Queries

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

Query:

```
SELECT
   customer_id,
   COUNT(DISTINCT driver_id) AS Distinct_no_of_drivers
FROM    bookings_data
GROUP BY    customer_id
ORDER BY    customer_id;
```

- This SQL query examines the bookings_data table's customer-driver interactions. It determines how many distinct drivers are connected to every customer. The query makes sure that each customer's records are aggregated independently by classifying the data according to customer_id. The number of different drivers each client has come across is then determined using the COUNT(DISTINCT driver_id) function.
- Lastly, a clear and structured representation of the number of drivers who served each client is provided by sorting the results by customer_id in ascending order. Understanding patterns of driver distribution and client involvement can be aided by this kind of information.

hadoop@ip-10-0-2-56:~

```
hive> SET hive.cli.print.header=true;
hive> SELECT
    >     customer_id,
    >     COUNT(DISTINCT driver_id) AS Distinct_no_of_drivers
    > FROM
    >     bookings_data
    > GROUP BY
    >     customer_id
    > ORDER BY
    >     customer_id;
Query ID = hadoop_20250625190454_497935d6-3596-4227-a0cc-4ea53751bca1
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1750873936410_0011)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 5.67 s
--------------------------------------------------------------------------------
OK
customer_id     distinct_no_of_drivers
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
```

hadoop@ip-10-0-2-56:~

```
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 5.67 s
--------------------------------------------------------------------------------
OK
customer_id     distinct_no_of_drivers
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
13262795        1
13356177        1
13387493        1
```

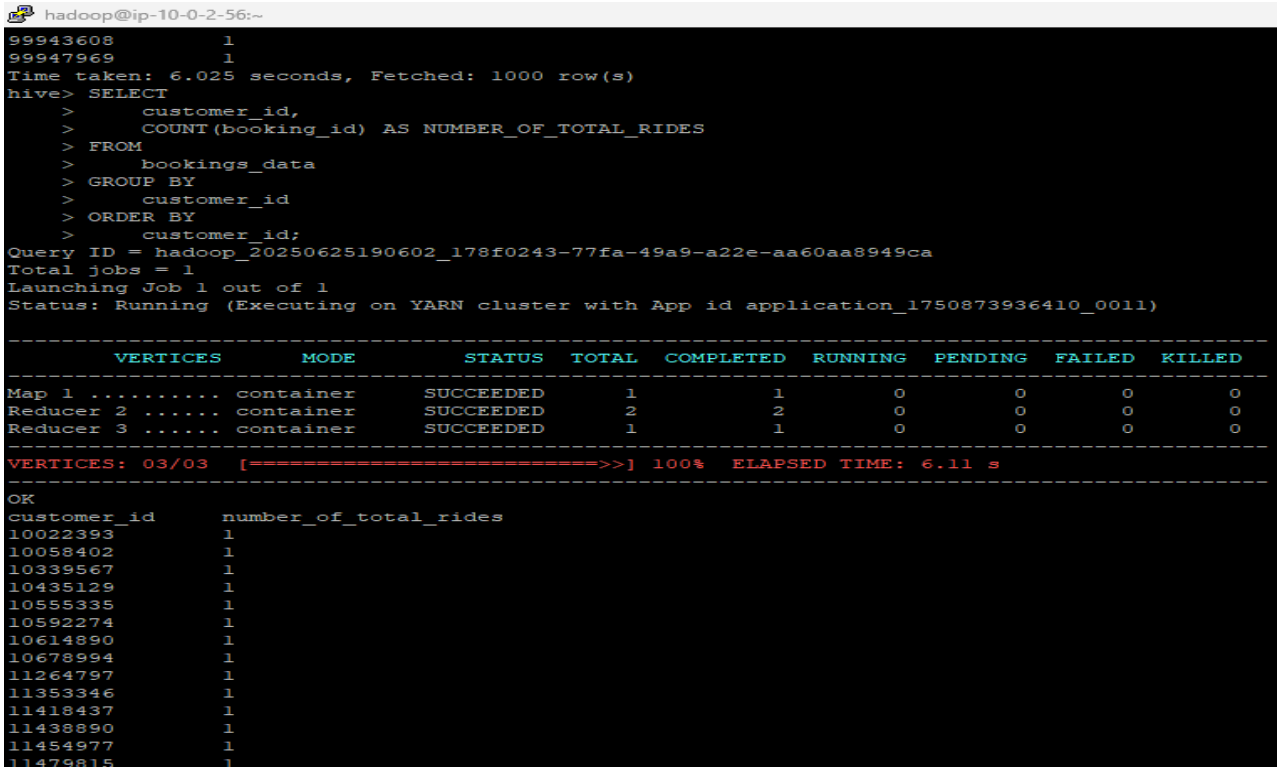
The following Output matches validation document.

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

Query:

```
SELECT
   customer_id,
   COUNT(booking_id) AS NUMBER_OF_TOTAL_RIDES
FROM    bookings_data
GROUP BY    customer_id
ORDER BY    customer_id;
```

- Using the bookings_data table, this SQL query determines how many rides each client has taken overall. After choosing the customer_id, it counts how many booking_id entries there are linked to each customer.
- The query organises all reservations under each distinct customer by utilising the GROUP BY clause on customer_id. A summary of the number of rides each customer has completed is provided by the outcome.
- It is simple to examine the total number of rides for every client in an orderly fashion because the final output is sorted by customer_id.



```
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 6.11 s
--------------------------------------------------------------------------------------------
OK
customer_id     number_of_total_rides
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
13262795        1
```

The above Output matched 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 can show 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 part of this task. Conversion ratio can be calculated as Total 'Book Now' Button Press/Total Visits made by customer on the booking page.

Query:

```
SELECT
  COUNT(CASE
    WHEN page_id = 'e7bc5fb2-1231-11eb-adc1-0242ac120002'
    AND is_page_view = 'Yes' THEN 1
```

```
    END) AS total_page_visits,

    COUNT(CASE
        WHEN button_id = 'fcba68aa-1231-11eb-adc1-0242ac120002'
        AND is_button_click = 'Yes' THEN 1
    END) AS total_button_pressed,

    ROUND(
        COUNT(CASE
            WHEN button_id = 'fcba68aa-1231-11eb-adc1-0242ac120002'
            AND is_button_click = 'Yes' THEN 1
        END) * 1.0 /
        COUNT(CASE
            WHEN page_id = 'e7bc5fb2-1231-11eb-adc1-0242ac120002'
            AND is_page_view = 'Yes' THEN 1
        END), 4
    ) AS conversion_ratio
FROM clickstream_data;
```

- The purpose of this SQL query is to assess user interaction effectiveness and engagement on a particular webpage.
- It uses the clickstream_data table to compute three important metrics. It first counts the records where the given page_id was viewed and tagged as 'Yes' in the is_page_view field in order to calculate the overall number of visits to a specific page.
- Second, it counts the number of times a particular button on that page was clicked; this is verified by the is_button_click field being set to 'Yes' and is recognised by a unique button_id.
- Lastly, by dividing the total number of button clicks by the total number of page visits and rounding the result to four decimal places, the query determines the conversion ratio.
- This conversion ratio offers important insights into user behaviour and the performance of the page design by showing how well the page drives user activities.

```
hadoop@ip-10-0-2-56:~
99943608        1
99947969        1
Time taken: 6.485 seconds, Fetched: 1000 row(s)
hive> SELECT
    >     COUNT(CASE
    >         WHEN page_id = 'e7bc5fb2-1231-11eb-adc1-0242ac120002'
    >         AND is_page_view = 'Yes' THEN 1
    >     END) AS total_page_visits,
    >
    >     COUNT(CASE
    >         WHEN button_id = 'fcba68aa-1231-11eb-adc1-0242ac120002'
    >         AND is_button_click = 'Yes' THEN 1
    >     END) AS total_button_pressed,
    >
    >     ROUND(
    >         COUNT(CASE
    >             WHEN button_id = 'fcba68aa-1231-11eb-adc1-0242ac120002'
    >             AND is_button_click = 'Yes' THEN 1
    >         END) * 1.0 /
    >         COUNT(CASE
    >             WHEN page_id = 'e7bc5fb2-1231-11eb-adc1-0242ac120002'
    >             AND is_page_view = 'Yes' THEN 1
    >         END), 4
    >     ) AS conversion_ratio
    > FROM clickstream_data;
Query ID = hadoop_20250625190919_1e2dd756-5549-4a65-a8a3-dfd0eebd4336
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1750873936410_0011)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED     10         10        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 12.83 s
--------------------------------------------------------------------------------
OK
total_page_visits       total_button_pressed    conversion_ratio
436010  422418  0.9688
Time taken: 13.416 seconds, Fetched: 1 row(s)
hive>
```

We got the same conversion ratio of 0.9688 as we have in validation document.

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

Query:

```
WITH black_cabs AS (
   SELECT *
   FROM bookings_data
   WHERE cab_color = 'black'
)
SELECT  COUNT(booking_id) AS TOTAL_TRIPS_BY_BLACK_CABS
FROM black_cabs;
```

- Using the bookings_data table, this SQL query determines the total number of trips made by black taxis.

- A Common Table Expression (CTE) called black_cabs is initially created, filtering and storing any records with the cab_color set to 'black'. The query then counts the total number of booking_id entries—which indicate the total number of trips made in black taxis—using this filtered dataset.
- The total number of trips made by black taxis is the only value displayed in the final result. This method assists in separating particular data subsets for targeted investigation.

```
OK
total_page_visits       total_button_pressed    conversion_ratio
436010  422418  0.9688
Time taken: 13.416 seconds, Fetched: 1 row(s)
hive> WITH black_cabs AS (
    >     SELECT *
    >     FROM bookings_data
    >     WHERE cab_color = 'black'
    > )
    >
    > SELECT
    >     COUNT(booking_id) AS TOTAL_TRIPS_BY_BLACK_CABS
    > FROM black_cabs;
Query ID = hadoop_20250625191221_5e695501-ddf9-4e22-b0c0-6aec0ac87e42
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1750873936410_0011)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 5.73 s
--------------------------------------------------------------------------------
OK
total_trips_by_black_cabs
72
Time taken: 6.062 seconds, Fetched: 1 row(s)
hive>
```

We got 72 Black Cab trips same as Validation documentation.

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

Query:

```
WITH daily_tips AS (
  SELECT
    TO_DATE(from_unixtime(CAST(pickup_timestamp / 1000 AS BIGINT))) AS TRIP_DATE,
    SUM(tip_amount) AS SUM_TIPS
  FROM
    bookings_data
  GROUP BY
    TO_DATE(from_unixtime(CAST(pickup_timestamp / 1000 AS BIGINT)))
```

)

```
SELECT
   TRIP_DATE,
   ROUND(SUM_TIPS, 0) AS TOTAL_TIP_AMOUNT
FROM
   daily_tips
ORDER BY
   TRIP_DATE;
```

- The bookings_data table's total daily tip amounts are determined using this SQL query.
- In order to transform each trip's timestamp from Unix time (in milliseconds) to a readable date format (TRIP_DATE), it first creates a Common Table Expression (CTE) named daily_tips. After that, the query groups the records by TRIP_DATE and adds up the tip amounts for each day.
- The query then rounds the total tips to the closest whole number for easier reading after choosing the travel dates and associated total tip amounts from the daily_tips CTE. Next, TRIP_DATE is used to arrange the results chronologically.
- This query offers insights into tipping trends over time by summarising the total tips collected by day.

```
hadoop@ip-10-0-2-56:~
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: fals
hive> SET hive.cli.print.header=true;
hive> WITH daily_tips AS (
    >     SELECT
    >         TO_DATE(from_unixtime(CAST(pickup_timestamp / 1000 AS BIGINT))) AS TRIP_DATE,
    >         SUM(tip_amount) AS SUM_TIPS
    >     FROM
    >         bookings_data
    >     GROUP BY
    >         TO_DATE(from_unixtime(CAST(pickup_timestamp / 1000 AS BIGINT)))
    > )
    >
    > SELECT
    >     TRIP_DATE,
    >     ROUND(SUM_TIPS, 0) AS TOTAL_TIP_AMOUNT
    > FROM
    >     daily_tips
    > ORDER BY
    >     TRIP_DATE;
Query ID = hadoop_20250625192402_f6aecefa-669e-49be-b31d-a89221b8270b
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1750873936410_0017)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........  container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......  container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ......  container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 5.85 s
----------------------------------------------------------------------------------------------
OK
trip_date       total_tip_amount
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
```

```
hadoop@ip-10-0-2-56:~
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 5.85 s
----------------------------------------------------------------------------------------------
OK
trip_date       total_tip_amount
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
2020-02-09      266
2020-02-10      115
```

The following Output matches the validation documentation.

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

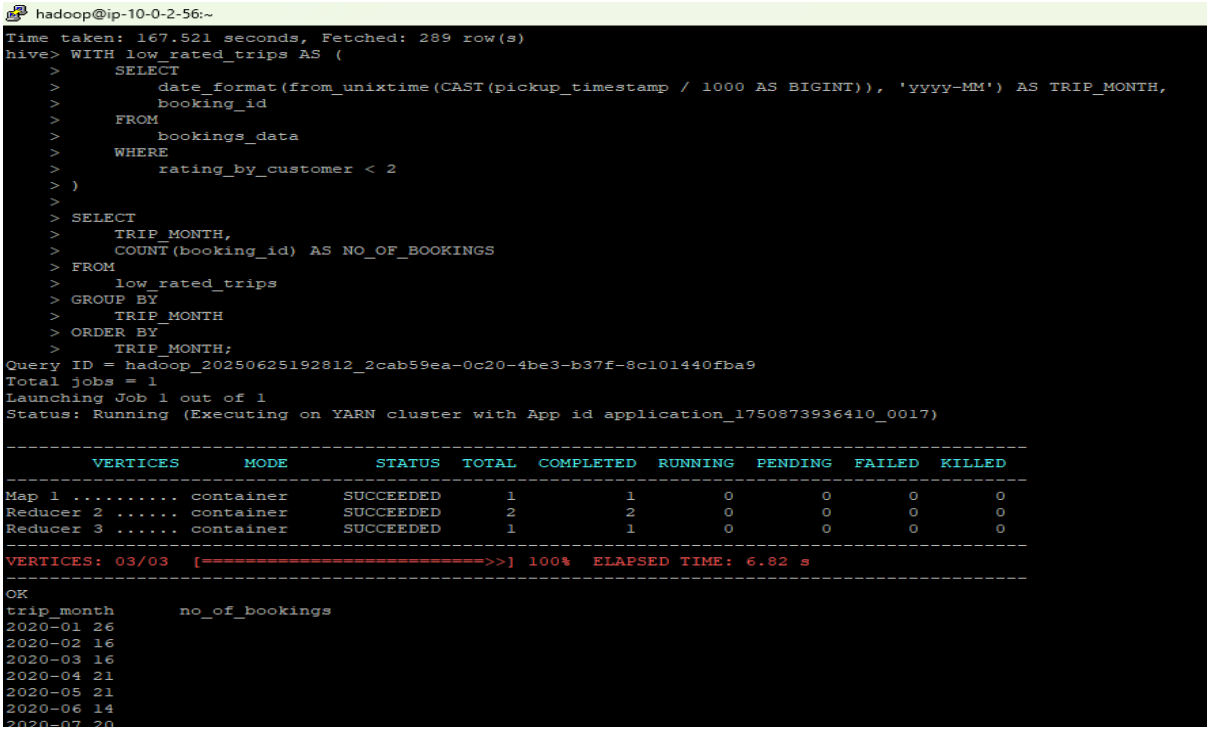
Query:

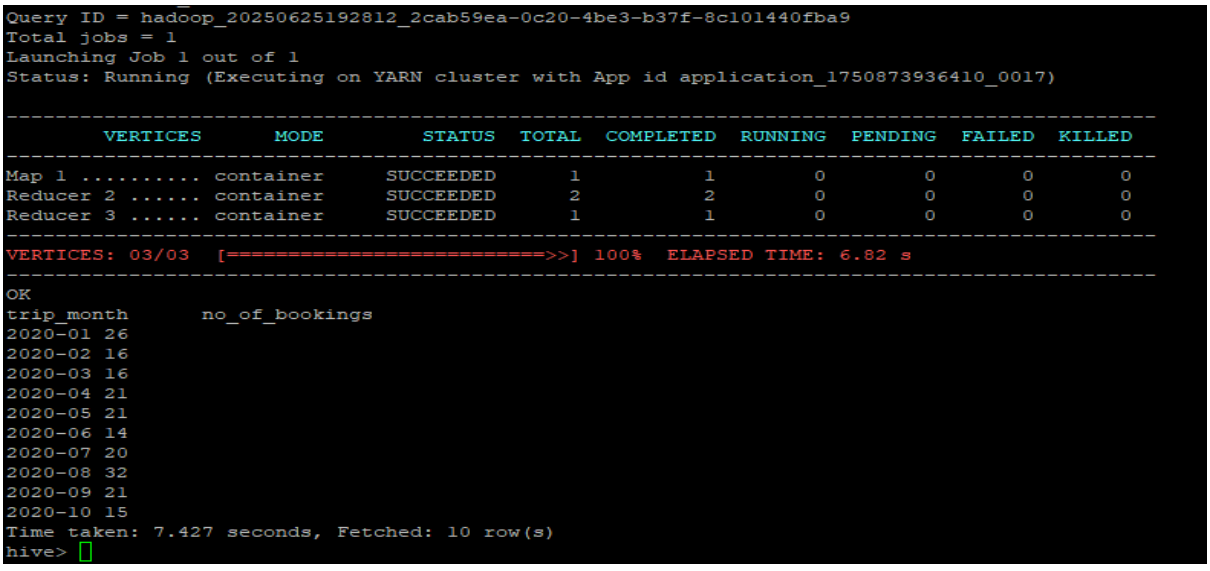
```
 WITH low_rated_trips AS (
   SELECT
      date_format(from_unixtime(CAST(pickup_timestamp / 1000 AS BIGINT)), 'yyyy-MM')
AS TRIP_MONTH,
      booking_id
   FROM
      bookings_data
   WHERE
      rating_by_customer < 2
)

SELECT
   TRIP_MONTH,
   COUNT(booking_id) AS NO_OF_BOOKINGS
FROM
   low_rated_trips
GROUP BY
   TRIP_MONTH
ORDER BY
   TRIP_MONTH;
```

- This SQL query is designed to analyze low-rated trips on a monthly basis from the bookings_data table.
- It starts by creating a Common Table Expression (CTE) named low_rated_trips, which filters trips where the rating_by_customer is less than 2, indicating poorly rated rides. For each of these trips, it extracts the trip month by converting the Unix timestamp (in

milliseconds) to a readable yyyy-MM format and selects the corresponding booking_id.

- In the main query, it groups these low-rated trips by TRIP_MONTH and counts the number of bookings for each month, providing the total number of low-rated trips per month. The results are ordered chronologically by month.
- This query helps track monthly trends in customer dissatisfaction, which can be valuable for identifying periods of poor service and potential areas for operational improvement.





The above output matches the Validation document.

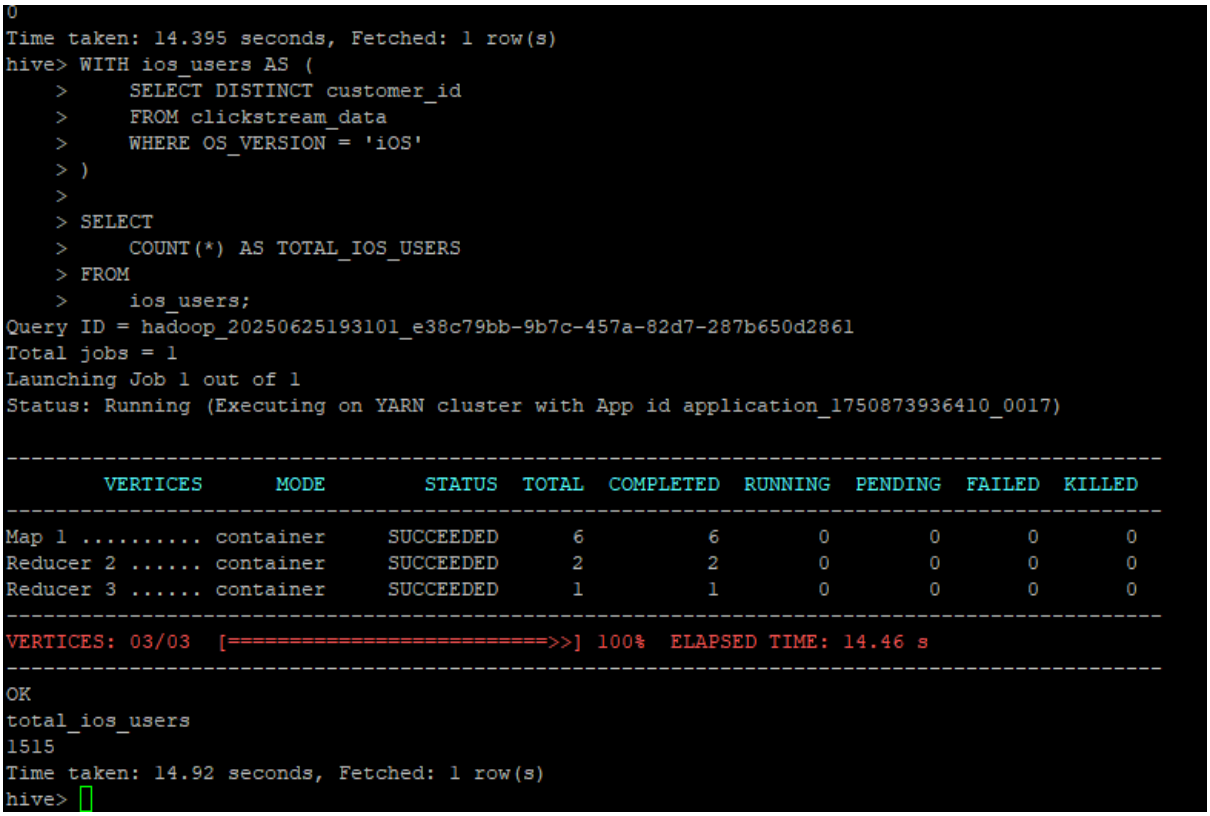
Task 11: Calculate the count of total iOS users.

Query:

```
WITH ios_users AS (
   SELECT DISTINCT customer_id
   FROM clickstream_data
   WHERE OS_VERSION = 'iOS'
)

SELECT
   COUNT(*) AS TOTAL_IOS_USERS
FROM
   ios_users;
```

- The clickstream_data dataset is used in this SQL query to determine the total number of distinct iOS users.
- A Common Table Expression (CTE) named ios_users is first created, and it picks all unique customer_id values where the OS_VERSION is 'iOS'. Even if an iOS user appears in the dataset more than once, this step guarantees that they are only counted once.
- In the last step, the query returns TOTAL_IOS_USERS, which is just the total number of distinct iOS users found in the CTE.
  For platform-specific user analysis or marketing campaigns aimed at iOS consumers, this query offers a clear and precise count of individual iOS users.

In the Output below we got 2 counts extra than the output from validation document. Looking more into this we found that since we got a little more of data ingested we got those 2 counts extra.



All the tasks from 1 to 11 are now successfully performed.