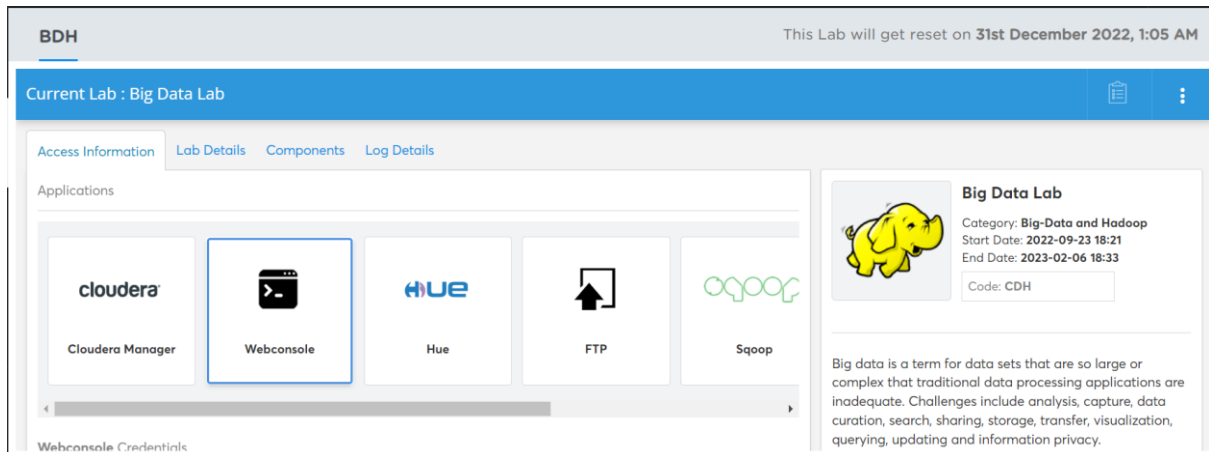


# YELLOW Taxi trip analysis using Hive

**First-Step:** Before Executing any queries inside HDFS, HIVE first we must prepare the Lab. In this project I have used Simplilearn big data cloudera distribution lab.



So from web-Console first login into the web console with proper user name and password.

Username: kaushikdey1984yahoo

Password : kaushikdey1984yahooodeqx

After that from web-console go to the hive shell with following screen-shot commands.

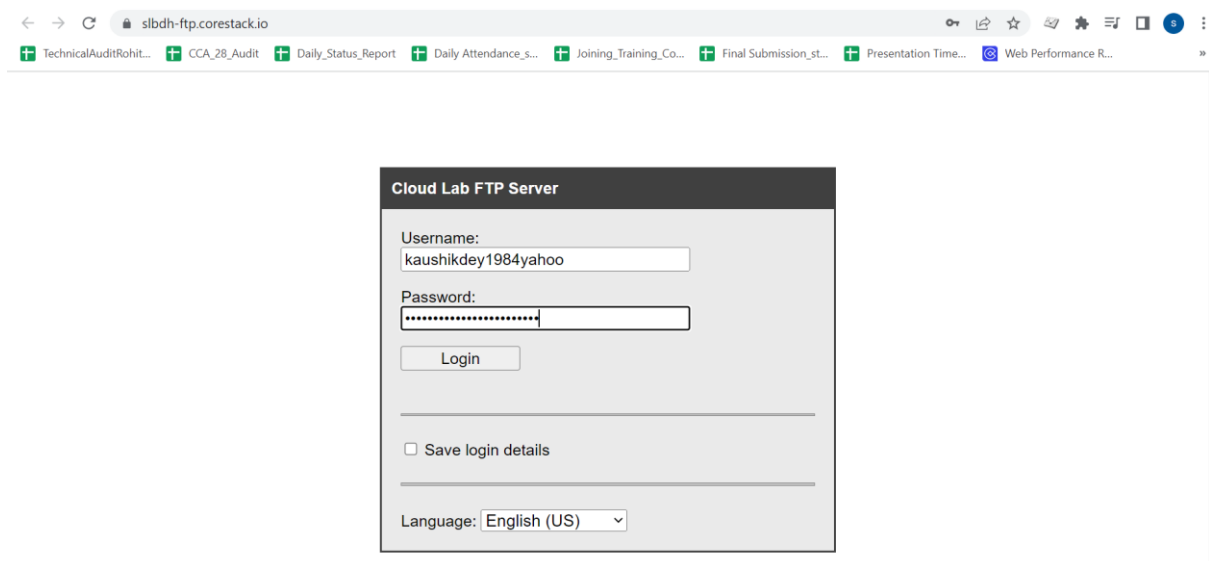
```
ip-10-0-41-79 login: kaushikdey1984yahoo
Password:
Last login: Tue Sep 27 15:44:28 on pts/1292
[kaushikdey1984yahoo@ip-10-0-41-79 ~]$ hive
WARNING: Use "yarn jar" to launch YARN applications.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/jars/log4j-slf4j-impl-2.8.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/jars/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]

Logging initialized using configuration in jar:file:/opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/jars/hive-common-2.1.1-cdh6.3.2.jar!/hive-log4j2.properties Async: false

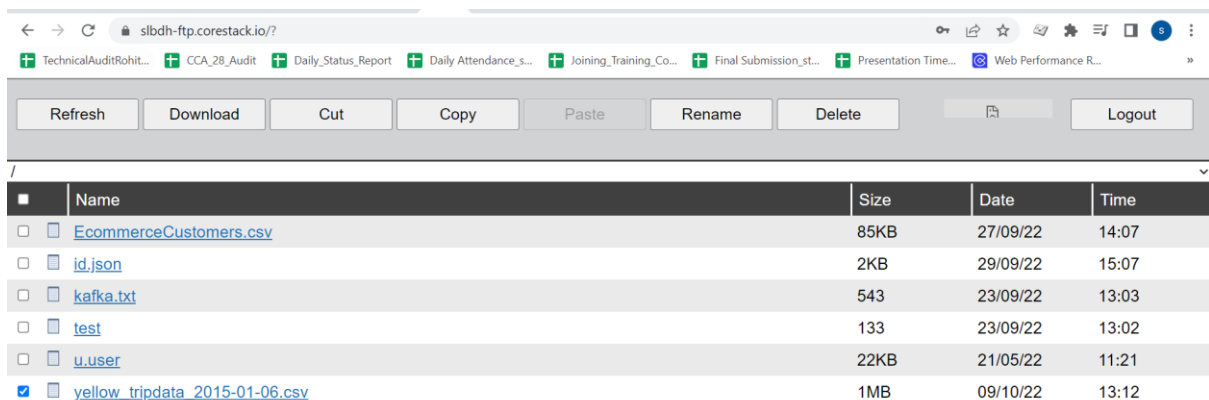
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> Create database mohita;
OK
Time taken: 2.519 seconds
hive> use mohita;
OK
Time taken: 0.064 seconds
hive> show tables;
OK
Time taken: 0.14 seconds
hive>
```

# YELLOW Taxi trip analysis using Hive

**Second-Step:** Load Yellow Taxi trip csv data from local system to hdfs via FTP.



Now the tick checkbox data is from yellow\_tripdata\_2015-01-06.csv.



We have to confirm that from the webconsole shell also. This is the following screen-shot.

```
ip-10-0-31-117 login: kaushikdey1984yahoo
Password:
[kaushikdey1984yahoo@ip-10-0-31-117 ~]$ pwd
/mnt/home/kaushikdey1984yahoo
[kaushikdey1984yahoo@ip-10-0-31-117 ~]$ ls
EcommerceCustomers.csv id.json kafka.txt test u.user yellow_tripdata_2015-01-06.csv
[kaushikdey1984yahoo@ip-10-0-31-117 ~]$ ls -ltr
total 1604
-rw-r--r-- 1 kaushikdey1984yahoo kaushikdey1984yahoo 22628 May 21 11:21 u.user
-rw-r--r-- 1 kaushikdey1984yahoo kaushikdey1984yahoo 133 Sep 23 13:02 test
-rw-r--r-- 1 kaushikdey1984yahoo kaushikdey1984yahoo 543 Sep 23 13:03 kafka.txt
-rw-r--r-- 1 kaushikdey1984yahoo kaushikdey1984yahoo 87360 Sep 27 14:07 EcommerceCustomers.csv
-rw-r--r-- 1 kaushikdey1984yahoo kaushikdey1984yahoo 1821 Sep 29 15:07 id.json
-rw-r--r-- 1 kaushikdey1984yahoo kaushikdey1984yahoo 1512490 Oct 9 13:12 yellow_tripdata_2015-01-06.csv
[kaushikdey1984yahoo@ip-10-0-31-117 ~]$
```

Now move this csv file from userpath to hdfs with following commands.

- 1) Hadoop fs -mkdir newDataFlair
- 2) Hadoop fs -put yellow\_tripdata\_2015-01-06.csv newDataFlair
- 3) Hadoop fs -ls newDataFlair

## YELLOW Taxi trip analysis using Hive

```
[kaushikdey1984yahoo@ip-10-0-41-79 ~]$ hadoop fs -ls
Found 5 items
drwxr-xr-x - kaushikdey1984yahoo hadoop 0 2022-09-24 14:00 .Trash
drwxr-xr-x - kaushikdey1984yahoo hadoop 0 2022-09-29 18:00 .sparkStaging
drwxr-xr-x - kaushikdey1984yahoo hadoop 0 2022-09-27 16:28 Desktop
drwxr-xr-x - kaushikdey1984yahoo hadoop 0 2022-09-29 15:09 newDataFlair
drwxr-xr-x - kaushikdey1984yahoo hadoop 0 2022-09-23 13:09 sample_101
[kaushikdey1984yahoo@ip-10-0-41-79 ~]$ hadoop fs -ls /newDataFlair
ls: '/newDataFlair': No such file or directory
[kaushikdey1984yahoo@ip-10-0-41-79 ~]$ hadoop fs -ls newDataFlair
Found 2 items
-rw-r--r-- 3 kaushikdey1984yahoo hadoop 87360 2022-09-27 16:02 newDataFlair/EcommerceCustomers.csv
-rw-r--r-- 3 kaushikdey1984yahoo hadoop 1821 2022-09-29 15:09 newDataFlair/id.json
[kaushikdey1984yahoo@ip-10-0-41-79 ~]$ hadoop fs -put yellow_tripdata_2015-01-06.csv newDataFlair
[kaushikdey1984yahoo@ip-10-0-41-79 ~]$ hadoop fs -ls newDataFlair
Found 3 items
-rw-r--r-- 3 kaushikdey1984yahoo hadoop 87360 2022-09-27 16:02 newDataFlair/EcommerceCustomers.csv
-rw-r--r-- 3 kaushikdey1984yahoo hadoop 1821 2022-09-29 15:09 newDataFlair/id.json
-rw-r--r-- 3 kaushikdey1984yahoo hadoop 1512490 2022-10-09 13:20 newDataFlair/yellow_tripdata_2015-01-06.csv
[kaushikdey1984yahoo@ip-10-0-41-79 ~]$
```

**Third-Step:** Now change the shell ( that is Hive) and execute the following commands:

- 1) Create database mohita;
- 2) Use mohita;
- 3) Hive>CREATE TABLE IF NOT EXISTS taxidata (vendor\_id string, pickup\_datetime string, dropoff\_datetime string, passenger\_count int, trip\_distance DECIMAL(9,6), pickup\_longitude DECIMAL(9,6), pickup\_latitude DECIMAL(9,6), rate\_code int, store\_and\_fwd\_flag string, dropoff\_longitude DECIMAL(9,6), dropoff\_latitude DECIMAL(9,6), payment\_type string, fare\_amount DECIMAL(9,6), extra DECIMAL(9,6), mta\_tax DECIMAL(9,6), tip\_amount DECIMAL(9,6), tolls\_amount DECIMAL(9,6), total\_amount DECIMAL(9,6), trip\_time\_in\_secs int ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED as TEXTFILE TBLPROPERTIES ("skip.header.line.count"="1");
- 4) Hive>Show tables;

```
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/jars/hive-common-2.1.1-cdh6.3.2.jar!/hive-log4j2.properties Async: false

WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> Create database mohita;
OK
Time taken: 2.519 seconds
hive> use mohita;
OK
Time taken: 0.064 seconds
hive> show tables;
OK
Time taken: 0.14 seconds
hive> CREATE TABLE IF NOT EXISTS taxidata (vendor_id string, pickup_datetime string, dropoff_datetime string, passenger_count int, trip_distance DECIMAL(9,6), pickup_longitude DECIMAL(9,6), pickup_latitude DECIMAL(9,6), rate_code int, store_and_fwd_flag string, dropoff_longitude DECIMAL(9,6), dropoff_latitude DECIMAL(9,6), payment_type string, fare_amount DECIMAL(9,6), extra DECIMAL(9,6), mta_tax DECIMAL(9,6), tip_amount DECIMAL(9,6), tolls_amount DECIMAL(9,6), total_amount DECIMAL(9,6), trip_time_in_secs int ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED as TEXTFILE TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.193 seconds
hive> show tables;
OK
taxidata
Time taken: 0.045 seconds, Fetched: 1 row(s)
hive>
```

Now we have to load data from hdf5 to hive shell.

- 5) Hive> LOAD DATA INPATH 'newDataFlair/yellow\_tripdata\_2015-01-06.csv' INTO TABLE taxidata;

Now we have to check that from Hive shell & Hue Editor. ( It is from UI SIDE)

- 6) Hive> select \* from mohita.taxidata LIMIT 5;

# YELLOW Taxi trip analysis using Hive

ScreenShot from Hive Shell;

```
Time taken: 0.023 seconds
hive> SELECT * from taxidata LIMIT 5;
OK
2      2015-01-08 22:44:09      2015-01-08 22:50:56      1      1.550000      -73.987686      40.724251      1      N      -73.973763      40.743378
2      7.500000      0.500000      0.500000      0.000000      0.000000      8.800000      5000
1      2015-01-08 22:44:09      2015-01-08 22:51:17      3      1.200000      -73.991570      40.726933      1      N      -74.004105      40.721081
2      7.000000      0.500000      0.500000      0.000000      0.000000      8.300000      5344860
1      2015-01-08 22:44:10      2015-01-08 22:55:27      1      2.400000      -73.981918      40.783443      1      N      -73.952354      40.798199
2      10.500000      0.500000      0.500000      0.000000      0.000000      11.800000      3345464
1      2015-01-08 22:44:10      2015-01-08 22:58:09      1      7.300000      -73.973122      40.743553      1      N      -73.919571      40.832001
2      21.500000      0.500000      0.500000      0.000000      0.000000      22.800000      893933
1      2015-01-08 22:44:12      2015-01-08 22:46:16      1      0.400000      -73.982948      40.766209      1      N      -73.984390      40.764053
2      3.500000      0.500000      0.500000      0.000000      0.000000      4.800000      36864
Time taken: 0.212 seconds, Fetched: 5 row(s)
hive> SELECT * from mohita.taxidata LIMIT 5;
OK
2      2015-01-08 22:44:09      2015-01-08 22:50:56      1      1.550000      -73.987686      40.724251      1      N      -73.973763      40.743378
2      7.500000      0.500000      0.500000      0.000000      0.000000      8.800000      5000
1      2015-01-08 22:44:09      2015-01-08 22:51:17      3      1.200000      -73.991570      40.726933      1      N      -74.004105      40.721081
2      7.000000      0.500000      0.500000      0.000000      0.000000      8.300000      5344860
1      2015-01-08 22:44:10      2015-01-08 22:55:27      1      2.400000      -73.981918      40.783443      1      N      -73.952354      40.798199
2      10.500000      0.500000      0.500000      0.000000      0.000000      11.800000      3345464
1      2015-01-08 22:44:10      2015-01-08 22:58:09      1      7.300000      -73.973122      40.743553      1      N      -73.919571      40.832001
2      21.500000      0.500000      0.500000      0.000000      0.000000      22.800000      893933
1      2015-01-08 22:44:12      2015-01-08 22:46:16      1      0.400000      -73.982948      40.766209      1      N      -73.984390      40.764053
2      3.500000      0.500000      0.500000      0.000000      0.000000      4.800000      36864
Time taken: 0.063 seconds, Fetched: 5 row(s)
hive>
```

ScreenShot from Hue Editor;

The screenshot shows the Hue Editor interface. The query editor contains the query: `SELECT * from mohita.taxidata LIMIT 5;`. The query history shows the query was executed successfully. The results pane displays the following data:

taxidata.vendor_id	taxidata.pickup_datetime	taxidata.dropoff_datetime	taxidata.passenger_count	taxidata.trip_di
1 2	2015-01-08 22:44:09	2015-01-08 22:50:56	1	1.550000
2 1	2015-01-08 22:44:09	2015-01-08 22:51:17	3	1.200000
3 1	2015-01-08 22:44:10	2015-01-08 22:55:27	1	2.400000
4 1	2015-01-08 22:44:10	2015-01-08 22:58:09	1	7.300000
5 1	2015-01-08 22:44:12	2015-01-08 22:46:16	1	0.400000

**Fourth-Step:** in Fourth step we have to analysis the following questions & answers.

**Perform taxi trip analysis by solving the questions below:**

1. What is the total Number of trips ( equal to the number of rows )?

SELECT count(\*) from mohita.taxidata;

Output : 10000

# YELLOW Taxi trip analysis using Hive

The screenshot shows the Hive interface with a query executed: `SELECT count(*) from taxidata;`. The result is 10000. The interface includes a sidebar with a table list, a central query editor and results pane, and a right-hand pane showing table schemas.

Table	Schema
mohita.taxidata	vendor_id string, pickup_datetime string, dropoff_datetime string, passenger_count int, trip_distance decimal(9,6), pickup_longitude decimal(9,6), pickup_latitude decimal(9,6), rate_code int, store_and_fwd_flag string, dropoff_longitude decimal(9,6), dropoff_latitude decimal(9,6), payment_type string, fare_amount decimal(9,6), extra decimal(9,6), mta_tax decimal(9,6), tip_amount decimal(9,6), tolls_amount decimal(9,6), total_amount decimal(9,6), trip_time_in_secs int

2) What is the total revenue generated by all the trips? The fare is stored in the column total\_amount.

Ans : `SELECT sum(total_amount) as total_revenue from taxidata;`

Output : 160546.810000

The screenshot shows the Hive interface with a query executed: `SELECT sum(total_amount) as total_revenue from taxidata;`. The result is 160546.810000. The interface is similar to the first screenshot, showing the query editor, results pane, and table schemas.

Table	Schema
mohita.taxidata	vendor_id string, pickup_datetime string, dropoff_datetime string, passenger_count int, trip_distance decimal(9,6), pickup_longitude decimal(9,6), pickup_latitude decimal(9,6), rate_code int, store_and_fwd_flag string, dropoff_longitude decimal(9,6), dropoff_latitude decimal(9,6), payment_type string, fare_amount decimal(9,6), extra decimal(9,6), mta_tax decimal(9,6), tip_amount decimal(9,6), tolls_amount decimal(9,6), total_amount decimal(9,6), trip_time_in_secs int

3) What fraction of the total is paid for tolls? The toll is stored in tolls\_amount.

Ans : `SELECT sum(tolls_amount) / sum(total_amount) from taxidata;`

Output : 0.0155530340341237549348

# YELLOW Taxi trip analysis using Hive

The screenshot shows the Hue interface with a Hive query executed. The query is:

```
1 SELECT count(*) from taxidata;  
2 SELECT sum(total_amount) as total_revenue from taxidata;  
3 SELECT sum(tolls_amount) / sum(total_amount) from taxidata;
```

The execution log shows the following information:

```
INFO : Total MapReduce CPU Time Spent: 7 seconds 438 ms  
INFO : Completed executing command(queryId=hive_20221007101706_000000_0000_1000_0  
fff-9b08e8d7623b); Time taken: 18.867 seconds  
INFO : OK
```

The results table shows one row with the value 0.0155530340341237549348.

Tables	Statement 3/3
Filter...	
taxidata	
vendor_id	string
pickup_datetime	string
dropoff_datetime	string
passenger_count	int
trip_distance	decimal(9,6)
pickup_longitude	decimal(9,6)
pickup_latitude	decimal(9,6)
rate_code	int
store_and_fwd_flag	string
dropoff_longitude	decimal(9,6)
dropoff_latitude	decimal(9,6)
payment_type	string
fare_amount	decimal(9,6)
extra	decimal(9,6)
mta_tax	decimal(9,6)
tip_amount	decimal(9,6)
tolls_amount	decimal(9,6)
total_amount	decimal(9,6)
trip_time_in_secs	int

- 4) What fraction of it is driver tips? The tip is stored in tip\_amount.  
Ans : `SELECT sum(tip_amount) / sum (total_amount) from taxidata;`  
Output : 0.1078520339332808917225

The screenshot shows the Hue interface with a Hive query executed. The query is:

```
1 SELECT count(*) from taxidata;  
2 SELECT sum(total_amount) as total_revenue from taxidata;  
3 SELECT sum(tolls_amount) / sum (total_amount) from taxidata;  
4 SELECT sum(tip_amount) / sum (total_amount) from taxidata;
```

The execution log shows the following information:

```
INFO : Total MapReduce CPU Time Spent: 7 seconds 80 msc  
INFO : Completed executing command(queryId=hive_20221007102247_24000000_0000_1000_0  
a75-0854b148bf99); Time taken: 17.812 seconds  
INFO : OK
```

The results table shows one row with the value 0.1078520339332808917225.

Tables	Statement 4/4
Filter...	
taxidata	
vendor_id	string
pickup_datetime	string
dropoff_datetime	string
passenger_count	int
trip_distance	decimal(9,6)
pickup_longitude	decimal(9,6)
pickup_latitude	decimal(9,6)
rate_code	int
store_and_fwd_flag	string
dropoff_longitude	decimal(9,6)
dropoff_latitude	decimal(9,6)
payment_type	string
fare_amount	decimal(9,6)
extra	decimal(9,6)
mta_tax	decimal(9,6)
tip_amount	decimal(9,6)
tolls_amount	decimal(9,6)
total_amount	decimal(9,6)
trip_time_in_secs	int

- 5) What is the average trip total amount?  
Ans : `SELECT avg(total_amount) as trip_amount from taxidata;`  
Output : 16.0546810000

The screenshot displays the Hive CLI environment. At the top, there's a search bar and navigation icons. On the left sidebar, various databases like 'amazon\_reviews\_table' and 'emp' are listed. The main area shows a SQL query being executed:

```

1 SELECT count(*) from taxidata;
2 DESCRIBE taxidata;
3 SELECT sum(total_amount) as total_revenue from taxidata;
4 SELECT sum(tolls_amount) /sum (total_amount) from taxidata;
5 SELECT sum(tip_amount) / sum (total_amount) from taxidata;
6 SELECT avg(total_amount) as trip_amount from taxidata;
7

```

The execution log indicates successful completion:

```

INFO : Total MapReduce CPU Time Spent: 6 seconds 640 ms
INFO : Completed executing command(queryId=hive_202210071710007030-1100-1200)
d42-951df532ad30); Time taken: 33.186 seconds
INFO : OK

```

The results section shows a single entry for 'trip\_amount':

trip_amount
1 16.0546810000

- Ans : `SELECT payment_type, avg(fare_amount) as average_fare,  
avg(tip_amount) as average_tip,  
avg(mta_tax) as average_tax  
from taxidata GROUP BY payment_type;`  
output :

payment_type		average_fare	average_tip	average_tax
payment_type		average_fare	average_tip	average_tax
1	1	13.5610182727	2.7042480087	0.4971107293
2	2	11.3933830986	0.0000000000	0.4988732394
3	3	13.2107894737	0.0000000000	0.4210526316
4	4	12.2222222222	0.0000000000	0.5000000000

# YELLOW Taxi trip analysis using Hive

The screenshot shows the Hive console interface. On the left is a sidebar with a 'Tables' list containing various database tables like 'amazon\_reviews\_table', 'emp1', 'emp2', etc. The main area displays a SQL query and its execution results. The query is as follows:

```
1 SELECT count(*) from taxidata;
2 DESCRIBE taxidata;
3 SELECT sum(total_amount) as total_revenue from taxidata;
4 SELECT sum(tolls_amount) / sum (total_amount) from taxidata;
5 SELECT sum(tip_amount) / sum (total_amount) from taxidata;
6 SELECT avg(total_amount) as trip_amount from taxidata;
7 SELECT payment_type, avg(fare_amount) as average_fare, avg(tip_amount) as average_tip,
8 avg(eta_tax) as average_tax
9 from taxidata GROUP BY payment_type;
```

The execution log shows the query completed successfully. The results are displayed in a table with 4 columns: payment\_type, average\_fare, average\_tip, and average\_tax.

payment_type	average_fare	average_tip	average_tax
1	13.5610182727	2.7042480087	0.4971107293
2	11.3933830986	0.0000000000	0.4988732394
3	13.2107894737	0.0000000000	0.4210526316
4	12.2222222222	0.0000000000	0.5000000000

- 7) On an average which hour of the day generates the highest revenue?  
 Ans : select h24 as hour,  
 avg(total\_amount) as avg\_revenue from (select hour(pickup\_datetime) as  
 h24,  
 total\_amount from taxidata) ff group by h24 order by avg\_revenue desc;

The screenshot shows the Hive console interface. The main area displays a SQL query and its execution results. The query is as follows:

```
1 SELECT count(*) from taxidata;
2 DESCRIBE taxidata;
3 SELECT sum(total_amount) as total_revenue from taxidata;
4 SELECT sum(tolls_amount) / sum (total_amount) from taxidata;
5 SELECT sum(tip_amount) / sum (total_amount) from taxidata;
6 SELECT avg(total_amount) as trip_amount from taxidata;
7 SELECT payment_type, avg(fare_amount) as average_fare, avg(tip_amount) as average_tip,
8 avg(eta_tax) as average_tax
9 from taxidata GROUP BY payment_type;
10
11 select h24 as hour,
12 avg(total_amount) as avg_revenue from (select hour(pickup_datetime) as h24,
13 total_amount from taxidata) ff group by h24 order by avg_revenue desc;
```

The execution log shows the query completed successfully. The results are displayed in a table with 2 columns: hour and avg\_revenue.

hour	avg_revenue
1	22
2	23
3	0

- 8) What is the average distance of the trips? Distance is stored in the column trip\_distance.  
 Ans : SELECT avg(trip\_distance) as avg\_distance from taxidata;  
 Output: 3.2530330000



# YELLOW Taxi trip analysis using Hive

[illegible]

9) How many different payment types are used? Column name – payment\_type. select distinct payment\_type from taxidata

Ans : `SELECT distinct payment_type from taxidata;`

Output :

[illegible]