Logic for Final Submission

1. **Configurations to run these analytical queries**

   A. **Enabling *SSH port 22* and *Custom TCP port 888* for Master/Core Security Groups to access *Hue User Interface* to write these analytical queries.**

Inbound rules (21)

Manage tags | Edit inbound rules

Filter security group rules

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
–	sgr-09568b1baf494b4fd	IPv4	Custom TCP	TCP	8443	72.21.198.64/29	–
–	sgr-0f2bc75c24db4a7db	–	All TCP	TCP	0 - 65535	sg-0ab99caeb73a442...	–
–	sgr-0a6d5f274763b9d7f	IPv4	Custom TCP	TCP	8082	0.0.0.0/0	–
–	sgr-09a390321d8a67...	IPv4	Custom TCP	TCP	8443	54.240.217.16/29	–
–	sgr-0b59c7c3a4e2a56dd	–	All UDP	UDP	0 - 65535	sg-0ab99caeb73a442...	–
–	sgr-0ba71e7f046e969db	IPv4	Custom TCP	TCP	8443	207.171.167.26/32	–
–	sgr-015ce3334768875...	–	All UDP	UDP	0 - 65535	sg-0c230da81897f91d...	–
–	sgr-0d0b2951ad8625...	IPv4	Custom TCP	TCP	8443	72.21.217.0/24	–
–	sgr-0881b45505e4e3...	IPv4	SSH	TCP	22	171.76.82.183/32	–
–	sgr-04a11d487205eb...	IPv4	Custom TCP	TCP	8443	54.240.217.8/29	–
–	sgr-0f618e0f32d2c84d0	IPv4	Custom TCP	TCP	8443	207.171.172.6/32	–
–	sgr-01264ff212f11f0c6	–	All ICMP - IPv4	ICMP	All	sg-0ab99caeb73a442...	–
–	sgr-02defb23325f6f08f	IPv4	Custom TCP	TCP	8888	171.76.82.183/32	–
–	sgr-0ff9a49abfae5f9ec	–	All TCP	TCP	0 - 65535	sg-0c230da81897f91d...	–
–	sgr-01085add762391...	–	All ICMP - IPv4	ICMP	All	sg-0c230da81897f91d...	–

   B. **Accessing Hue interface URL from "Application user interface" tab.**

Summary | Application user interfaces | Monitoring | Hardware | Configurations | Events | Steps | Bootstrap actions

Persistent application user interfaces

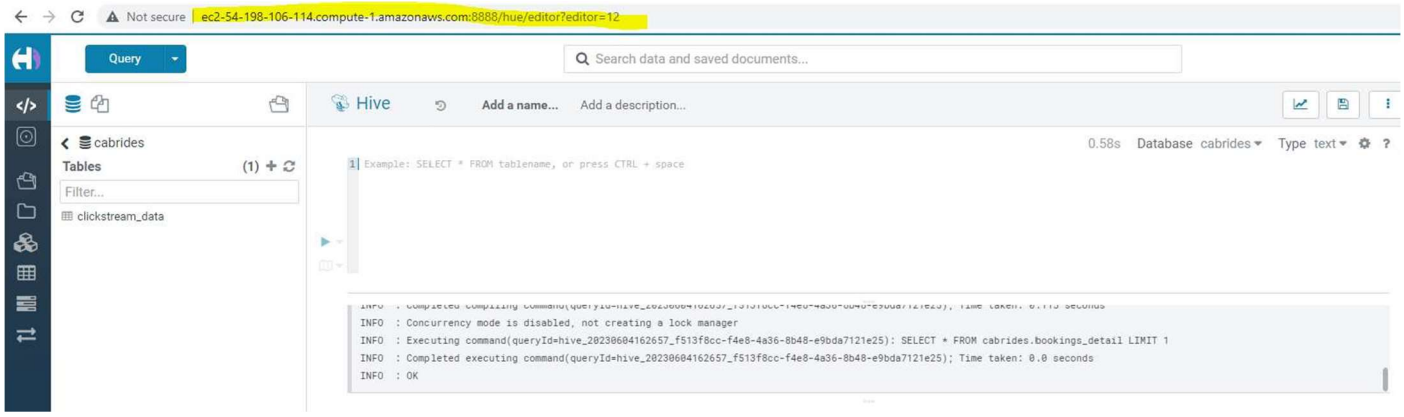
Applications installed on the Amazon EMR cluster publish user interfaces (UI) as web sites to monitor cluster activity. Persistent UI logs are available for 30 days after an applica cluster.

Application user interface
YARN timeline server
Tez UI
Spark history server

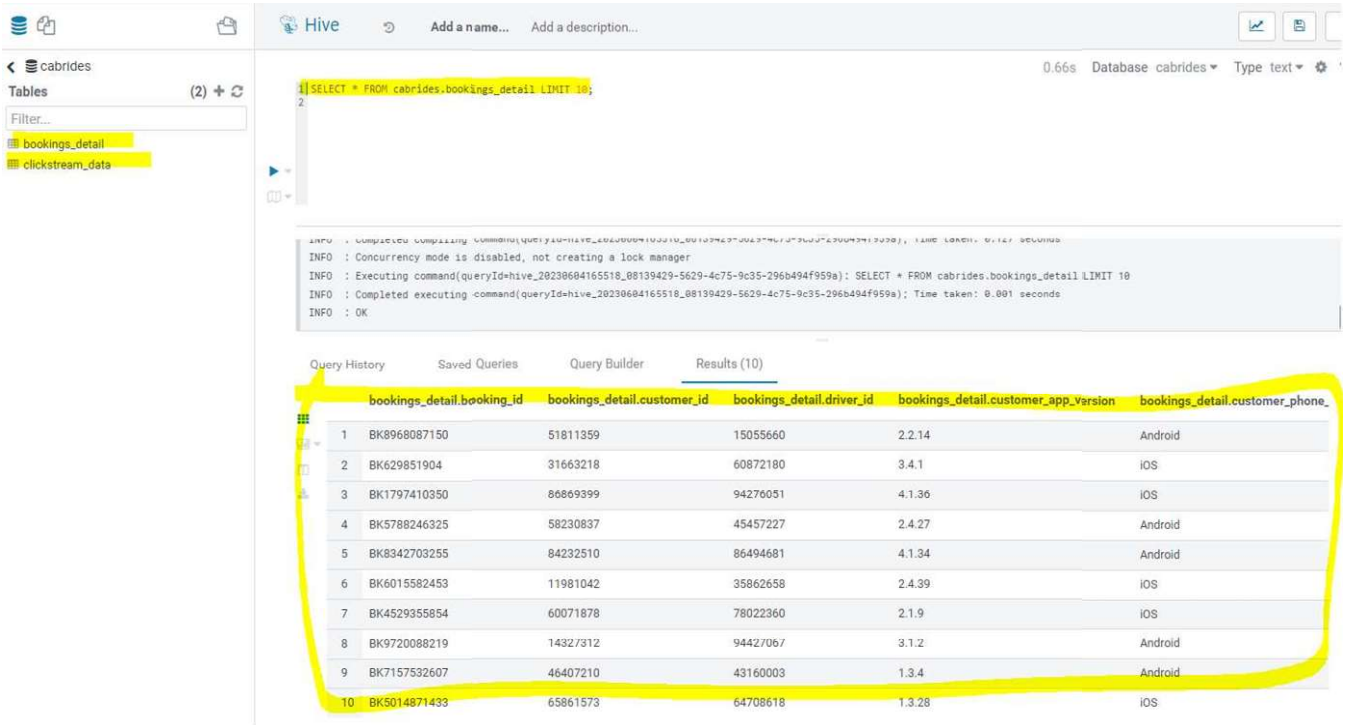
On-cluster application user interfaces

On-cluster UI are available only while clusters are running. Because they are hosted on the master node, on-cluster UI require a connection via SSH tunneling. Set up SSH tunn

Application	User interface URL
HDFS Name Node	http://ec2-54-198-106-114.compute-1.amazonaws.com:50070/
Hue	http://ec2-54-198-106-114.compute-1.amazonaws.com:8888/
Spark History Server	http://ec2-54-198-106-114.compute-1.amazonaws.com:18080/
Resource Manager	http://ec2-54-198-106-114.compute-1.amazonaws.com:8088/



C. Verifying that Hue UI is connected to Hive and able to access created Bookings and Streaming tables.



	bookings_detail.booking_id	bookings_detail.customer_id	bookings_detail.driver_id	bookings_detail.customer_app_version	bookings_detail.customer_phone_
1	BK8968087150	51811359	15055660	2.2.14	Android
2	BK629851904	31663218	60872180	3.4.1	iOS
3	BK1797410350	86869399	94276051	4.1.36	iOS
4	BK5788246325	58230837	45457227	2.4.27	Android
5	BK8342703255	84232510	86494681	4.1.34	Android
6	BK6015582453	11981042	35862658	2.4.39	iOS
7	BK4529355854	60071878	78022360	2.1.9	iOS
8	BK9720088219	14327312	94427067	3.1.2	Android
9	BK7157532607	46407210	43160003	1.3.4	Android
10	BK5014871433	65861573	64708618	1.3.28	iOS

2. Queries with output and explanation:

Task 5: Calculate the total number of different drivers for each customer.

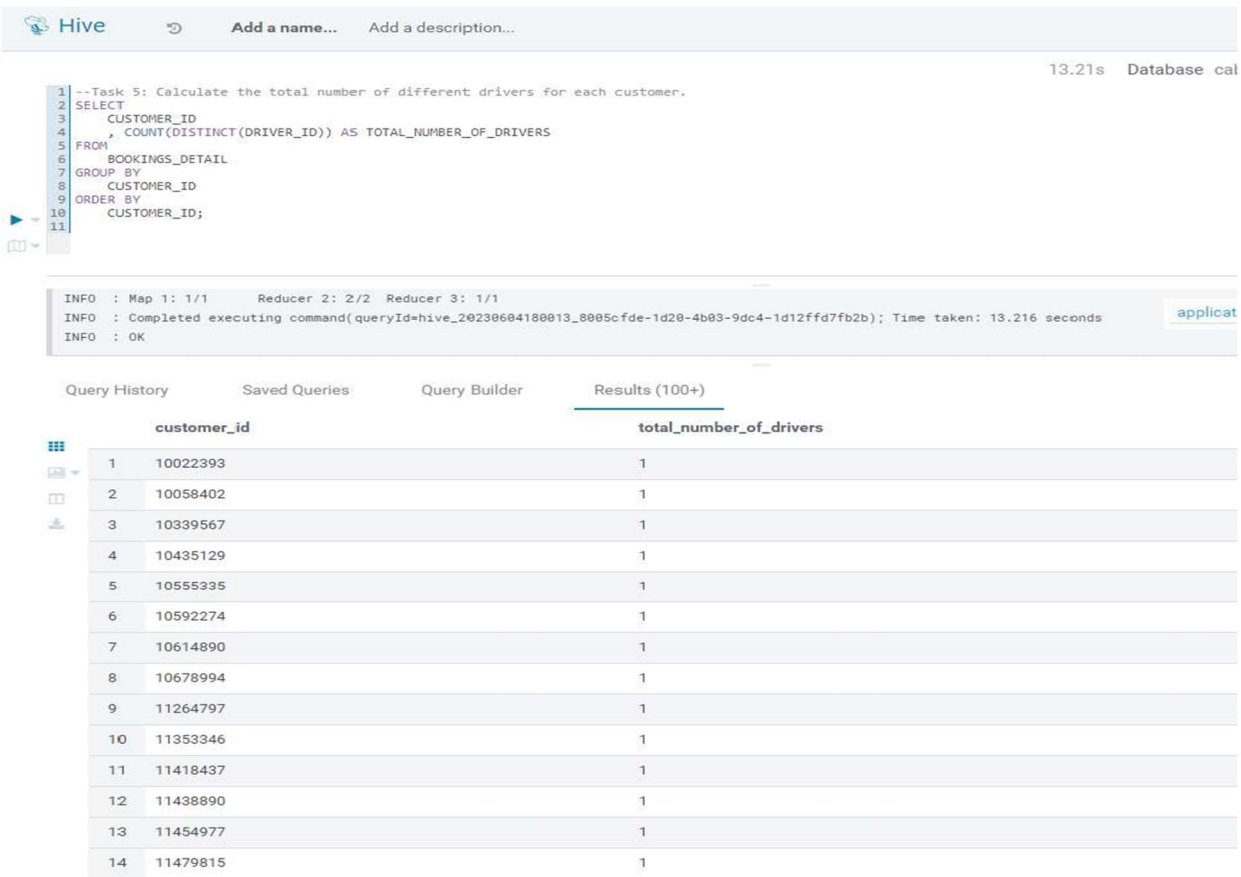
Query:-

```
SELECT
        CUSTOMER_ID
        , COUNT(DISTINCT(DRIVER_ID)) AS TOTAL_NUMBER_OF_DRIVERS
FROM
        BOOKINGS_DETAIL
GROUP BY
        CUSTOMER_ID
ORDER BY
        CUSTOMER_ID;
```

Explanation: -

1. Counting unique drivers (driver_id) grouped by Customers (costomer_id) give an insight that how many times a customer taken ride with the same driver. This may help to find a pattern like favorite/loved rides, a driver drive cabin a specific area during specific time etc..
2. Here order by clause shows customers in older to newer order (customer_id in ascending order), assuming that Customer ID is auto generated in OLTP system and allocated based on account created date.

Output:



	customer_id	total_number_of_drivers
1	10022393	1
2	10058402	1
3	10339567	1
4	10435129	1
5	10555335	1
6	10592274	1
7	10614890	1
8	10678994	1
9	11264797	1
10	11353346	1
11	11418437	1
12	11438890	1
13	11454977	1
14	11479815	1

Validation:

```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-11-17 12:23:06,034 Stage-1 map = 0%,  reduce = 0%
2020-11-17 12:23:12,394 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.27 sec
2020-11-17 12:23:20,727 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.69 sec
MapReduce Total cumulative CPU time: 7 seconds 690 msec
Ended Job = job_1605615116654_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.69 sec   HDFS Read: 43007 HDFS Write: 11000 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 690 msec
OK
10022393        1
10058402        1
10339567        1
10435129        1
10555335        1
10592274        1
10614890        1
10678994        1
11264797        1
11353346        1
11418437        1
11438890        1
11454977        1
11479815        1
11518953        1
11580321        1
11596512        1
11608791        1
11655671        1
11757536        1
11764909        1
11860278        1
11981042        1
12106105        1
12142182        1
12312603        1
12334699        1
12367832        1
12856708        1
12885363        1
12913608        1
12914577        1
12966909        1
13015449        1
13229062        1
```

Note: Expected output is exactly matching with output given in the validation document.

Task 6: Calculate the total rides taken by each customer.

Query:-

```
SELECT
        CUSTOMER_ID
        , COUNT(BOOKING_ID) AS TOTAL_RIDES
FROM
        BOOKINGS_DETAIL
GROUP BY
        CUSTOMER_ID;
ORDER BY
        CUSTOMER_ID;
```

Explanation: -

1. Counting bookings (booking_id) ***GROUPED BY*** Customers (costomer_id) give an insight that how many rides arebooked by each customer.
2. This could help to find out most valuable customers for the company and company could roll out offer for such frequent rides to generated more revenue and customer clustering can be done based on no of rides booked/taken.
3. Here order by clause shows customers in older to newer order (customer_id in ascending order), assuming thatCustomer ID is auto generated in OLTP system and allocated based on account created date.

Output:

Hive ☆ Add a name... Add a description...

6.36s Dat

```
--Task 6: Calculate the total rides taken by each customer.
SELECT
    CUSTOMER_ID
    , COUNT(BOOKING_ID) AS TOTAL_RIDES
FROM
    BOOKINGS_DETAIL
GROUP BY
    CUSTOMER_ID;
ORDER BY
    CUSTOMER_ID;
```

```
INFO  : Map 1: 1/1      Reducer 2: 2/2
INFO  : Completed executing command(queryId=hive_20230604180210_6c863202-4127-4cb2-99e5-d34bc232f2a3); Time taken: 6.35 seconds
INFO  : OK
```

Query History | Saved Queries | Query Builder | Results (100+)

	customer_id	total_rides
1	10022393	1
2	10555335	1
3	10592274	1
4	10678994	1
5	11264797	1
6	11418437	1
7	11438890	1
8	11518953	1
9	11580321	1
10	11764909	1
11	11860278	1
12	12312603	1
13	12334699	1
14	12367832	1

ook/editor?type=hive

Validation:

```
Ended Job = job_1605615116654_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.65 sec   HDFS Read: 38721 HDFS Write: 11000 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 650 msec
OK
10022393        1
10058402        1
10339567        1
10435129        1
10555335        1
10592274        1
10614890        1
10678994        1
11264797        1
11353346        1
11418437        1
11438890        1
11454977        1
11479815        1
11518953        1
11580321        1
11596512        1
11608791        1
11655671        1
11757536        1
11764909        1
11860278        1
11981042        1
12106105        1
12142182        1
12312603        1
```

Note: Expected output is exactly matching with output given in the validation document.

Task 7: Find the total visits made by each customer on the booking page and the total 'Book Now' button presses. This canshow the conversion ratio.
The booking page id is 'e7bc5fb2-1231-11eb-adc1-0242ac120002'.
The Book Now button id is 'fcba68aa-1231-11eb-adc1-0242ac120002'. You also need to calculate the conversion ratio as partof this task.
Conversion ratio can be calculated as **Total 'Book Now' Button Press/Total Visits made by customer on the booking page**.

Query:-

```
SELECT
        SUM(CASE WHEN PAGE_ID = 'e7bc5fb2-1231-11eb-adc1-0242ac120002' THEN 1 ELSE 0 END) AS TOTAL_PAGE_VISITS,
        SUM(CASE WHEN BUTTON_ID = 'fcba68aa-1231-11eb-adc1-0242ac120002' THEN   1 ELSE 0 END) AS TOTAL_BUTTON_PRESSED,
        ROUND(CAST(SUM(CASE WHEN BUTTON_ID = 'fcba68aa-1231-11eb-adc1-0242ac120002'
        THEN 1 ELSE 0 END) AS FLOAT) /
        CAST(SUM(CASE WHEN PAGE_ID = 'e7bc5fb2-1231-11eb-adc1-0242ac120002' THEN 1 ELSE
        0 END) AS FLOAT), 4) AS CONVERSION_RATIO
FROM CLICKSTREAM_DATA;
```

Explanation: -

1. This analysis can help to view customer's behavior like when they visit on booking page how many times they book ride.
2. Counting page visit events (PAGE_ID = 'e7bc5fb2-1231-11eb-adc1-0242ac120002') gives total page visits.
3. Counting booking events (BUTTON_ID = 'fcba68aa-1231-11eb-adc1-0242ac120002') gives total rides booked.
4. Conversion ratio is nothing but ***total bookings divided by total page visit***.
5. Conversion ratio is a critical KPI for the company which indicates that there is a high probability of booking if any customer visits on booking page white is ***~98%*** in this case.

Output:-

Validation:-

3. When you run the query to get the conversion ratio, you should get the conversion ratio as **0.9688.**

Note: Slightly difference in conversion ratio that is because ~16 more event were captured from kafkastream kept open for ~10 minutes.

Task 8: Calculate the count of all trips done on black cabs.

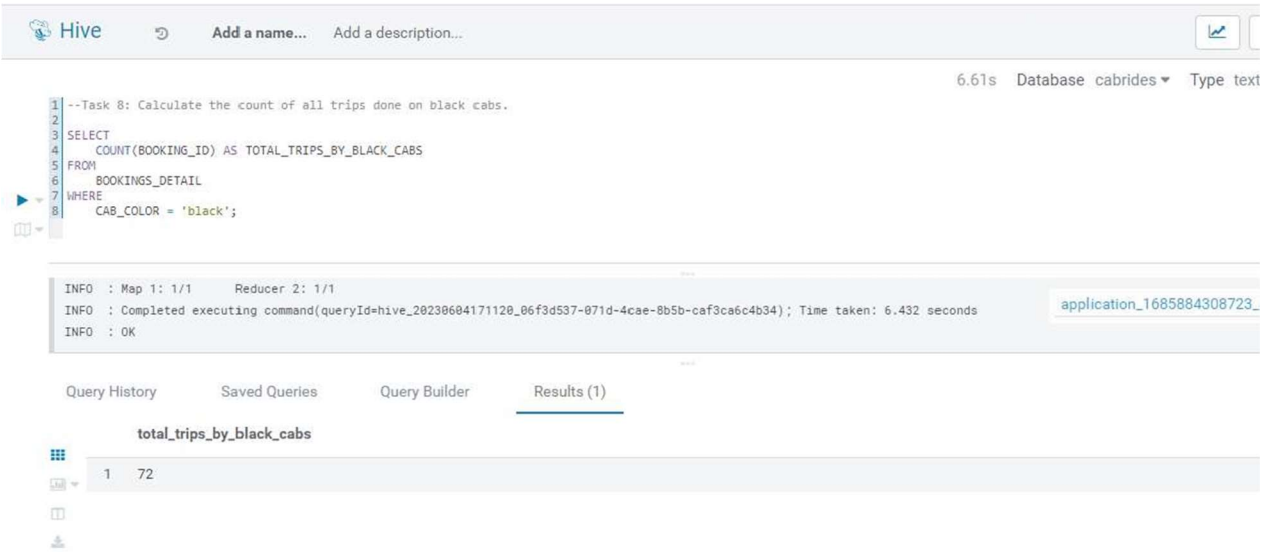
Query:-

```
SELECT
        COUNT(BOOKING_ID) AS TOTAL_TRIPS_BY_BLACK_CABS
FROM
        BOOKINGS_DETAIL
WHERE
        CAB_COLOR = black;
```

Explanation: -

1. This analysis helps to identify total trips done by ***Black*** cabs.
2. Further grouping by ***CAB_COLOR*** can give total number or percentage of rides done by cabs with specific color.
3. This can unhide any hidden booking pattern with cab color like customer prefer to book cab with any specific color.

Output:



Validation:

3. When you run the query to get the conversion ratio, you should get the conversion ratio as **0.9688.**
4. Count of all trips done on black cabs - **72.**
5. When you run the query to get the total amount of tips given date wise to all drivers by customers, you would get an output as shown below:

Note: Number of trips done by the black cabs is exactly matching with count given in the validation document.

Task 9: Calculate the total amount of tips given date wise to all drivers by customers.

Query:-

```
SELECT
        DATE(PICKUP_TIMESTAMP) TRIP_DATE
        , ROUND(SUM(TIP_AMOUNT),0) AS TOTAL_TIP_AMOUNT
FROM
        BOOKINGS_DETAIL
GROUP BY
        DATE(PICKUP_TIMESTAMP)
ORDER BY
        TRIP_DATE;
```

Explanation: -

1. Date function extract date only value from ***datetime_stamp*** value and returns as ***Pickup Date***.
2. ***SUM*** function with ***GROUP BY*** Pickup Date is used to get total tip amount by Pickup date.
3. ***ORDER BY*** clause with Pickup Date alias is used to show output in ascending pickup date
4. This analysis could help to understand if customers give more tips on any specific occasion or any specific day.
5. Management can rollout offers for customer/driver based on this analysis because looks happy on specific day ifthey give more tip amount.

Output:

Hive Add a name... Add a description...

6.60s Database cabr

```
--Task 9: Calculate the total amount of tips given date wise to all drivers by customers.
SELECT
    DATE(PICKUP_TIMESTAMP) TRIP_DATE
    , ROUND(SUM(TIP_AMOUNT),0) AS TOTAL_TIP_AMOUNT
FROM
    BOOKINGS_DETAIL
GROUP BY
    DATE(PICKUP_TIMESTAMP)
ORDER BY
    TRIP_DATE;
```

```
INFO  : Map 1: 1/1      Reducer 2: 2/2  Reducer 3: 1/1
INFO  : Completed executing command(queryId=hive_20230604175432_a2240e14-5494-48d1-ad8d-2bb17209c5b0); Time taken: 6.591 seconds
INFO  : OK
```

applicatio

Query History Saved Queries Query Builder Results (100+)

	trip_date	total_tip_amount
1	2020-01-01	59
2	2020-01-02	95
3	2020-01-03	11
4	2020-01-04	123
5	2020-01-05	134
6	2020-01-06	189
7	2020-01-07	148
8	2020-01-08	111
9	2020-01-09	48
10	2020-01-10	77
11	2020-01-11	81
12	2020-01-12	109
13	2020-01-14	142
14	2020-01-15	338
15	2020-01-16	155

Validation:

2020-01-01	59
2020-01-02	95
2020-01-03	11
2020-01-04	123
2020-01-05	134
2020-01-06	189
2020-01-07	148
2020-01-08	111
2020-01-09	48
2020-01-10	77
2020-01-11	81
2020-01-12	109
2020-01-14	142
2020-01-15	338
2020-01-16	155
2020-01-17	296
2020-01-18	240
2020-01-20	210
2020-01-21	5
2020-01-23	148
2020-01-24	472
2020-01-25	98
2020-01-26	209
2020-01-27	231
2020-01-28	567
2020-01-29	123
2020-01-30	112
2020-01-31	256
2020-02-01	317
2020-02-02	338
2020-02-03	191
2020-02-04	258
2020-02-05	212
2020-02-06	154
2020-02-07	91
2020-02-08	270

Note: Total amount of tips by pickup date is exactly matching with output given in the validation document.

Task 10: Calculate the total count of all the bookings with ratings lower than 2 as given by customers in a particularmonth.

Query:-

```
SELECT
        DATE_FORMAT(PICKUP_TIMESTAMP, 'yyyy-MM') TRIP_MONTH
        , COUNT(BOOKING_ID) AS NO_OF_BOOKINGS
FROM
        BOOKINGS_DETAIL
WHERE
        RATING_BY_CUSTOMER < 2
GROUP BY
        DATE_FORMAT(PICKUP_TIMESTAMP, 'yyyy-MM')
ORDER BY
        TRIP_MONTH;
```

Explanation: -

1. ***DATE_FORMAT*** function formats datetimestamp value in the specified format like yyyy-MM in this case whichresults like 2023-06.
2. ***WHERE*** clause is used to filter bookings where rating given by customers is ***less than 2*** which indicates customers dissatisfaction.
3. ***ORDER BY*** clause with Trip month alias is used to show output in ascending order of pickup month.
4. This analysis could help to understand number of trips by month where customers were not happy.
5. Also could give insight or a hidden pattern in dissatisfactory rides in a specific month or period which could be n number of factors like low rating because of AC was not on during summary time, cab reached late on pickup pointdue to traffic on a rainy day/season etc.
6. Based on this analysis, instructions can be given to driver to make customers happy and take care of things whichcould lead to low customer rating.

Output:

Hive Add a name... Add a description...

6.59s Database

```
--Task 10: Calculate the total count of all the bookings with ratings lower than 2 as given by customers in a particular month.

SELECT
    DATE_FORMAT(PICKUP_TIMESTAMP, 'yyyy-MM') TRIP_MONTH
    , COUNT(BOOKING_ID) AS NO_OF_BOOKINGS
FROM
    BOOKINGS_DETAIL
WHERE
    RATING_BY_CUSTOMER < 2
GROUP BY
    DATE_FORMAT(PICKUP_TIMESTAMP, 'yyyy-MM')
ORDER BY
    TRIP_MONTH;
```

```
INFO  : Map 1: 1/1      Reducer 2: 2/2  Reducer 3: 1/1
INFO  : Completed executing command(queryId=hive_20230604182704_fc36cbb6-1096-4692-ad8f-93f842f89e84); Time taken: 6.583 seconds
INFO  : OK
```

appl

Query History Saved Queries Query Builder Results (10)

	trip_month	no_of_bookings
1	2020-01	26
2	2020-02	16
3	2020-03	16
4	2020-04	21
5	2020-05	21
6	2020-06	14
7	2020-07	20
8	2020-08	32
9	2020-09	21
10	2020-10	15

Validation:

```
Total MapReduce CPU Time Spent: 7 seconds 970 msec
OK
2020-01 26
2020-02 16
2020-03 16
2020-04 21
2020-05 21
2020-06 14
2020-07 20
2020-08 32
2020-09 21
2020-10 15
```

Note: Count of bookings with low customer rating by month is exactly matching with output given in thevalidation document.

Task 11: Calculate the count of total iOS users.

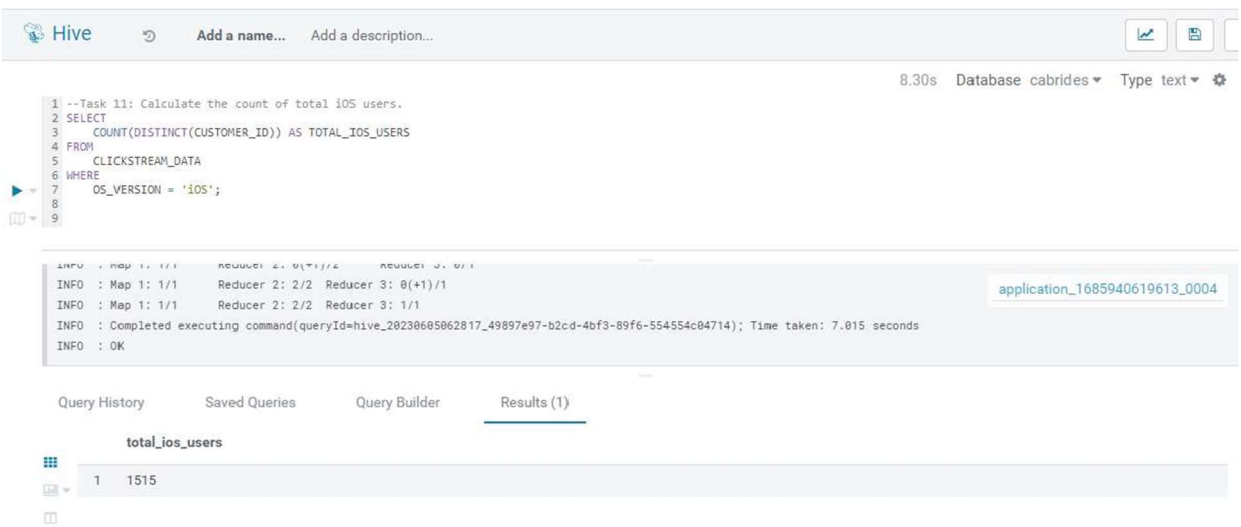
Query:-

```
SELECT
        COUNT(DISTINCT(CUSTOMER_ID)) AS TOTAL_IOS_USERS
FROM
        CLICKSTREAM_DATA
WHERE
        OS_VERSION = 'iOS';
```

Explanation: -

1. ***COUNT*** function ***DISTINCT*** is used to get unique customers who are using ***iOS*** devices.
2. ***WHERE*** clause is used to filter events which are generated out of ***iOS*** devices.
3. ***ORDER BY*** clause with Trip month alias is used to show output in ascending order of pickup month.
4. This analysis gives a high level insight that how many or percentage of customers using specific type of device or OS.
5. Like if company makes any update on iOS and Androids mobile app then how many customers would be impactedand so on.

Output:-



Validation:

7. You should get the count of all iOS users as **1503**.

Note: 3004 event were captured from kafka stream where in validation documents its 2984 hence iOS userare slightly more in our analysis.