## Logic For Final Submission

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

**select customer\_id , count(distinct driver\_id) from booking\_data group by customer\_id order by customer\_id asc;**

This query will group the results by customer\_id and then order the output by customer\_id in ascending order.

Explanation of each part:

SELECT customer\_id: This specifies the column customer\_id that you want to retrieve in the results.

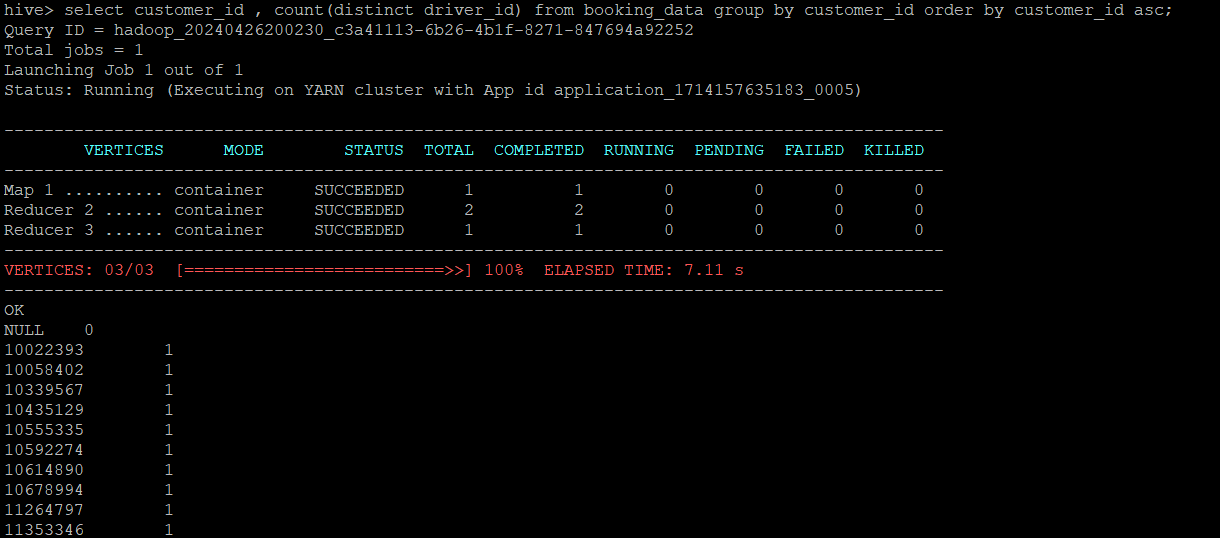
COUNT(DISTINCT driver\_id) AS num\_unique\_drivers: This calculates the count of distinct driver\_id values for each customer\_id group. The alias num\_unique\_drivers gives a name to this computed value in the result set.

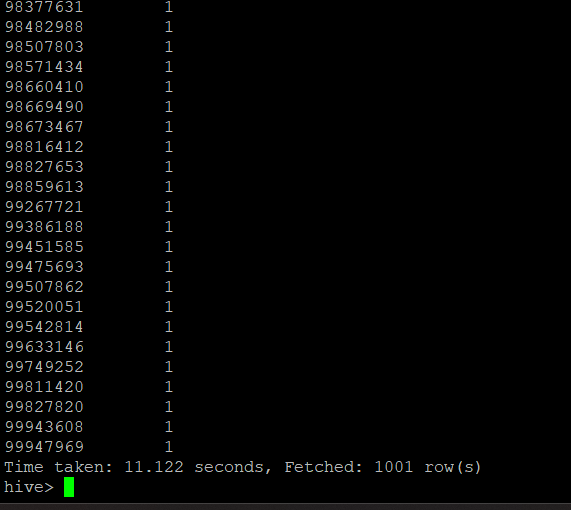
FROM booking\_data: This specifies the table booking\_data from which the data is being queried.

GROUP BY customer\_id: This groups the rows of the booking\_data table by customer\_id.

ORDER BY customer\_id ASC: This orders the result set by customer\_id in ascending order (ASC).

screenshot:





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

**select customer\_id ,count(distinct booking\_id) from booking\_data group by customer\_id order by customer\_id asc;**

This query will output a list of customer\_id values along with the count of distinct booking\_id values associated with each customer, with the results ordered by customer\_id. Each row in the output will represent a unique customer and the number of distinct bookings they have made.

Explanation of each part:

SELECT customer\_id, COUNT(DISTINCT booking\_id): This part of the query selects two columns:

customer\_id: The unique identifier for each customer.

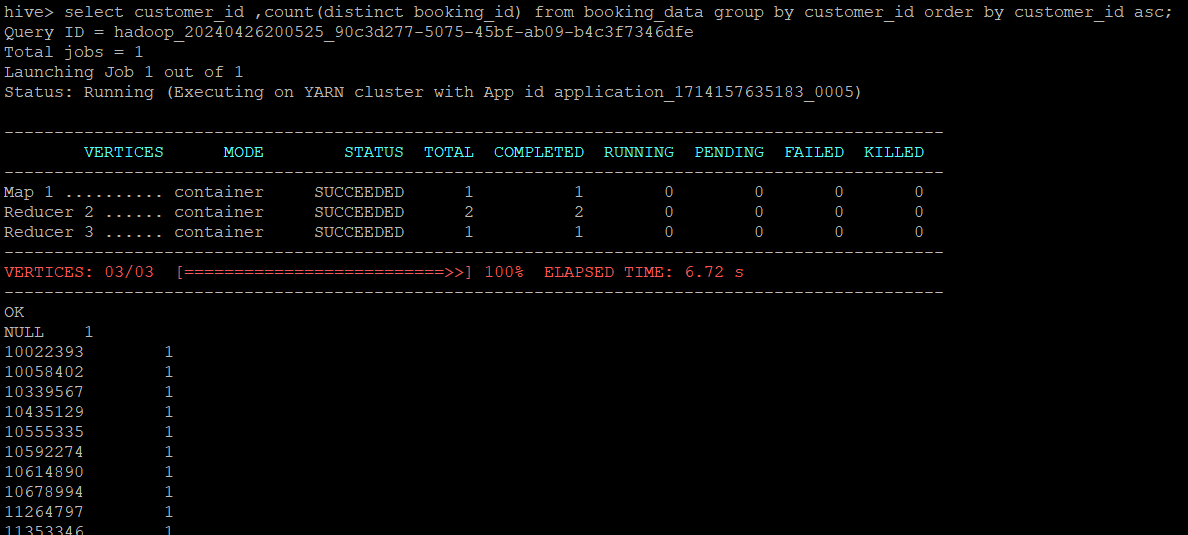
COUNT(DISTINCT booking\_id): Calculates the number of distinct booking\_id values associated with each customer\_id. Using COUNT(DISTINCT ...), it ensures that each booking is only counted once per customer.

FROM booking\_data: Specifies the source table booking\_data from which the data is being retrieved.

GROUP BY customer\_id: This clause groups the data by customer\_id. The COUNT(DISTINCT booking\_id) function is then applied within each group of customer\_id to count the unique booking IDs per customer.

ORDER BY customer\_id ASC: Finally, this orders the result set by customer\_id in ascending order (ASC), which means the results will be listed from the smallest customer\_id to the largest.

screenshot:



**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.**

**select (sum(case when button\_id = "fcba68aa-1231-11eb-adc1-0242ac120002" and**

**is\_button\_click = 'Yes' then 1 end) / sum(case when page\_id = "e7bc5fb2-1231-11eb-adc1-0242ac120002" and is\_page\_view = 'Yes' then 1 end)) as conversion\_ratio from clickstream\_data;**

This query will calculate the conversion ratio, which typically represents the ratio of button clicks (meeting specific criteria) to page views (meeting specific criteria) within your clickstream\_data table. The result will be a single value representing this conversion ratio.

Explanation of each part:

SELECT: This indicates that we want to select and calculate a value based on the subsequent expression.

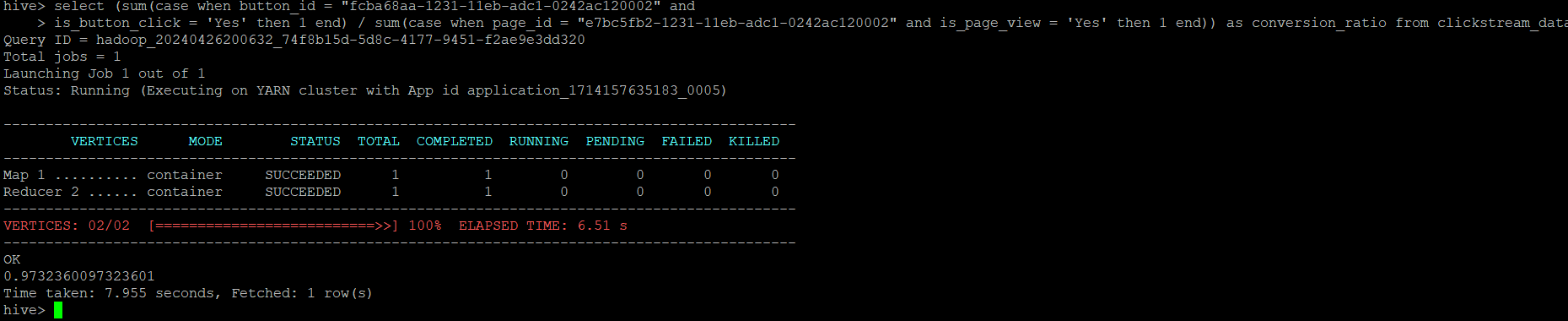
SUM(CASE WHEN button\_id = 'fcba68aa-1231-11eb-adc1-0242ac120002' AND is\_button\_click = 'Yes' THEN 1 END): This part of the query calculates the total count of records where button\_id matches 'fcba68aa-1231-11eb-adc1-0242ac120002' and is\_button\_click is 'Yes'. The CASE statement assigns a value of 1 for each matching record, and SUM adds these values together.

SUM(CASE WHEN page\_id = 'e7bc5fb2-1231-11eb-adc1-0242ac120002' AND is\_page\_view = 'Yes' THEN 1 END): This part of the query calculates the total count of records where page\_id matches 'e7bc5fb2-1231-11eb-adc1-0242ac120002' and is\_page\_view is 'Yes'. Similarly, the CASE statement assigns a value of 1 for each matching record, and SUM adds these values together.

The division (/) operator is used to divide the sum of button clicks (is\_button\_click = 'Yes') by the sum of page views (is\_page\_view = 'Yes') to compute the conversion ratio.

AS conversion\_ratio: This assigns an alias conversion\_ratio to the calculated value in the result set.

screenshot:



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

**select cab\_color ,count(distinct driver\_id ) from booking\_data where cab\_color in ('black') group by cab\_color ;**

The output of this query will be a single row (since we're grouping by cab\_color) showing the count of distinct driver\_id values for bookings where the cab\_color is 'black'. The cab\_color itself will be 'black' in the result**.**

Explanation of each part:

SELECT cab\_color, COUNT(DISTINCT driver\_id): This part of the query selects two columns:

cab\_color: The color of the cab for which we want to count the distinct driver\_id.

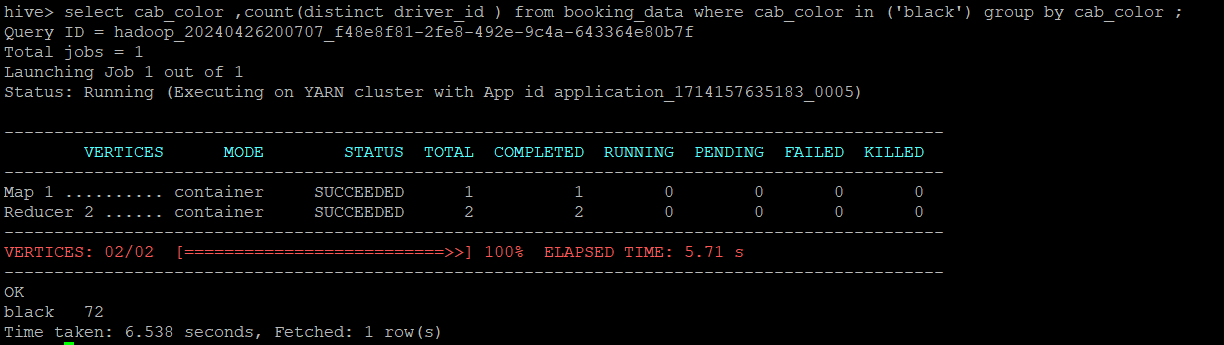
COUNT(DISTINCT driver\_id): Calculates the number of distinct driver\_id values associated with bookings where the cab\_color is 'black'.

FROM booking\_data: Specifies the source table booking\_data from which the data is being retrieved.

WHERE cab\_color IN ('black'): This clause filters the rows from booking\_data where the cab\_color is specifically 'black'. The IN operator allows specifying multiple values (though in this case, only one value is provided).

GROUP BY cab\_color: Groups the data by cab\_color ('black' in this case). The COUNT(DISTINCT driver\_id) function is then applied within this group to count the unique driver\_id values associated with the specified cab\_color

screenshot:



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

**select pickup\_date, sum(tip\_amount) from booking\_data group by pickup\_date order by pickup\_date asc;**

The output of this query will be a list of pickup\_date values along with the total sum of tip\_amount for each date, ordered by pickup\_date. Each row in the output will represent a unique pickup\_date and the corresponding total tip amount for that date.

Explanation of each part:

SELECT pickup\_date, SUM(tip\_amount): This part of the query selects two columns:

pickup\_date: The date on which the booking was made or the ride was picked up.

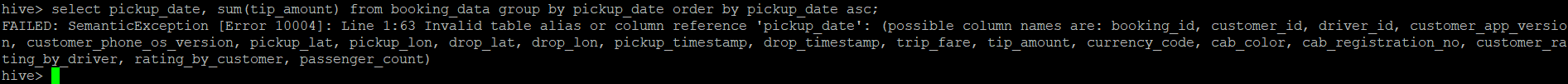
SUM(tip\_amount): Calculates the total sum of tip\_amount for each pickup\_date. This aggregation function adds up all the tip amounts associated with each pickup\_date.

FROM booking\_data: Specifies the source table booking\_data from which the data is being retrieved.

GROUP BY pickup\_date: Groups the data by pickup\_date. The SUM(tip\_amount) function is applied within each group of pickup\_date to calculate the total tip amount for that date.

ORDER BY pickup\_date ASC: Finally, this orders the result set by pickup\_date in ascending order (ASC), which means the results will be listed chronologically from the earliest pickup\_date to the latest.

screenshot:



**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') ,count( rating\_by\_customer) from booking\_data where rating\_by\_customer < 2 group by date\_format(pickup\_timestamp,'yyyy-MM');**

The output of this query will be a list of year-month combinations (yyyy-MM) along with the count of ratings less than 2 for each month, based on the pickup\_timestamp of the bookings. The results will be grouped by month and will only include months where there are ratings below 2.

Explanation of each part:

SELECT DATE\_FORMAT(pickup\_timestamp, 'yyyy-MM'), COUNT(rating\_by\_customer):

DATE\_FORMAT(pickup\_timestamp, 'yyyy-MM'): This function formats the pickup\_timestamp column to display the year and month (yyyy-MM format).

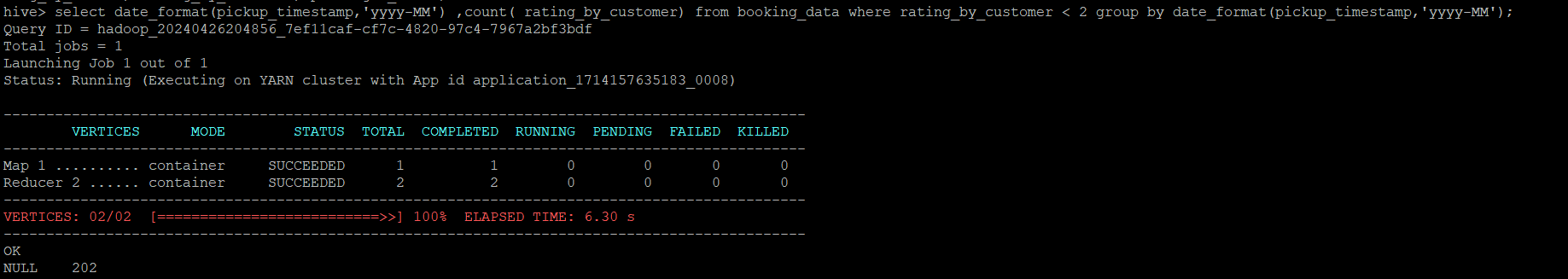
COUNT(rating\_by\_customer): Calculates the count of rating\_by\_customer values that are less than 2.

FROM booking\_data: Specifies the source table booking\_data from which the data is being retrieved.

WHERE rating\_by\_customer < 2: This clause filters the rows where the rating\_by\_customer is less than 2. Only bookings with ratings lower than 2 are considered in the count.

GROUP BY DATE\_FORMAT(pickup\_timestamp, 'yyyy-MM'): Groups the data by the formatted yyyy-MM pickup\_timestamp. The COUNT(rating\_by\_customer) function is then applied within each month group to count the number of ratings less than 2 for that specific month.

screenshot:



**Task 11: Calculate the count of total iOS users.**

**select os\_version ,count(distinct customer\_id) from clickstream\_data where os\_version in ('iOS') group by os\_version;**

This query is useful for analyzing the distribution of unique customers across different versions of the iOS operating system recorded in your clickstream data.

Explanation of each part:

SELECT os\_version, COUNT(DISTINCT customer\_id): This part of the query selects the os\_version column and counts the distinct customer\_id values from the clickstream\_data table.

os\_version: This is the column name representing the operating system version.

COUNT(DISTINCT customer\_id): This function calculates the number of unique customer\_id values for each os\_version.

FROM clickstream\_data: This specifies the source table clickstream\_data from which data is being queried.

WHERE os\_version IN ('iOS'): This part of the query filters the rows to include only those where the os\_version is 'iOS'. The IN ('iOS') condition checks if the os\_version matches the specified value 'iOS'.

GROUP BY os\_version: This groups the results by os\_version, meaning that the counts will be aggregated and displayed for each unique os\_version value ('iOS' in this case).

screenshot:

