Лабораторная работа №4 Кокарев Д. В. РИМ-201211

Задание 0

1. Установить на свой кластер hadoop 3.3 СУБД HIVE 3.1.2

Кластер hadoop уже установлен и настроен из лабораторной работы № 2, поэтому переходим к установки hive.

Скачиваем архив:

```
wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz

tar -xzvf apache-hive-3.1.2-bin.tar.gz
```

Перемещаем в каталог, где находится hadoop:

```
sudo mv apache-hive-3.1.2-bin /usr/local/hive
```

Даем права:

```
sudo chown -R hduser:hadoop /usr/local/hive
```

Добавим следующие строки в файл bashrc:

```
export HIVE_HOME=/usr/local/hive

export CLASSPATH=$CLASSPATH:$HIVE_HOME/lib:$HADOOP_HOME/share/hadoop/common/lib

export PATH=$PATH:$HIVE_HOME/bin
```

```
hduser@master: ~
  GNU nano 4.8                    /home/hduser/.bashrc                    Modified
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=${JAVA_HOME}/bin:${PATH}
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar

export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

export HIVE_HOME=/usr/local/hive
export CLASSPATH=$CLASSPATH:$HIVE_HOME/lib:$HIVE_HOME/share/hadoop/common/lib
export PATH=$PATH:$HIVE_HOME/bin

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

После редактирования, не забываем про source ~/.bashrc

Далее добавляем строки в файл profile:

```
hduser@master: ~
  GNU nano 4.8                    /home/hduser/.profile
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export HIVE_HOME=/usr/local/hive

export CLASSPATH=$CLASSPATH:$HIVE_HOME/lib:$HADOOP_HOME/share/hadoop/common/lib
PATH="$JAVA_HOME/bin:$HADOOP_HOME/bin:$HIVE_HOME/bin:$PATH"

                                [ Read 34 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line
```

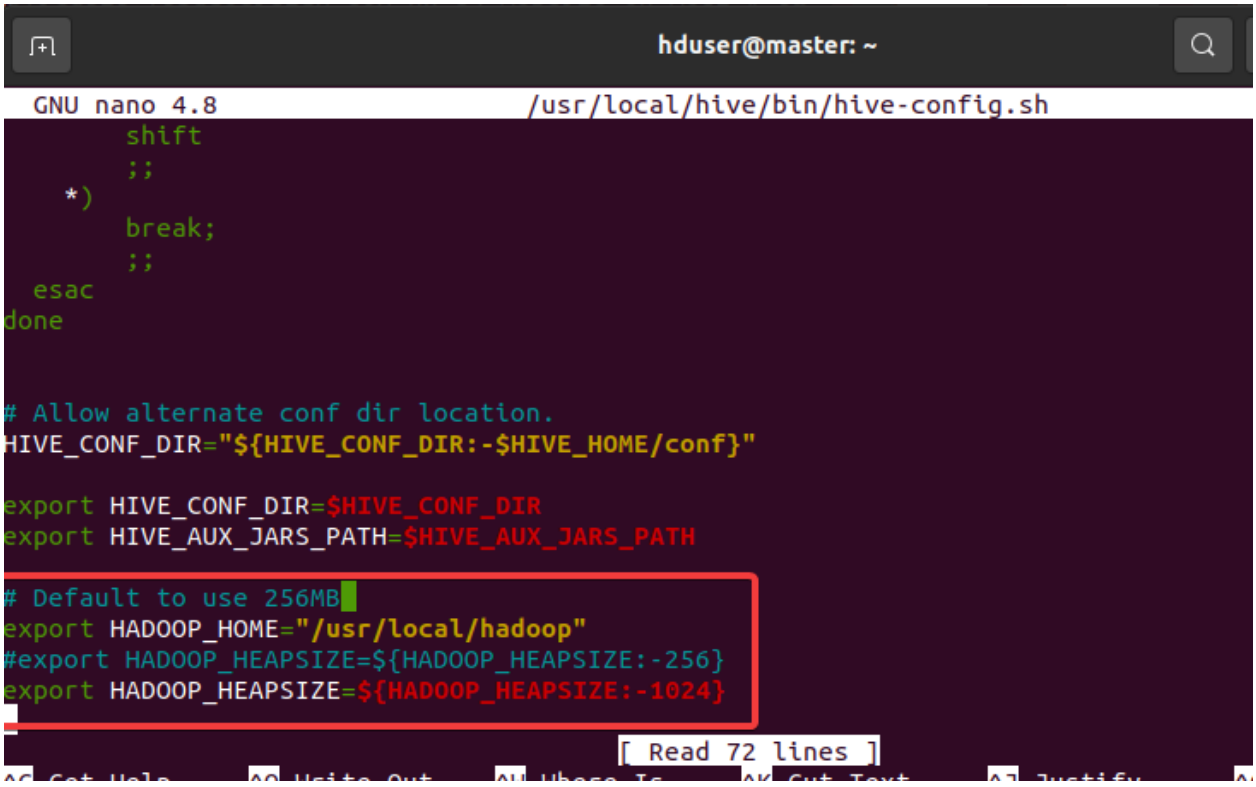
И так же, после редактирования этого файла, вызываем команду: source ~/.profile

Затем редактируем файл hive-config.sh:

export HADOOP_HOME="/usr/local/hadoop-3.3.0"

#export HADOOP_HEAPSIZE=${HADOOP_HEAPSIZE:-256}

export HADOOP_HEAPSIZE=${HADOOP_HEAPSIZE:-1024}



После настройки конфигурационных файлов запускаем инициализацию схемы БД:

schematool -dbType derby –initSchema

Встретилась ошибка:

```
hduser@master:/usr/local/hive/bin$ schematool -dbType derby --initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/St
aticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Exception in thread "main" java.lang.NoSuchMethodError: com.google.common.base.Preconditions.checkAr
gument(ZLjava/lang/String;Ljava/lang/Object;)V
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:1380)
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:1361)
        at org.apache.hadoop.mapred.JobConf.setJar(JobConf.java:536)
        at org.apache.hadoop.mapred.JobConf.setJarByClass(JobConf.java:554)
        at org.apache.hadoop.mapred.JobConf.<init>(JobConf.java:448)
        at org.apache.hadoop.hive.conf.HiveConf.initialize(HiveConf.java:5141)
        at org.apache.hadoop.hive.conf.HiveConf.<init>(HiveConf.java:5104)
        at org.apache.hive.beeline.HiveSchemaTool.<init>(HiveSchemaTool.java:96)
        at org.apache.hive.beeline.HiveSchemaTool.main(HiveSchemaTool.java:1473)
        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 org.apache.hadoop.util.RunJar.run(RunJar.java:323)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:236)
```

После поиска в интернете выяснилось, что это из-за разных версий файлов guava. Необходимо файл, с наименьшей версией, заменить на наибольшую. В моем этот java файл у Hadoop был 27 версии, а у hive – 19. Файлы располагаются в каталогах: /share/Hadoop/hdfs/lib и /usr/local/hive/lib соответственно.

И все заработало:

```
hduser@master:/usr/local/hive/bin$ schematool -dbType derby --initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.j
ar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting metastore schema initialization to 3.1.0
Initialization script hive-schema-3.1.0.derby.sql
```

2. Войти под пользователем hive и запустить консольную утилиту hive

Поскольку, уже был создан пользователь, для работы с Hadoop, то с ним и продолжаем работать и запускать этот сервис на нем же.

Сначала запускаем hiveserver2, предварительно должны быть запущены dfs и yarn:

```
hduser@master:~$ hiveserver2
2021-12-26 08:49:59: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.ja
r!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4
j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 1d9c0514-29c7-4864-b1bf-f6382e0185b9
Hive Session ID = 4ceddb70-ad1a-4fc1-aed2-c34bc8102180
Hive Session ID = 9a6f86a9-fe77-48ab-a69a-465ec2ca7dfd
Hive Session ID = 30854e82-2a38-4c6f-a113-16f23db7adc0
```

Затем запускаем hive:

```
hduser@master:/usr/local/hive/bin$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.j
ar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = e7a479d1-495b-4ae8-b4f0-adebeb958ebf

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-3.1.2.jar!/hiv
e-log4j2.properties Async: true
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.
Hive Session ID = a9a8fda1-6328-4ac4-b457-05d1483c4519
```

3. Выполнить команду select version(); и записать в отчет полученный ответ:

```
hive> select version();
OK
Time taken: 1.691 seconds
```

Версия hive не была получена, потому что не хватает прав, на созданную директорию в кластере Hadoop. Поэтому останавливаем hive и hiveserver2 и прописываем права на папку, которая создалась автоматически при первом запуске:

hadoop fs -chmod -R 777 /tmp

Снова запускаем hiveserver2 и hive, и пробуем получить версию:

```
hive> select version();
OK
3.1.2 r8190d2be7b7165effa62bd21b7d60ef81fb0e4af
Time taken: 3.622 seconds, Fetched: 1 row(s)
hive>
```

Успешно!

Задание 1

1. Воспроизведите примеры из справки раздел DDL Operations

CREATE TABLE pokes (foo INT, bar STRING);

CREATE TABLE invites (foo INT, bar STRING) PARTITIONED BY (ds STRING);

SHOW TABLES;

SHOW TABLES '.*s';

DESCRIBE invites;

```
hive> CREATE TABLE pokes (foo INT, bar STRING);
OK
Time taken: 0.596 seconds
hive> CREATE TABLE invites (foo INT, bar STRING) PARTITIONED BY (ds STRING);
OK
Time taken: 0.097 seconds
hive> SHOW TABLES;
OK
invites
pokes
Time taken: 0.052 seconds, Fetched: 2 row(s)
hive> SHOW TABLES '.*s';
OK
invites
pokes
Time taken: 0.059 seconds, Fetched: 2 row(s)
hive> DESCRIBE invites;
OK
foo                     int
bar                     string
ds                      string

# Partition Information
# col_name              data_type               comment
ds                      string
Time taken: 0.13 seconds, Fetched: 7 row(s)
```

ALTER TABLE pokes ADD COLUMNS (new_col INT);

```
hive> DESCRIBE pokes;
OK
foo                     int
bar                     string
new_col                 int
Time taken: 0.089 seconds, Fetched: 3 row(s)
```

ALTER TABLE invites ADD COLUMNS (new_col2 INT COMMENT 'a comment');

ALTER TABLE invites REPLACE COLUMNS (foo INT, bar STRING, baz INT COMMENT 'baz replaces new_col2');

```
hive> ALTER TABLE invites ADD COLUMNS (new_col2 INT COMMENT 'a comment');
OK
Time taken: 0.158 seconds
hive> ALTER TABLE invites REPLACE COLUMNS (foo INT, bar STRING, baz INT COMMENT 'baz replaces new_col2');
OK
Time taken: 0.194 seconds
hive> DESCRIBE invites;
OK
foo                     int
bar                     string
baz                     int                     baz replaces new_col2
ds                      string
```

ALTER TABLE invites REPLACE COLUMNS (foo INT COMMENT 'only keep the first column');

```
hive> ALTER TABLE invites REPLACE COLUMNS (foo INT COMMENT 'only keep the first column');
OK
Time taken: 0.162 seconds
hive> DESCRIBE invites;
OK
foo                     int                     only keep the first column
ds                      string
```

DROP TABLE pokes;

```
hive> DROP TABLE pokes;
OK
Time taken: 0.147 seconds
hive> show tables;
OK
invites
Time taken: 0.045 seconds, Fetched: 1 row(s)
hive>
```

Задание 2

1. Загрузите тестовый массив данных в текущую папку (файл большой и в облаке, может качаться долго).
2. С помощью команд head и wc -l изучите его содержимое

```
hduser@master:~$ cat pp-complete.csv | wc -l
26541204
```

```
hduser@master:~$ cat pp-complete.csv | head -10
"{F887F88E-7D15-4415-804E-52EAC2F10958}","70000","1995-07-07 00:00","MK15 9HP","D","N","F","31","","ALDRICH DRIVE","WILLEN","MILTON KEYNES","MILTON KEYNES","MILTON KEYNES","A","A"
"{40FD4DF2-5362-407C-92BC-566E2CCE89E9}","44500","1995-02-03 00:00","SR6 0AQ","T","N","F","50","","HOWICK PARK","SUNDERLAND","SUNDERLAND","SUNDERLAND","TYNE AND WEAR","A","A"
"{7A99F89E-7D81-4E45-ABD5-566E49A045EA}","56500","1995-01-13 00:00","CO6 1SQ","T","N","F","19","","BRICK KILN CLOSE","COGGESHALL","COLCHESTER","BRAINTREE","ESSEX","A","A"
"{28225260-E61C-4E57-8B56-566E5285B1C1}","58000","1995-07-28 00:00","B90 4TG","T","N","F","37","","RAINSBROOK DRIVE","SHIRLEY","SOLIHULL","SOLIHULL","WEST MIDLANDS","A","A"
"{444D34D7-9BA6-43A7-B695-4F48980E0176}","51000","1995-06-28 00:00","DY5 1SA","S","N","F","59","","MERRY HILL","BRIERLEY HILL","BRIERLEY HILL","DUDLEY","WEST MIDLANDS","A","A"
"{AE76CAF1-F8CC-43F9-8F63-4F48A2857D41}","17000","1995-03-10 00:00","S65 1QJ","T","N","L","22","","DENMAN STREET","ROTHERHAM","ROTHERHAM","ROTHERHAM","SOUTH YORKSHIRE","A","A"
"{709FB471-3690-4945-A9D6-4F48CE65AAB6}","58000","1995-04-28 00:00","PE7 3AL","D","Y","F","4","","BROOK LANE","FARCET","PETERBOROUGH","PETERBOROUGH","CAMBRIDGESHIRE","A","A"
"{5FA8692E-537B-4278-8C67-5A060540506D}","19500","1995-01-27 00:00","SK10 2QW","T","N","L","38","","GARDEN STREET","MACCLESFIELD","MACCLESFIELD","MACCLESFIELD","CHESHIRE","A","A"
"{E78710AD-ED1A-4B11-AB99-5A0614D519AD}","20000","1995-01-16 00:00","SA6 5AY","D","N","F","592","","CLYDACH ROAD","YNYSTAWE","SWANSEA","SWANSEA","SWANSEA","A","A"
"{1DFBF83E-53A7-4813-A37C-5A06247A09A8}","137500","1995-03-31 00:00","NR2 2NQ","D","N","F","26","","LIME TREE ROAD","NORWICH","NORWICH","NORWICH","NORFOLK","A","A"
hduser@master:~$ cat pp-complete.csv | tail -10
"{CFC9085C-6DCD-9A70-E053-6B04A8C09D6A}","125000","2021-06-24 00:00","SA10 6PJ","S","N","F","22","","HIGHLAND GARDENS","","NEATH","NEATH PORT TALBOT","NEATH PORT TALBOT","A","A"
"{CFC9085C-6DCE-9A70-E053-6B04A8C09D6A}","294995","2021-01-22 00:00","CH4 0LN","D","Y","F","6","","LLYS BUCKLER","PENYMYNYDD","CHESTER","FLINTSHIRE","FLINTSHIRE","A","A"
"{CFC9085C-6DCF-9A70-E053-6B04A8C09D6A}","380000","2021-03-25 00:00","SA19 7LS","D","N","F","LLWYDCOED ISAF","","","PENYBANC","LLANDEILO","CARMARTHENSHIRE","CARMARTHENSHIRE","A","A"
"{CFC9085C-6DD0-9A70-E053-6B04A8C09D6A}","197500","2021-03-22 00:00","LL11 2UY","D","N","F","12","","SNOWDON DRIVE","","WREXHAM","WREXHAM","WREXHAM","A","A"
"{CFC9085C-6DD1-9A70-E053-6B04A8C09D6A}","351495","2021-04-01 00:00","CF64 5WE","D","Y","F","39","","FLAT HOLM WALK","SULLY","PENARTH","THE VALE OF GLAMORGAN","THE VALE OF GLAMORGAN","A","A"
"{CFC9085C-6DD2-9A70-E053-6B04A8C09D6A}","299995","2021-04-01 00:00","CF64 5WE","D","Y","F","40","","FLAT HOLM WALK","SULLY","PENARTH","THE VALE OF GLAMORGAN","THE VALE OF GLAMORGAN","A","A"
"{CFC9085C-6DD4-9A70-E053-6B04A8C09D6A}","250000","2021-03-25 00:00","LL17 0PY","D","N","F","LINDERIC, 2B","","PANT GLAS","","ST ASAPH","DENBIGHSHIRE","DENBIGHSHIRE","A","A"
"{CFC9085C-6DD5-9A70-E053-6B04A8C09D6A}","278995","2021-03-29 00:00","NP12 2QU","D","Y","F","3","","CLOS OAKDALE","GELLIHAF","BLACKWOOD","CAERPHILLY","CAERPHILLY","A","A"
"{CFC9085C-6DD6-9A70-E053-6B04A8C09D6A}","310000","2021-03-31 00:00","CF64 5WD","D","Y","F","32","","MELROSE WALK","SULLY","PENARTH","THE VALE OF GLAMORGAN","THE VALE OF GLAMORGAN","A","A"
"{CFC9085C-6DD7-9A70-E053-6B04A8C09D6A}","335950","2021-03-31 00:00","NP7 5DX","F","Y","L","PLAS ELYRCH","FLAT 1","TUDOR STREET","","ABERGAVENNY","MONMOUTHSHIRE","MONMOUTHSHIRE","A","A"
```

3. С помощью команды head -n сделайте 3 файла содержащие 100к, 1М и 10М строк.

```
hduser@master:~$ cat pp-complete.csv | head -100000 > pp-100k.csv
hduser@master:~$ cat pp-100k.csv | wc -l
100000
hduser@master:~$
hduser@master:~$ cat pp-complete.csv | head -1000000 > pp-1m.csv
hduser@master:~$ cat pp-1m.csv | wc -l
1000000
hduser@master:~$
hduser@master:~$ cat pp-complete.csv | head -10000000 > pp-10m.csv
hduser@master:~$ cat pp-10m.csv | wc -l
10000000
```

4. Создайте тестовую таблицу при помощи кода в примере 1 и загрузите в неё данные записав в отчёт скорость записи каждого файла (для каждого следующего файла таблицу можно удалять или создавать новую с другим именем), количество строк и скорость выполнения запроса count(*).

Создаем таблицу:

CREATE TABLE price_paid (id STRING, price STRING, dt STRING) row format delimited fields terminated by ",";

```
hive>
    > CREATE TABLE price_paid (id STRING, price STRING, dt STRING) row format delimited fields terminated by ",";
OK
Time taken: 0.67 seconds
```

Загружаем файл с 100к записями:

LOAD DATA LOCAL INPATH 'pp-100k.csv' OVERWRITE INTO TABLE price_paid;

```
hive> LOAD DATA LOCAL INPATH 'pp-100k.csv' OVERWRITE INTO TABLE price_paid;
Loading data to table default.price_paid
OK
Time taken: 1.258 seconds
```

Время выполнения: 1.258 сек.

Получаем количество записей:

SELECT count(*) FROM price_paid;

```
hive> SELECT count(*) FROM price_paid;
Query ID = hduser_20211227075705_ad7d2e1d-e62c-44af-8094-c0773956cb8c
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_1640619926322_0002, Tracking URL = http://master:8088/proxy/application_1640619926322_0002/
Kill Command = /usr/local/hadoop/bin/mapred job  -kill job_1640619926322_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-12-27 07:57:18,189 Stage-1 map = 0%,  reduce = 0%
2021-12-27 07:57:26,475 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.12 sec
2021-12-27 07:57:33,661 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.44 sec
MapReduce Total cumulative CPU time: 3 seconds 440 msec
Ended Job = job_1640619926322_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.44 sec   HDFS Read: 17456798 HDFS Write: 106 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 440 msec
OK
100000
```

Аналогично подгружаем таблицы с 1м и 10м записями:

LOAD DATA LOCAL INPATH 'pp-1m.csv' OVERWRITE INTO TABLE price_paid;

```
hive> LOAD DATA LOCAL INPATH 'pp-1m.csv' OVERWRITE INTO TABLE price_paid;
Loading data to table default.price_paid
OK
Time taken: 8.148 seconds
```

```
hive>
    > SELECT count(*) FROM price_paid;
Query ID = hduser_20220110075627_77c14f70-62a0-4a68-a0aa-8bc80a4e33f8
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_1641829100460_0001, Tracking URL = http://master:8088/proxy/application_1641829100460_0001/
Kill Command = /usr/local/hadoop/bin/mapred job  -kill job_1641829100460_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-01-10 07:56:52,617 Stage-1 map = 0%,  reduce = 0%
2022-01-10 07:57:04,025 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.52 sec
2022-01-10 07:57:10,209 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.77 sec
MapReduce Total cumulative CPU time: 3 seconds 770 msec
Ended Job = job_1641829100460_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.77 sec   HDFS Read: 174947411 HDFS Write: 107 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 770 msec
OK
1000000
Time taken: 43.675 seconds, Fetched: 1 row(s)
```

LOAD DATA LOCAL INPATH 'pp-10m.csv' OVERWRITE INTO TABLE price_paid;

```
hive> LOAD DATA LOCAL INPATH 'pp-10m.csv' OVERWRITE INTO TABLE price_paid;
Loading data to table default.price_paid
OK
Time taken: 188.474 seconds
```

```
hive> SELECT count(*) FROM price_paid;
Query ID = hduser_20220110080108_8f80b44e-ab6c-4605-aba8-a6691d29d71e
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_1641829100460_0002, Tracking URL = http://master:8088/proxy/application_1641829100460_0002/
Kill Command = /usr/local/hadoop/bin/mapred job  -kill job_1641829100460_0002
Hadoop job information for Stage-1: number of mappers: 7; number of reducers: 1
2022-01-10 08:02:03,021 Stage-1 map = 0%,  reduce = 0%
2022-01-10 08:02:47,154 Stage-1 map = 14%,  reduce = 0%, Cumulative CPU 2.91 sec
2022-01-10 08:03:04,506 Stage-1 map = 19%,  reduce = 0%, Cumulative CPU 6.19 sec
2022-01-10 08:03:59,436 Stage-1 map = 29%,  reduce = 0%, Cumulative CPU 9.4 sec
2022-01-10 08:04:09,865 Stage-1 map = 33%,  reduce = 0%, Cumulative CPU 9.4 sec
2022-01-10 08:04:27,502 Stage-1 map = 43%,  reduce = 0%, Cumulative CPU 9.92 sec
2022-01-10 08:04:33,934 Stage-1 map = 43%,  reduce = 14%, Cumulative CPU 10.11 sec
2022-01-10 08:04:36,118 Stage-1 map = 57%,  reduce = 14%, Cumulative CPU 14.42 sec
2022-01-10 08:04:41,148 Stage-1 map = 57%,  reduce = 19%, Cumulative CPU 14.48 sec
2022-01-10 08:04:44,688 Stage-1 map = 71%,  reduce = 19%, Cumulative CPU 17.81 sec
2022-01-10 08:04:46,023 Stage-1 map = 71%,  reduce = 24%, Cumulative CPU 17.88 sec
2022-01-10 08:05:00,339 Stage-1 map = 81%,  reduce = 24%, Cumulative CPU 25.41 sec
2022-01-10 08:05:02,723 Stage-1 map = 90%,  reduce = 24%, Cumulative CPU 33.71 sec
2022-01-10 08:05:04,960 Stage-1 map = 90%,  reduce = 29%, Cumulative CPU 33.73 sec
2022-01-10 08:05:05,981 Stage-1 map = 100%,  reduce = 29%, Cumulative CPU 34.39 sec
2022-01-10 08:05:08,029 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 35.74 sec
MapReduce Total cumulative CPU time: 35 seconds 740 msec
Ended Job = job_1641829100460_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 7  Reduce: 1   Cumulative CPU: 35.74 sec   HDFS Read: 1751934282 HDFS Write: 108 SUCCESS
Total MapReduce CPU Time Spent: 35 seconds 740 msec
OK
10000000
Time taken: 241.624 seconds, Fetched: 1 row(s)
```

Задание 3

1. Дополнив оставшимися колонками пример ниже загрузите данные в таблицы HIVE, замерьте время загрузки и запишите в отчёт

CREATE TABLE price (
id STRING,
price INT,
datetime TIMESTAMP,
postcode STRING,
property_type STRING,
new_build_flag STRING,
tenure_type STRING,
paon STRING,
saon STRING,
street STRING,
locality STRING,
town_city STRING,
district STRING,
county STRING,
ppd STRING,
rs STRING)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' WITH SERDEPROPERTIES ("separatorChar" = ",", "quoteChar"="\"", "escapeChar"="\\") STORED AS TEXTFILE;

```
hive> CREATE TABLE price (
    > id STRING,
    > price INT,
    > datetime TIMESTAMP,
    > postcode STRING,
    > property_type STRING,
    > new_build_flag STRING,
    > tenure_type STRING,
    > paon STRING,
    > saon STRING,
    > street STRING,
    > locality STRING,
    > town_city STRING,
    > district STRING,
    > county STRING,
    > ppd STRING, rs STRING )
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' WITH SERDEPROPERTIES ("separatorChar" = ",", "quoteChar"="\"", "escapeChar"="\\") STORED AS TEXTFILE;
OK
Time taken: 2.422 seconds
```

```
hive>
    > LOAD DATA LOCAL INPATH 'pp-complete.csv' OVERWRITE INTO TABLE price;
Loading data to table default.price
OK
Time taken: 1073.622 seconds
```

2. В итоговой таблице должно содержаться 16 колонок и 26_541_204 строк.

SELECT count(*) FROM price;

```
Total MapReduce CPU Time Spent: 3 minutes 36 seconds 880 msec
OK
26541204
Time taken: 520.248 seconds, Fetched: 1 row(s)
hive>
```

3. Напишите запросы к загруженным данным, выполните их и запишите в отчёт: текст запроса, результат выполнения, время выполнения:

3.1. Средняя цена за год

```
select date_format(datetime, 'yyyy'),cast(avg(price) as INT)
from price
group by date_format(datetime, 'yyyy')
order by date_format(datetime, 'yyyy');
```

Результат в файле res_1.txt.

Время выполнения:

```
OK
1995    67931
1996    71506
1997    78532
1998    85436
1999    96037
2000    107483
2001    118885
2002    137942
2003    155888
2004    178886
2005    189352
2006    203528
2007    219378
2008    217056
2009    213419
2010    236109
2011    232804
2012    238366
2013    256923
2014    279938
2015    297266
2016    313222
2017    346095
2018    350275
2019    351488
2020    370677
2021    383662
Time taken: 1731.12 seconds, Fetched: 27 row(s)
```

3.2. Средняя цена за год в Городе

```
select date_format(datetime, 'yyyy'),
town_city,
cast(avg(price) as INT)
from price
group by date_format(datetime, 'yyyy'), town_city
order by date_format(datetime, 'yyyy');
```

Результат в файле res_2.txt

Время выполнения:

```
Time taken: 2502.293 seconds, Fetched: 31180 row(s)
```

3.3. Самые дорогие районы

```
select district,
```

```
cast(avg(price) as INT)
from price
group by district
order by cast(avg(price) as INT) DESC;
```

Результат в файле res_3.txt.

Время выполнения:

```
Time taken: 572.826 seconds, Fetched: 463 row(s)
hive>
```