Data Science

Lab - 4 *18/03/2021*

Name: Manikandan P
RegNo: 2019202030

HADOOP INSTALLATION:

Download the file according to your operating system. Keep the java folder directly under the local disk directory (C:\Java\jdk1.8.0_162) rather than in Program Files (C:\Program Files\Java\jdk1.8.0_162) as it can create errors afterwards

Configurations:

Now we need to edit some files located in the hadoop directory of the etc folder where we installed hadoop. The files that need to be edited have been highlighted.

1. Edit the file core-site.xml in the hadoop directory. Copy this xml property in the configuration in the file

2. Edit mapred-site.xml and copy this property in the configuration

3. Create a folder 'data' in the hadoop directory

4. Edit the file hdfs-site.xml and add below property in the configuration

Note: The path of namenode and datanode across value would be The path of the datanode and namenode folders you just created.

5. Edit the file yarn-site.xml and add below property in the configuration

6. Edit hadoop-env.cmd and replace %JAVA_HOME% with the path of the java folder where your jdk 1.8 is installed2
Hadoop needs Windows OS specific files which does not come with default download of hadoop.

Installing Hadoop :

To install Apache Hive, you must have a Hadoop Cluster installed and running: You can refer to our previously published step-by-step guide to install Hadoop 2.7.6 on Windows 10.

Downloading Apache Hive binaries :

In order to download Apache Hive binaries, you should go to the following website: https://downloads.apache.org/hive/hive-3.1.2/. Then, download the apache-hive-2.7.6.-bin.tar.gz file.

Configuring hive-site.xml:

Now, we should go to the Apache Hive configuration directory (E:\hadoop-env\apache-hive-3.1.2\conf) create a new file "hive-site.xml". We should paste the following XML code within this file:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
   <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>
   <property>
      <name>
         javax.jdo.option.ConnectionDriverName
      </name>

      <value>
         org.apache.derby.jdbc.ClientDriver
      </value>
```

```
        <description>
           Driver class name for a JDBC metastore
        </description>
     </property>
     <property>
        <name>
           hive.server2.enable.doAs
        </name>
        <description>
           Enable user impersonation for HiveServer2
        </description>
        <value>
              True
        </value>
     </property>
     <property>
        <name>
           hive.server2.authentication
        </name>
        <value>
           NONE
        </value>
        <description>

           Client authentication types. NONE: no authentication check LDAP:
           LDAP/AD   based   authentication   KERBEROS:   Kerberos/GSSAPI
           authentication CUSTOM: Custom authentication provider (Use with
           propertyhive.server2.custom.authentication.class)

        </description>
     </property>
     <property>
        <name>
           datanucleus.autoCreateTables
        </name>
        <value>
           True
        </value>
     </property>
  </configuration>
```

Connect python to hive:

step 1:

install bitarray using above link based on your python version download the bitarray-*.whl file suppose python3.8 download this "bitarray-1.7.1-cp38-cp38-win_amd64.whl" download-> save to local path pip install "path-to-your-download-file"

step 2:

pip install impyla

step 3:

```
from impala.dbapi import connect

con= connect(port=10000, auth_mechanism="PLAIN")

cur= con.cursor()
```

Edit hive-site.xml property:

```
<property>
    <name>
        hive.server2.enable.doAs
    </name>
    <description>
        Enable user impersonation for HiveServer2
    </description>
    <value>
        false
    </value>
</property>
```

Commands:

* start-all.cmd

* startNetworkServer -h 0.0.0.07

* hive --service hiveserver2

Hive shell commands:

CREATE A DATABASE IN HIVE:

* create database db;

```
hive> show databases;
OK
default
1 row selected (1.526 seconds)
hive> create database db;
OK
No rows affected (0.613 seconds)
hive> show databases;
OK
db
default
2 rows selected (0.084 seconds)
hive>
```

USE A DATABASE IN HIVE:

* use db;

CREATE A TABLE IN HIVE:

* create table students(rollno int, name string, age int) comment 'student details' row format delimited fields terminated by '\t' lines terminated by '\n' stored as textfile;

```
hive> create table students(rollno int, name string, age int)
. . > comment 'student details'
. . > row format delimited
. . > fields terminated by '\t'
. . > lines terminated by '\n'
. . > stored as textfile;
OK
No rows affected (1.205 seconds)
```

DESCRIBE A TABLE IN HIVE:

* describe students;

```
hive> describe students;
OK
rollno int
name string
age int
3 rows selected (0.791 seconds)
```

LOAD A TEXTFILE IN (students)TABLE:

* load data local inpath 'data.txt' overwrite into table students;

```
hive> load data local inpath 'data.txt' overwrite into table students;
Loading data to table mca.students
OK
No rows affected (2.32 seconds)
```

SHOW ALL DATA FROM THE FILE:

* select * from students;

CREATE A TABLE FOR OUR DATASET:

create table StudentsPerformance(gender string, ethnicity string, parental_level_of_education string, lunch string, test_preparation_course string, math_score int, reading_score int, writing_score int) row format delimited fields terminated by ',' lines terminated by '\n' stored as textfile;

```
hive> create table StudentsPerformance(gender string,
. . > ethnicity string,
. . > parental_level_of_education string,
. . > lunch string,
. . > test_preparation_course string,
. . > math_score int,
. . > reading_score int,
. . > writing_score int)
. . > row format delimited
. . > fields terminated by ','
. . > lines terminated by '\n'
. . > stored as textfile;
OK
No rows affected (0.199 seconds)
```

LOAD DATASET TO THE TABLE:

load data local inpath 'StudentsPerformance.csv' overwrite into table StudentsPerformance;

```
hive> load data local inpath 'StudentsPerformance.csv' overwrite into table StudentsPerformance;
Loading data to table mca.studentsperformance
OK
No rows affected (0.573 seconds)
```

COUNT THE TUPLES IN THE DATASET:

* select count(1) from StudentsPerformance;

```
hive> select count(1) from StudentsPerformance;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = Kalaiselvan_20210320104351_c7197a01-d316-43e1-8af5-337044e78396
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1616212935685_0001, Tracking URL = http://LAPTOP-SVE8K7GI:8088/proxy/application_1616212935685_0001/
Kill Command = C:\hadoop-2.7.6\bin\hadoop.cmd job  -kill job_1616212935685_0001
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-03-20 10:44:14,262 Stage-1 map = 0%,  reduce = 0%
2021-03-20 10:44:26,823 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.718 sec
2021-03-20 10:44:38,049 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.653 sec
MapReduce Total cumulative CPU time: 7 seconds 653 msec
Ended Job = job_1616212935685_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.653 sec   HDFS Read: 65766 HDFS Write: 104 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 653 msec
OK
1001
1 row selected (49.072 seconds)
```

PICK MAX(SELECTED COLUMN) IN TABLE:

* select max(math_score) from StudentsPerformance;

```
hive> select max(math_score) from StudentsPerformance;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = Kalaiselvan_20210320105008_3854bbb2-87d9-4e25-afbc-48840ee03100
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1616212935685_0002, Tracking URL = http://LAPTOP-SVE8K7GI:8088/proxy/application_1616212935685_0002/
Kill Command = C:\hadoop-2.7.6\bin\hadoop.cmd job  -kill job_1616212935685_0002
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-03-20 10:50:23,656 Stage-1 map = 0%,  reduce = 0%
2021-03-20 10:50:32,517 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.841 sec
2021-03-20 10:50:43,496 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.527 sec
MapReduce Total cumulative CPU time: 6 seconds 527 msec
Ended Job = job_1616212935685_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.527 sec   HDFS Read: 66168 HDFS Write: 103 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 527 msec
OK
100
1 row selected (35.739 seconds)
```

CONNECTION QUERY:

```
from impala.dbapi import connect
import pandas as pd
conn=connect(host="localhost",database="mca",port=10000,
auth_mechanism="PLAIN")
cursor= conn.cursor()
```

```
def convert_dataframe(data):
    if data=="No result set":
        print(data)
        return
    try:
        if len(cursor.description[0][0].split("."))>0:
            get_colnames= [x[0].split(".")[1] for x in cursor.description]
        else:
            raise()
    except:
        get_colnames= [x[0] for x in cursor.description]
    return pd.DataFrame(data, columns=get_colnames)

def hasResultSet():
    return cursor.has_result_set

def get_query(query):
    cursor.execute(query)
    if not hasResultSet():
        return str("No result Set")
    return cursor.fetchall()
def query(query):
    data = get_query(query)

return convert_dataframe(data)
```

In [1]:
```
from impala.dbapi import connect
import pandas as pd
conn=connect(host="localhost",database="mca", port=10000, auth_mechanism="PLAIN")
cursor= conn.cursor()

def convert_dataframe(data):
    if data=="No result set":
        print(data)
        return
    try:
        if len(cursor.description[0][0].split("."))>0:
            get_colnames= [x[0].split(".")[1] for x in cursor.description]
        else:
            raise()
    except:
        get_colnames= [x[0] for x in cursor.description]
    return pd.DataFrame(data, columns=get_colnames)

def hasResultSet():
    return cursor.has_result_set

def get_query(query):
    cursor.execute(query)
    if not hasResultSet():
        return str("No result Set")
    return cursor.fetchall()
def query(query):
    data = get_query(query)
    return convert_dataframe(data)
```

In [2]:
```
query('desc StudentsPerformance')
```

Out[2]:

	col_name	data_type	comment
0	gender	string	
1	ethnicity	string	
2	parental_level_of_education	string	
3	lunch	string	
4	test_preparation_course	string	
5	math_score	int	
6	reading_score	int	
7	writing_score	int	

\THANK YOU MAM !!\