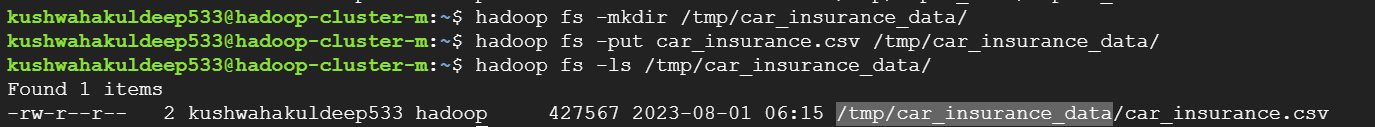
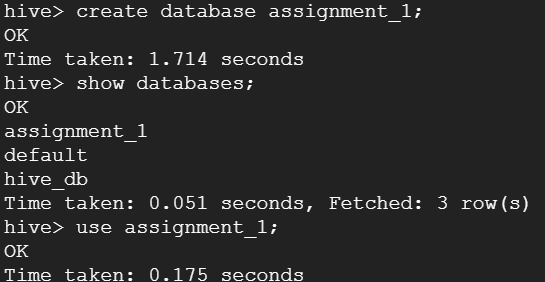
Problem 1: Data Loading





**Problem 1:**

Data Loading 1. Load the data into a Hive table. Create an external table with the given schema and load the data into the table from a text file or HDFS path;

hive>

CREATE EXTERNAL TABLE car\_insurance\_data (

Id INT,Age INT,Job STRING,

Marital STRING,Education STRING,

Default INT,Balance INT,HHInsurance INT,

CarLoan INT,Communication STRING,

LastContactDay INT, LastContactMonth string,

NoOfContacts INT,DaysPassed INT,

PrevAttempts INT,Outcome STRING,

CallStart STRING,CallEnd STRING,

CarInsurance INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

location '/tmp/car\_insurance\_data/';

**Problem 2: Data Exploration**

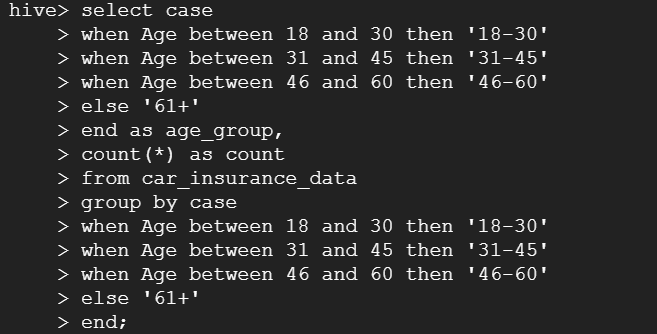
1.How many records are there in the dataset?

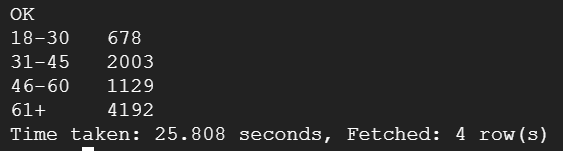
🡪select count(\*) from car\_insurance\_data;

2.How many unique job categories are there?

🡪 select count(distinct(job)) from car\_insurance\_data;

3. What is the age distribution of customers in the dataset? Provide a breakdown by age group: 18-30, 31-45, 46-60, 61+.





4. Count the number of records that have missing values in any field.

🡪

hive>

SELECT COUNT(\*)

WHERE Id IS NULL

OR Age IS NULL

OR Job IS NULL

OR Marital IS NULL

OR Education IS NULL

OR Default IS NULL

OR Balance IS NULL

OR HHInsurance IS NULL

OR CarLoan IS NULL

OR Communication IS NULL

OR LastContactDay IS NULL

OR LastContactMonth IS NULL

OR NoOfContacts IS NULL

OR DaysPassed IS NULL

OR PrevAttempts IS NULL

OR Outcome IS NULL

OR CallStart IS NULL

OR CallEnd IS NULL

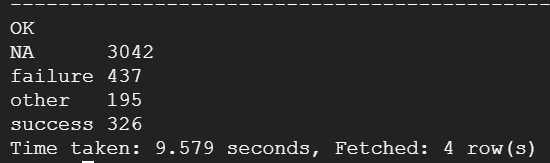
OR CarInsurance IS NULL

FROM car\_insurance\_data;



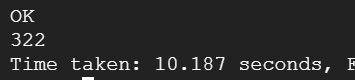
5. Determine the number of unique 'Outcome' values and their respective counts.

hive> select outcome,count(\*) from car\_insurance\_data group by outcome;



6. Find the number of customers who have both a car loan and home insurance.

🡪 select count(\*) from car\_insurance\_data where HHInsurance=1 and carloan=1;

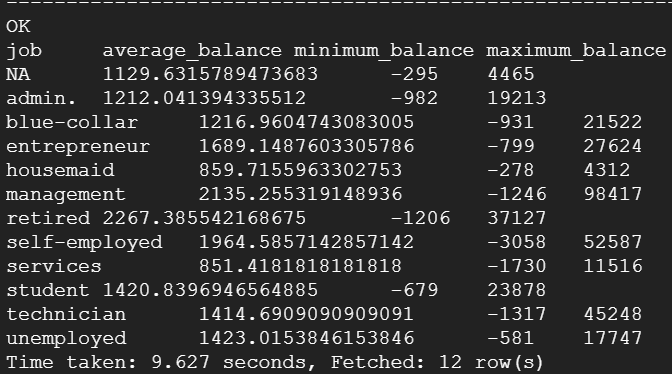


**Problem 3: Aggregations**

1. What is the average, minimum, and maximum balance for each job category?

🡪

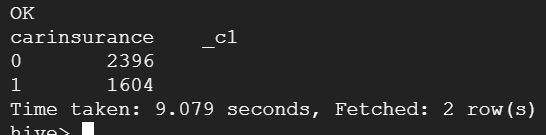
select job,avg(balance) as average\_balance,min(balance)as minimum\_balance ,max(balance)as maximum\_balance from car\_insurance\_data group by job;



2. Find the total number of customers with and without car insurance.

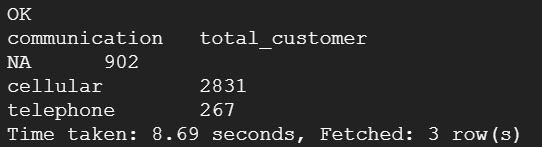
🡪

select carinsurance,count(\*) from car\_insurance\_data group by carinsurance;



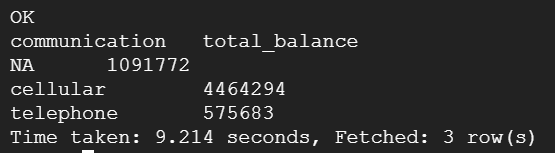
3. Count the number of customers for each communication type.

* select communication ,count(\*)as Total\_customer from car\_insurance\_data group by communication;



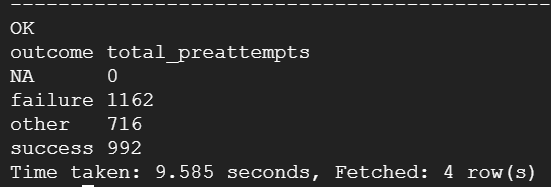
4. Calculate the sum of 'Balance' for each 'Communication' type.

* select communication ,sum(balance)as Total\_balance from car\_insurance\_data group by communication;



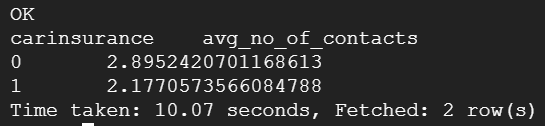
5. Count the number of 'PrevAttempts' for each 'Outcome' type.

* select outcome ,sum(PrevAttempts)as Total\_PreAttempts from car\_insurance\_data group by outcome;



6. Calculate the average 'NoOfContacts' for people with and without 'CarInsurance'.

* select carinsurance ,avg(NoOfContacts)as avg\_no\_of\_contacts from car\_insurance\_data group by carinsurance;



**Problem 4: Partitioning and Bucketing**

1. Create a partitioned table on 'Education' and 'Marital' status. Load data from the original table to this new partitioned table.

CREATE TABLE car\_insurance\_data\_partitioned (

Id INT,

Age INT,

Job STRING,

Default INT,

Balance INT,

HHInsurance INT,

CarLoan INT,

Communication STRING,

LastContactDay INT,

LastContactMonth INT,

NoOfContacts INT,

DaysPassed INT,

PrevAttempts INT,

Outcome STRING,

CallStart STRING,

CallEnd STRING,

CarInsurance INT)

PARTITIONED BY (Education STRING, Marital STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

**hive>** INSERT OVERWRITE TABLE

car\_insurance\_data\_partitioned PARTITION(Education,

Marital)

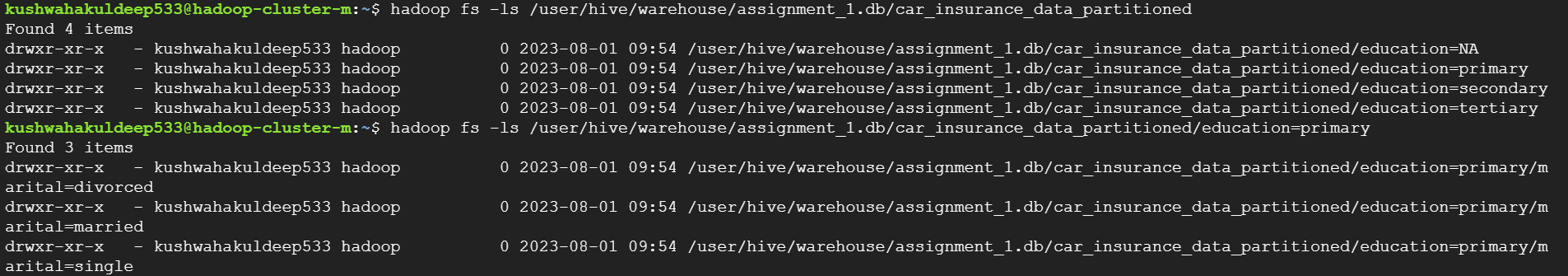
SELECT Id, Age, Job, Default, Balance, HHInsurance, CarLoan, Communication, LastContactDay,

LastContactMonth, NoOfContacts, DaysPassed,

PrevAttempts, Outcome, CallStart, CallEnd,

CarInsurance, Education, Marital

FROM car\_insurance\_data;



2. Create a bucketed table on 'Age', bucketed into 4 groups (as per the age groups mentioned above). Load data from the original table into this bucketed table.

🡪

hive> set hive.enforce.bucketing=true;

hive>

create table car\_insurance\_data\_bucketed(

Id INT,

Age INT,

Job STRING,

Marital String,

Education String,

Default INT,

Balance INT,

HHInsurance INT,

CarLoan INT,

Communication STRING,

LastContactDay INT,

LastContactMonth INT,

NoOfContacts INT,

DaysPassed INT,

PrevAttempts INT,

Outcome STRING,

CallStart STRING,

CallEnd STRING,

CarInsurance INT)

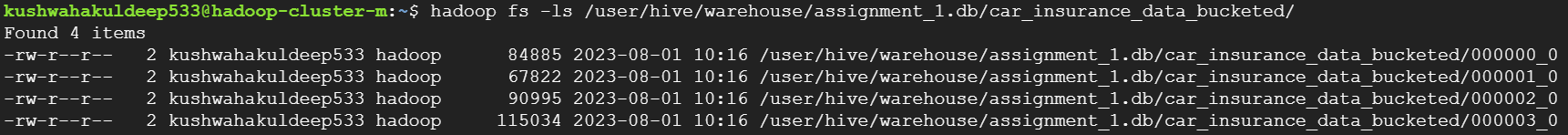
clustered by(Age) into 4 buckets

row format delimited

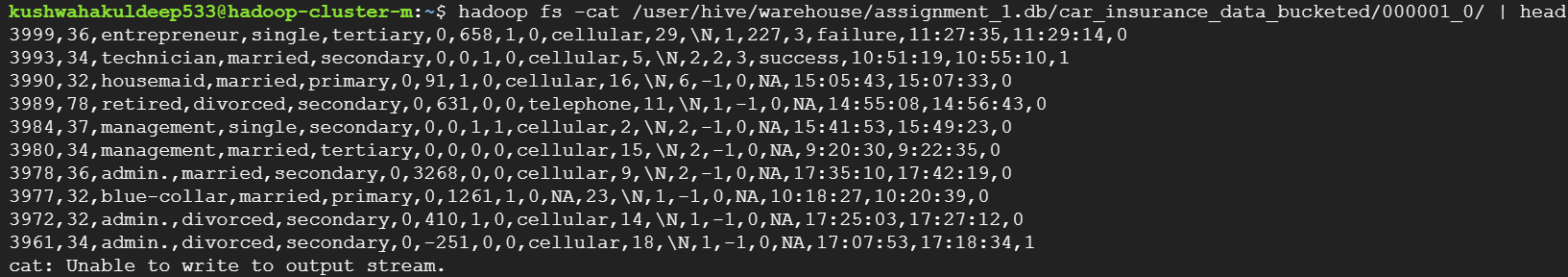
fields terminated by ','

stored as textfile;

hive> insert overwrite table car\_insurance\_data\_bucketed select \* from car\_insurance\_data;



hadoop fs -cat /user/hive/warehouse/assignment\_1.db/car\_insurance\_data\_bucketed/000001\_0/ | head



Imp 3. Add an additional partition on 'Job' to the partitioned table created earlier and move the data accordingly.

* If you want to add an additional partition on 'Job' to the previously created partitioned table, you actually have to create a new table as Hive does not allow altering the partitioning of existing tables. However, it's a straightforward task to create a new partitioned table and move the data accordingly.

hive> set hive.exec.dynamic.partition.mode=nonstrict;

CREATE TABLE car\_insurance\_data\_partitioned\_new ( Id INT, Age INT, Default INT, Balance INT, HHInsurance INT, CarLoan INT, Communication STRING, LastContactDay INT, LastContactMonth INT, NoOfContacts INT, DaysPassed INT, PrevAttempts INT, Outcome STRING, CallStart STRING, CallEnd STRING, CarInsurance INT) PARTITIONED BY (Education STRING, Marital STRING, Job STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

INSERT OVERWRITE TABLE car\_insurance\_data\_partitioned\_new

PARTITION(Education, Marital, Job) SELECT Id, Age, Default, Balance,

HHInsurance, CarLoan, Communication, LastContactDay,

LastContactMonth, NoOfContacts, DaysPassed,

PrevAttempts, Outcome, CallStart, CallEnd,

CarInsurance, Education, Marital, Job

FROM car\_insurance\_data\_partitioned;

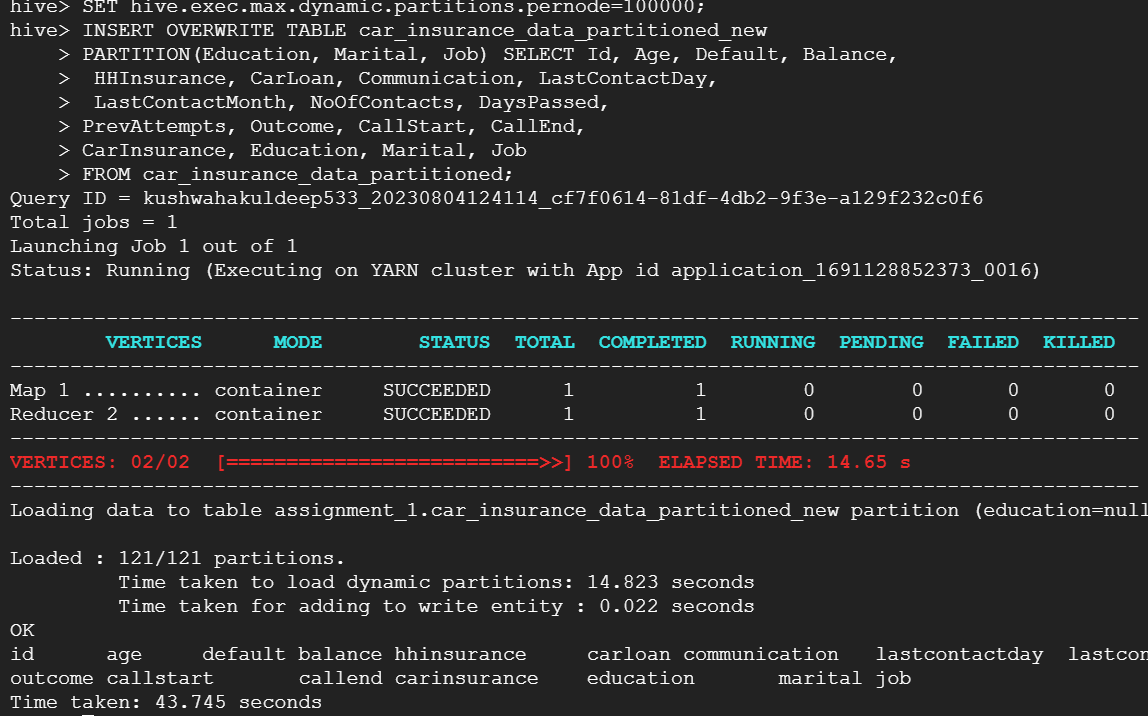
Error:

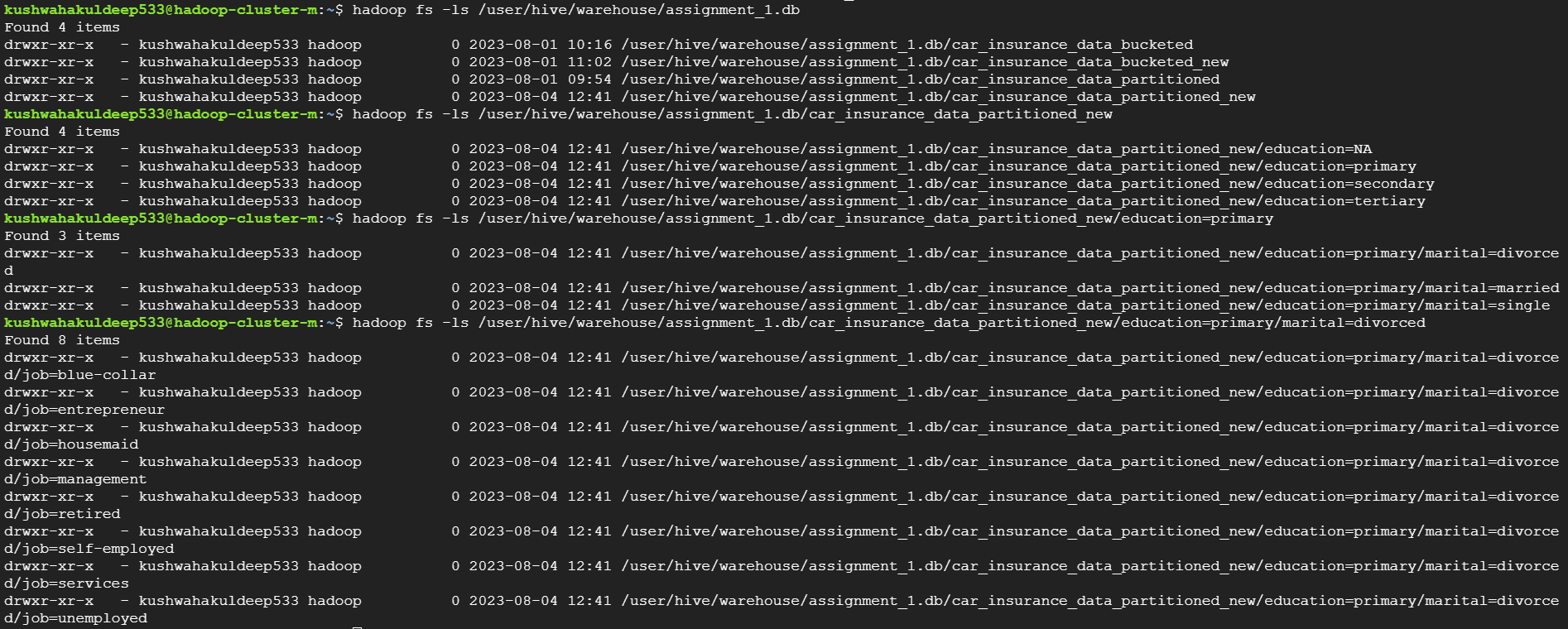
Caused by: org.apache.hadoop.hive.ql.metadata.HiveFatalException: [Error 20004]: Fatal error occurred when node tried to create too many dynamic partitions. The maximum number of dynamic partitions is controlled by hive.exec.max.dynamic.partitions and hive.exec.max.dynamic.partitions.pernode. Maximum was set to 100 partitions per node, number of dynamic partitions on this node: 101

Try setting those properties to higher values.

SET hive.exec.max.dynamic.partitions=100000;

SET hive.exec.max.dynamic.partitions.pernode=100000;





4. Increase the number of buckets in the bucketed table to 10 and redistribute the data.

* In Hive, once a table is bucketed, the number of buckets cannot be changed. The process of bucketing happens at the time of table creation and is immutable. Therefore, in order to increase the number of buckets, you will need to create a new table with the desired number of buckets and then insert data into the new table from the existing one

Hive> set hive.enforce.bucketing=true;

Hive>

create table car\_insurance\_data\_bucketed\_new(

Id INT,

Age INT,

Job STRING,

Marital String,

Education String,

Default INT,

Balance INT,

HHInsurance INT,

CarLoan INT,

Communication STRING,

LastContactDay INT,

LastContactMonth INT,

NoOfContacts INT,

DaysPassed INT,

PrevAttempts INT,

Outcome STRING,

CallStart STRING,

CallEnd STRING,

CarInsurance INT)

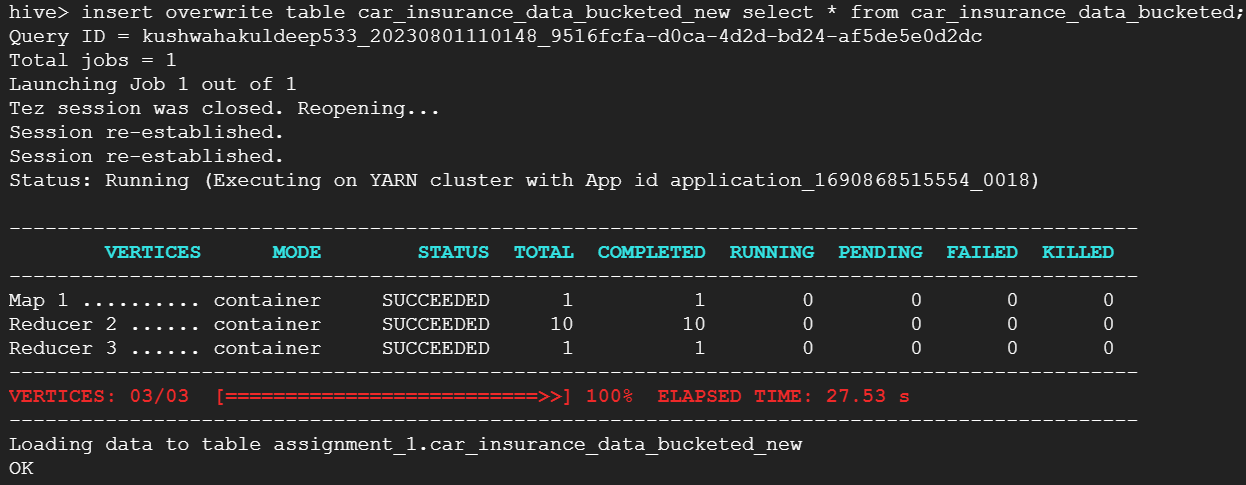
clustered by(Age) into 10 buckets

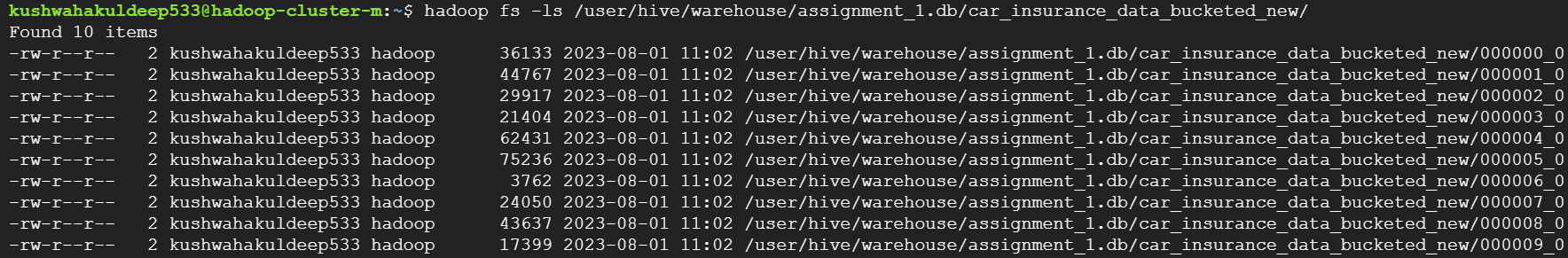
row format delimited

fields terminated by ','

stored as textfile;

Hive> insert overwrite table car\_insurance\_data\_bucketed\_new select \* from car\_insurance\_data\_bucketed;





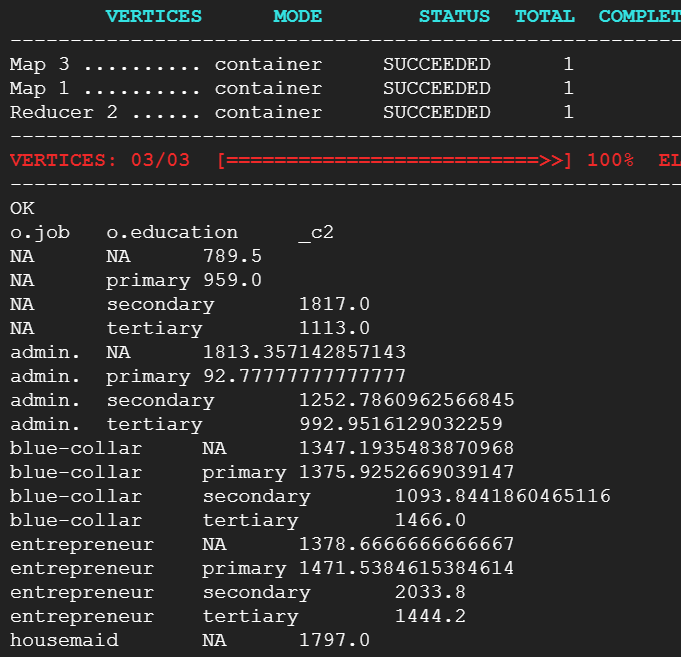
**Problem 5: Optimized Joins**

1. Join the original table with the partitioned table and find out the average 'Balance' for each 'Job' and 'Education' level.

* select o.job,o.education,avg(o.balance)

from car\_insurance\_data o join car\_insurance\_data\_partitioned p

on p.id=o.id group by o.job,o.education;



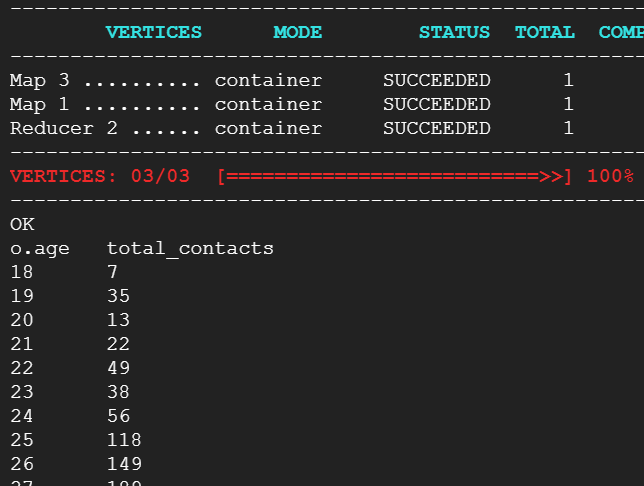
2. Join the original table with the bucketed table and calculate the total 'NoOfContacts' for each 'Age' group.

Select o.Age,sum(o. NoOfContacts) as Total\_Contacts

from car\_insurance\_data o

Join car\_insurance\_data\_bucketed\_new p

on o.id=p.id group by o.age;



3. Join the partitioned table and the bucketed table based on the 'Id' field and find the total balance for each education level and marital status for each age group.

Select o.Age,o.education,o.marital,sum(o.balance)

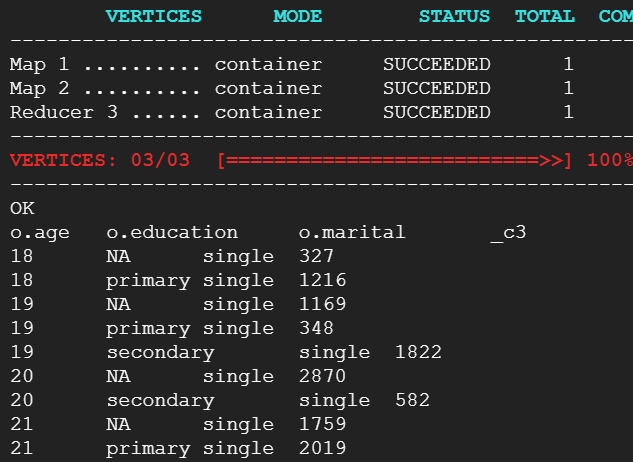
From car\_insurance\_data\_partitioned o

Join car\_insurance\_data\_bucketed p

On o.id=p.id

group by

o.Age,o.education;



**Problem 6: Window Function**

1.Calculate the cumulative sum of 'NoOfContacts' for each 'Job' category, ordered by 'Age'.

🡪

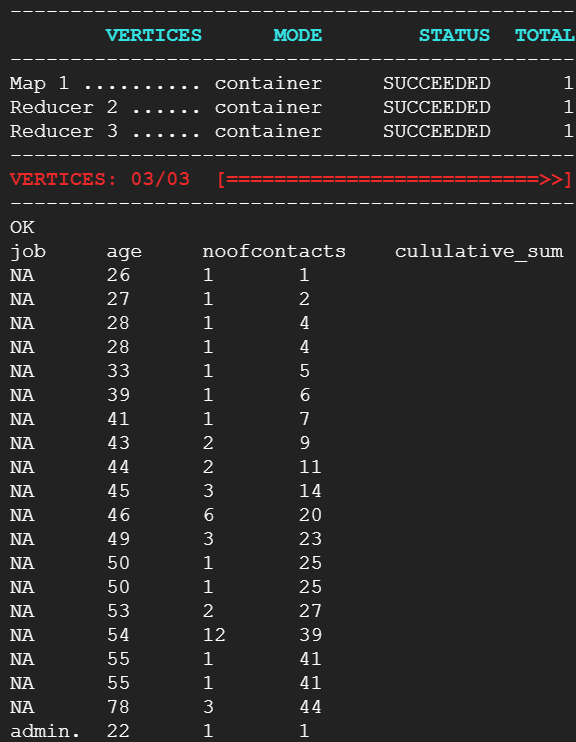
Select job,age, NoOfContacts,sum(NoOfContacts)

over(partition by job order by age)

as cululative\_sum

From car\_insurance\_data

order by job,age;



2. Calculate the running average of 'Balance' for each 'Job' category, ordered by 'Age'.

🡪

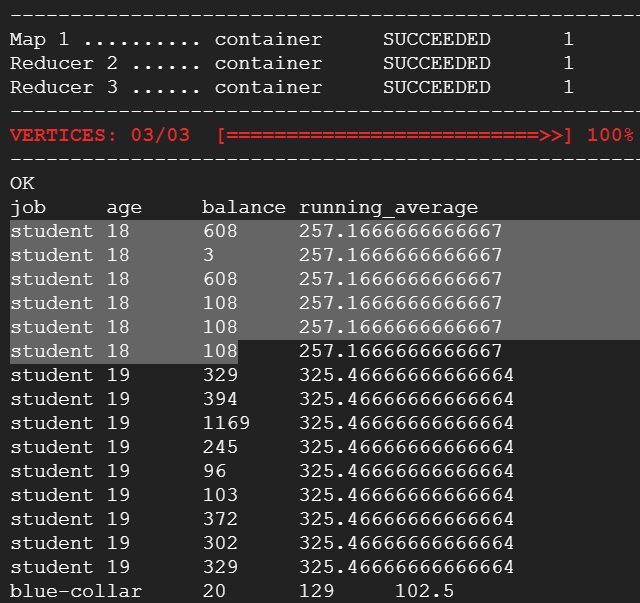
Select job,age,balance, AVG(balance)

over(partition by job order by age)

as running\_average

From car\_insurance\_data

order by age,job;



3. For each 'Job' category, find the maximum 'Balance' for each 'Age' group using window functions.

🡪

Select age, job, balance from

(

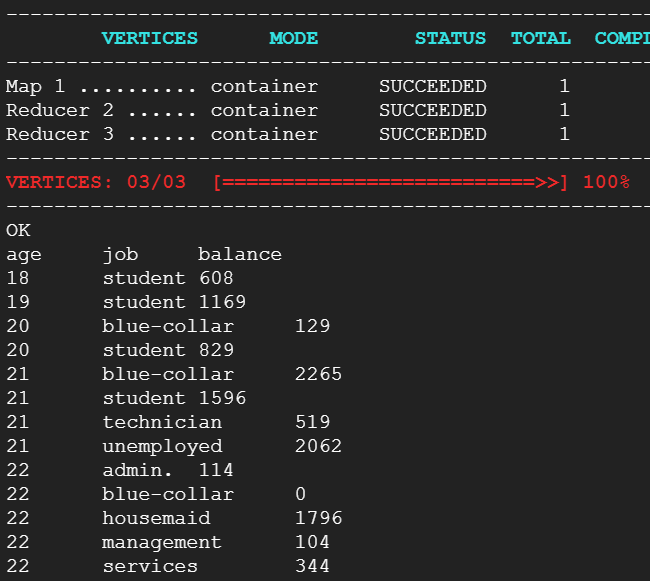
Select age, job, balance,

row\_number ()

over(partition by job, age order by balance desc)

as rn From car\_insurance\_data ) t

Where rn=1 order by age, job;



4. Calculate the rank of 'Balance' within each 'Job' category, ordered by 'Balance' descending.

🡪

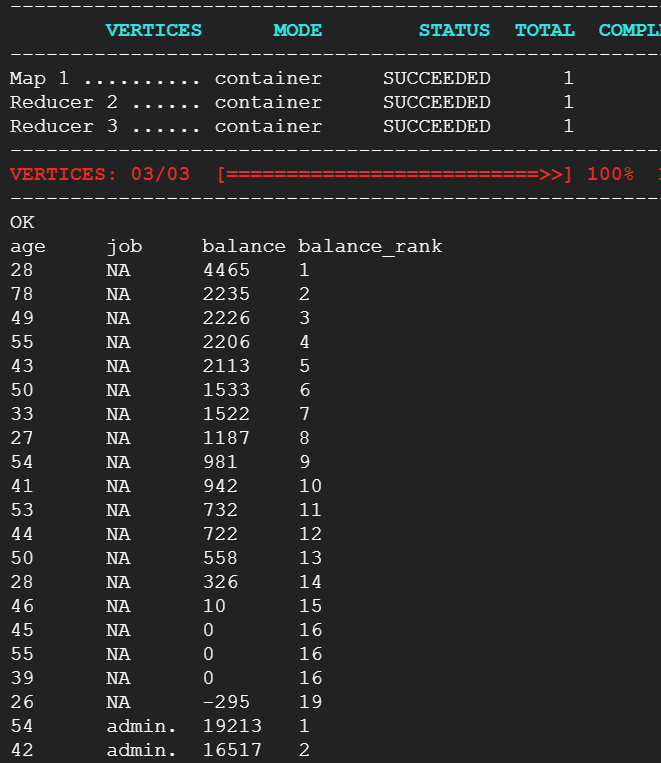
Select age, job, balance,

RANK ()

over(partition by job order by balance desc)

as Balance\_Rank From car\_insurance\_data

order by job,balance desc;



**Problem 7: Advanced Aggregations**

1. Find the job category with the highest number of car insurances.

🡪

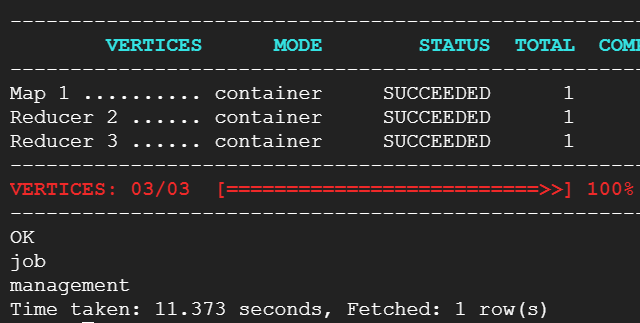
Select job from (select job,count(\*) as car\_insurance\_count

from car\_insurance\_data

where carinsurance=1

group by job) t

order by car\_insurance\_count desc limit 1;



2. Which month has seen the highest number of last contacts?

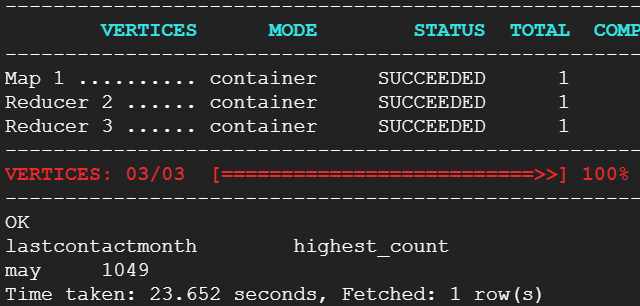
🡪

Select LastContactMonth,count(\*) as highest\_count

from car\_insurance\_data

group by LastContactMonth

order by highest\_count desc limit 1;



3. Calculate the ratio of the number of customers with car insurance to the number of customers without car insurance for each job category.

🡪

Select t1.job, t1. car\_ins\_count/t2. no\_car\_ins\_count

as car\_insurance\_ratio from

(Select job, count(\*) as car\_ins\_count from car\_insurance\_data

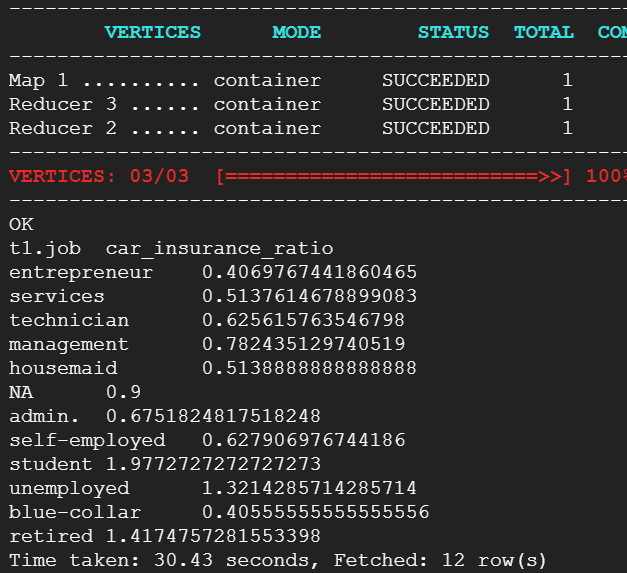
where carinsurance=1 Group by job)t1

join

(Select job, count(\*) as no\_car\_ins\_count from car\_insurance\_data

where carinsurance=0 Group by job)t2

on t1.job=t2.job;



4. Find out the 'Job' and 'Education' level combination which has the highest number of car insurances.

🡪

Select job, education from

(select job, education ,count(\*) as car\_insurance\_count

From car\_insurance\_data

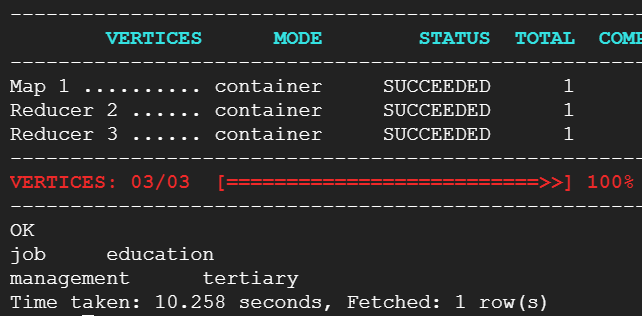
where carinsurance=1

group by job, education

) t

Order by car\_insurance\_count desc

Limit 1;



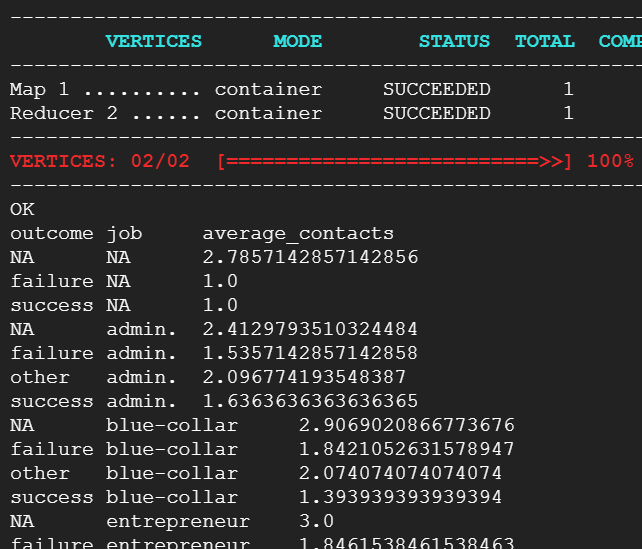
5. Calculate the average 'NoOfContacts' for each 'Outcome' and 'Job' combination.

🡪

SELECT Outcome, Job, AVG(NoOfContacts) AS average\_contacts

FROM car\_insurance\_data

GROUP BY Outcome, Job;



6. Determine the month with the highest total 'Balance' of customers.

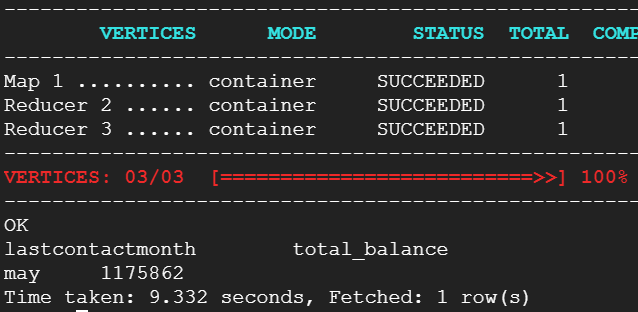
🡪

SELECT LastContactMonth, SUM(Balance) AS total\_balance

FROM car\_insurance\_data

GROUP BY LastContactMonth

ORDER BY total\_balance DESC LIMIT 1;



**Problem 8: Complex joins and aggregations**

1. For customers who have both a car loan and home insurance, find out the average 'Balance' for each 'Education' level.

🡪

Select education,avg(balance) as average\_balance

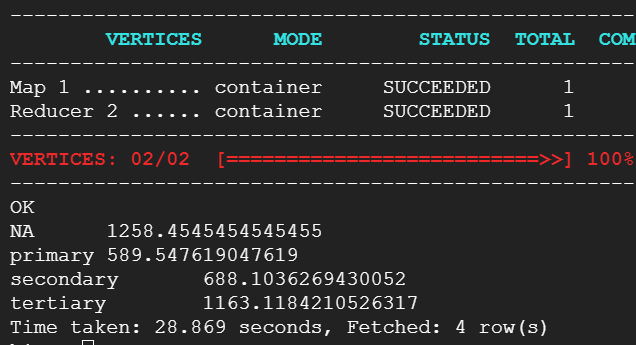
From(

Select education,balance from car\_insurance\_data

Where CarLoan=1 and HHInsurance=1

)t

Group by education;



2. Identify the top 3 'Communication' types for customers with 'CarInsurance', and display their average 'NoOfContacts'.

Select communication, avg(NoOfContacts) as Avg\_Contact from

(

Select communication, NoOfContacts

From car\_insurance\_data

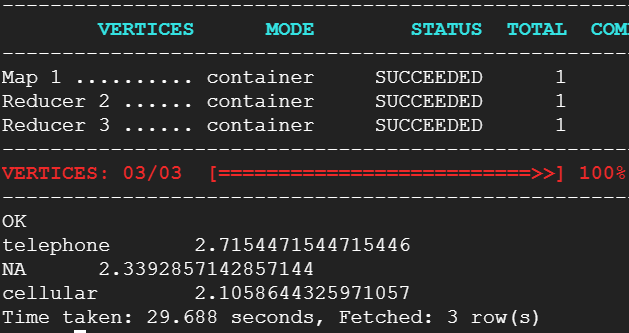
Where carinsurance=1

)t

Group by communication

Order by Avg\_contact desc

Limit 3;



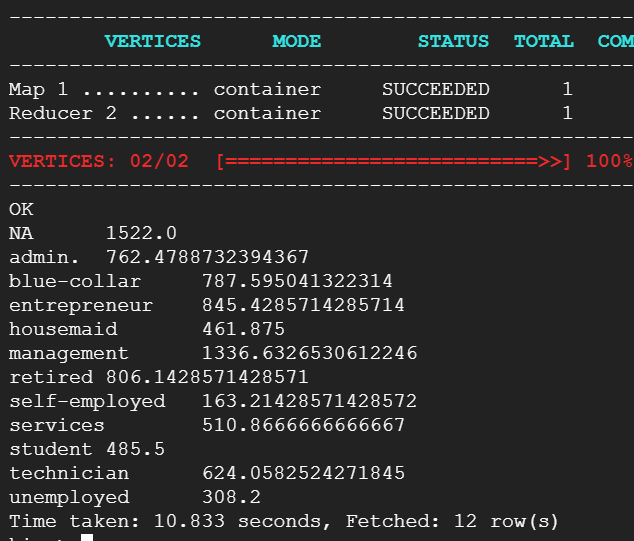
3. For customers who have a car loan, calculate the average balance for each job category.

SELECT Job, AVG(Balance) AS average\_balance

FROM car\_insurance\_data

WHERE CarLoan = 1

Group by job;



4. Identify the top 5 job categories that have the most customers with a 'default', and show their average 'balance'.

SELECT Job, AVG(Balance) AS average\_balance

FROM

( SELECT Job, Balance

FROM car\_insurance\_data

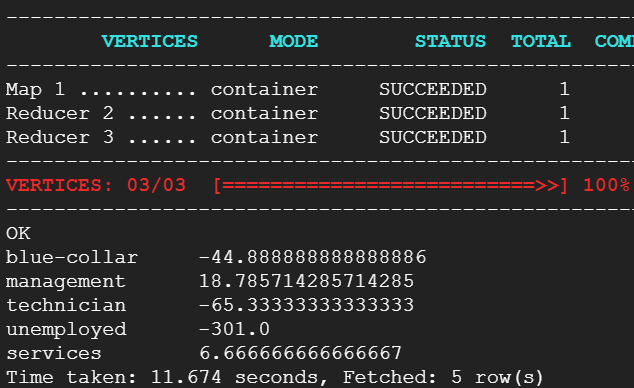
WHERE Default = 1

) t

GROUP BY Job

ORDER BY COUNT(\*) DESC

LIMIT 5;



Select job, avg(balance) as average\_balance,

sum(default) as most\_cust

From (

select job, balance,default from car\_insurance\_data

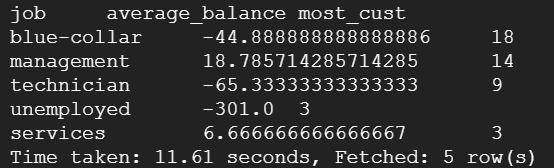
where default=1

)t

Group by job

Order by most\_cust desc

Limit 5;



**Problem 9: Advanced Window Functions**

1. Calculate the difference in 'NoOfContacts' between each customer and the customer with the next highest number of contacts in the same 'Job' category.

🡪

Select c1.id, c1.job, c1.NoOfContacts,

c1.NoOfContacts - c2.NextHighestContact as ContactDifference

from Car\_insurance\_data c1

Join

(select c1.job, c1.NoOfContacts, min(c2.NoOfContacts) as nextHighestContact

From car\_insurance\_data c1

Left join car\_insurance\_data c2

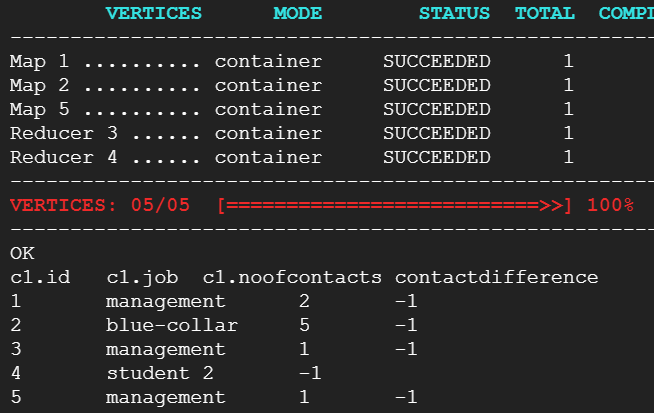
On c1.job=c2.job and c1.NoOfContacts < c2.NoOfContacts

GROUP BY c1.Job, c1.NoOfContacts

)c2

On c1.job=c2.job and c1.NoOfContacts = c2.NoOfContacts

Order by c1.id asc;



2. For each customer, calculate the difference between their 'balance' and the average 'balance' of their 'job' category.

🡪

Select c1.id, c1.job, c1.Balance, c1.balance - j.AvgBalance as balance\_difference

From car\_insurance\_data c1

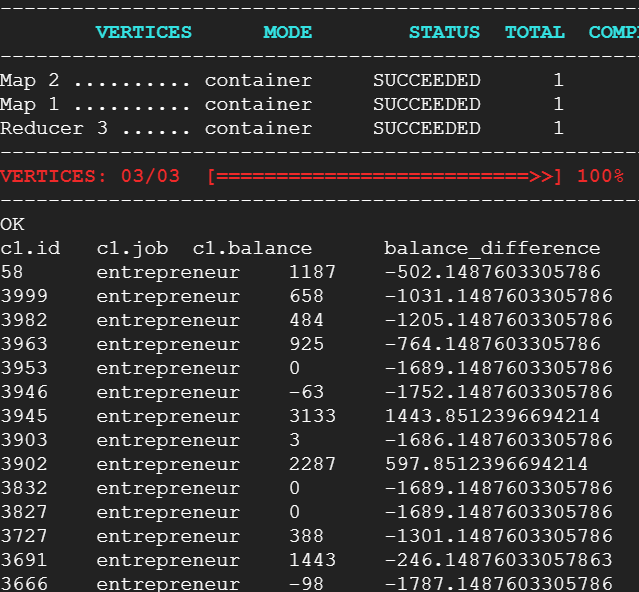
JOIN ( Select job, avg(balance) as AvgBalance

From car\_insurance\_data

Group by job

) j

On c1.job=j.job;



3. For each 'Job' category, find the customer who had the longest call duration.

🡪

Select id, job, t.callduration

From (select id, job, (callend - callstart)as callduration,

Row\_number() over(partition by job)

As rn from car\_insurance\_data

order by callduration desc

)t where rn=1;

4. Calculate the moving average of 'NoOfContacts' within each 'Job' category, using a window frame of the current row and the two preceding rows. Problem

🡪

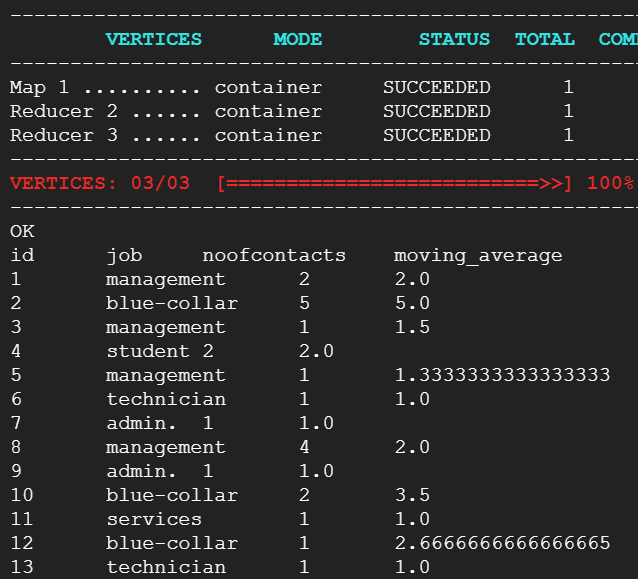
Select id, job,NoOfContacts, avg(NoOfContacts)

Over (partition by job order by id rows between

2 preceding and current row) as moving\_average

From car\_insurance\_data

Order by id limit 20;



**10: Performance Tuning**

1. Experiment with different file formats (like ORC, Parquet) and measure their impact on the performance of your Hive queries.

🡪

CREATE EXTERNAL TABLE car\_data\_parquet (

Id INT,Age INT,Job STRING,

Marital STRING,Education STRING,

Default INT,Balance INT,HHInsurance INT,

CarLoan INT,Communication STRING,

LastContactDay INT, LastContactMonth string,

NoOfContacts INT,DaysPassed INT,

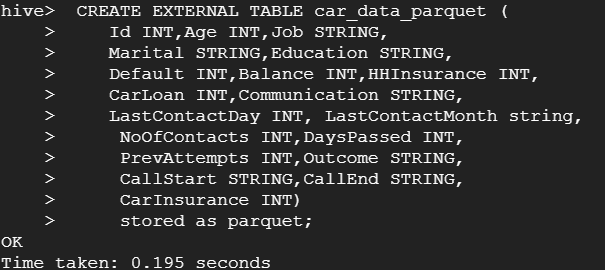
PrevAttempts INT,Outcome STRING,

CallStart STRING,CallEnd STRING,

CarInsurance INT)

stored as parquet;

from car\_data insert overwrite table car\_data\_parquet select \*;



Select c1.id, c1.job, c1.Balance, c1.balance - j.AvgBalance as balance\_difference

From car\_data\_parquet c1

JOIN ( Select job, avg(balance) as AvgBalance

From car\_data\_parquet

Group by job

) j

On c1.job=j.job;



**ORC**

CREATE EXTERNAL TABLE car\_data\_orc (

Id INT,Age INT,Job STRING,

Marital STRING,Education STRING,

Default INT,Balance INT,HHInsurance INT,

CarLoan INT,Communication STRING,

LastContactDay INT, LastContactMonth string,

NoOfContacts INT,DaysPassed INT,

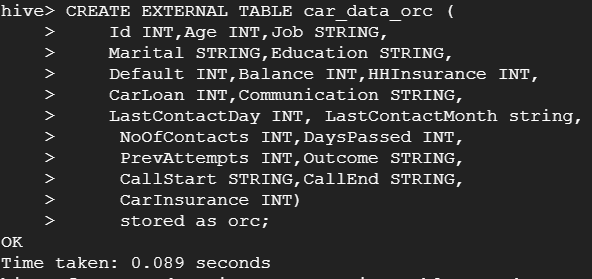
PrevAttempts INT,Outcome STRING,

CallStart STRING,CallEnd STRING,

CarInsurance INT)

stored as orc;

from car\_data insert overwrite table car\_data\_orc select \*;



Select c1.id, c1.job, c1.Balance, c1.balance - j.AvgBalance as balance\_difference

From car\_data\_orc c1

JOIN ( Select job, avg(balance) as AvgBalance

From car\_data\_orc

Group by job

) j

On c1.job=j.job;



2. Use different levels of compression and observe their effects on storage and query performance.

3. Compare the execution time of join queries with and without bucketing.

4. Optimize your Hive queries using different Hive optimization techniques (for example, predicate pushdown, map-side joins, etc.). Discuss the difference in performance.