INDEX

Ex. No:1

Date :

Downloading and installing Hadoop; Understanding different Hadoop modes. Startup Scripts, Configuration files

Aim:

To Install Hadoop with the required steps to set up the one node Hadoop cluster.

Procedure:

Step 1:

Installing Java is the main prerequisite for Hadoop. Install java1.7.

```
$sudo apt-get update
$sudo apt-get install openjdk-11-jdk
$sudo apt-get install openjdk-11-jre
$ java -version
```

```
java version "1.7.0_79"

OpenJDK Runtime Environment (IcedTea 2.5.6) (7u79-2.5.6-0ubuntu1.14.04.1) OpenJDK 64-Bit
Server VM (build 24.79-b02, mixed mode)
```

Step 2:

SSH Server accepting password authentication (at least for the setup

time). To install, run:

```
root@a4it196:/home/student# sudo apt-get install openssh-server
```

Step 3:

Generate the ssh key

```
root@a4it196:/home/student# ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
```

```
Generating public/private rsa key
pair. Created directory '/root/.ssh'.

Your identification has been saved in
/root/.ssh/id_rsa. Your public key has been saved
in /root/.ssh/id_rsa.pub.

The key fingerprint is:
77:a1:20:bb:db:95:6d:89:ce:44:25:32:b6:81:5d:d5
root@a4it196 The key's random art image is:

+--[ RSA 2048] ---+

|          .... |
|     o .   E|
|    o B . o  |
|     + * + . |
|    . S + .  |
|     . o = . |
|    . = +    |
|     o = .   |
|    . . o    |
+___________+
```

Step 4:

If the master also acts a slave (`ssh localhost` should work without a password)

root@a4it196:/home/student# cat $HOME/.ssh/id_rsa.pub>>$HOME/.ssh/authorized_keys

Step 5:

Create hadoop group and user:

Step 5.1 root@a4it196:/home/student# **sudo addgroup hadoop**

```
Adding group `hadoop' (GID
1003) ... Done.
```

Step 5.2 root@a4it196:/home/student# **sudo adduser --ingroup hadoop hadoop**

```
Adding user `hadoop' ...

Adding new user `hadoop' (1003) with group
`hadoop' ... Creating home directory
`/home/hadoop' ...

Copying files from `/etc/skel'
... Enter new UNIX
password:


Retype new UNIX password:

passwd: password updated
successfully Changing the user
information for hadoop

Enter the new value, or press ENTER for the
         default Full Name []:

         Room   Number
         []:  Work  Phone
         []: Home Phone
         []: Other []:

Is the information correct?
[Y/n] y
```

```
root@a4it196:/home/student#
```

Step 6:

Copy your .tar file to home.(hadoop-2.7.0.tar.gz)

Step 7:

Extracting the tar file.

```
root@a4it196:/home/student# sudo tar -xzvf hadoop-2.7.0.tar.gz -C /usr/local/lib/
```

Step 8:

Changing the Ownership

```
root@a4it196:/home/student# sudo chown -R hadoop:hadoop /usr/local/lib/hadoop-2.7.0
```

Step 9:

Create HDFS directories:

```
root@a4it196:/home/student# sudo mkdir -p /var/lib/hadoop/hdfs/namenode

root@a4it196:/home/student# sudo mkdir -p /var/lib/hadoop/hdfs/datanode

root@a4it196:/home/student# sudo chown -R hadoop /var/lib/hadoop
```

Step 10:

Check where your Java is installed: root@a4it196:/home/student# **readlink -f /usr/bin/java**

```
/usr/lib/jvm/java-11-openjdk-amd64/jre/bin/java
```

Step 11:

Open gedit and do it root@a4it196:/home/student# **gedit ~/.bashrc**

Add to ~/.bashrc file:

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_INSTALL=/usr/local/lib/hadoop-2.7.0
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib/native"
```

Step 12:

Reload source

```
root@a4it196:/home/student# source ~/.bashrc
```

Step 13:

Modify JAVA_HOME in /usr/local/lib/hadoop-2.7.0/etc/hadoop/hadoop-env.sh

```
: root@a4it196:/home/student# cd /usr/local/lib/hadoop-2.7.0/etc/hadoop
```

root@a4it196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# **sudo gedit hadoop-env.sh**

export JAVA_HOME=${ JAVA_HOME}

Changed this to below path

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

Step 14:

Modify /usr/local/lib/hadoop-2.7.0/etc/hadoop/core-site.xml to have something like:

root@a4it196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# **sudo gedit core-site.xml**

```
<configuration>
 <property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</value>
 </property>
</configuration>
```

Step 15:

Modify /usr/local/lib/hadoop-2.7.0/etc/hadoop/yarn-site.xml to have something like:
root@a4it196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# **sudo gedit yarn-site.xml**

```
<configuration>
 <property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
 </property>
 <property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>
```

```
</configuration>
```

Step 16:

Create /usr/local/lib/hadoop-2.7.0/etc/hadoop/mapred-site.xml from template:
root@a4it196:/usr/local/lib/hadoop-2.7.0/etc/hadoop#

sudo mv mapred-site.xml.template mapred-site.xml

Step 17:

Modify /usr/local/lib/hadoop-2.7.0/etc/hadoop/mapred-site.xml to have something like:
root@a4it196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# sudo **gedit mapred-site.xml**

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Step 18:

Modify /usr/local/lib/hadoop-2.7.0/etc/hadoop/hdfs-site.xml to

have something like: root@a4it196:/usr/local/lib/hadoop-

2.7.0/etc/hadoop# sudo **gedit hdfs-site.xml**

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
```

```
   <value>file:/var/lib/hadoop/hdfs/namenode</value>
  </property>
  <property>
 <name>dfs.datanode.data.dir</name>
 <value>file:/var/lib/hadoop/hdfs/datanode</value>
  </property>
</configuration>
```

Step 19:

Make changes in /etc/profile

$**gedit /etc/profile**

```
JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
PATH=$PATH:$JAVA_HOME/bin

export JAVA_HOME

export PATH
```

$**source /etc/profile**

Step 20:

/usr/local/lib/hadoop-2.7.0/etc/hadoop# **sudo chmod 777 /usr/local/lib/hadoop-2.7.0**

:/usr/local/lib/hadoop-2.7.0/etc/hadoop# **sudo chmod 777 /var/lib/hadoop/hdfs/datanode**

:/usr/local/lib/hadoop-2.7.0/etc/hadoop# **sudo chmod 777 /var/lib/hadoop/hdfs/namenode**

:/usr/local/lib/hadoop-2.7.0/etc/hadoop# **hdfs namenode -format**

Step 21:

Switch to hadoop user
root:/home/hadoop **:#start-dfs.sh**

```
root:/home/hadoop

:#start-yarn.sh

root:/home/hadoop# jps

 6334 SecondaryNameNode

6498 ResourceManager

6927 Jps

6142 DataNode

5990 NameNode

6696 NodeManager
```

Step 22:

Browse the web interface for the **Name Node;** by default it is available at:

http://localhost:50070

Result:

Thus the procedure to set up the one node Hadoop cluster was successfully done.

Ex. No : 2

Date :

Installation of Hive along with practice examples

Aim

To install the Hive with the required steps along with practice examples.

Procedure

Step 1: Installing Hive

The following steps are required for installing Hive on your system. Let us assume the Hive archive is downloaded onto the /Downloads directory.

Extracting and verifying Hive Archive

The following command is used to verify the download and extract the hive archive:

```
$ tar zxvf apache-hive-0.14.0-bin.tar.gz
$ ls
```

On successful download, you get to see the following response:

```
apache-hive-0.14.0-bin apache-hive-0.14.0-bin.tar.gz
```

Copying files to /usr/local/hive directory

We need to copy the files from the super user "su -". The following commands are used to copy the files from the extracted directory to the /usr/local/hive" directory.

```
$ su -
passwd:

# cd /home/user/Download
# mv apache-hive-0.14.0-bin /usr/local/hive
# exit
```

Setting up environment for Hive

You can set up the Hive environment by appending the following lines to **~/.bashrc** file:

```
export HIVE_HOME=/usr/local/hive
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/*:.
```

The following command is used to execute ~/.bashrc file.

```
$ source ~/.bashrc
```

Step 2: Configuring Hive

To configure Hive with Hadoop, you need to edit the **hive-env.sh** file, which is placed in the **$HIVE_HOME/conf** directory. The following commands redirect to Hive **config** folder and copy the template file:

```
$ cd $HIVE_HOME/conf
$ cp hive-env.sh.template hive-env.sh
```

Edit the **hive-env.sh** file by appending the following line:

```
export HADOOP_HOME=/usr/local/hadoop
```

Hive installation is completed successfully. Now you require an external database server to configure Metastore. We use Apache Derby database.

Step 3: Downloading and Installing Apache Derby

Follow the steps given below to download and install Apache Derby:

Downloading Apache Derby

The following command is used to download Apache Derby. It takes some time to download.

```
$ cd ~
$ wget http://archive.apache.org/dist/db/derby/db-derby-
10.4.2.0/db-derby-10.4.2.0-bin.tar.gz
```

The following command is used to verify the download:

```
$ ls
```

On successful download, you get to see the following response:

```
db-derby-10.4.2.0-bin.tar.gz
```

Extracting and verifying Derby archive

The following commands are used for extracting and verifying the Derby archive:

```
$ tar zxvf db-derby-10.4.2.0-bin.tar.gz
$ ls
```

On successful download, you get to see the following response:

```
db-derby-10.4.2.0-bin db-derby-10.4.2.0-bin.tar.gz
```

Copying files to /usr/local/derby directory

We need to copy from the super user "su -". The following commands are used to copy the files from the extracted directory to the /usr/local/derby directory:

```
$ su -
passwd:
# cd /home/user
# mv db-derby-10.4.2.0-bin /usr/local/derby
# exit
```

Setting up environment for Derby

You can set up the Derby environment by appending the following lines to **~/.bashrc** file:

```
export DERBY_HOME=/usr/local/derby
export PATH=$PATH:$DERBY_HOME/bin
Apache Hive
18
export
CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/d
erbytools.jar
```

The following command is used to execute **~/.bashrc** file:

```
$ source ~/.bashrc
```

Create a directory to store Metastore

Create a directory named data in $DERBY_HOME directory to store Metastore data.

```
$ mkdir $DERBY_HOME/data
```

Derby installation and environmental setup is now complete.

Step 4: Configuring Metastore of Hive

Configuring Metastore means specifying to Hive where the database is stored. You can do this by editing the hive-site.xml file, which is in the $HIVE_HOME/conf directory. First of all, copy the template file using the following command:

```
$ cd $HIVE_HOME/conf
$ cp hive-default.xml.template hive-site.xml
```

Edit **hive-site.xml** and append the following lines between the <configuration> and </configuration> tags:

```
<property>
   <name>javax.jdo.option.ConnectionURL</name>
   <value>jdbc:derby://localhost:1527/metastore_db;create=true
</value>
   <description>JDBC connect string for a JDBC metastore
</description>
</property>
```

Create a file named jpox.properties and add the following lines into it:

```
javax.jdo.PersistenceManagerFactoryClass =
org.jpox.PersistenceManagerFactoryImpl
org.jpox.autoCreateSchema = false
org.jpox.validateTables = false
org.jpox.validateColumns = false
org.jpox.validateConstraints = false
org.jpox.storeManagerType = rdbms
org.jpox.autoCreateSchema = true
org.jpox.autoStartMechanismMode = checked
org.jpox.transactionIsolation = read_committed
javax.jdo.option.DetachAllOnCommit = true
javax.jdo.option.NontransactionalRead = true
javax.jdo.option.ConnectionDriverName =
org.apache.derby.jdbc.ClientDriver
javax.jdo.option.ConnectionURL =
jdbc:derby://hadoop1:1527/metastore_db;create = true
javax.jdo.option.ConnectionUserName = APP
javax.jdo.option.ConnectionPassword = mine
```

Step 5: Verifying Hive Installation

Before running Hive, you need to create the **/tmp** folder and a separate Hive folder in HDFS. Here, we use the **/user/hive/warehouse** folder. You need to set write permission for these newly created folders as shown below:

```
chmod g+w
```

Now set them in HDFS before verifying Hive. Use the following commands:

```
$ $HADOOP_HOME/bin/hadoop fs -mkdir /tmp
$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp
$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse
```

The following commands are used to verify Hive installation:

```
$ cd $HIVE_HOME
$ bin/hive
```

On successful installation of Hive, you get to see the following response:

```
Logging initialized using configuration in
jar:file:/home/hadoop/hive-0.9.0/lib/hive-common-
0.9.0.jar!/hive-log4j.properties
Hive history
file=/tmp/hadoop/hive_job_log_hadoop_201312121621_1494929084.txt
…………………….
hive>
```

The following sample command is executed to display all the tables:

```
hive> show tables;
OK
Time taken: 2.798 seconds
hive>
```

Result:

Thus, the installation of Hive was studied and installed successfully.

Ex. No : 3

Date :

Installation of Sqoop

Aim

To install the Sqoop with the required steps for installation.

Procedure

```
wget http://mirrors.estointernet.in/apache/sqoop/1.4.7/sqoop-1.4.7.tar.gz
tar -xvf sqoop-1.4.7.tar.gz

GO to .bashrc and set env variables
====================================
nano ~/.bashrc
export SQOOP_HOME=/home/hdoop/sqoop-1.4.7
export PATH=$PATH:$SQOOP_HOME/bin
source ~/.bashrc
Go to sqoop conf and set hadoop path
=======================================
$ cd $SQOOP_HOME/conf
$ mv sqoop-env-template.sh sqoop-env.sh
$ nano sqoop-env.sh
export HADOOP_COMMON_HOME=/home/hdoop/hadoop-3.2.1
export HADOOP_MAPRED_HOME=/home/hdoop/hadoop-3.2.1
Configure mysql connector
===================
$ wget http://ftp.ntu.edu.tw/MySQL/Downloads/Connector-J/mysql-connector-java-8.0.21.tar.gz
$ tar -xvf mysql-connector-java-8.0.21.tar.gz
$ mv mysql-connector-java-8.0.21/mysql-connector-java-8.0.21.jar /$SQOOP_HOME/lib
Download and put sqoop jar in sqoop home
=========================================
wget https://talend-
update.talend.com/nexus/content/repositories/libraries/org/apache/sqoop/sqoop/1.4.7/sqoop-
1.4.7.jar
cp sqoop-1.4.7.jar sqoop-1.4.7/
Verify the installation
=============
$ sqoop version
```

Result:

Thus, the installation of Hive was studied and installed successfully.

OVERVIEW OF THE PROJECT

This project is about analyzing global COVID-19 data with the help of Big Data tools like Hadoop, Sqoop, Hive, and Spark. The task is broken down into three main parts:

1. Data Ingestion with Sqoop (from MySQL to HDFS)
2. Data Analysis with Hive
3. Data Processing with Spark

TASK 1: INGESTING DATA FROM MYSQL TO HDFS USING SQOOP

Step 1: Prepare the MySQL Database and Table

1. **Create a MySQL Database**

In MySQL, create a database to store the COVID data:

```
sql

CREATE DATABASE CovidDB;
USE CovidDB;
```

2. Create the CovidData Table Create the CovidData table with the appropriate schema:

```
sql

CREATE TABLE CovidData (
    id INT PRIMARY KEY,
    location VARCHAR(100),
    date_current DATE,
    total_cases INT,
    total_deaths INT,
    vaccinated INT,
    diabetes_prevalence DECIMAL(5,2),
    continent VARCHAR(50)
);
```

Step 2: Load Data into MySQL

To load data from the .csvfile into the MySQL table, use the LOAD DATA LOCAL INFILE command:

sql

```sql
LOAD DATA LOCAL INFILE '/home/user/DataSheet.csv'
INTO TABLE CovidData
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Step 3: Create an HDFS Directory for Data

Now, create an HDFS directory where we'll store the data:

bash

```bash
hdfs dfs -mkdir /CovidHDFS
```

Step 4: Import Data from MySQL to HDFS using Sqoop

Now we will use Sqoop to import data from the MySQL table to HDFS:

bash

```bash
sqoop import \
--connect jdbc:mysql://localhost/CovidDB \
--table CovidData \
--target-dir /CovidHDFS \
--num-mappers 1;
```

This command imports the CovidData table from MySQL into HDFS into the /CovidHDFS directory with a single mapper.

Step 5: Backup the Imported Data

After the data is imported, back it up into another directory:

bash

```
hdfs dfs -mkdir /CovidData_Backup
hdfs dfs -cp /CovidHDFS/* /CovidData_Backup/
```

Step 6: View the First and Last 1KB of Data

To view the first 1KB of data:

bash

```
hdfs dfs -cat /CovidHDFS/part-m-00000 | head -c 1024
```

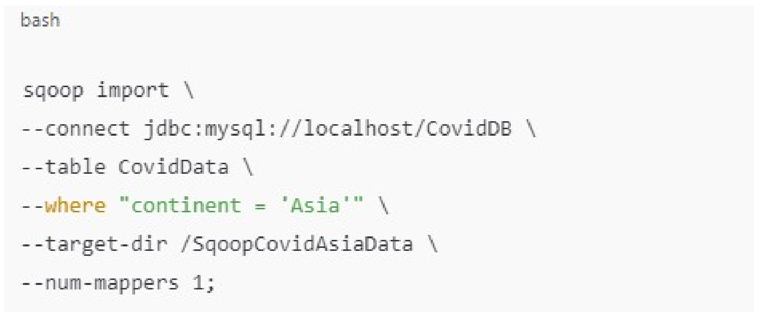
To view the last 1KB of data:

bash

```
hdfs dfs -cat /CovidHDFS/part-m-00000 | tail -c 1024
```

Step 7: Filter Data for Asia

To import only the data related to Asia:

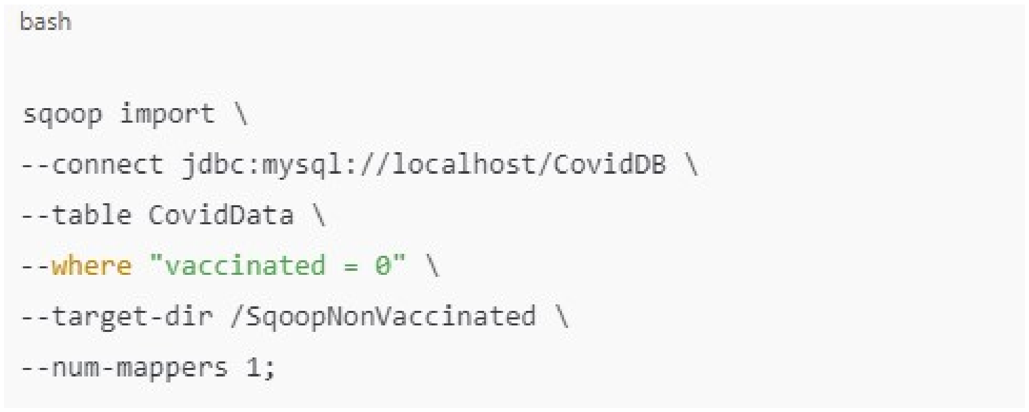


bash

```
sqoop import \
--connect jdbc:mysql://localhost/CovidDB \
--table CovidData \
--where "continent = 'Asia'" \
--target-dir /SqoopCovidAsiaData \
--num-mappers 1;
```

Step 8: Filter Data for Non-Vaccinated People

To import data for people who have not been vaccinated:



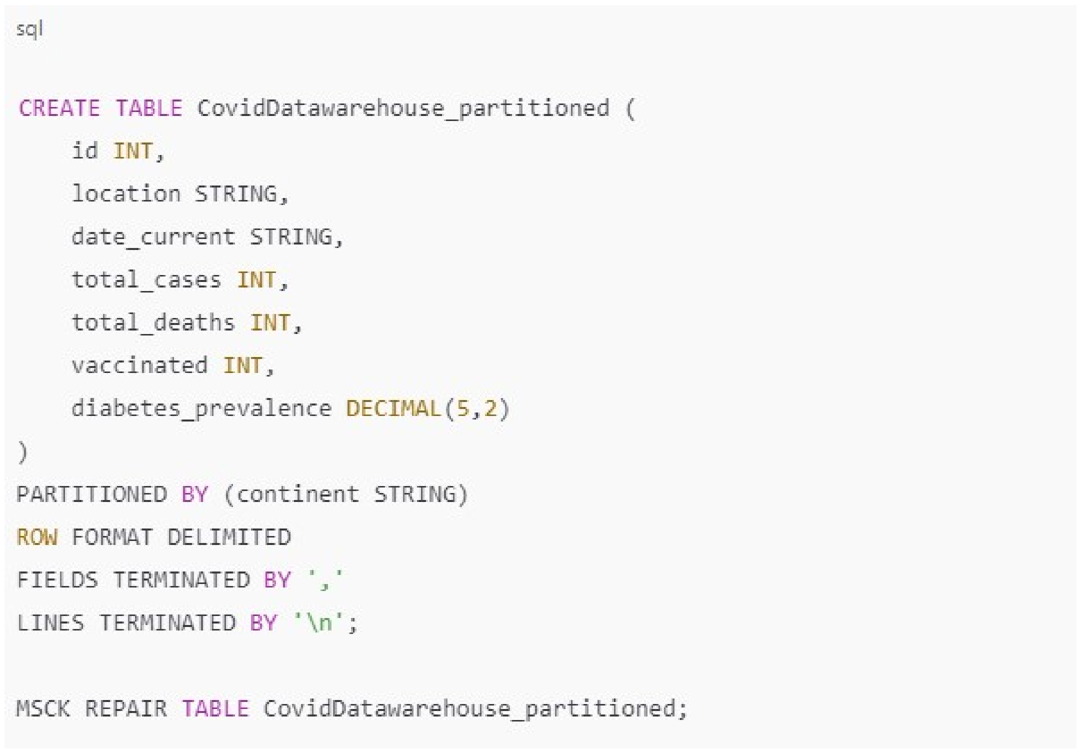
bash

```
sqoop import \
--connect jdbc:mysql://localhost/CovidDB \
--table CovidData \
--where "vaccinated = 0" \
--target-dir /SqoopNonVaccinated \
--num-mappers 1;
```

<u>Task 2: Data Analysis Using Hive</u>

Step 1: Create Hive Table for Global Data

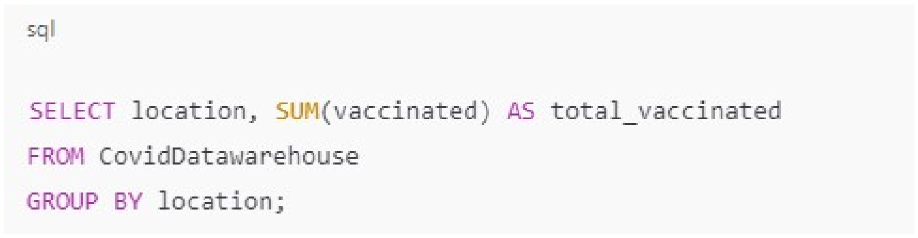
First, create a Hive internal table CovidDatawarehousefor the dataset CovidGlobalData.csv. We'll store date_currentas a string and convert it to a date type later.



```sql
CREATE TABLE CovidDatawarehouse_partitioned (
    id INT,
    location STRING,
    date_current STRING,
    total_cases INT,
    total_deaths INT,
    vaccinated INT,
    diabetes_prevalence DECIMAL(5,2)
)
PARTITIONED BY (continent STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n';

MSCK REPAIR TABLE CovidDatawarehouse_partitioned;
```

Step 2: Count the Number of Vaccinations by Location

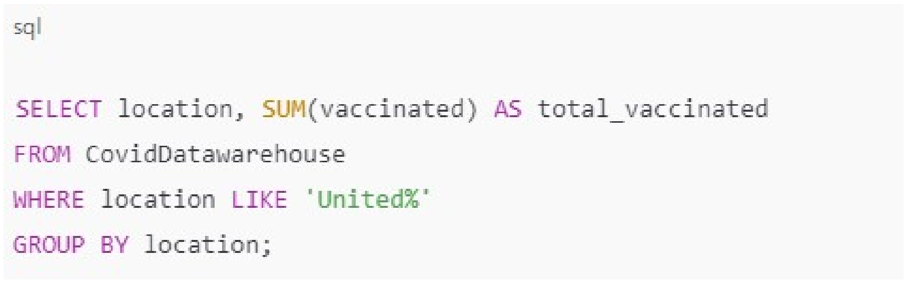
To count the number of vaccinations available in each location:



```sql
SELECT location, SUM(vaccinated) AS total_vaccinated
FROM CovidDatawarehouse
GROUP BY location;
```

Step 3: List Locations Starting with 'United'

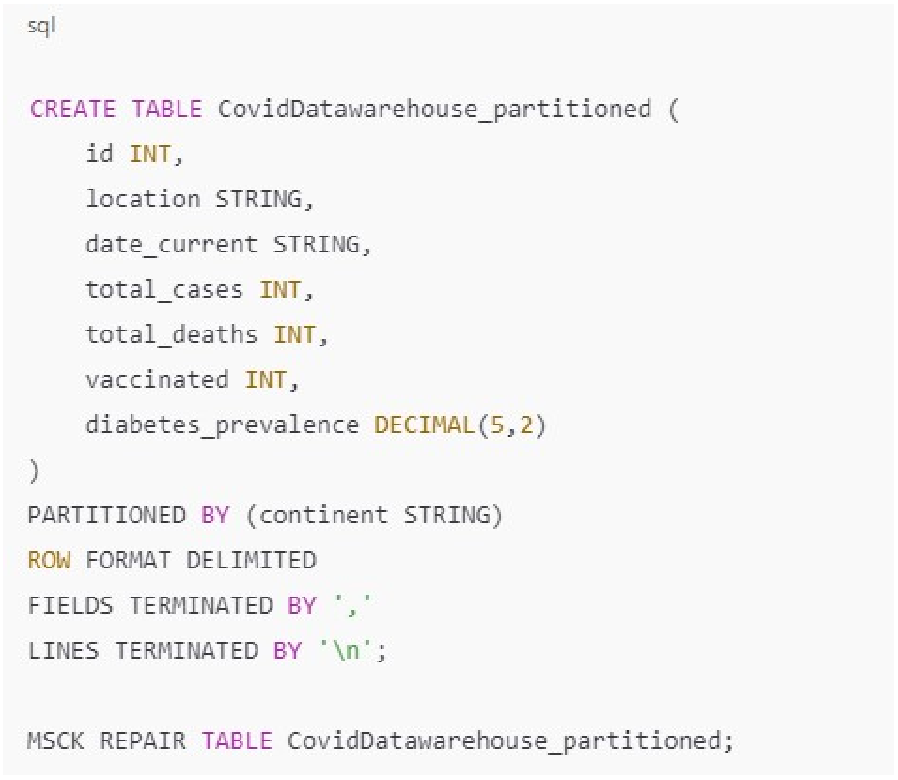
To list the number of vaccinations available in locations starting with "United":



```sql
SELECT location, SUM(vaccinated) AS total_vaccinated
FROM CovidDatawarehouse
WHERE location LIKE 'United%'
GROUP BY location;
```

Step 4: Partition the Data by Continent

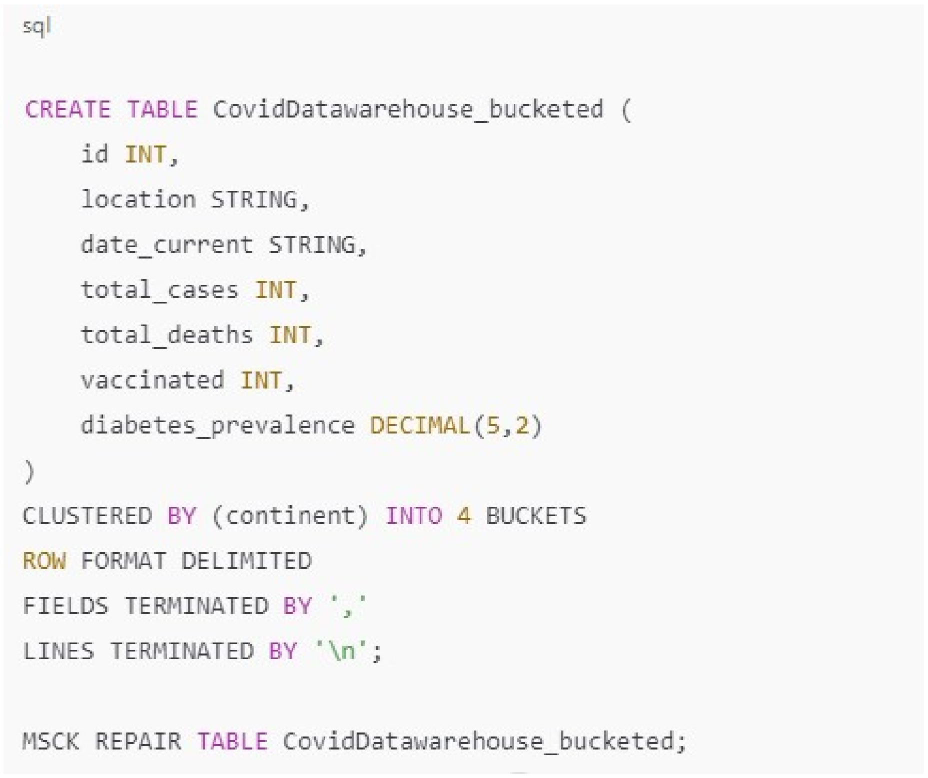
We will partition the data by continent:



```sql
CREATE TABLE CovidDatawarehouse_partitioned (
    id INT,
    location STRING,
    date_current STRING,
    total_cases INT,
    total_deaths INT,
    vaccinated INT,
    diabetes_prevalence DECIMAL(5,2)
)
PARTITIONED BY (continent STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n';

MSCK REPAIR TABLE CovidDatawarehouse_partitioned;
```

Step 5: Bucket Data into 4 Buckets Based on Continent

Now, distribute the data into 4 buckets:

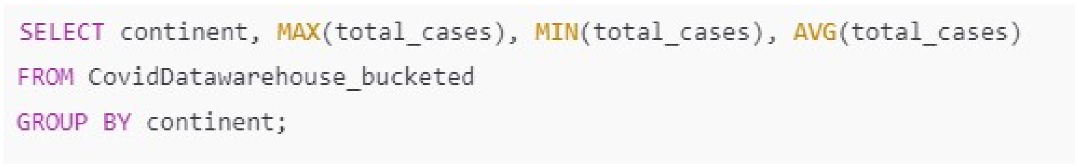


sql

```sql
CREATE TABLE CovidDatawarehouse_bucketed (
    id INT,
    location STRING,
    date_current STRING,
    total_cases INT,
    total_deaths INT,
    vaccinated INT,
    diabetes_prevalence DECIMAL(5,2)
)
CLUSTERED BY (continent) INTO 4 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n';

MSCK REPAIR TABLE CovidDatawarehouse_bucketed;
```

Step 6: Find Max, Min, and Avg COVID Cases in Each Bucket

sql



```sql
SELECT continent, MAX(total_cases), MIN(total_cases), AVG(total_cases)
FROM CovidDatawarehouse_bucketed
GROUP BY continent;
```

Step 7: Find Deaths by Continent

```sql
sql

SELECT continent, SUM(total_deaths)
FROM CovidDatawarehouse
GROUP BY continent;
```

Step 8: Find Average Diabetes Prevalence for Israel

```sql
sql

SELECT AVG(diabetes_prevalence)
FROM CovidDatawarehouse
WHERE location = 'Israel';
```

<u>Task 3: Data Processing with Spark</u>

Step 1: Load Data into Spark

First, let's load the CSV into Spark as a

DataFrame: python

```python
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("CovidAnalysis").getOrCreate()

# Load data into DataFrame
df = spark.read.csv("/path/to/CovidGlobalData.csv", header=True, inferSchema=True)
```

Step 2: Find Total Cases in Each Continent

```python
python

df.groupBy("continent").agg({"total_cases": "sum"}).show()
```

Step 3: Find Total Deaths in Each Location

python

```python
df.groupBy("location").agg({"total_deaths": "sum"}).show()
```

Step 4: Find Max Deaths for Europe and Asia

python

```python
df.filter(df["continent"] == "Europe").agg({"total_deaths": "max"}).show()
df.filter(df["continent"] == "Asia").agg({"total_deaths": "max"}).show()
```

Step 5: Find Total Vaccinations by Continent

python

```python
df.groupBy("continent").agg({"vaccinated": "sum"}).show()
```

Step 6: Find Country-wise Vaccination for January 2021

python

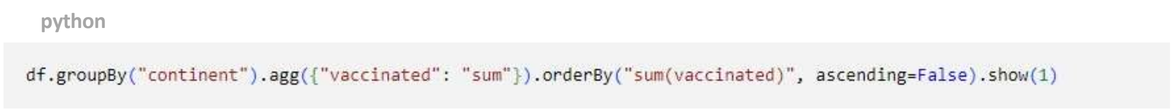
```python
from pyspark.sql.functions import month, year

df = df.withColumn("month", month("date_current")).withColumn("year", year("date_current"))

df.filter((df["month"] == 1) & (df["year"] == 2021)).groupBy("location").agg({"vaccinated": "sum"}).show()
```

Step 7: Find Average Total Cases Across All Locations

python

```python
df.agg({"total_cases": "avg"}).show()
```

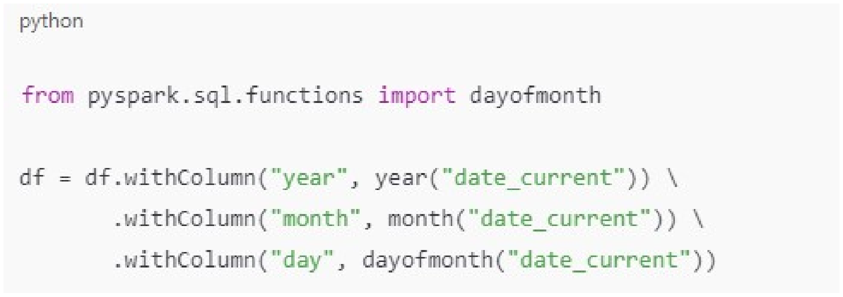
Step 8: Find the Continent with the Highest Vaccinations



python

```python
df.groupBy("continent").agg({"vaccinated": "sum"}).orderBy("sum(vaccinated)", ascending=False).show(1)
```

Step 9: Extract Year, Month, and Day from date_current



python

```python
from pyspark.sql.functions import dayofmonth

df = df.withColumn("year", year("date_current")) \
       .withColumn("month", month("date_current")) \
       .withColumn("day", dayofmonth("date_current"))
```