Assignment 1

Create "ratings" managed table

```
create table ratings(userID int,movieID int,rating float,timestamps string)
row format delimited
fields terminated by ','
lines terminated by '\n';
```

Display the ratings data

```
select * from ratings limit 10;
```

```
0: jdbc:hive2://> select * from ratings limit 10;
OK
+-----------------+------------------+-----------------+---------------------+
| ratings.userid  | ratings.movieid  | ratings.rating  | ratings.timestamps  |
+-----------------+------------------+-----------------+---------------------+
| NULL            | NULL             | NULL            | timestamp           |
| 1               | 1                | 4.0             | 964982703           |
| 1               | 3                | 4.0             | 964981247           |
| 1               | 6                | 4.0             | 964982224           |
| 1               | 47               | 5.0             | 964983815           |
| 1               | 50               | 5.0             | 964982931           |
| 1               | 70               | 3.0             | 964982400           |
| 1               | 101              | 5.0             | 964980868           |
| 1               | 110              | 4.0             | 964982176           |
| 1               | 151              | 5.0             | 964984041           |
+-----------------+------------------+-----------------+---------------------+
```

Display rating wise count

```
select rating,count(movieid) as counts from ratings group by rating;
```

```
+---------+---------+
| rating  | counts  |
+---------+---------+
| NULL    | 0       |
| 0.5     | 1370    |
| 1.0     | 2811    |
| 1.5     | 1791    |
| 2.0     | 7551    |
| 2.5     | 5550    |
| 3.0     | 20047   |
| 3.5     | 13136   |
| 4.0     | 26818   |
| 4.5     | 8551    |
| 5.0     | 13211   |
+---------+---------+
```

Assignment 2

Create "weather" external table under /user/training/weather

```
create external table weather(wbanno int, lst_date date, crx_vn decimal(4,3), longitude decimal(4,2),
latitude decimal(4,2), t_daily_max float, t_daily_min float)
row format delimited
fields terminated by ' '
lines terminated by '\n'
location '/user/training/weather';
```

Load the data

```
load data local inpath '/home/miles/futurense_hadoop-pyspark/labs/dataset/weather/weather_fine.txt'
overwrite into table weather;
```

Display the weather data

```
0: jdbc:hive2://> select * from weather limit 5;
23/02/19 16:57:30 [31587b9a-6c4d-4fbb-8fa2-3089a772a473 main]: WARN metastore.ObjectStore: datanucleus.autoStartMechanismMode is set to unsupported value null . Setting it to value: ignored
OK
23/02/19 16:57:30 [31587b9a-6c4d-4fbb-8fa2-3089a772a473 main]: WARN lazy.LazyStruct: Extra bytes detected at the end of the row! Ignoring similar problems.
+----------------+-----------------+---------------+------------------+-----------------+--------------------+--------------------+
| weather.wbanno | weather.lst_date | weather.crx_vn | weather.longitude | weather.latitude | weather.t_daily_max | weather.t_daily_min |
+----------------+-----------------+---------------+------------------+-----------------+--------------------+--------------------+
| 23907          | 2015-01-01      | 2.423         | -98.08           | 30.62           | 2.2                | -0.6               |
| 23907          | 2015-01-02      | 2.423         | -98.08           | 30.62           | 3.5                | 1.3                |
| 23907          | 2015-01-03      | 2.423         | -98.08           | 30.62           | 15.9               | 2.3                |
| 23907          | 2015-01-04      | 2.423         | -98.08           | 30.62           | 9.2                | -1.3               |
| 23907          | 2015-01-05      | 2.423         | -98.08           | 30.62           | 10.9               | -3.7               |
+----------------+-----------------+---------------+------------------+-----------------+--------------------+--------------------+
```

Display Max, Min weather

```
select max(t_daily_max), min(t_daily_min) from weather;
```

```
+--------+--------+
|  max   |  min   |
+--------+--------+
| 36.0   | -7.9   |
+--------+--------+
```

Display month-wise Max and Min weather

```
select month(lst_date) as month,max(t_daily_max) as max,min(t_daily_min) as min from weather group
by month(lst_date);
```

month	max	min
1	26.5	-7.9
2	26.6	-3.5
3	29.1	-3.2
4	30.8	8.0
5	31.1	14.3
6	33.6	0.0
7	36.0	19.8

Assignment 3

Create "customers" and "transactions" tables

```
create table Customers(cust_id int, last_name string, first_name string, age int, profession string)
row format delimited
fields terminated by ','
lines terminated by '\n'
location '/user/training/retail';

create table Transactions(trans_id int, trans_date date, cust_id int, amount double, category string, desc string, city string, state string, pymt_mode string)
row format delimited
fields terminated by ','
lines terminated by '\n'
location '/user/training/retail';
```

Load the data

```
load data local inpath '/home/miles/futurense_hadoop-pyspark/labs/dataset/retail/customers.txt' overwrite into table Customers;

load data local inpath '/home/miles/futurense_hadoop-pyspark/labs/dataset/retail/transactions1.txt' overwrite into table Transactions;
```

1Q) No of transactions by customer

```
select cust_id,count(trans_id) as transactions from transactions group by cust_id;
```

cust_id	transactions
4000001	8
4000002	6
4000003	3
4000004	5
4000005	5
4000006	5
4000007	6
4000008	10
4000009	6
4000010	6

2Q) Total transaction amount by customer

select cust_id,sum(amount) as amount from transactions group by cust_id;

cust_id	amount
4000001	651.05
4000002	706.97
4000003	527.5899999999999
4000004	337.06
4000005	325.15
4000006	539.38
4000007	699.5500000000001
4000008	859.42
4000009	457.83
4000010	447.09000000000003

3Q) Get top 3 customers by transaction amount

select cust_id,sum(amount) as amount from transactions group by cust_id order by sum(amount) desc limit 3;

cust_id	amount
4000008	859.42
4000002	706.97
4000007	699.5500000000001

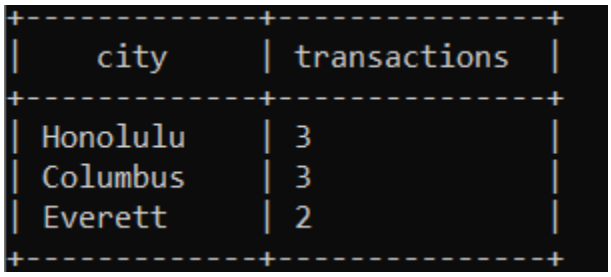
4Q) No of transactions by customer and mode of payment

select cust_id,count(trans_id) as transactions,pymt_mode from transactions group by cust_id,pymt_mode;

cust_id	transactions	pymt_mode
4000001	1	cash
4000001	7	credit
4000002	1	cash
4000002	5	credit
4000003	3	credit
4000004	4	cash
4000004	1	credit
4000005	1	cash
4000005	4	credit
4000006	5	credit
4000007	6	credit
4000008	10	credit
4000009	6	credit
4000010	6	credit

5) Get top 3 cities which has more transactions

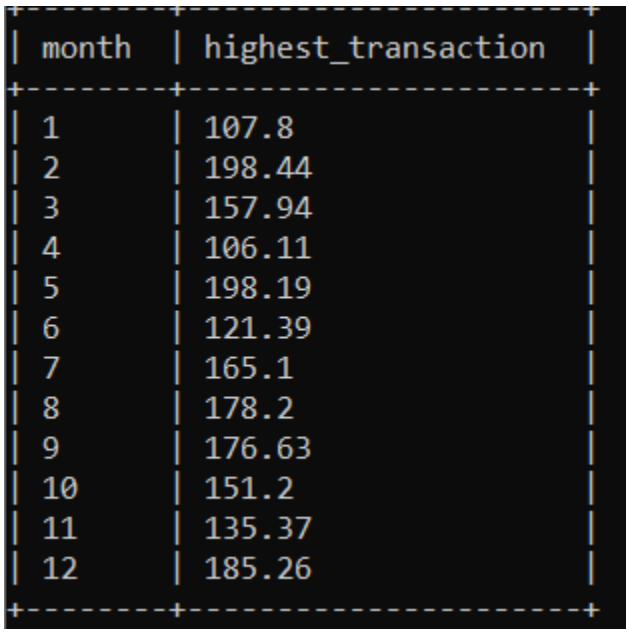
select city, count(trans_id) as transactions from transactions group by city order by count(trans_id) desc limit 3;



city	transactions
Honolulu	3
Columbus	3
Everett	2

6) Get month wise highest transaction

select month(trans_date) as month, max(amount) as highest_transaction from transactions group by month(trans_date);



month	highest_transaction
1	107.8
2	198.44
3	157.94
4	106.11
5	198.19
6	121.39
7	165.1
8	178.2
9	176.63
10	151.2
11	135.37
12	185.26

7) Get sample transactions

create table Transactions1(trans_id int, trans_date date, cust_id int, amount double, category string, desc string, city string, state string, pymt_mode string)
clustered by (cust_id) into 5 buckets
row format delimited
fields terminated by ','
lines terminated by '\n'
location '/user/training/retail';

load data local inpath '/home/miles/futurense_hadoop-pyspark/labs/dataset/retail/transactions1.txt' overwrite into table Transactions;

SELECT * FROM transactions1 TABLESAMPLE (bucket 3 out of 5 on cust_id);

transactions1.trans_id	transactions1.trans_date	transactions1.cust_id	transactions1.amount	transactions1.category	transactions1.desc	transactions1.city	transactions1.state	transactions1.pymt_mode
55	2012-04-12	4000006	106.11	Water Sports	Swimming	New York	New York	credit
39	2011-12-03	4000006	174.36	Outdoor Play Equipment	Swing Sets	Pittsburgh	Pennsylvania	credit
9	2012-05-05	4000006	152.46	Jumping	Bungee Jumping	St. Petersburg	Florida	credit
8	2012-05-01	4000006	10.44	Winter Sports	Snowmobiling	Des Moines	Iowa	credit
7	2012-02-07	4000006	96.01	Outdoor Play Equipment	Sandboxes	Columbus	Ohio	credit
22	2011-10-10	4000009	19.64	Water Sports	Kitesurfing	Saint Paul	Minnesota	credit
26	2011-11-10	4000009	31.58	Combat Sports	Wrestling	Orange	California	credit
23	2011-02-05	4000009	99.5	Gymnastics	Gymnastics Rings	Springfield	Illinois	credit
25	2012-02-10	4000009	144.2	Indoor Games	Darts	Phoenix	Arizona	credit
12	2011-08-02	4000009	41.52	Indoor Games	Bowling	San Francisco	California	credit
11	2012-06-06	4000009	121.39	Outdoor Play Equipment	Swing Sets	Columbus	Ohio	credit
7	2012-02-07	4000006	96.01	Outdoor Play Equipment	Sandboxes	Columbus	Ohio	credit
8	2012-05-01	4000006	10.44	Winter Sports	Snowmobiling	Des Moines	Iowa	credit
9	2012-05-05	4000006	152.46	Jumping	Bungee Jumping	St. Petersburg	Florida	credit
11	2012-06-06	4000009	121.39	Outdoor Play Equipment	Swing Sets	Columbus	Ohio	credit
12	2011-08-02	4000009	41.52	Indoor Games	Bowling	San Francisco	California	credit
22	2011-10-10	4000009	19.64	Water Sports	Kitesurfing	Saint Paul	Minnesota	credit
23	2011-02-05	4000009	99.5	Gymnastics	Gymnastics Rings	Springfield	Illinois	credit
25	2012-02-10	4000009	144.2	Indoor Games	Darts	Phoenix	Arizona	credit
26	2011-11-10	4000009	31.58	Combat Sports	Wrestling	Orange	California	credit