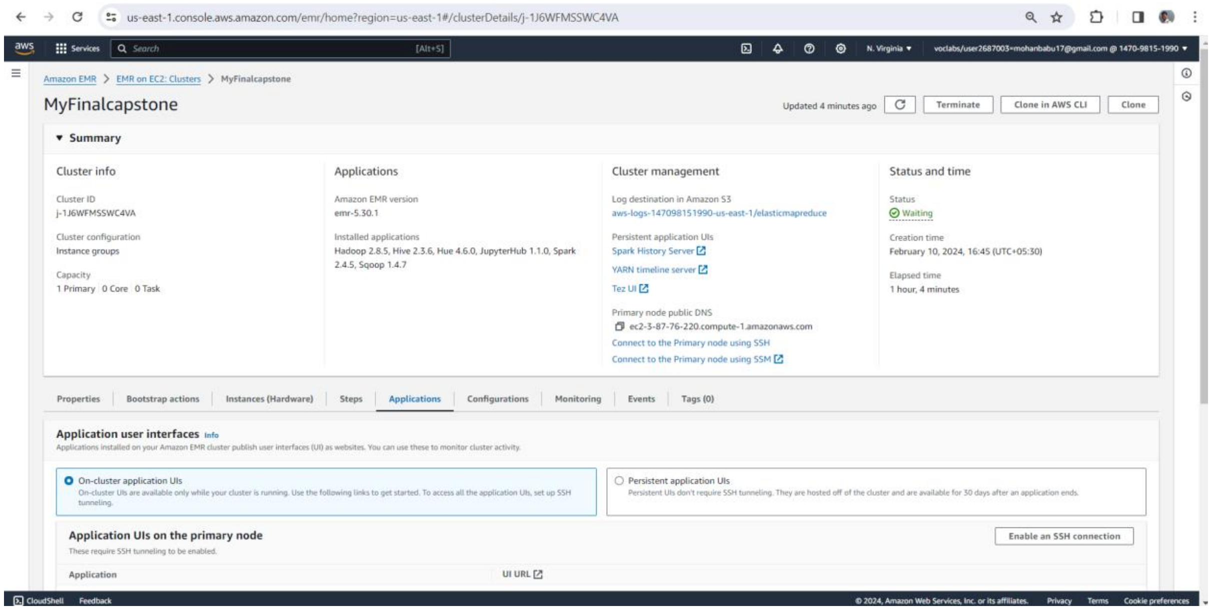
Logic For Final Submission

Active EMR cluster:



us-east-1.console.aws.amazon.com/emr/home?region=us-east-1#/clusterDetails/j-1J6WFMSSWC4VA

N. Virginia | voclabs/user2687003=mohanbabu17@gmail.com @ 1470-9815-1990

Amazon EMR > EMR on EC2: Clusters > MyFinalcapstone

MyFinalcapstone

Updated 4 minutes ago | Terminate | Clone in AWS CLI | Clone

▼ Summary

Cluster info	Applications	Cluster management	Status and time
Cluster ID: j-1J6WFMSSWC4VA	Amazon EMR version: emr-5.30.1	Log destination in Amazon S3: aws-logs-147098151990-us-east-1/elasticmapreduce	Status: Waiting
Cluster configuration: Instance groups	Installed applications: Hadoop 2.8.5, Hive 2.3.6, Hue 4.6.0, JupyterHub 1.1.0, Spark 2.4.5, Sqoop 1.4.7	Persistent application UIs: Spark History Server, YARN timeline server, Tez UI	Creation time: February 10, 2024, 16:45 (UTC+05:30)
Capacity: 1 Primary, 0 Core, 0 Task		Primary node public DNS: ec2-3-87-76-220.compute-1.amazonaws.com	Elapsed time: 1 hour, 4 minutes
		Connect to the Primary node using SSH	
		Connect to the Primary node using SSM	

Properties | Bootstrap actions | Instances (Hardware) | Steps | Applications | Configurations | Monitoring | Events | Tags (0)

Application user interfaces Info

Applications installed on your Amazon EMR cluster publish user interfaces (UI) as websites. You can use these to monitor cluster activity.

[✓ On-cluster application UIs] On-cluster UIs are available only while your cluster is running. Use the following links to get started. To access all the application UIs, set up SSH tunneling.

Persistent application UIs: Persistent UIs don't require SSH tunneling. They are hosted off of the cluster and are available for 30 days after an application ends.

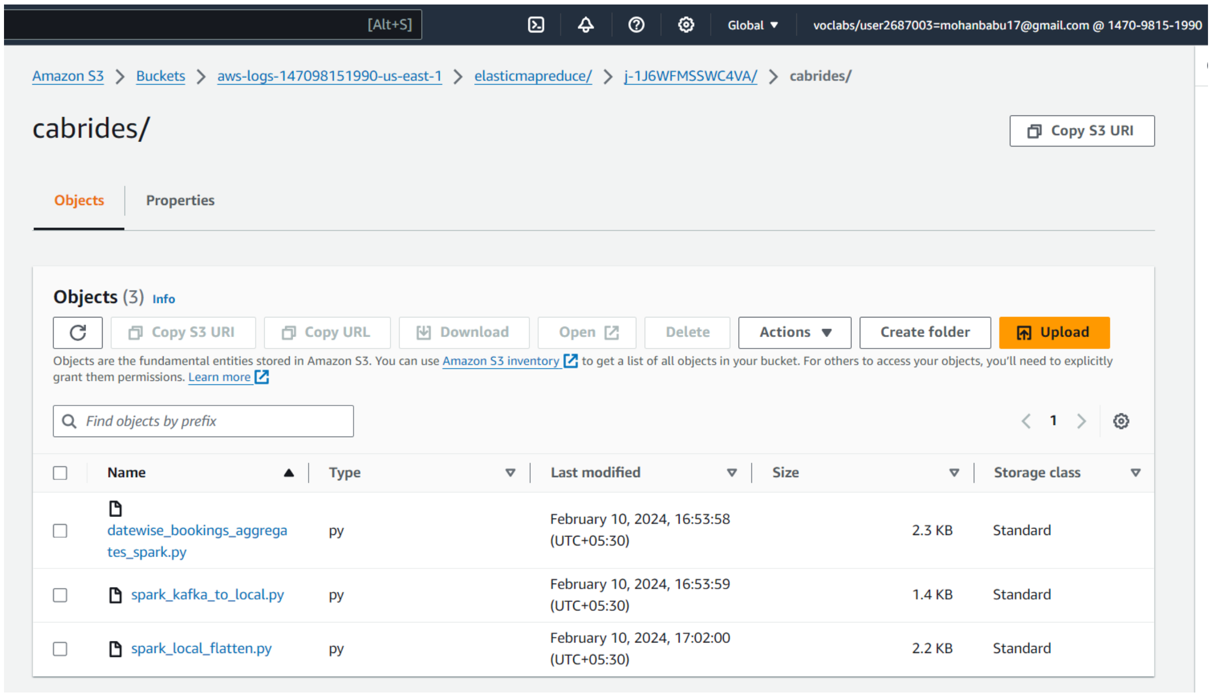
Application UIs on the primary node — Enable an SSH connection

These require SSH tunneling to be enabled.

Application	UI URL

CloudShell | Feedback | © 2024, Amazon Web Services, Inc. or its affiliates. Privacy | Terms | Cookie preferences

S3 files:



[Alt+S] | Global | voclabs/user2687003=mohanbabu17@gmail.com @ 1470-9815-1990

Amazon S3 > Buckets > aws-logs-147098151990-us-east-1 > elasticmapreduce/ > j-1J6WFMSSWC4VA/ > cabrides/

cabrides/

Copy S3 URI

Objects | Properties

Objects (3) Info

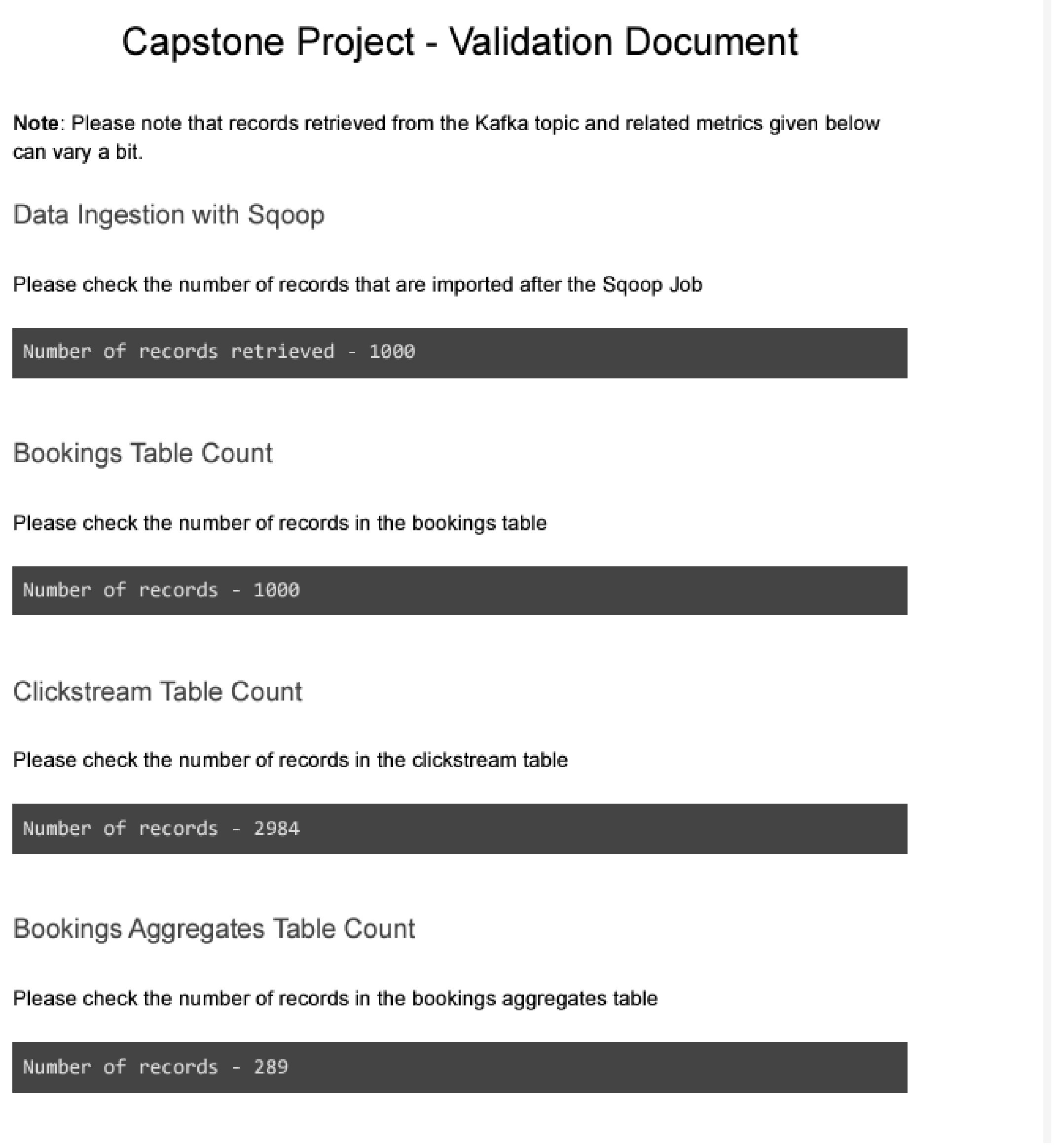
Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Find objects by prefix

Name	Type	Last modified	Size	Storage class
datewise_bookings_aggregates_spark.py	py	February 10, 2024, 16:53:58 (UTC+05:30)	2.3 KB	Standard
spark_kafka_to_local.py	py	February 10, 2024, 16:53:59 (UTC+05:30)	1.4 KB	Standard
spark_local_flatten.py	py	February 10, 2024, 17:02:00 (UTC+05:30)	2.2 KB	Standard

Validation already completed for task 1 to task 4 in LogicFirstSubmission.pdf



Capstone Project - Validation Document

Note: Please note that records retrieved from the Kafka topic and related metrics given below can vary a bit.

Data Ingestion with Sqoop

Please check the number of records that are imported after the Sqoop Job

```
Number of records retrieved - 1000
```

Bookings Table Count

Please check the number of records in the bookings table

```
Number of records - 1000
```

Clickstream Table Count

Please check the number of records in the clickstream table

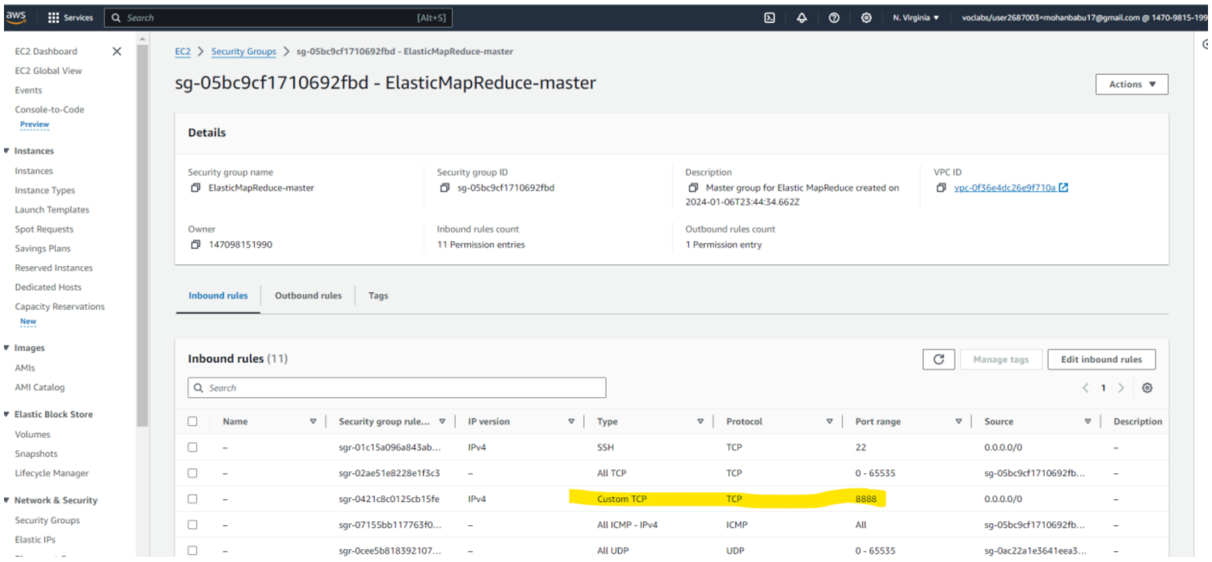
```
Number of records - 2984
```

Bookings Aggregates Table Count

Please check the number of records in the bookings aggregates table

```
Number of records - 289
```

Enable HUE UI for Hive execution: add port 8888 in both master and slave security group



EC2 > Security Groups > sg-05bc9cf1710692fbd - ElasticMapReduce-master

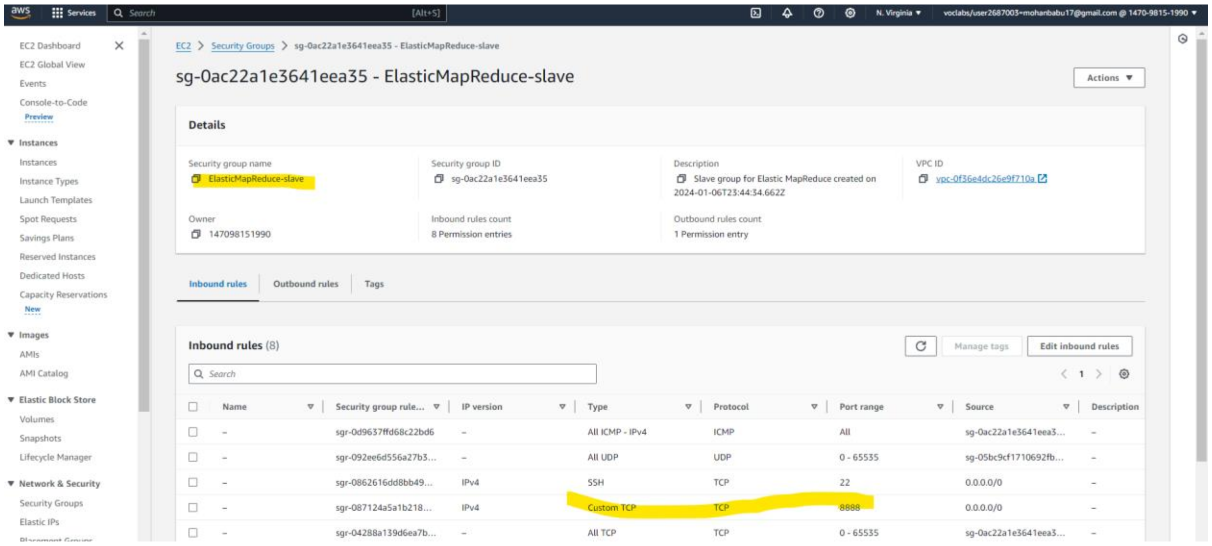
sg-05bc9cf1710692fbd - ElasticMapReduce-master

Details

Security group name	Security group ID	Description	VPC ID
ElasticMapReduce-master	sg-05bc9cf1710692fbd	Master group for Elastic MapReduce created on 2024-01-06T23:44:34.662Z	vpc-0f36e4dc26e9f710a
Owner	**Inbound rules count**	**Outbound rules count**	
147098151990	11 Permission entries	1 Permission entry	

Inbound rules (11)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
–	sgr-01c15a096a843ab...	IPv4	SSH	TCP	22	0.0.0.0/0	–
–	sgr-02ae51e8228e1f3c3	–	All TCP	TCP	0 - 65535	sg-05bc9cf1710692fb...	–
–	sgr-0421c8c0125cb15fe	IPv4	Custom TCP	TCP	8888	0.0.0.0/0	–
–	sgr-07155bb117763f0...	–	All ICMP - IPv4	ICMP	All	sg-05bc9cf1710692fb...	–
–	sgr-0cee5b818392107...	–	All UDP	UDP	0 - 65535	sg-0ac22a1e3641eea3...	–



EC2 > Security Groups > sg-0ac22a1e3641eea35 - ElasticMapReduce-slave

sg-0ac22a1e3641eea35 - ElasticMapReduce-slave

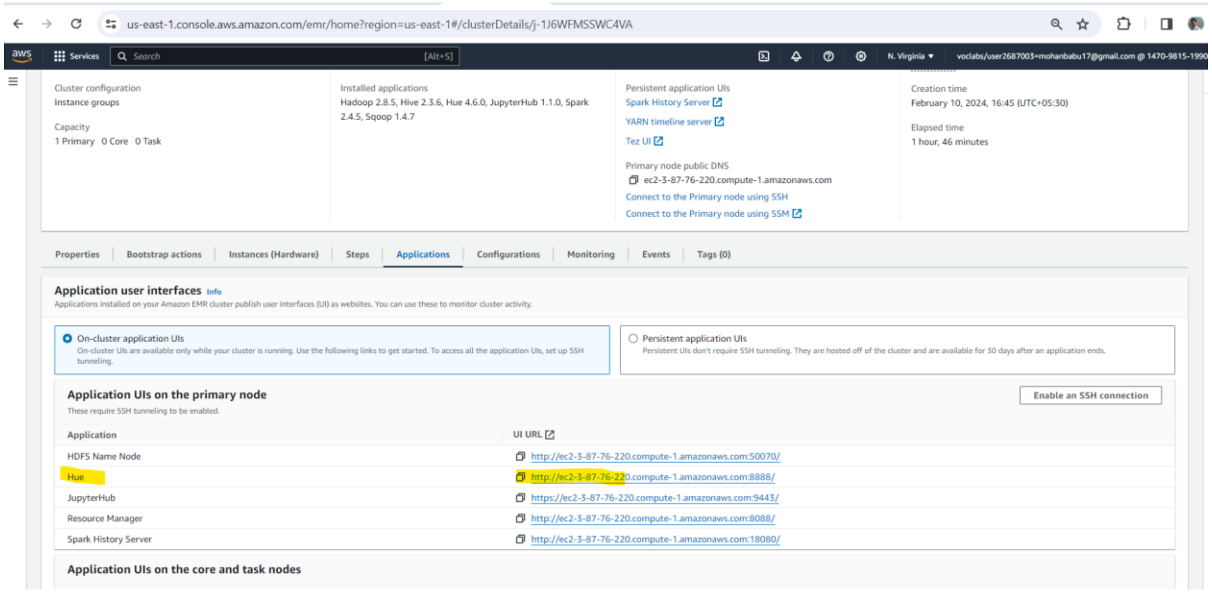
Details

Security group name	Security group ID	Description	VPC ID
ElasticMapReduce-slave	sg-0ac22a1e3641eea35	Slave group for Elastic MapReduce created on 2024-01-06T23:44:34.662Z	vpc-0f36e4dc26e9f710a
Owner	**Inbound rules count**	**Outbound rules count**	
147098151990	8 Permission entries	1 Permission entry	

Inbound rules (8)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
–	sgr-0d9637ffd68c22bd6	–	All ICMP - IPv4	ICMP	All	sg-0ac22a1e3641eea3...	–
–	sgr-092ee6d556a27b3...	–	All UDP	UDP	0 - 65535	sg-05bc9cf1710692fb...	–
–	sgr-0862616dd8bb49...	IPv4	SSH	TCP	22	0.0.0.0/0	–
–	sgr-087124a5a1b218...	IPv4	Custom TCP	TCP	8888	0.0.0.0/0	–
–	sgr-04288a139d6ea7b...	–	All TCP	TCP	0 - 65535	sg-0ac22a1e3641eea3...	–

Open HUE link:



us-east-1.console.aws.amazon.com/emr/home?region=us-east-1#/clusterDetails/j-1J6WFMSSWC4VA

Cluster configuration
Instance groups

Capacity
1 Primary | 0 Core | 0 Task

Installed applications
Hadoop 2.8.5, Hive 2.3.6, Hue 4.6.0, JupyterHub 1.1.0, Spark 2.4.5, Sqoop 1.4.7

Persistent application UIs
Spark History Server
YARN timeline server
Tez UI

Primary node public DNS
ec2-3-87-76-220.compute-1.amazonaws.com
Connect to the Primary node using SSH
Connect to the Primary node using SSM

Creation time
February 10, 2024, 16:45 (UTC+05:30)

Elapsed time
1 hour, 46 minutes

Properties | Bootstrap actions | Instances (Hardware) | Steps | Applications | Configurations | Monitoring | Events | Tags (0)

Application user interfaces Info
Applications installed on your Amazon EMR cluster publish user interfaces (UI) as websites. You can use these to monitor cluster activity.

● On-cluster application UIs
On-cluster UIs are available only while your cluster is running. Use the following links to get started. To access all the application UIs, set up SSH tunneling.

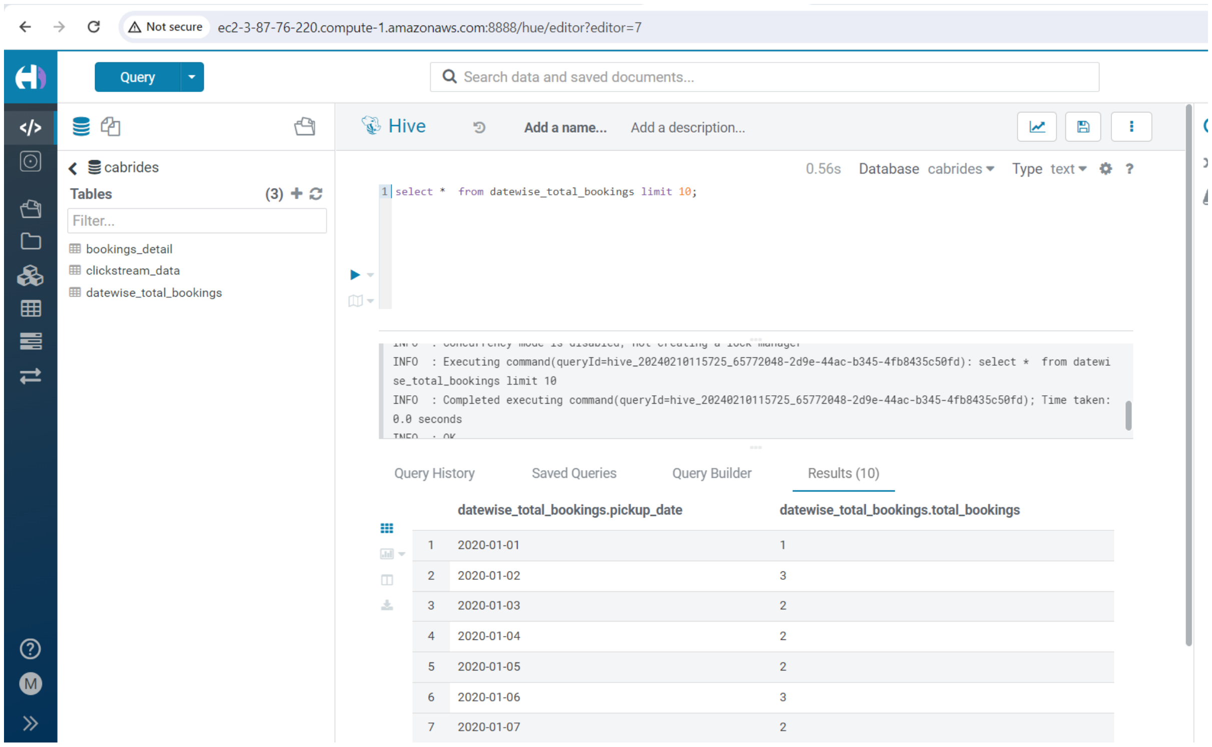
○ Persistent application UIs
Persistent UIs don't require SSH tunneling. They are hosted off of the cluster and are available for 30 days after an application ends.

Application UIs on the primary node
These require SSH tunneling to be enabled.

Application	UI URL
HDFS Name Node	http://ec2-3-87-76-220.compute-1.amazonaws.com:50070/
Hue	http://ec2-3-87-76-220.compute-1.amazonaws.com:8888/
JupyterHub	https://ec2-3-87-76-220.compute-1.amazonaws.com:9443/
Resource Manager	http://ec2-3-87-76-220.compute-1.amazonaws.com:8088/
Spark History Server	http://ec2-3-87-76-220.compute-1.amazonaws.com:18080/

Application UIs on the core and task nodes

HUE: query check in UI, select cabrides database and execute table query



ec2-3-87-76-220.compute-1.amazonaws.com:8888/hue/editor?editor=7

cabrides
Tables (3)
bookings_detail
clickstream_data
datewise_total_bookings

Hive — 0.56s Database cabrides Type text

```
select *  from datewise_total_bookings limit 10;
```

```
INFO  : Executing command(queryId=hive_20240210115725_65772048-2d9e-44ac-b345-4fb8435c50fd): select *  from datewise_total_bookings limit 10
INFO  : Completed executing command(queryId=hive_20240210115725_65772048-2d9e-44ac-b345-4fb8435c50fd); Time taken: 0.0 seconds
INFO  : OK
```

Query History | Saved Queries | Query Builder | Results (10)

	datewise_total_bookings.pickup_date	datewise_total_bookings.total_bookings
1	2020-01-01	1
2	2020-01-02	3
3	2020-01-03	2
4	2020-01-04	2
5	2020-01-05	2
6	2020-01-06	3
7	2020-01-07	2

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

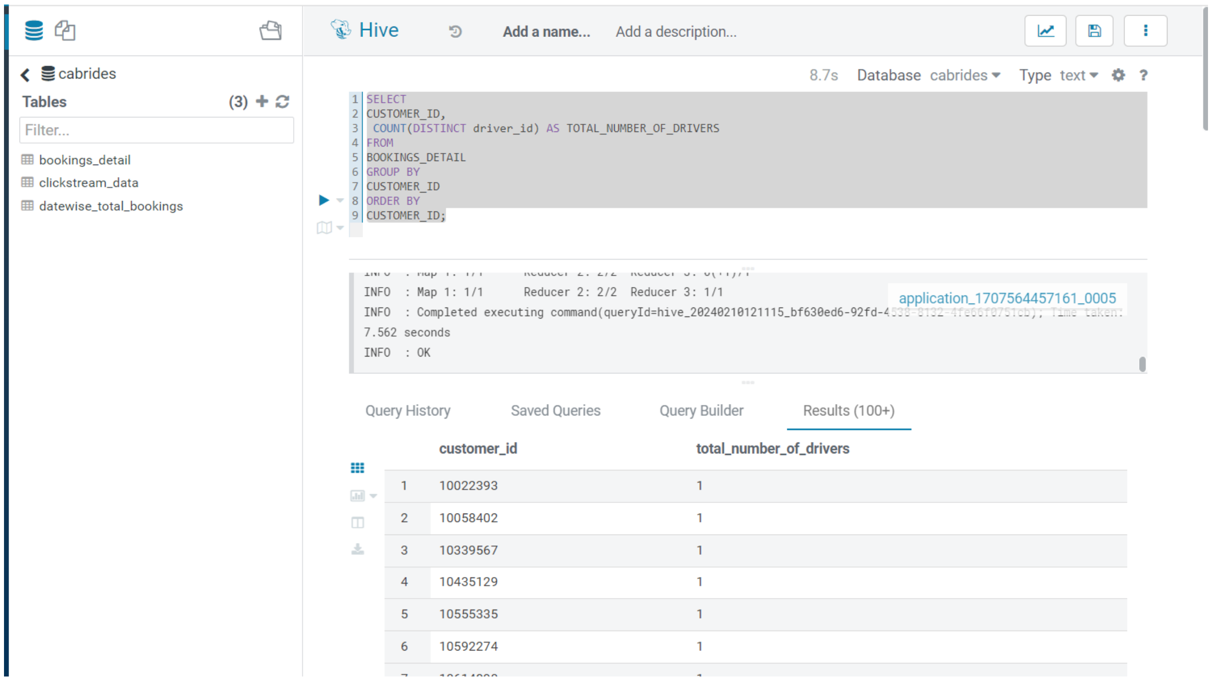
QUERY:

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

EXPLANATION:

Check multiple drivers for each customer by grouping customer and count total individual drivers for each and sort by customer id

OUTPUT: each customer is having only one driver , so total number of drivers is "1"



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

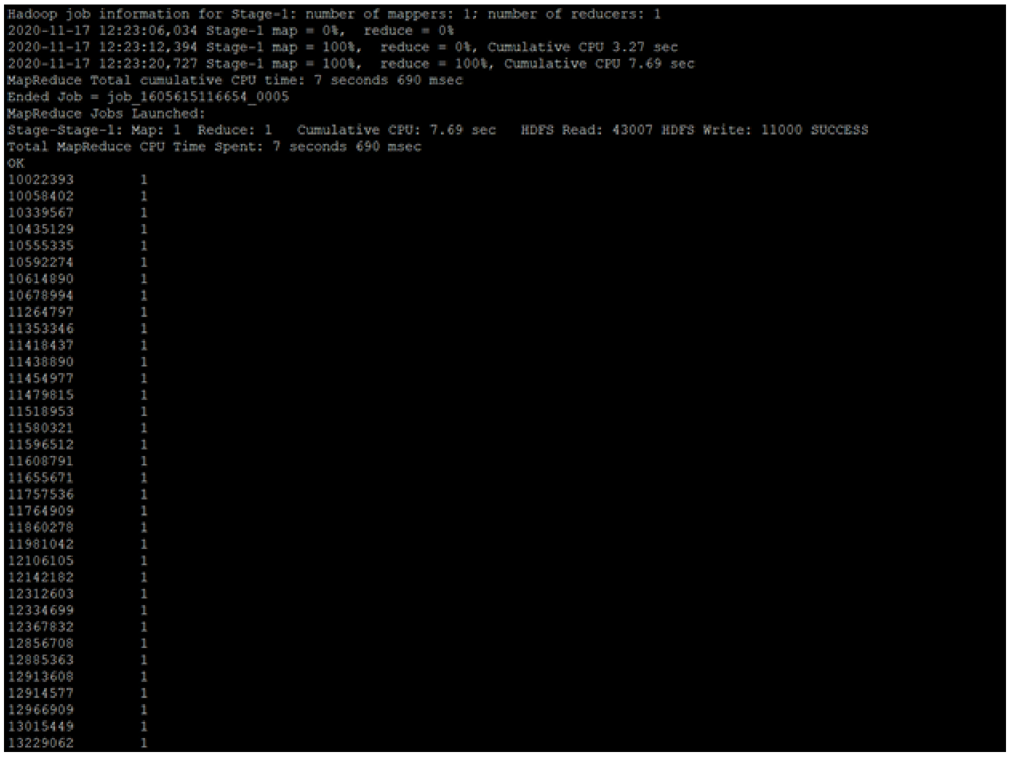
```
INFO  : Map 1: 1/1      Reducer 2: 2/2  Reducer 3: 1/1
INFO  : Completed executing command(queryId=hive_20240210121115_bf630ed6-92fd-4538-8132-4fe66f0751cb); Time taken:
7.562 seconds
INFO  : OK
```

application_1707564457161_0005

	customer_id	total_number_of_drivers
1	10022393	1
2	10058402	1
3	10339567	1
4	10435129	1
5	10555335	1
6	10592274	1

VALIDATION: Exact Match

1. When you run the query to calculate the total number of different drivers for each customer, you would get an output as shown below:



```
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
```

- **Task 6:**

Calculate the total rides taken by each customer.

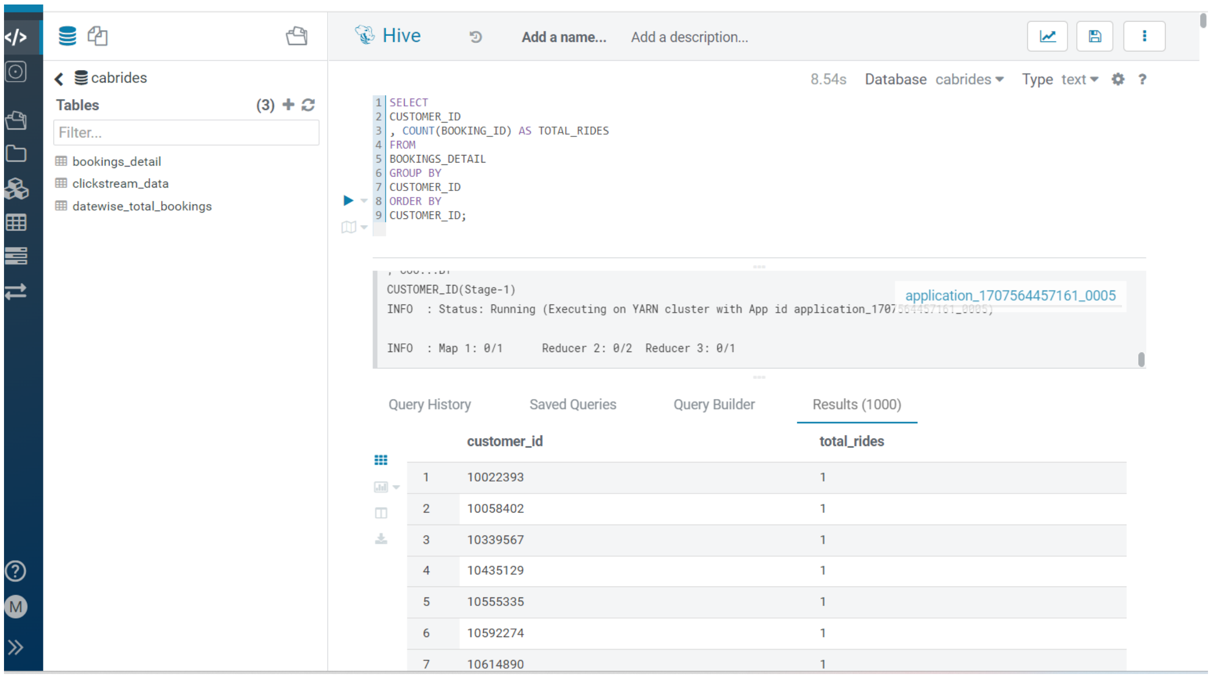
QUERY:

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

EXPLANATION:

Check total rides for each customer by grouping customer, count total rides and sort by customer id

OUTPUT: each customer is having only one ride , so total ride is "1" for each customer



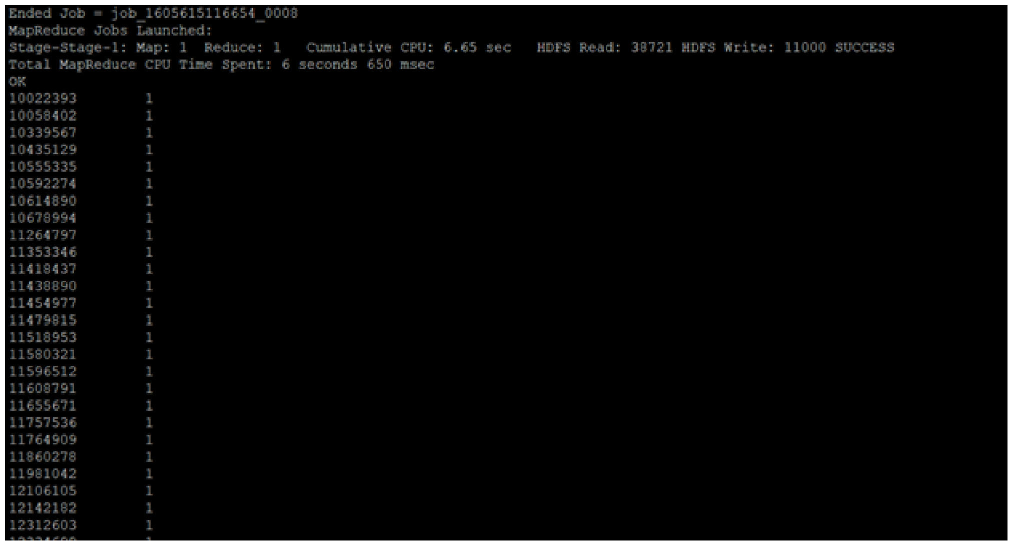
```
SELECT
CUSTOMER_ID
, COUNT(BOOKING_ID) AS TOTAL_RIDES
FROM
BOOKINGS_DETAIL
GROUP BY
CUSTOMER_ID
ORDER BY
CUSTOMER_ID;
```

```
CUSTOMER_ID(Stage-1)
INFO  : Status: Running (Executing on YARN cluster with App id application_1707564457161_0005)

INFO  : Map 1: 0/1      Reducer 2: 0/2  Reducer 3: 0/1
```

	customer_id	total_rides
1	10022393	1
2	10058402	1
3	10339567	1
4	10435129	1
5	10555335	1
6	10592274	1
7	10614890	1

VALIDATION: Exact Match



```
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
```

- **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.

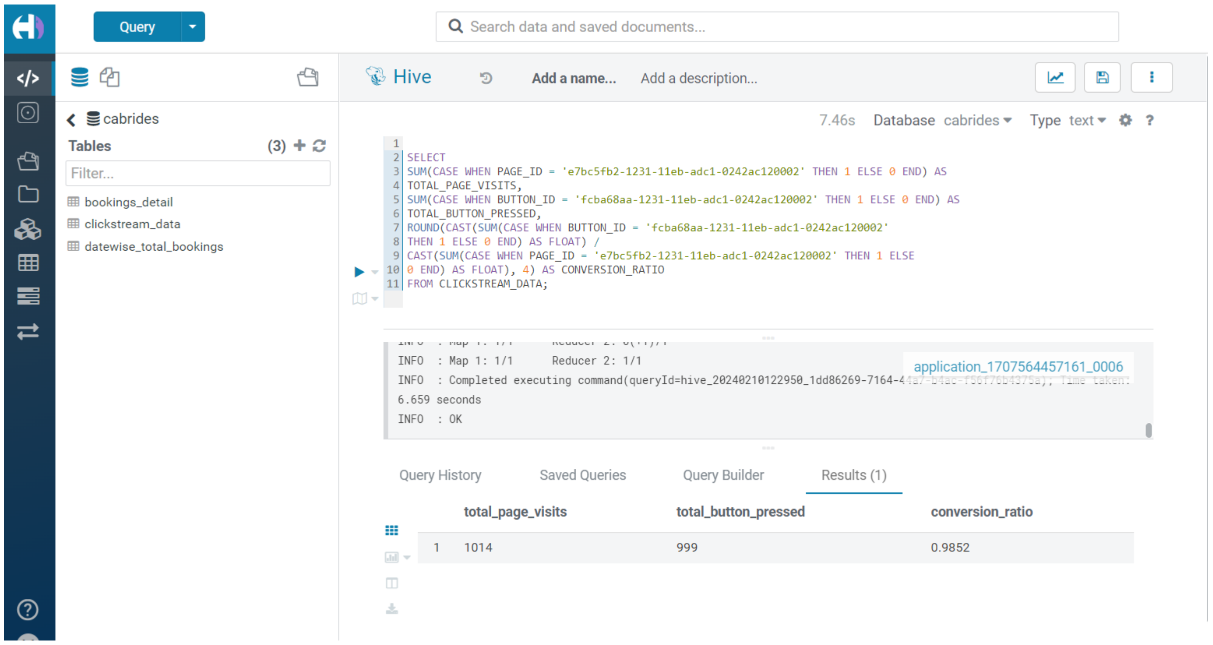
QUERY:

```
SELECT
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:

- This analysis is useful for understanding customer behavior, such as how often they book a ride after visiting the booking page.
- Tracking the number of times the booking page is visited (PAGE_ID = 'e7bc5fb2-1231-11eb-adc1-0242ac120002') provides the total count of page visits.
- Recording the number of times the booking button is clicked (BUTTON_ID = 'fc6ba68a-1231-11eb-adc1-0242ac120002') indicates the total number of rides that have been booked.
- The conversion ratio is simply the number of total bookings divided by the number of total page visits.
- The conversion ratio is an essential Key Performance Indicator (KPI) for the company, showing a high likelihood of booking when a customer visits the booking page, which is approximately 98% in this instance.

OUTPUT:



```
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;
```

```
INFO  : Map 1: 1/1      Reducer 2: 1/1
INFO  : Completed executing command(queryId=hive_20240210122950_1dd86269-7164-44a7-b4ac-f56f76b4375a); Time taken:
6.659 seconds
INFO  : OK
```

	total_page_visits	total_button_pressed	conversion_ratio
1	1014	999	0.9852

VALIDATION: close to Match (0.9852), since Kafka had extra 16 records compare to validation it should get the conversion ratio as 0.9688.

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

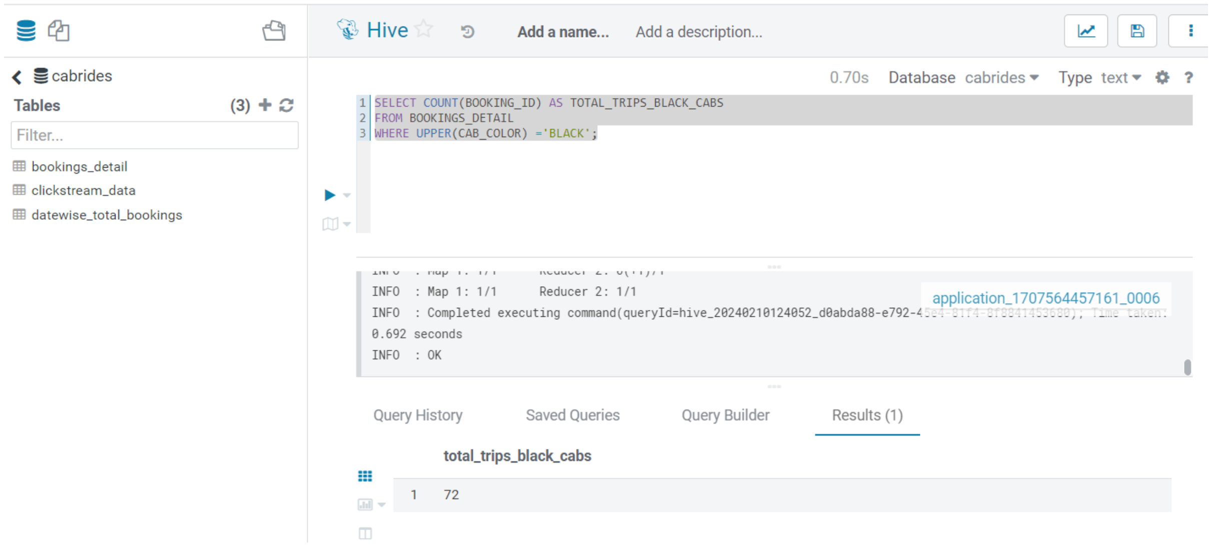
QUERY:

SELECT COUNT(BOOKING_ID) AS TOTAL_TRIPS_BLACK_CABS
FROM BOOKINGS_DETAIL
WHERE UPPER(CAB_COLOR) ='BLACK';

EXPLANATION:

- This assessment aids in determining the overall number of journeys made using Black taxis.
- Categorizing by CAB_COLOR can reveal the total count or proportion of trips completed by taxis of a particular color.
- This might reveal any obscure trends in booking related to the color of the taxis, such as a customer's preference for booking taxis of a certain color.

OUTPUT: since case sensitive, handles upper from cab colour



```
SELECT COUNT(BOOKING_ID) AS TOTAL_TRIPS_BLACK_CABS
FROM BOOKINGS_DETAIL
WHERE UPPER(CAB_COLOR) ='BLACK';
```

```
INFO  : Map 1: 1/1      Reducer 2: 1/1
INFO  : Completed executing command(queryId=hive_20240210124052_d0abda88-e792-45e4-81f4-8f8841453680); Time taken:
0.692 seconds
INFO  : OK
```

```
   total_trips_black_cabs
1  72
```

VALIDATION: Exact Match

Count of all trips done on black cabs -72.

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

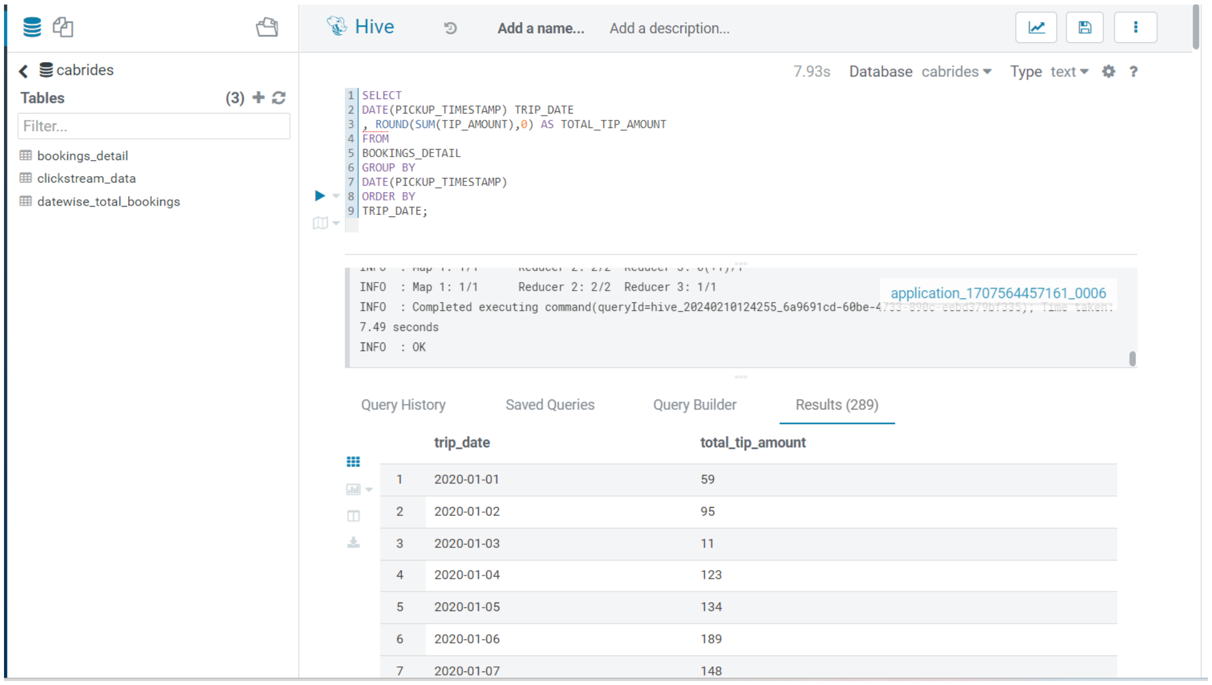
QUERY:

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:

- The date function isolates the date from the datetime_stamp value and labels it as Pickup_Date.
- The SUM function, in conjunction with GROUP BY Pickup Date, calculates the aggregate tip amount for each Pickup date.
- The ORDER BY clause, applied to the Pickup Date alias, arranges the results in chronological order based on the date of pickup.
- Analyzing this data can provide insights into whether customers tend to tip more on certain occasions or specific days.
- Based on these insights, management can introduce promotions for customers or drivers, inferring that a higher tip amount correlates with greater customer satisfaction on particular days.

OUTPUT:

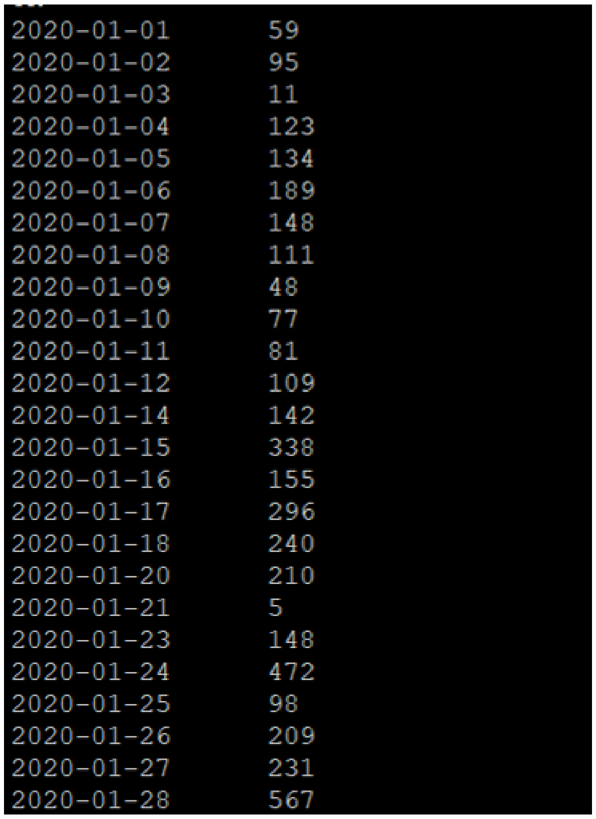


```
SELECT
DATE(PICKUP_TIMESTAMP) TRIP_DATE
, ROUND(SUM(TIP_AMOUNT),0) AS TOTAL_TIP_AMOUNT
FROM
BOOKINGS_DETAIL
GROUP BY
DATE(PICKUP_TIMESTAMP)
ORDER BY
TRIP_DATE;
```

```
INFO  : Map 1: 1/1      Reducer 2: 2/2  Reducer 3: 1/1
INFO  : Completed executing command(queryId=hive_20240210124255_6a9691cd-60be-4733-890c-eebd379bf335); Time taken:
7.49 seconds
INFO  : OK
```

	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

VALIDATION: Exact Match



```
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
```

- **Task 10**:

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

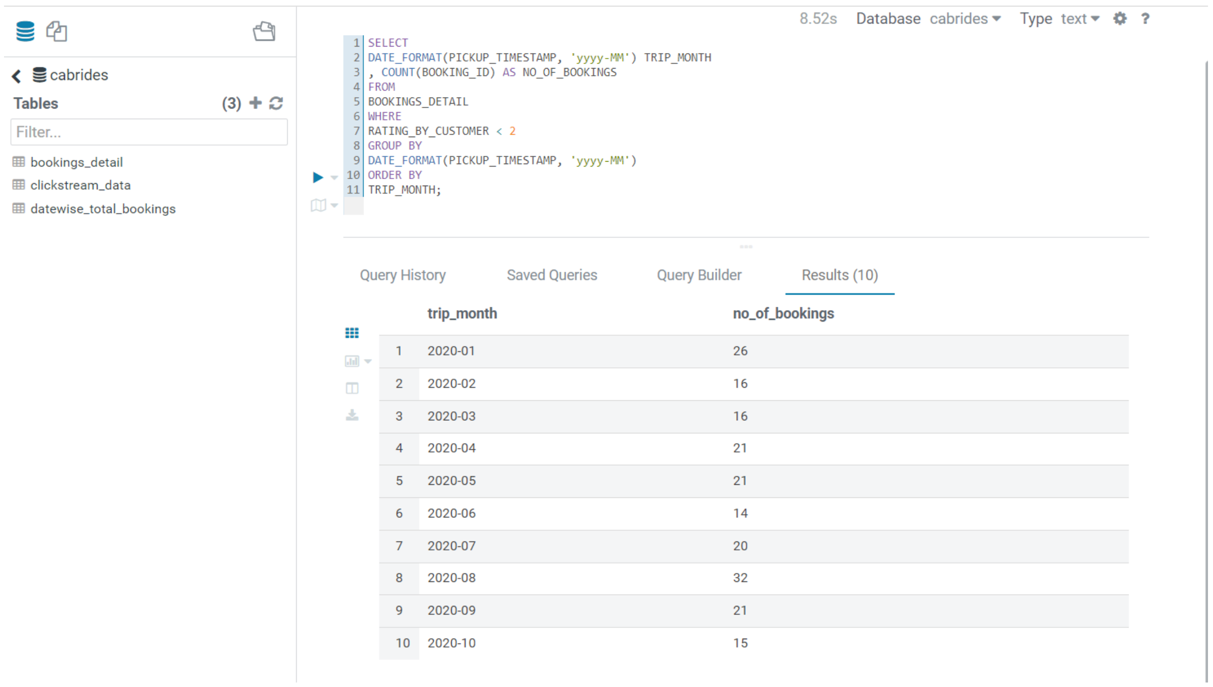
QUERY:

```
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:

- DATE_FORMAT function formats datetimestamp value in the specified format like yyyy-MM in this case which results like 2023-06.
- WHERE clause is used to filter bookings where rating given by customers is less than 2 which indicates customers dissatisfaction.
- ORDER BY clause with Trip month alias is used to show output in ascending order of pickup month.
- This analysis could help to understand number of trips by month where customers were not happy.
- 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 summery time, cab reached late on pickup point due to traffic on a rainy day/season etc.
- Based on this analysis, instructions can be given to driver to make customers happy and take care of things which could lead to low customer rating.

OUTPUT:



8.52s Database cabrides Type text

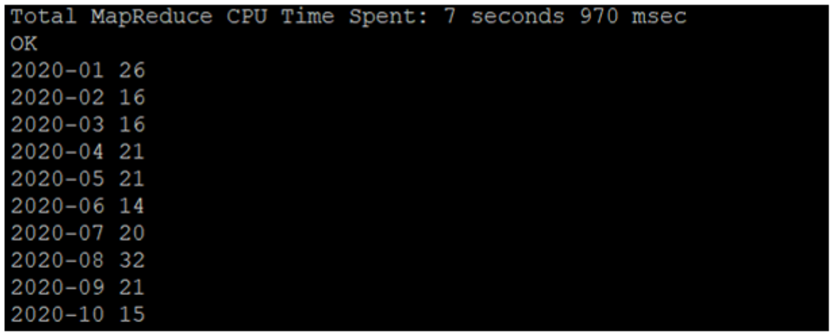
```
SELECT
DATE_FORMAT(PICKUP_TIMESTAMP, 'yyyy-MM') TRIP_MONTH
, COUNT(BOOKING_ID) AS NO_OF_BOOKINGS
FROM
BOOKINGS_DETAIL
WHERE
RATING_BY_CUSTOMER < 2
GROUP BY
DATE_FORMAT(PICKUP_TIMESTAMP, 'yyyy-MM')
ORDER BY
TRIP_MONTH;
```

cabrides
Tables (3)
bookings_detail
clickstream_data
datewise_total_bookings

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: Exact Match



```
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
```

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

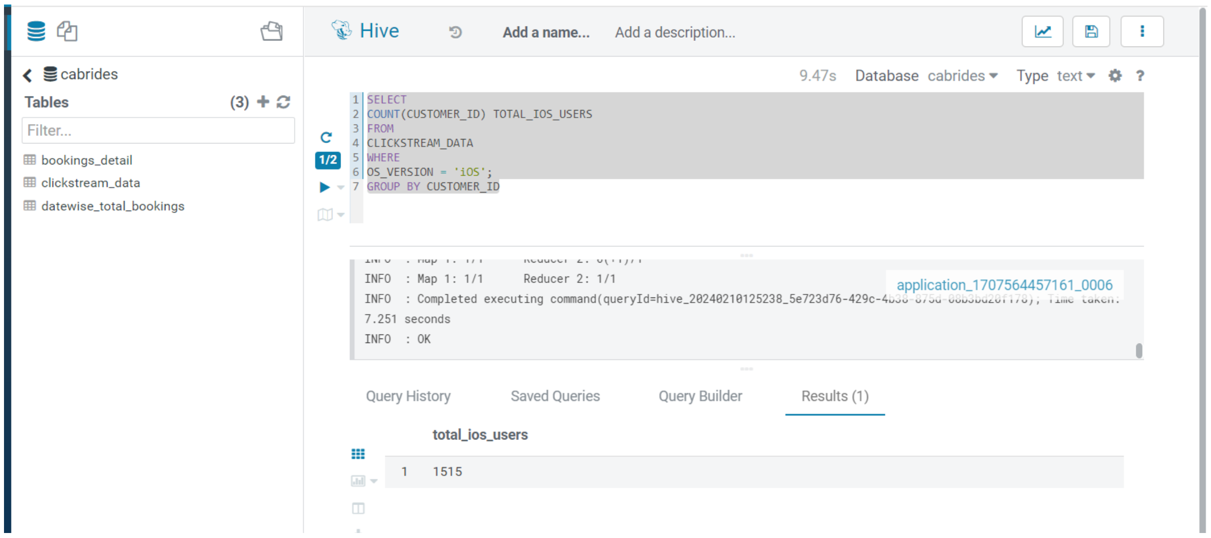
QUERY:

```
SELECT COUNT(CUSTOMER_ID) TOTAL_IOS_USERS
FROM CLICKSTREAM_DATA
WHERE  OS_VERSION = 'iOS';
GROUP BY CUSTOMER_ID;
```

EXPLANATION:

- The DISTINCT option within the COUNT function is utilized to tally the distinct number of customers using iOS devices.
- The WHERE clause filters for events originating from iOS devices.
- The ORDER BY clause organizes the data in chronological order by the pickup month, using Trip month as an alias.
- This analysis offers a broad overview of the quantity or proportion of customers who use a particular type of device or operating system.
- For instance, should the company roll out updates to the iOS and Android mobile applications, this data could estimate the customer base that would be affected.

OUTPUT:



```
SELECT
COUNT(CUSTOMER_ID) TOTAL_IOS_USERS
FROM
CLICKSTREAM_DATA
WHERE
OS_VERSION = 'iOS';
GROUP BY CUSTOMER_ID
```

```
INFO  : Map 1: 1/1      Reducer 2: 1/1
INFO  : Completed executing command(queryId=hive_20240210125238_5e723d76-429c-4b38-875d-08b3bd20f178); Time taken:
7.251 seconds
INFO  : OK
```

application_1707564457161_0006

```
   total_ios_users
1  1515
```

VALIDATION: close to Match(1515) , since Kafka had extra 16 records compare to validation
You should get the count of all iOS users as 1503.