory:6144, vCores:4>	0

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinalTime	State	FinalStatus	Running Containers	Allocated CPU vCores	Allocated Memory MB	Allocated GPUs	Reserved CPU vCores	Reserved Memory MB	Reserved GPUs	% of Cluster	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1628614480454_0010	hadoop	Total_revenue_report	SPARK	default	0	Tue Aug 10 23:24:44 +0550	Tue Aug 10 23:24:44 +0550	Tue Aug 10 23:25:03 +0550	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0
application_1628614480454_0009	hadoop	TEZ	TEZ	default	0	Tue Aug 10 23:17:48 +0550	Tue Aug 10 23:17:48 +0550	Tue Aug 10 23:22:28 +0550	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0
application_1628614480454_0008	hadoop	Tel_debut_Compaction	SPARK	default	0	Tue Aug 10 23:16:59 +0550	Tue Aug 10 23:16:59 +0550	Tue Aug 10 23:17:24 +0550	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0
application_1628614480454_0007	hadoop	Tel_debut_Compaction	SPARK	default	0	Tue Aug 10 23:06:02 +0550	Tue Aug 10 23:06:02 +0550	Tue Aug 10 23:06:32 +0550	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0
application_1628614480454_0006	hadoop	hdfsview_sub_details	SPARK	default	0	Tue Aug 10 23:03:33 +0550	Tue Aug 10 23:03:33 +0550	Tue Aug 10 23:05:01 +0550	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0
application_1628614480454_0005	hadoop	HIVE-080448a-d579-453b-886a-003037c5052a	TEZ	default	0	Tue Aug 10 23:02:43 +0550	Tue Aug 10 23:02:43 +0550	Tue Aug 10 23:12:22 +0550	FINISHED	ENDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0
application_1628614480454_0004	hadoop	hdfsview	SPARK	default	0	Tue Aug 10 22:59:28 +0550	Tue Aug 10 22:59:28 +0550	Tue Aug 10 22:59:58 +0550	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0
application_1628614480454_0003	hadoop	hdfsview_sub_details	SPARK	default	0	Tue Aug 10 22:58:32 +0550	Tue Aug 10 22:58:32 +0550	Tue Aug 10 22:59:03 +0550	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0
application_1628614480454_0002	hadoop	HIVE-080448a-d579-453b-886a-003037c5052a	TEZ	default	0	Tue Aug 10 22:44:57 +0550	Tue Aug 10 22:44:57 +0550	Tue Aug 10 22:50:08 +0550	FINISHED	ENDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0
application_1628614480454_0001	hadoop	HIVE-616212b7-0733-41f0-b1d0-4b384711dd9	TEZ	default	0	Tue Aug 10 22:44:43 +0550	Tue Aug 10 22:44:43 +0550	Tue Aug 10 22:44:45 +0550	KILLED	KILLED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	100%	History	0

Showing 1 to 10 of 10 entries

## Application Logs :

### Application application\_1628614480454\_0010

Cluster: NEW, NEW SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED

User: hadoop

Name: Total\_revenue\_report

Application Type: SPARK

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: 10sec

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

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
attempt_1628614480454_0010_000001	Tue Aug 10 23:24:44 +0550 2021	http://ec2-172-31-17-143.ap-south-1.compute.internal:8042	Logs	0	0

Showing 1 to 1 of 1 entries

## Check Log on Terminal :

```

EEEEEEEEEEEEEEEEEEEE MMMMMMM      MMMMMMM RRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M      M::::::::M R::::::::::::R
EE::::::::EEEEEEEEEE::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 M::::M M::::M R::::R      R::::R
  E::::EEEEEEEEEE M::::M M::::M M::::M M::::M R::::RRRRRR::::R
  E::::EEEEEEEEEE M::::M M::::M M::::M M::::M R::::RRRRRR::::R
  E::::E      M::::M M::::M M::::M M::::M R::::R      R::::R
  E::::E      EEEEE M::::M      MMM M::::M R::::R      R::::R
EE::::::::EEEEEEEEEE::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$$anonfun$$apply(JDBCOptions.scala:99)
    at org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions$$anonfun$$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:749)

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$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/08/10 18:11:00 INFO SparkContext: Successfully stopped SparkContext
21/08/10 18:11:00 INFO ShutdownHookManager: Shutdown hook called
21/08/10 18:11:00 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-6faa3d5f-f74a-47fb-bed2-d9fc449be06a
21/08/10 18:11:00 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-ceed1d44-cl91-4634-806b-lcl8639cec2c/pyspark-a61096cf-394d-4345-a23d-d8c8255eb710
21/08/10 18:11:00 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-ceed1d44-cl91-4634-806b-lcl8639cec2c
[hadoop@ip-172-31-17-143 ~]$

```

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)	<a href="http://master-public-dns-name:8082/">http://master-public-dns-name:8082/</a>
Ganglia	<a href="http://master-public-dns-name/ganglia/">http://master-public-dns-name/ganglia/</a>

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