

Name :- Mihir Manoj Jawale
Student_id =240840325033

HIVE

Question 1

1.

select a.name from airport a join routes r on r.src_airport_id=a.airport_id join routes on r.dest.airport_id=a.airport_id;

2.

select equipment from routes r join airline a on a.airline_id= r.airline_id where a.stops=max(stops)

3.

select count(*) from airline a join routes r on a.airline_id = r.airline_id where stops!=0;

```
hive (cdac_mj)> select count(*) from airlines a join routes r on a.airline_id = r.airline_id where stops!=0;
Query ID = cdacuser72325_20241121084936_8f0e20b0-1619-4974-98af-1d213c6e2783
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Defaulting to jobconf value of: 4
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_1732089968849_2293, Tracking URL = http://master:6318/proxy/application_1732089968849_2293/
```



```

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_1732089968849_2296, Tracking URL = http://master:6318/proxy/application_1732089968849_2296/
Kill Command = /opt/hadoop/bin/mapred job -kill job_1732089968849_2296
Hadoop job information for Stage-2: number of mappers: 2; number of reducers: 1
2024-11-21 08:50:17,410 Stage-2 map = 0%, reduce = 0%
2024-11-21 08:50:22,539 Stage-2 map = 50%, reduce = 0%, Cumulative CPU 2.55 sec
2024-11-21 08:50:24,602 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 5.13 sec
2024-11-21 08:50:29,717 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 8.33 sec
MapReduce Total cumulative CPU time: 8 seconds 330 msec
Ended Job = job_1732089968849_2296
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2 Reduce: 4 Cumulative CPU: 27.02 sec HDFS Read: 2727222 HDFS Write: 456 SUCCESS
Stage-Stage-2: Map: 2 Reduce: 1 Cumulative CPU: 8.33 sec HDFS Read: 11122 HDFS Write: 102 SUCCESS
Total MapReduce CPU Time Spent: 35 seconds 350 msec
OK
11
Time taken: 55.374 seconds, Fetched: 1 row(s)
hive (cdac_mj)>
>
>
>
>
>
>

```

Question 2

1.

create table source_id (airline string , airline_id int, src_airport_id int , dest_airport_iata String ,dest_airport_id int , codeshare string , stops int equipment string) partitioned by (src_airport_iata string)

```

hive (cdac_mj)> create table source_id (airline string , airline_id int, src_airport_id int , dest_airport_iata String ,dest_airport_id int , codeshare string , stops i
nt, equipment string) partitioned by ,(src_airport_iata string);
OK
Time taken: 0.154 seconds
hive (cdac_mj)> set hive.exec.dynamic.partition.mode=nonstrict;
hive (cdac_mj)>
> set hive.exec.dynamic.partition=true;
hive (cdac_mj)> insert into table source_id partitioned(src_airport_iata) airline , airline_id ,src_airport_iata, src_airport_id, dest_airport_iata, dest_airport_id ,
codeshaer, stops , equipment from routes;
NoViableAltException(225@[])
at org.apache.hadoop.hive.q1.parse.HiveParser.regularBody(HiveParser.java:39572)
at org.apache.hadoop.hive.q1.parse.HiveParser.queryStatementExpressionBody(HiveParser.java:38900)
at org.apache.hadoop.hive.q1.parse.HiveParser.queryStatementExpression(HiveParser.java:38788)
at org.apache.hadoop.hive.q1.parse.HiveParser.execStatement(HiveParser.java:2396)
at org.apache.hadoop.hive.q1.parse.HiveParser.statement(HiveParser.java:1420)
at org.apache.hadoop.hive.q1.parse.ParseDriver.parse(ParseDriver.java:220)
at org.apache.hadoop.hive.q1.parse.ParseUtils.parse(ParseUtils.java:74)
at org.apache.hadoop.hive.q1.parse.ParseUtils.parse(ParseUtils.java:67)
at org.apache.hadoop.hive.q1.Driver.compile(Driver.java:616)
at org.apache.hadoop.hive.q1.Driver.compileInternal(Driver.java:1826)
at org.apache.hadoop.hive.q1.Driver.compileAndRespond(Driver.java:1773)

```

```
insert into table source_id select * from routes;
```

2.

```
insert into table source_id partitioned(src_airport_iata) (airline ,
airline_id, src_airport_id , dest_airport_iata ,dest_airport_id , codeshare , stops, equipment)
from routes where src_airport_iata='JFK' ;
```

3.

```
insert into table source_id partitioned(src_airport_iata) (airline ,  
airline_id, src_airport_id , dest_airport_iata ,dest_airport_id , codeshare , stops, equipment)  
from routes where src_airport_iata='LAX' ;
```

4.

```
source_id.getPartitionis()
```

Spark

```
data=sc.textFile("/user/cdacuser72325/airlines1.csv")
```

```
data.count()
```

```
header =data.first()
```

```
clear = data.filter(lambda a : a!=header)
```

```
for line in clear.take(10):  
    print(line)
```

Question 1:

1.

2.

```
split = clear.map(lambda a:
(a.split(",")[0],a.split(",")[1],a.split(",")[2],int(a.split(",")[3])
))
>>> for line in split.take(10):
...     print(line)
```

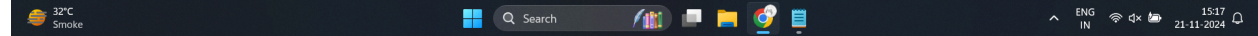
```
>>> split = clear.map(lambda a: (a.split(",")[0],a.split(",")[1],a.split(",")[2],int(a.split(",")[3])))
>>> for line in split.take(10):
...     print(line)
...
('1995', '1', '296.9', 46561)
('1995', '2', '296.8', 37443)
('1995', '3', '287.51', 34128)
('1995', '4', '287.78', 30388)
('1996', '1', '283.97', 47808)
('1996', '2', '275.78', 43020)
('1996', '3', '269.49', 38952)
('1996', '4', '278.33', 37443)
('1997', '1', '283.4', 35067)
('1997', '2', '289.44', 46565)
>>>
```

Question 2

1.

```
s=split.map(lambda a:a[2])
>>> print(s.mean())
329.7475
>>> print(s.max())
396.37
>>> print(s.min())
269.49
```

```
>>> split = clear.map(lambda a: (a.split(",")[0],a.split(",")[1],float(a.split(",")[2]),int(a.split(",")[3])))
>>> s=split.map(lambda a:a[2]).min()
>>> for line in s.take(1):
...     print(line)
...
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'float' object has no attribute 'take'
>>> s=split.map(lambda a:a[2])
>>> print(s.mean())
329.7475
>>> print(s.max())
396.37
>>> print(s.min())
269.49
>>> █
```



2.

```
>>> s1=split.map(lambda a:a[2] >290 )
>>> s.count()
84
>>> s1.count()
84
```

```
>>> s.count()
84
>>> s1=split.map(lambda a:a[2] >290 )
>>> s.count()
84
>>> s1.count()
84
>>> for line in s1.take(10):
...     print(line)
...
True
True
False
False
False
False
False
False
False
False
>>>
```



3.

```
>>> split = clear.map(lambda a:
(a.split(",")[0],int(a.split(",")[3])))
>>> for line in split.take(10):
```

```
...     print(line)
...
('1995', 46561)
('1995', 37443)
('1995', 34128)
('1995', 30388)
('1996', 47808)
('1996', 43020)
>>> for line in reduce.collect():
...     print(line)
...
('1995', 148520)
('2002', 152195)
('2003', 156153)
('2004', 164800)
('2007', 176299)
('2010', 163741)
('2011', 142647)
('2012', 166076)
('2013', 173676)
('2014', 159823)
('2015', 165438)
('1996', 167223)
('1997', 157972)
('1998', 135678)
('1999', 150000)
('2000', 154376)
('2001', 173598)
('2005', 150610)
```

```

>>> split = clear.map(lambda a: (a.split(",")[0],int(a.split(",")[3])))
>>> for line in split.take(10):
...     print(line)
...
('1995', 46561)
('1995', 37443)
('1995', 34128)
('1995', 36388)
('1996', 47898)
('1996', 43828)
('1996', 38952)
('1996', 37443)
('1997', 35067)
('1997', 46565)
>>> reduce = split.reduceByKey(lambda a,b a+b)
File "<stdin>", line 1
      reduce = split.reduceByKey(lambda a,b a+b)
                                   ^
SyntaxError: invalid syntax
>>> reduce = split.reduceByKey(lambda a,b: a+b)
>>> for line in reduce.take(10):
...     print(line)
...
('1995', 148520)
('2002', 152195)
('2003', 156153)
('2004', 164800)
('2007', 176299)
('2010', 163741)
('2011', 142647)
('2012', 166076)
('2013', 173676)
('2014', 159823)

```



```

>>> for line in reduce.collect():
...     print(line)
...
('1995', 148520)
('2002', 152195)
('2003', 156153)
('2004', 164800)
('2007', 176299)
('2010', 163741)
('2011', 142647)
('2012', 166076)
('2013', 173676)
('2014', 159823)
('2015', 165438)
('1996', 167223)
('1997', 157972)
('1998', 135678)
('1999', 150000)
('2000', 154376)
('2001', 173598)
('2005', 150610)
('2006', 153789)
('2008', 166897)
('2009', 150308)
>>>

```



4.

```
>>> split = clear.map(lambda a: (a.split(",")[0])
... )
>>> split.collect()
['1995', '1995', '1995', '1995', '1996', '1996', '1996', '1996', '1996', '1997', '1997', '1997', '1997', '1997', '1998', '1998', '1998', '1998', '1999', '1999', '1999', '1999', '2000',
'2000', '2000', '2000', '2001', '2001', '2001', '2001', '2002', '2002', '2002', '2002', '2003', '2003', '2003', '2003', '2004', '2004', '2004', '2004', '2005', '2005',
'2005', '2005', '2006', '2006', '2006', '2006', '2007', '2007', '2007', '2007', '2008', '2008', '2008', '2008', '2009', '2009', '2009', '2009', '2010', '2010', '2010',
'2010', '2011', '2011', '2011', '2011', '2012', '2012', '2012', '2012', '2013', '2013', '2013', '2013', '2014', '2014', '2014', '2014', '2015', '2015', '2015', '2015']
>>> combine = split.reduce
```

```
reduce =split.reduceByKey(lambda a,b :a+b)
```

```
for line in reduce.collect():
    print(line)
```

Output :

```
1995
1996
1997
1998
1999
...
2014
```

5.

```
split =clear.map(lambda a:
a.split(",")[0],float(a.split(",")[2]),int(a.split(",")[3]))
```

```
Revenue =split.map(lambda x: x[2] * x[3])
```

```
reduce =revenue.reduceByKey(lambda a,b : a+b)
```

```
For line in reduce.collect():
    print(line)
```