HIVE CHALLENGE
(Scenario Based questions)

Table of Contents

SCENARIO 1 2
SCENARIO 2 2
SCENARIO 3 2
SCENARIO 4 2
SCENARIO 5 3
SCENARIO 6 3
SCENARIO 7 3
SCENARIO 8 4
SCENARIO 9 4
HIVE JOIN OPERATIONS 5
BUILD A DATA PIPELINE WITH HIVE 6
HIVE OPERATION WITH PYTHON 12

SCENARIO 1

Will the reducer work or not if you use “Limit 1” in any HiveQL query?

Reducer(s) will be applied in HQL with aggregate functions including group by, order by, and so on. No reducers are called for simple select HQL query.

SCENARIO 2

Suppose I have installed Apache Hive on top of my Hadoop cluster using default metastore configuration. Then, what will happen if we have multiple clients trying to access Hive at the same time?

The default metastore configuration allows only one Hive session, and cannot be accessed by multiple clients. In this case, if multiple clients try to access will throw exception.
We have to use a standalone metastore, i.e., Local or remote metastore configuration in Apache Hive for allowing access to multiple clients concurrently. Example: MySQL database

SCENARIO 3

Suppose, I create a table that contains details of all the transactions done by the customers:
CREATE TABLE transaction_details (cust_id INT, amount FLOAT, month STRING, country STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ ;
Now, after inserting 50,000 records in this table, I want to know the total revenue generated for each month. But Hive is taking too much time in processing this query. How will you solve this problem and list the steps that I will be taking in order to do so?

Query latency can be resolved by partitioning the table based on month.

Steps to follow:
1. Create a partitioned table:

```
CREATE TABLE transaction_parts (cust_id int, amount float, country string)
PARTITIONED BY (month string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ‘,’ ;
```

2. Enable dynamic partitioning in Hive:

```
set hive.exec.dynamic.partition = true;
set hive.exec.dynamic.partition.mode = nonstrict;
```

3. Transfer the data from transaction_details to transaction_parts:

```
insert OVERWRITE table transaction_parts
PARTITION (month)
SELECT cust_id, amount, country, month FROM transaction_details;
```

SCENARIO 4

How can you add a new partition for the month December in the above partitioned table?

New partition can be added using Alter statement

```
ALTER TABLE transaction_parts
ADD PARTITION (month='Dec')
LOCATION '<hive path>/transaction_parts';
```

SCENARIO 5

I am inserting data into a table based on partitions dynamically. But I received an error - FAILED ERROR IN SEMANTIC ANALYSIS: Dynamic partition strict mode requires at least one static partition column. How will you remove this error?

We have to execute following command to resolve the issues

```
set hive.exec.dynamic.partition = true;
set hive.exec.dynamic.partition.mode = nonstrict;
```

SCENARIO 6

Suppose, I have a CSV file – 'sample.csv' present in '/temp' directory with the following entries:
id first_name last_name email gender ip_address
How will you consume this CSV file into the Hive warehouse using built-in SerDe?

```
# CREATE TABLE AS CSV SerDe
  create table csv_serde_tbl
  (
  id int,
  first_name string,
  last_name string,
  email string,
  gender string,
  ip_address string
  )
  row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  stored as textfile;

# LOAD DATA INTO csv_serde_tbl FROM LOCAL
  load data local inpath 'file:///tmp/sample.csv' into table csv_serde_tbl;
```

SCENARIO 7

Suppose, I have a lot of small CSV files present in the input directory in HDFS and I want to create a single Hive table corresponding to these files. The data in these files are in the format: {id, name, e-mail, country}. Now, as we know, Hadoop performance degrades when we use lots of small files.
So, how will you solve this problem where we want to create a single Hive table for lots of small files without degrading the performance of the system?

This problem can be resolved using SEQUENCEFILE format which will combine smaller files into one sequence file.

Below steps must be followed to achieve the result.

```
# Create a csv table:
  CREATE TABLE small_csv_tbl
  (
  id int,
  name string,
  e-mail string,
  country string
  )
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  STORED AS TEXTFILE;
```

Load the data into small_csv_tbl:

```
Load data inpath '/input' into table small_csv_tbl;
```

Create a table that will store data in SequenceFile format:

```
CREATE TABLE seq_csv_tbl
(
id int,
name string,
e-mail string,
country string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS SEQUENCEFILE;
```

Transfer the data from the small_csv_tbl table into the seq_csv_tbl table:

```
INSERT OVERWRITE TABLE seq_csv_tbl SELECT * FROM small_csv_tbl;
```

Here, we have managed to create a single sequence file containing all the small csv files.

SCENARIO 8

The following statement failed to execute. What can be the cause?
LOAD DATA LOCAL INPATH 'Home/country/state/' OVERWRITE INTO TABLE address;

The local inpath should contain the entire file path. Here, filename is missing.

SCENARIO 9

Is it possible to add 100 nodes when we already have 100 nodes in Hive? If yes, how?

Yes, we can add another 100 nodes following below steps:

1. Take a new system; create a new username and password
2. Install SSH and with the master node setup SSH connections
3. Add ssh public_rsa id key to the authorized keys file
4. Add the new DataNode hostname, IP address, and other details in /etc/hosts slaves file:
   192.168.1.102 slave3.in slave3
5. Start the DataNode on a new node
6. Login to the new node like suhadoop or:
   ssh -X hadoop@192.168.1.103
7. Start HDFS of the newly added slave node by using the following command:
   ./bin/hadoop-daemon.sh start data node
8. Check the output of the jps command on the new node

HIVE JOIN OPERATIONS

Create a table named CUSTOMERS (ID | NAME | AGE | ADDRESS | SALARY)
Create a Second table ORDER (OID | DATE | CUSTOMER_ID | AMOUNT)

Now perform different joins operations on top of these tables
(Inner JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN)

```
# CREATE CUSTOMER TABLE
create table customer
(
ID int,
NAME string,
AGE int,
ADDRESS string,
SALARY float
)
row format delimited
fields terminated by ','
STORED AS TEXTFILE;

# CREATE ORDER TABLE
create table order
(
OID int,
DATE date,
CUSTOMER_ID int,
AMOUNT float
)
row format delimited
fields terminated by ','
STORED AS TEXTFILE;

# INNER JOIN ON CUSTOMER AND ORDER TABLES
select cust.ID, cust.NAME, cust.AGE, cust.ADDRESS, ord .ID, ord.DATE, ord.AMOUNT
from customer cust
JOIN order ord
on (cust.ID=ord.CUSTOMER_ID)

# LEFT OUTER JOIN ON CUSTOMER AND ORDER TABLES
select cust.ID, cust.NAME, cust.AGE, cust.ADDRESS, ord .ID, ord.DATE, ord.AMOUNT
from customer cust
LEFT OUTER JOIN order ord
on (cust.ID=ord.CUSTOMER_ID)

# RIGHT OUTER JOIN ON CUSTOMER AND ORDER TABLES
select cust.ID, cust.NAME, cust.AGE, cust.ADDRESS, ord .ID, ord.DATE, ord.AMOUNT
from customer cust
RIGHT OUTER JOIN order ord
on (cust.ID=ord.CUSTOMER_ID)
```

```
# FULL OUTER JOIN ON CUSTOMER AND ORDER TABLES
select cust.ID, cust.NAME, cust.AGE, cust.ADDRESS, ord .ID, ord.DATE, ord.AMOUNT
from customer cust
FULL OUTER JOIN order ord
on (cust.ID=ord.CUSTOMER_ID)
```

BUILD A DATA PIPELINE WITH HIVE

Download a data from the given location - https://archive.ics.uci.edu/ml/machine-learning-databases/00360/

"AirQualityUCI.csv" file was downloaded from the above location
The file was then copied into local cloudera path "/home/cloudera/sourav/data" using FileZilla

1. Create a hive table as per given schema in your dataset

```
# CONNECT TO HIVE
  > hive

# CONNECT TO AN EXISTING DATABASE / CREATE A NEW DATABASE
  > user hive_demo;

# CREATE INTERNAL TEMPORARY TABLE
   create table temp_AirQualityUCI_tbl
   (
   Date string,
   Time string,
   CO string,
   PT08S1 int,
   NMHC int,
   C6H6 string,
   PT08S2 int,
   NOx int,
   PT08S3 int,
   NO2 int,
   PT08S4 int,
   PT08S5 int,
   T string,
   RH string,
   AH string,
   Ex1 string,
   Ex2 string
   )
   row format delimited
   fields terminated by '\;'
   tblproperties("skip.header.line.count"="1");
```

2. try to place a data into table location

COPY THE RAW DATA FROM THE CSV FILE INTO THE HIVE TEAMPORARY TABLE

```
LOAD DATA LOCAL INPATH '/home/cloudera/sourav/data/AirQualityUCI.csv' OVERWRITE INTO TABLE
temp_AirQualityUCI_tbl;
```

PRINT THE FIRST 5 ROWS FROM THE TEMPORARY TABLE

```
Select * from temp_AirQualityUCI_tbl limit 5;
```

temp_airqualityuci_tbl.date	temp_airqualityuci_tbl.time	temp_airqualityuci_tbl.co	temp_airqualityuci_tbl.pt08s1	temp_airqualityuci_tbl.nmhc	temp_airqualityuci_tbl.c6h6	temp_airqualityuci_tbl.pt08s2	temp_airqualityuci_tbl.nox	temp_airqualityuci_tbl.pt08s3	temp_airqualityuci_tbl.no2	temp_airqualityuci_tbl.pt08s4	temp_airqualityuci_tbl.pt08s5	temp_airqualityuci_tbl.t	temp_airqualityuci_tbl.rh	temp_airqualityuci_tbl.ah	temp_airqualityuci_tbl.ex1	temp_airqualityuci_tbl.ex2
10/03/2004	18.00.00	2,6	1360	150	11,9	1046	166	1056	113	1692	1268	13,6	48,9	0,7578		
10/03/2004	19.00.00	2	1292	112	9,4	955	103	1174	92	1559	972	13,3	47,7	0,7255		
10/03/2004	20.00.00	2,2	1402	88	9,0	939	131	1140	114	1555	1074	11,9	54,0	0,7502		
10/03/2004	21.00.00	2,2	1376	80	9,2	948	172	1092	122	1584	1203	11,0	60,0	0,7867		
10/03/2004	22.00.00	1,6	1272	51	6,5	836	131	1205	116	1490	1110	11,2	59,6	0,7888		

CREATE A NEW HIVE TABLE IN ORC FORMAT

```
create table AirQualityUCI_ORC_tbl
(
Date string,
Time string,
CO float,
PT08S1 int,
NMHC int,
C6H6 float,
PT08S2 int,
NOx int,
PT08S3 int,
NO2 int,
PT08S4 int,
PT08S5 int,
T float,
RH float,
AH float
)
STORED AS ORC;
```

TRANSFER CLEAN DATA FROM THE TEMPORARY TABLE INTO THE ORC TABLE

REPLACE COMMA (‘,’) WITH DOT (‘.’) USING REGEXP_REPLACE FUNCTION

```
from temp_AirQualityUCI_tbl insert overwrite table AirQualityUCI_ORC_tbl
select
DATE, Time,
regexp_replace(CO,',','.'),
PT08S1, NMHC,
regexp_replace(C6H6,',','.'),
PT08S2, NOx, PT08S3, NO2, PT08S4, PT08S5,
regexp_replace(T,',','.') as T,
regexp_replace(RH,',','.') as RH,
regexp_replace(AH,',','.') as AH;
```

3. Perform a select operation.

```
# PRINT THE FIRST 5 ROWS FROM THE ORC TABLE
## THIS TABLE CONTAINS CLEAN DATA
  Select * from AirQualityUCI_ORC_tbl limit 5;
```

airqualityuci_orc_tbl.date	airqualityuci_orc_tbl.time	airqualityuci_orc_tbl.co	airqualityuci_orc_tbl.pt08s1	airqualityuci_orc_tbl.nmhc	airqualityuci_orc_tbl.c6h6	airqualityuci_orc_tbl.pt08s2	airqualityuci_orc_tbl.nox	airqualityuci_orc_tbl.pt08s3	airqualityuci_orc_tbl.no2	airqualityuci_orc_tbl.pt08s4	airqualityuci_orc_tbl.pt08s5	airqualityuci_orc_tbl.t	airqualityuci_orc_tbl.rh	airqualityuci_orc_tbl.ah
10/03/2004	18.00.00	2.6	1360	150	11.9	1046	166	1056	113	1692	1268	13.6	48.9	0.7578
10/03/2004	19.00.00	2.0	1292	112	9.4	955	103	1174	92	1559	972	13.3	47.7	0.7255
10/03/2004	20.00.00	2.2	1402	88	9.0	939	131	1140	114	1555	1074	11.9	54.0	0.7502
10/03/2004	21.00.00	2.2	1376	80	9.2	948	172	1092	122	1584	1203	11.0	60.0	0.7867
10/03/2004	22.00.00	1.6	1272	51	6.5	836	131	1205	116	1490	1110	11.2	59.6	0.7888

4. Fetch the result of the select operation in your local as a csv file.

```
# TRANSFER DATA FROM HIVE TABLE INTO LOCAL FOLDER AS A CSV FILE
  INSERT OVERWRITE LOCAL DIRECTORY '/home/cloudera/mydata'
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  select * from AirQualityUCI_ORC_tbl;
```

```
[cloudera@quickstart mydata]$ cat /home/cloudera/mydata/* > /home/cloudera/sourav/AirQualityUCI_ORC_tbl_output.csv;
[cloudera@quickstart mydata]$ head /home/cloudera/sourav/AirQualityUCI_ORC_tbl_output.csv;
10/03/2004,18.00.00,2.6,1360,150,11.9,1046,166,1056,113,1692,1268,13.6,48.9,0.7578
10/03/2004,19.00.00,2.0,1292,112,9.4,955,103,1174,92,1559,972,13.3,47.7,0.7255
10/03/2004,20.00.00,2.2,1402,88,9.0,939,131,1140,114,1555,1074,11.9,54.0,0.7502
10/03/2004,21.00.00,2.2,1376,80,9.2,948,172,1092,122,1584,1203,11.0,60.0,0.7867
10/03/2004,22.00.00,1.6,1272,51,6.5,836,131,1205,116,1490,1110,11.2,59.6,0.7888
10/03/2004,23.00.00,1.2,1197,38,4.7,750,89,1337,96,1393,949,11.2,59.2,0.7848
11/03/2004,00.00.00,1.2,1185,31,3.6,690,62,1462,77,1333,733,11.3,56.8,0.7603
11/03/2004,01.00.00,1.0,1136,31,3.3,672,62,1453,76,1333,730,10.7,60.0,0.7702
11/03/2004,02.00.00,0.9,1094,24,2.3,609,45,1579,60,1276,620,10.7,59.7,0.7648
11/03/2004,03.00.00,0.6,1010,19,1.7,561,-200,1705,-200,1235,501,10.3,60.2,0.7517
```

5. Perform group by operation.

```
# SELECT HIVE TABLE USING GROUP BY
  select date,
  avg(CO) as Average_CO_GT,
  avg(PT08S1) as Average_PT08_S1,
  avg(NMHC) as Average_NMHC_CT
  from AirQualityUCI_ORC_tbl
  group by date;
```

7. Perform filter operation at least 5 kinds of filter examples.

```
# GROUP BY
  select date, avg(CO) as Ave_CO, from AirQualityUCI_ORC_tbl group by date;
# EQUAL CLAUSE
  select * from AirQualityUCI_ORC_tbl where date = '03/10/2004';
# LIKE OPERATOR
  select * from AirQualityUCI_ORC_tbl where date like concat('%','4/10/','%');
# NOT NULL
  select * from AirQualityUCI_ORC_tbl where CO is not null;
# RLIKE OPERATOR
  select * from AirQualityUCI_ORC_tbl where CO RLIKE '^(-)';
```

8. show an example of regex operation

SELECT FIRST 10 RECORDS OF HIVE TABLE WITH CO AS NEGATIVE VALUE USING REGEX OPERATION

```
select * from AirQualityUCI_ORC_tbl where CO RLIKE '^(-)' limit 10;
```

```
hive> select * from AirQualityUCI_ORC_tbl where CO RLIKE '^(-)' limit 10;
OK
airqualityuci_orc_tbl.date      airqualityuci_orc_tbl.time      airqualityuci_orc_tbl.co        airqualityuci_orc_tbl.pt08s1    airqualityuci_
hc      airqualityuci_orc_tbl.c6h6      airqualityuci_orc_tbl.pt08s2    airqualityuci_orc_tbl.nox       airqualityuci_orc_tbl.pt08s3    airqua
c_tbl.no2       airqualityuci_orc_tbl.pt08s4    airqualityuci_orc_tbl.pt08s5    airqualityuci_orc_tbl.t airqualityuci_orc_tbl.rh        airqua
c_tbl.ah
11/03/2004      04.00.00        -200.0  1011    14      1.3     527     21      1818    34      1197    445     10.1    60.5    0.7465
12/03/2004      04.00.00        -200.0  831     10      1.1     506     21      1893    32      1134    384     6.1     65.9    0.6248
12/03/2004      09.00.00        -200.0  1545    -200    22.1    1353    -200    767     -200    2058    1588    9.2     56.2    0.6561
13/03/2004      04.00.00        -200.0  1147    56      6.2     821     109     1132    83      1412    992     7.0     71.1    0.7158
14/03/2004      04.00.00        -200.0  1130    56      5.2     773     70      1130    82      1452    1051    12.1    61.1    0.8603
15/03/2004      04.00.00        -200.0  1078    44      4.0     711     66      1150    71      1468    1013    12.3    65.4    0.9351
16/03/2004      04.00.00        -200.0  941     25      2.6     626     59      1316    59      1373    840     12.3    66.2    0.945
17/03/2004      04.00.00        -200.0  883     17      1.9     577     54      1396    60      1303    808     12.7    57.9    0.8447
18/03/2004      04.00.00        -200.0  954     28      2.9     645     60      1260    78      1334    925     11.6    61.9    0.8442
19/03/2004      04.00.00        -200.0  934     -200    1.8     569     20      1440    32      1280    397     12.3    67.7    0.9665
```

9. alter table operation

ALTER TABLE TO RENAME THE TABLE NAME

```
ALTER TABLE temp_AirQualityUCI_tbl RENAME TO renamed_temp_AirQualityUCI_tbl;
```

```
hive> select * from temp_AirQualityUCI_tbl limit 1;
OK
10/03/2004      18.00.00        2,6     1360    150     11,9    1046    166     1056    113     1692    1268    13,6    48,9    0,7578
Time taken: 0.072 seconds, Fetched: 1 row(s)
hive> ALTER TABLE temp_AirQualityUCI_tbl RENAME TO renamed_temp_AirQualityUCI_tbl;
OK
Time taken: 0.088 seconds
hive> select * from temp_AirQualityUCI_tbl limit 1;
FAILED: SemanticException [Error 10001]: Line 1:14 Table not found 'temp_AirQualityUCI_tbl'
hive> select * from renamed_temp_AirQualityUCI_tbl limit 1;
OK
10/03/2004      18.00.00        2,6     1360    150     11,9    1046    166     1056    113     1692    1268    13,6    48,9    0,7578
```

10. drop table operation

DROP A HIVE TABLE

```
DROP TABLE IF EXISTS renamed_temp_AirQualityUCI_tbl;
```

12. order by operation.

RETURN ROWS FROM HIVE TABLE WITH ORDER BY CLAUSE

```
select * from AirQualityUCI_ORC_tbl order by date desc;
```

```
01/01/2005      00.00.00        -200.0  1046    -200    4.2     724     -200    848     -200    898     1201    8.2     40.1    0.4375
01/01/2005      01.00.00        1.6     1275    -200    8.8     930     215     649     106     1024    1617    5.3     50.7    0.4564
01/01/2005      02.00.00        2.5     1173    -200    7.5     878     300     738     129     1002    1355    5.9     50.0    0.4689
01/01/2005      03.00.00        2.7     1163    -200    7.6     881     -200    748     -200    1001    1296    4.9     53.9    0.4693
01/01/2005      04.00.00        1.9     1054    -200    5.6     791     253     830     126     967     1131    4.3     55.3    0.465
```

13. where clause operations you have to perform.

RETURN ROWS FROM HIVE TABLE USING WHERE CLAUSE

```
select * from AirQualityUCI_ORC_tbl
where date = '03/10/2004' and C6H6 > 15;
```

```
03/10/2004      19.00.00        -200.0  1524    -200    20.8    1320    -200    514     -200    1925    1481    23.7    55.6    1.6074
03/10/2004      20.00.00        -200.0  1540    -200    21.4    1336    -200    498     -200    1929    1590    22.7    59.6    1.6261
```

14. sorting operation you have to perform.

RETURN ROWS FROM HIVE TABLE USING SORT BY OPERATION

```
select * from AirQualityUCI_ORC_tbl
where date = '03/10/2004' and C6H6 < 8
sort by time;
```

03/10/2004	01.00.00	-200.0	1037	-200	6.5	836	-200	788	-200	1467	904	20.7	63.2	1.524
03/10/2004	02.00.00	-200.0	982	-200	5.1	768	-200	850	-200	1416	835	21.2	61.5	1.5241
03/10/2004	03.00.00	-200.0	944	-200	4.5	740	-200	881	-200	1412	781	20.3	64.6	1.518
03/10/2004	04.00.00	-200.0	929	-200	3.9	703	-200	908	-200	1391	761	20.2	64.4	1.5035
03/10/2004	05.00.00	-200.0	906	-200	3.3	672	-200	948	-200	1377	742	20.1	65.3	1.5153
03/10/2004	06.00.00	-200.0	907	-200	3.2	665	-200	972	-200	1370	779	19.9	66.0	1.5141
03/10/2004	07.00.00	-200.0	1013	-200	5.4	781	-200	829	-200	1469	894	19.8	66.4	1.5145
03/10/2004	08.00.00	-200.0	1055	-200	5.6	795	-200	800	-200	1476	927	20.4	64.1	1.5178
03/10/2004	09.00.00	-200.0	1138	-200	7.5	880	-200	727	-200	1538	1034	21.6	60.8	1.5496
03/10/2004	14.00.00	-200.0	1092	-200	6.0	814	-200	794	-200	1446	777	27.4	42.8	1.5344
03/10/2004	15.00.00	-200.0	1109	-200	6.7	843	-200	769	-200	1458	743	27.1	43.3	1.5255
03/10/2004	16.00.00	-200.0	1133	-200	6.2	823	-200	775	-200	1449	777	26.9	44.4	1.551
03/10/2004	17.00.00	-200.0	1192	-200	7.7	887	-200	716	-200	1488	819	26.9	44.4	1.5538
03/10/2004	23.00.00	-200.0	1079	-200	7.8	890	-200	745	-200	1522	1000	20.5	64.9	1.5473

15. distinct operation you have to perform.

RETURN DATA FROM A HIVE TABLE USING DISTINCT OPERATION

```
select distinct time from AirQualityUCI_ORC_tbl;
```

16. like operation you have to perform.

RETURN DATA FROM A HIVE TABLE USING LIKE OPERATION

```
select * from AirQualityUCI_ORC_tbl
where date like concat('%','4/10/','%') and time like concat('02','%');
```

```
hive> select * from AirQualityUCI_ORC_tbl where date like concat('%','4/10/','%') and time like concat('02','%');
OK
```

04/10/2004	02.00.00	-200.0	890	-200	3.2	666	-200	969	-200	1323	728	18.9	68.6	1.4818
14/10/2004	02.00.00	-200.0	740	-200	1.1	506	-200	1333	-200	1074	469	14.9	56.4	0.9474
24/10/2004	02.00.00	2.1	1155	-200	10.9	1012	243	633	66	1577	1124	19.4	71.4	1.5959

17. union operation you have to perform.

RETURN DATA FROM A HIVE TABLE USING UNION OPERATION

```
select * from AirQualityUCI_ORC_tbl
where date = '02/04/2004' and time < '05.00.00'
UNION ALL
select * from AirQualityUCI_ORC_tbl
where date = '02/04/2005' and time < '05.00.00'
```

02/04/2004	00.00.00	2.0	1139	157	8.0	899	126	921	104	1514	1067	14.1	54.4	0.8701
02/04/2004	01.00.00	1.3	1072	88	5.6	795	84	986	93	1442	1048	13.3	58.8	0.8922
02/04/2004	02.00.00	1.0	954	68	3.2	667	58	1180	60	1350	718	13.3	60.9	0.9229
02/04/2004	03.00.00	0.9	951	57	3.3	673	-200	1137	-200	1381	797	11.8	64.3	0.8869
02/04/2004	04.00.00	-200.0	926	36	2.8	638	37	1195	52	1342	749	11.2	65.9	0.8775
02/04/2005	00.00.00	1.5	965	-200	5.8	803	280	802	174	951	889	13.1	32.0	0.4817
02/04/2005	01.00.00	1.1	838	-200	1.9	580	167	1055	127	791	516	12.7	33.6	0.4902
02/04/2005	02.00.00	0.6	835	-200	1.5	546	102	1111	81	768	391	12.2	35.6	0.5038
02/04/2005	03.00.00	0.5	820	-200	1.4	536	92	1137	70	777	374	11.9	35.6	0.4945
02/04/2005	04.00.00	0.4	815	-200	0.9	488	63	1223	49	742	326	11.2	38.3	0.5099

18. table view operation you have to perform.

CREATE VIEW FROM A HIVE TABLE

```
create view vw_AirQualityUCI_ORC as
select DATE,Time,CO,PT08S1 from AirQualityUCI_ORC_tbl
where date = '02/04/2004' and time < '05.00.00';
```

```
hive> create view vw_AirQualityUCI_ORC as
    > select DATE,Time,CO,PT08S1 from AirQualityUCI_ORC_tbl
    > where date = '02/04/2004' and time < '05.00.00';
OK
Time taken: 0.074 seconds
hive> select * from vw_AirQualityUCI_ORC;
OK
02/04/2004      00.00.00        2.0     1139
02/04/2004      01.00.00        1.3     1072
02/04/2004      02.00.00        1.0     954
02/04/2004      03.00.00        0.9     951
02/04/2004      04.00.00        -200.0  926
Time taken: 0.093 seconds, Fetched: 5 row(s)
```

HIVE OPERATION WITH PYTHON

Create a python application that connects to the Hive database for extracting data, creating sub tables for data processing, drops temporary tables. Fetch rows to python itself into a list of tuples and mimic the join or filter operations