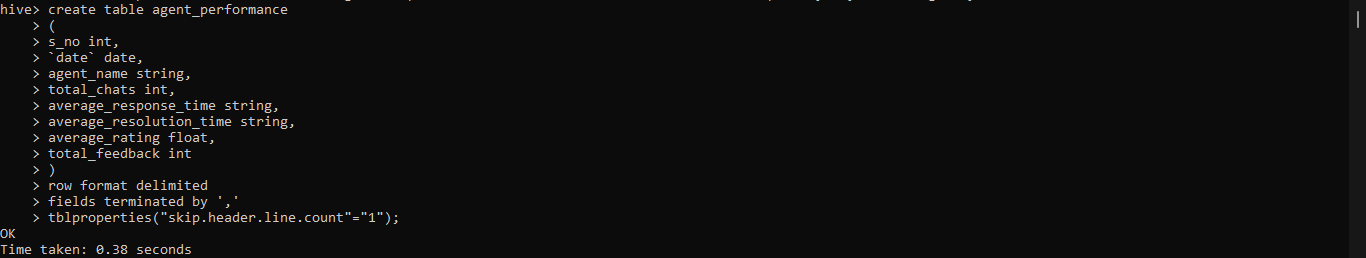
Hive mini project\_01 solution:

As hive is installed using docker image. At first we need to go hive server location and copy the files from local to hadoop using following commands:

* docker exec -it 773c55b5faa6286a691211f5260375ef30bbc2d67a8b5f432675cb6a010fa8f2 /bin/bash
* docker cp AgentLogingReport.csv 773c55b5faa6286a691211f5260375ef30bbc2d67a8b5f432675cb6a010fa8f2:/home
* docker cp AgentPerformance.csv 773c55b5faa6286a691211f5260375ef30bbc2d67a8b5f432675cb6a010fa8f2:/home

Q1) Create a schema based on the given dataset

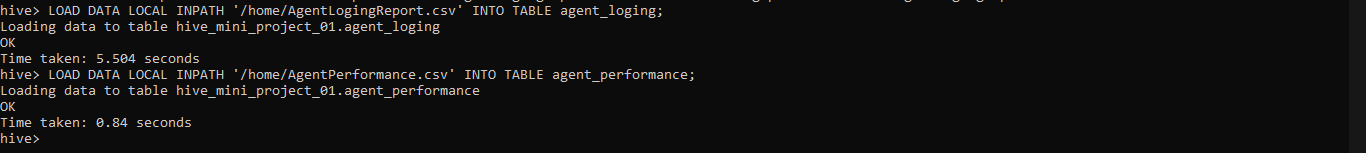




Q2) Dump the data inside the hdfs in the given schema location.

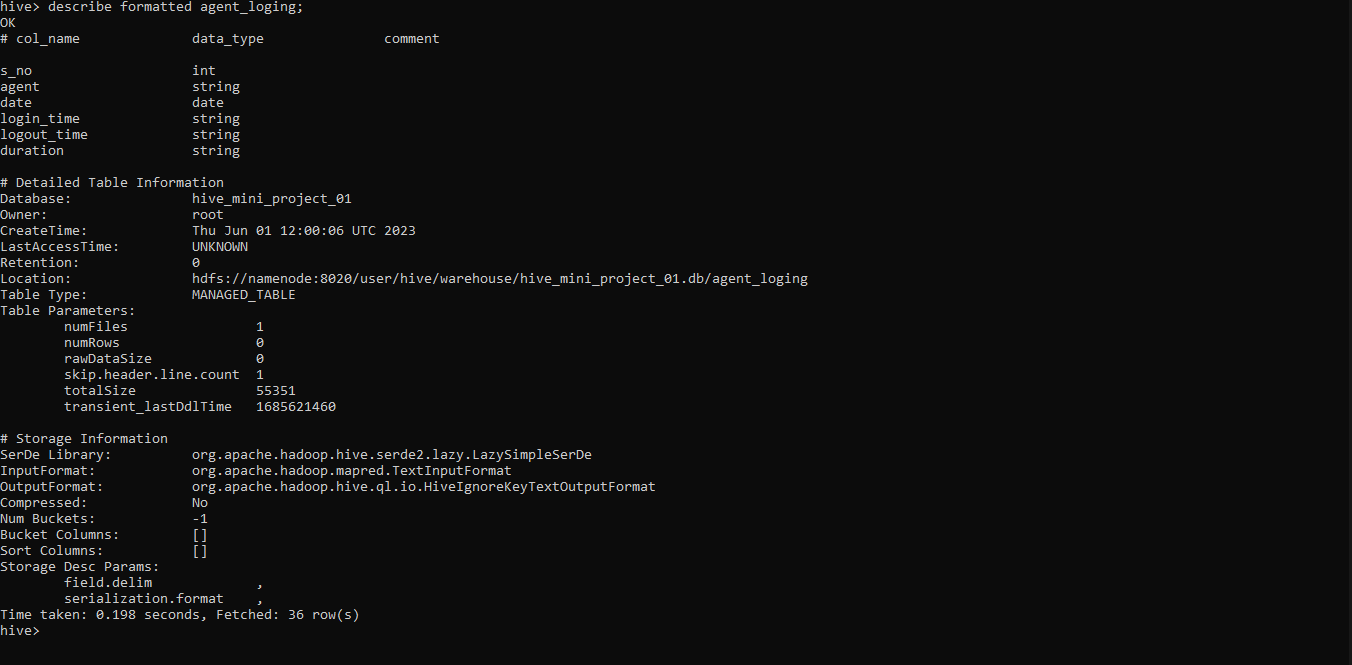
Use the following commands to load data from Hadoop to tables

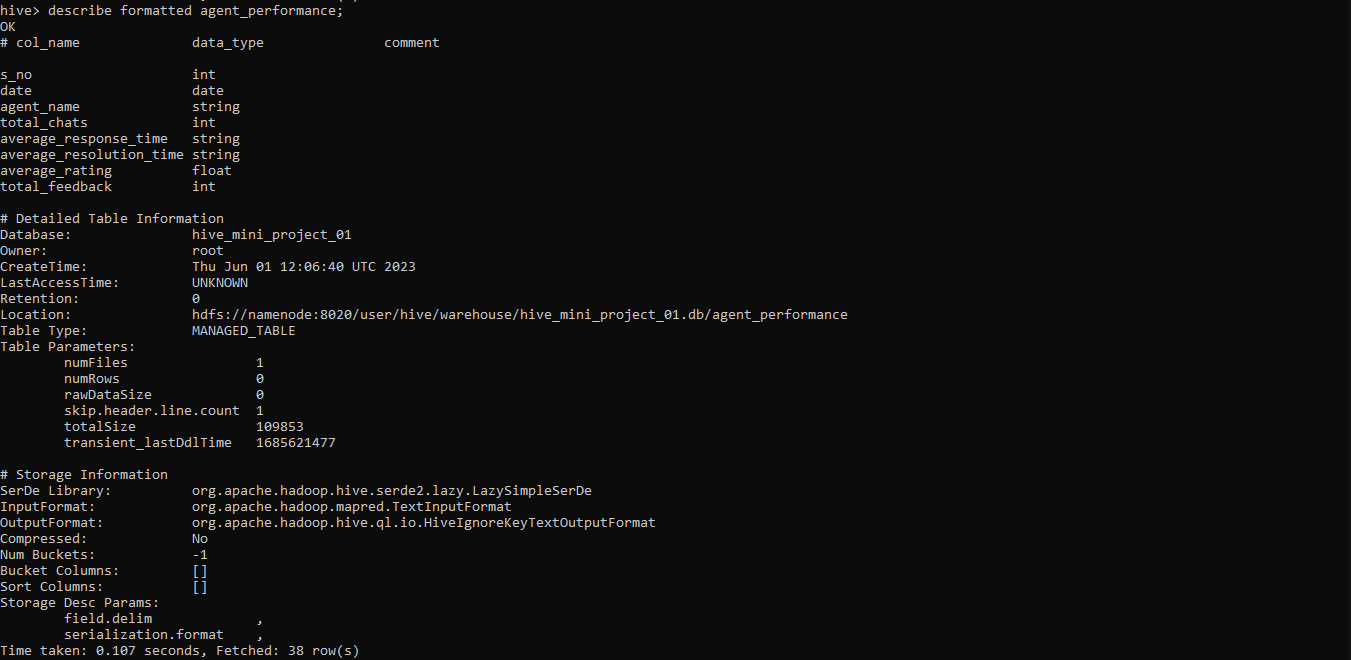
* LOAD DATA LOCAL INPATH '/home/AgentLogingReport' INTO TABLE agent\_loging;
* LOAD DATA LOCAL INPATH '/home/AgentPerformance.csv' INTO TABLE agent\_performance;

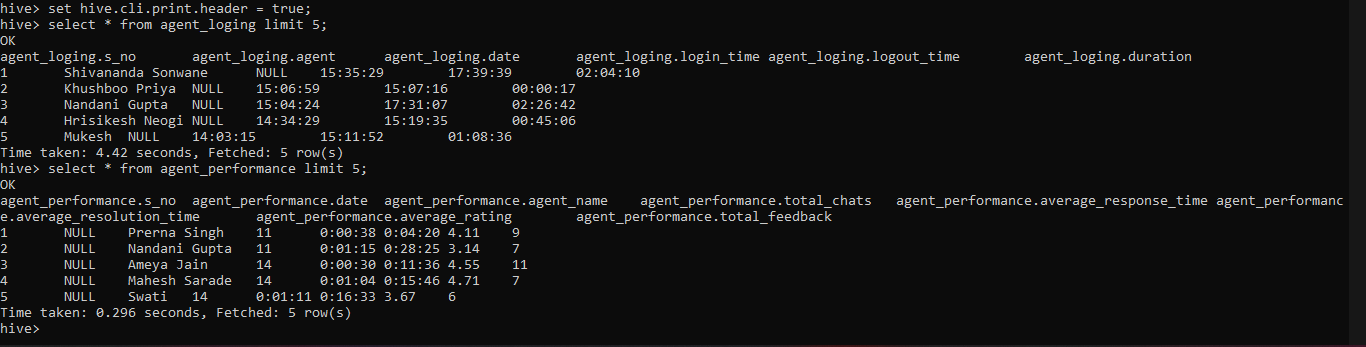


In order to get information about tables, we can use following:

* describe formatted agent\_loging
* describe formatted agent\_performance

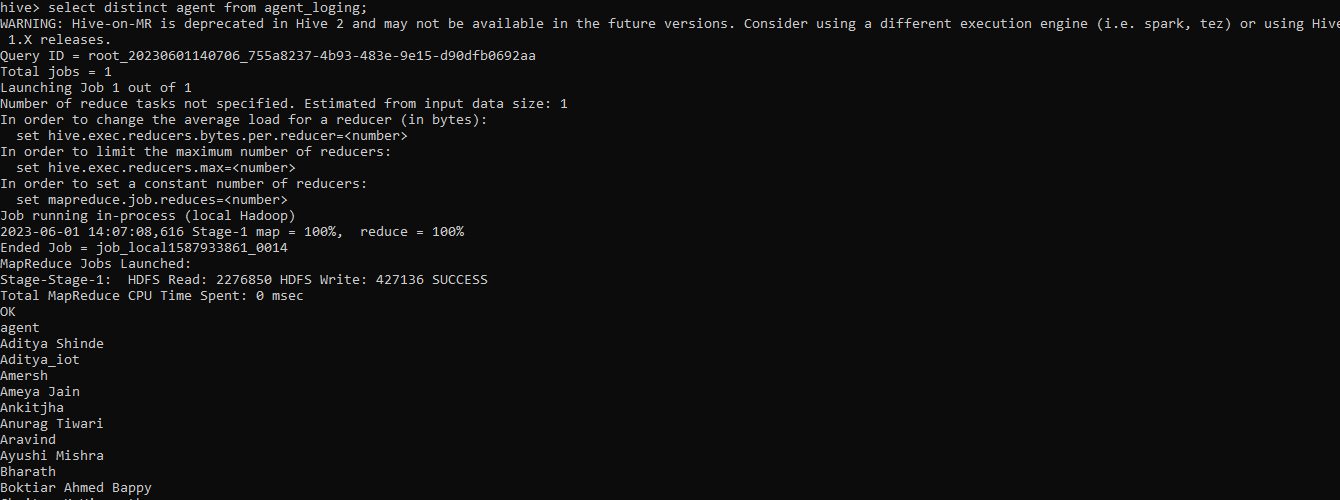






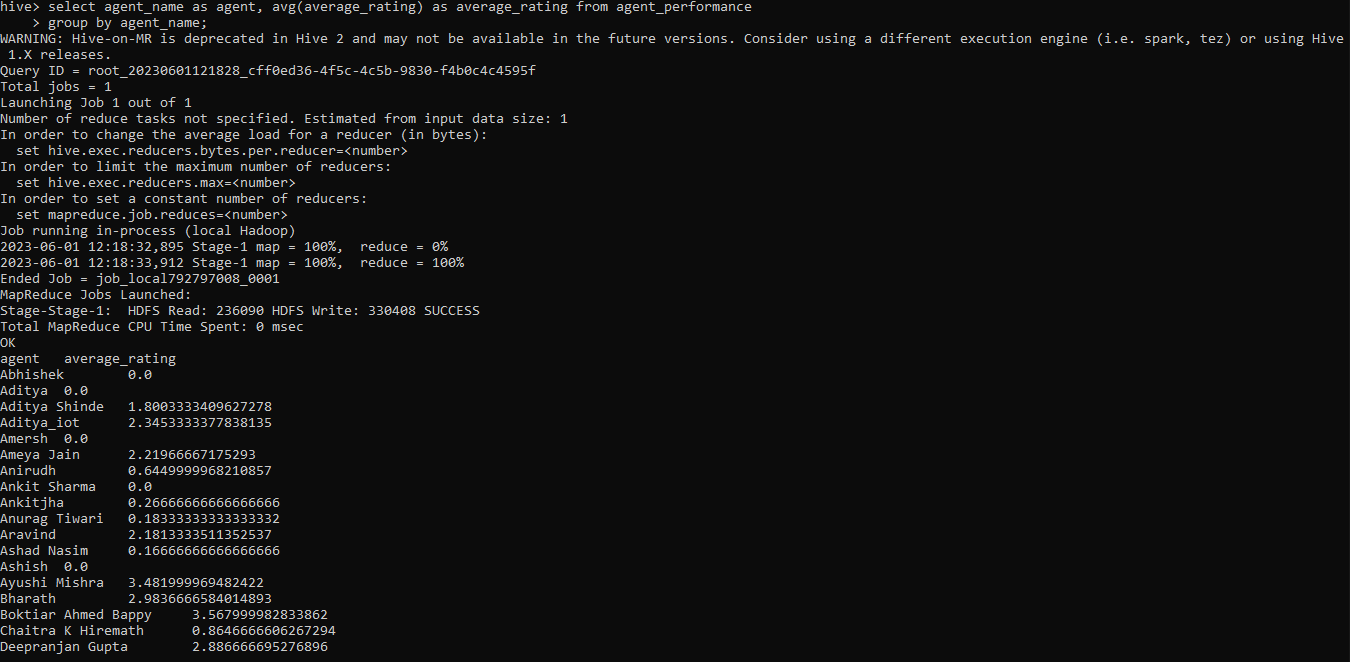
3. List of all agents' names.

* select distinct agent from agent\_loging;



4. Find out agent average rating.

* select agent\_name as agent, avg(average\_rating) as average\_rating from agent\_performance group by agent\_name;

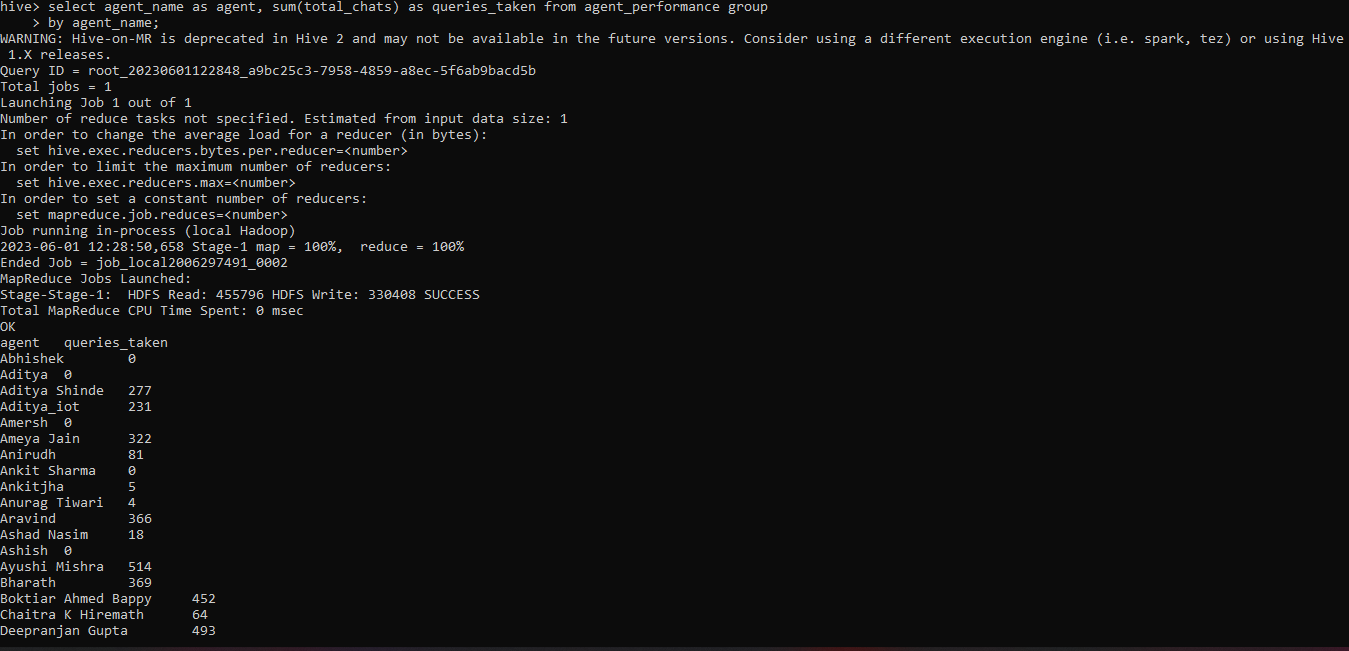


5. Total working days for each agent.

* select agent\_name as agent, count(distinct date) as number\_of\_working\_days from agent\_performance group by agent\_name;

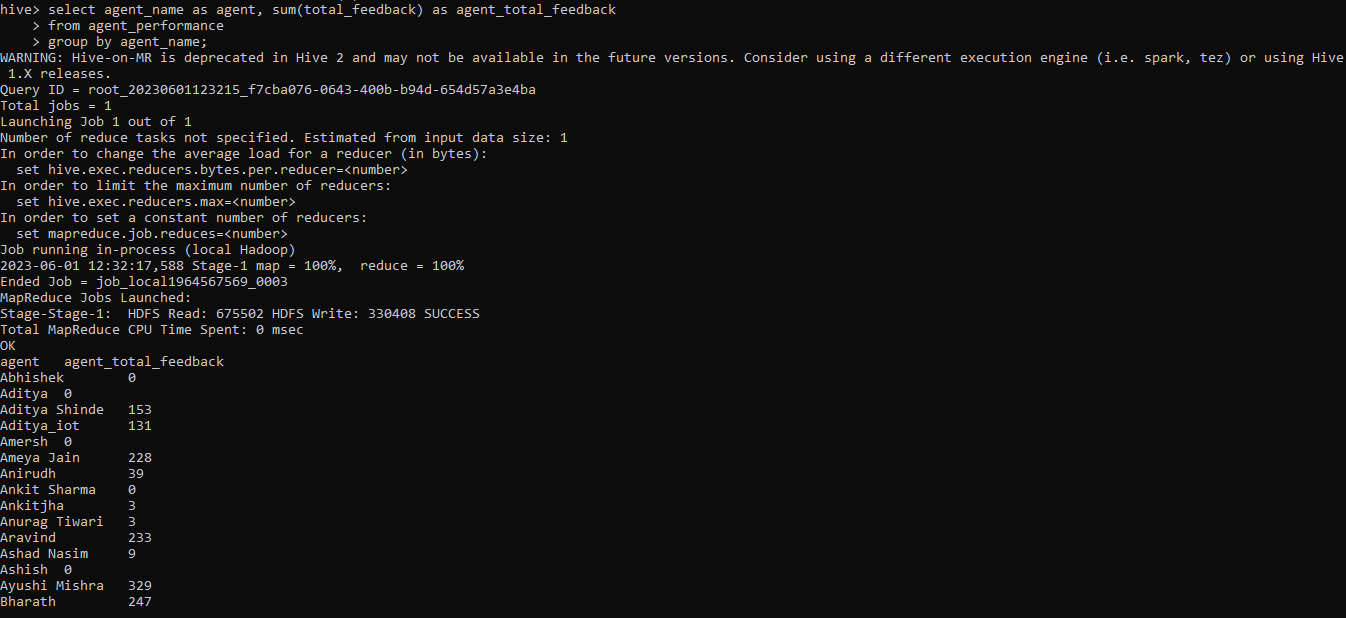
6. Total query that each agent has taken

* select agent\_name as agent, sum(total\_chats) as queries\_taken from agent\_performance group by agent\_name;



7. Total Feedback that each agent has received Solution

* select agent\_name as agent, sum(total\_feedback) as feedbacks\_received from agent\_performance group by agent\_name;



8. Agent name who have average rating between 3.5 to 4

* select agent\_name as agent

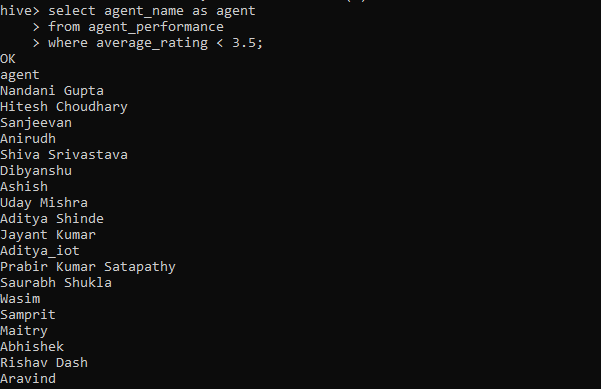
from agent\_performance

where average\_rating between 3.5 and 4;



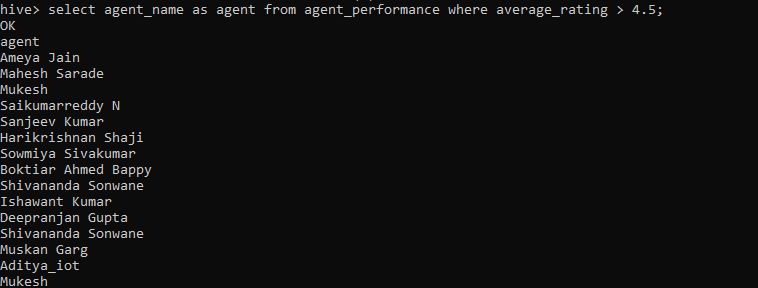
9. Agent name who have rating less than 3.5

* select agent\_name as agent from agent\_performance where average\_rating < 3.5;



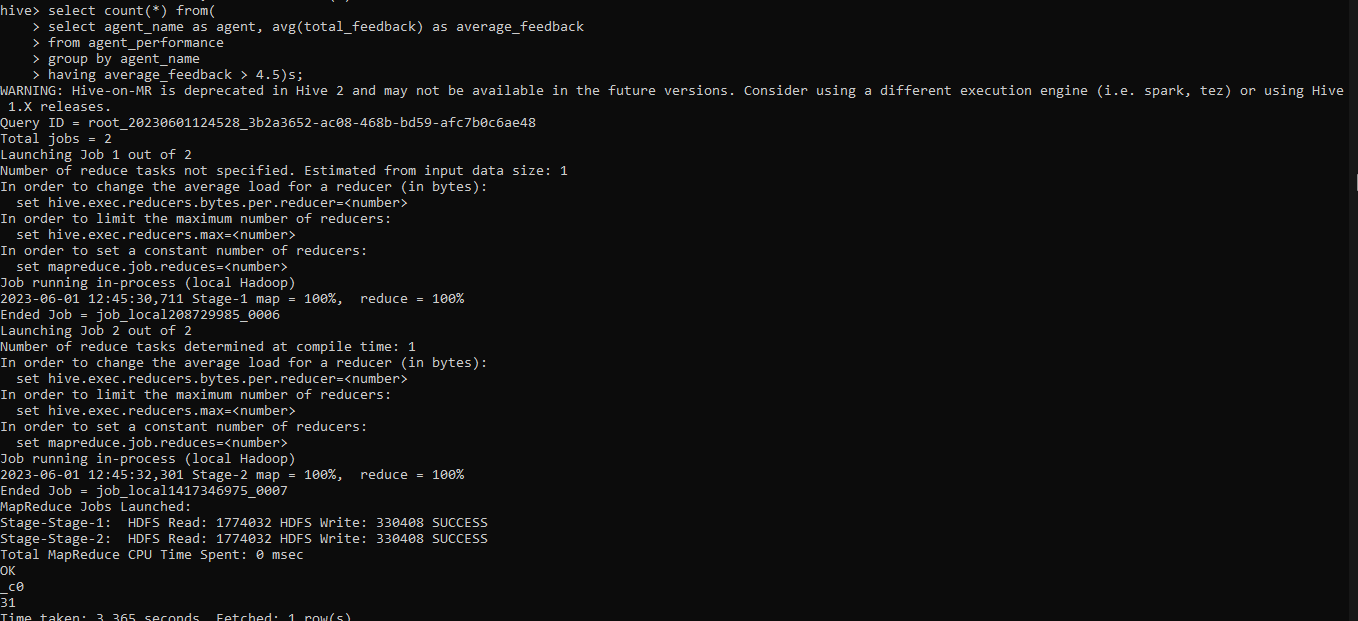
10. Agent name who have rating more than 4.5

* select agent\_name as agent from agent\_performance where average\_rating > 4.5;



11. How many feedback agents have received more than 4.5 average

* select count(\*) from(select agent\_name as agent, avg(total\_feedback) as average\_feedback from agent\_performance group by agent\_name having average\_feedback > 4.5)s;



12. Average weekly response time for each agent.

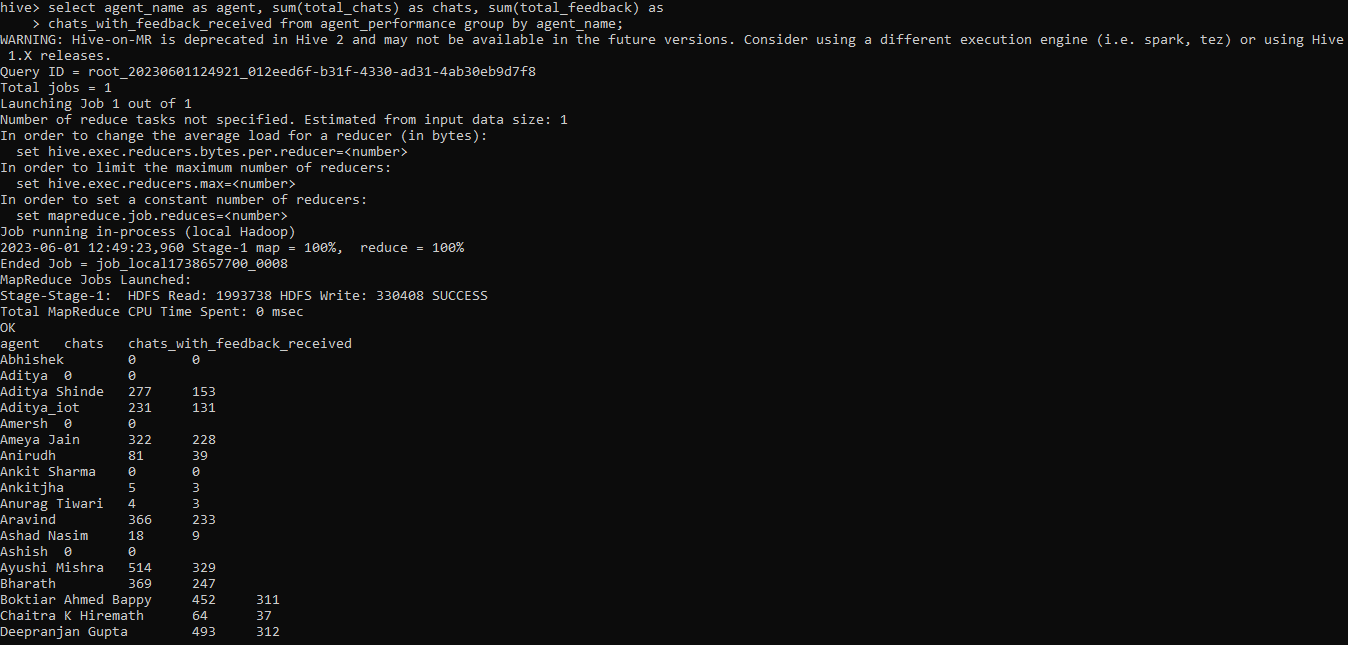
* select agent, avg(weekly\_response\_time\_in\_sec) as avg\_weekly\_response\_time\_in\_sec from (select week, agent, sum((time[0]\*3600+time[1]\*60+time[2])) as weekly\_response\_time\_in\_sec from(select agent\_name as agent, weekofyear(date) as week, split(average\_response\_time,':') as time from agent\_performance) t group by agent, week)s group by agent;

13. average weekly resolution time for each agents.

* select agent, avg(weekly\_resolution\_time\_in\_sec) as avg\_weekly\_resolution\_time\_in\_sec from (select week, agent, sum((time[0]\*3600+time[1]\*60+time[2])) as weekly\_resolution\_time\_in\_sec from(select agent\_name as agent, weekofyear(date) as week, split(average\_resolution\_time,':') as time from agent\_performance) t group by agent, week)s group by agent;

14. Find the number of chats on which they have received a feedback

* select agent\_name as agent, sum(total\_chats) as chats, sum(total\_feedback) as chats\_with\_feedback\_received from agent\_performance group by agent\_name;



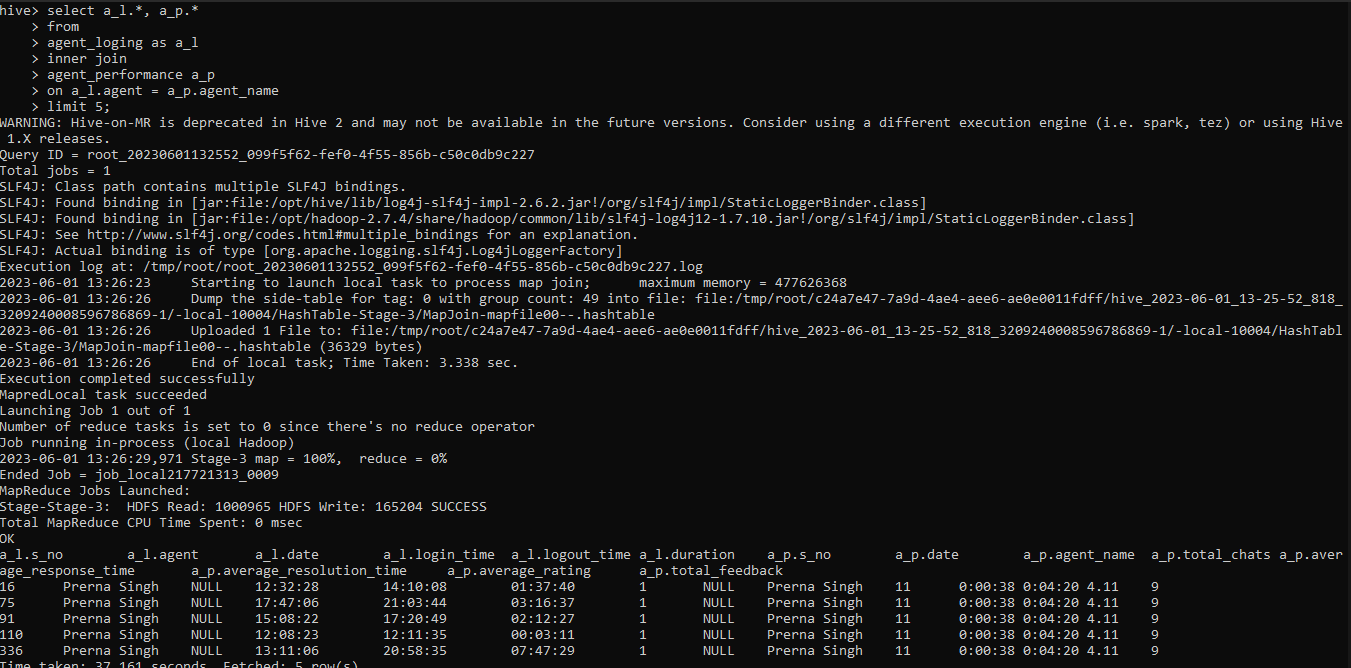
15. Total contribution hour for each and every agents weekly basis.

* select week, agent, sum((time[0]\*3600+time[1]\*60+time[2])/3600) as total\_hrs\_contributed from(select agent, weekofyear(date) as week, split(duration,':') as time from agent\_loging) t group by agent, week;

16. Perform inner join, left join and right join based on the agent column and after joining the table export that data into your local system.

a. inner join

* INSERT OVERWRITE DIRECTORY '/tmp/agent\_data'
* ROW FORMAT DELIMITED
* FIELDS TERMINATED BY ','
* select a\_l.\*, a\_p.\*
* from
* agent\_loging as a\_l
* inner join
* agent\_performance a\_p
* on a\_l.agent = a\_p.agent\_name
* limit 5

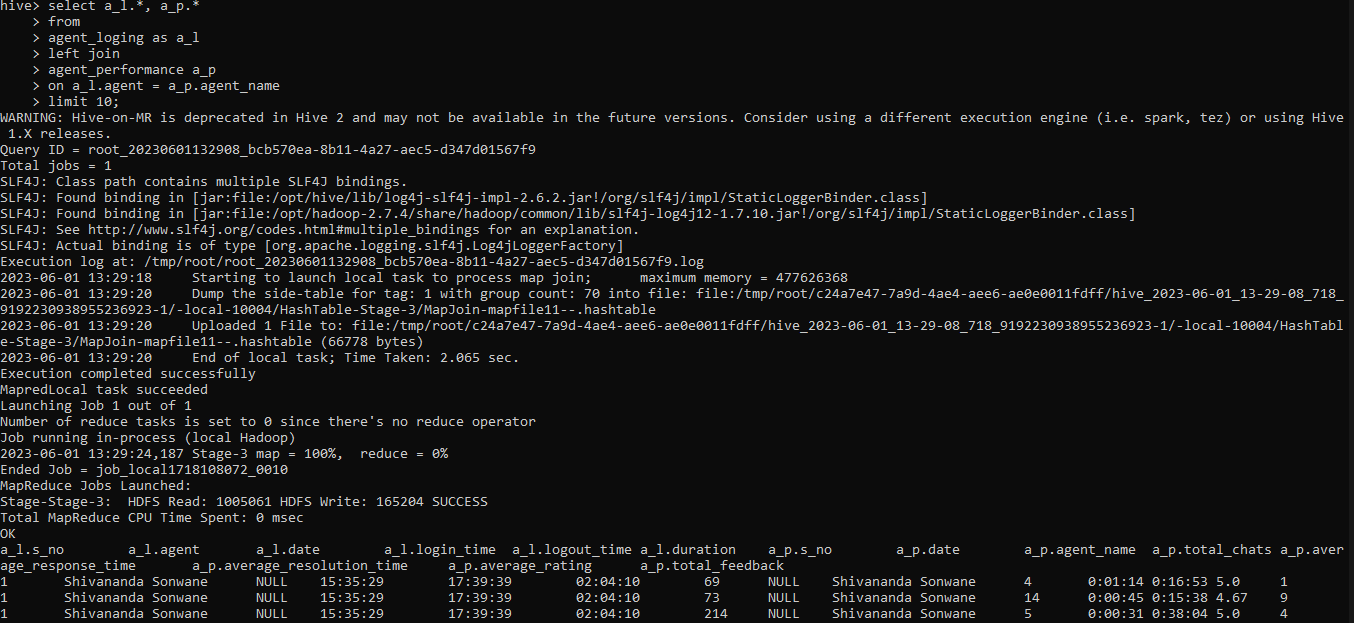


* hadoop fs -getmerge /tmp/agent\_data result.csv

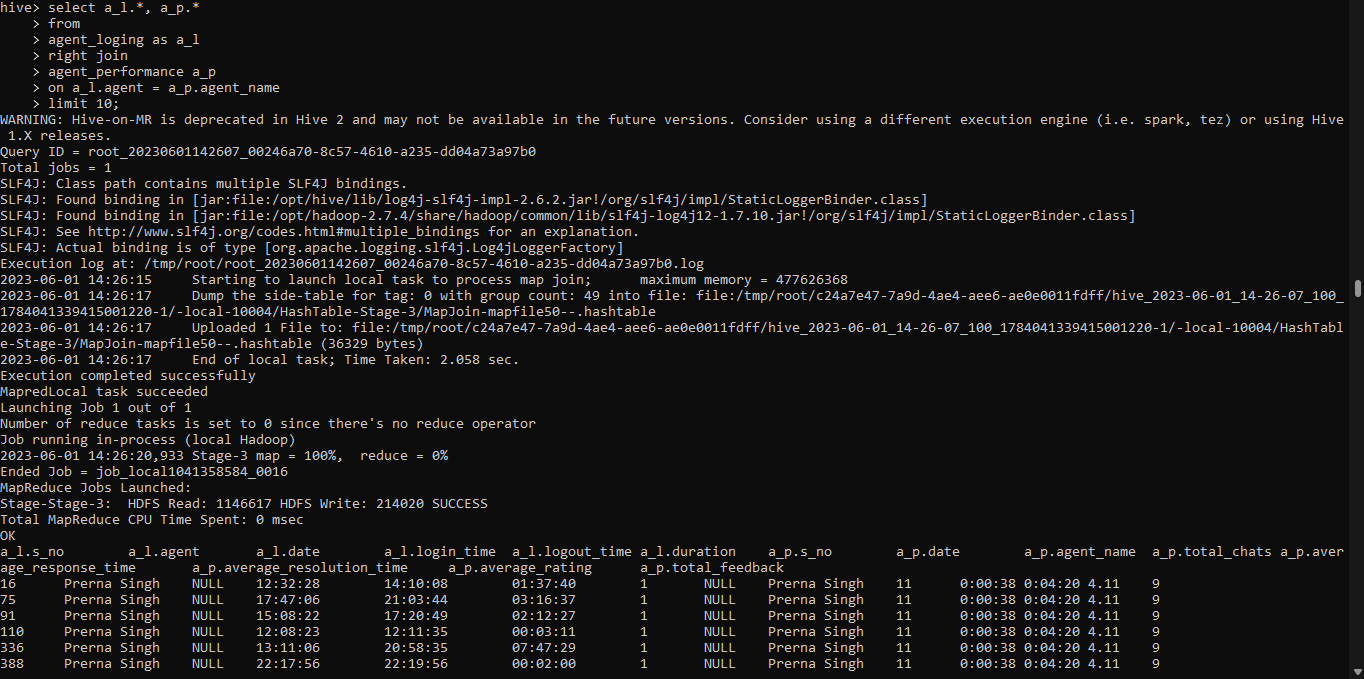


b. left join

* INSERT OVERWRITE DIRECTORY '/tmp/agent\_data'
* ROW FORMAT DELIMITED
* FIELDS TERMINATED BY ','
* select a\_l.\*, a\_p.\*
* from
* agent\_loging as a\_l
* inner join
* agent\_performance a\_p
* on a\_l.agent = a\_p.agent\_name
* limit 10

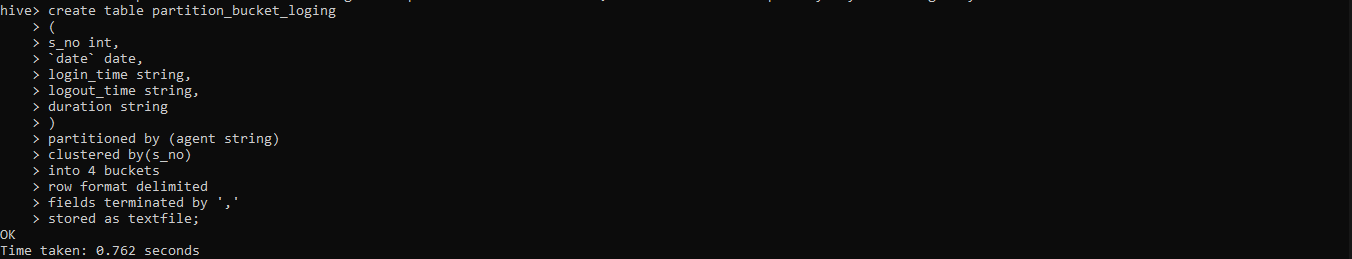


c. Right join



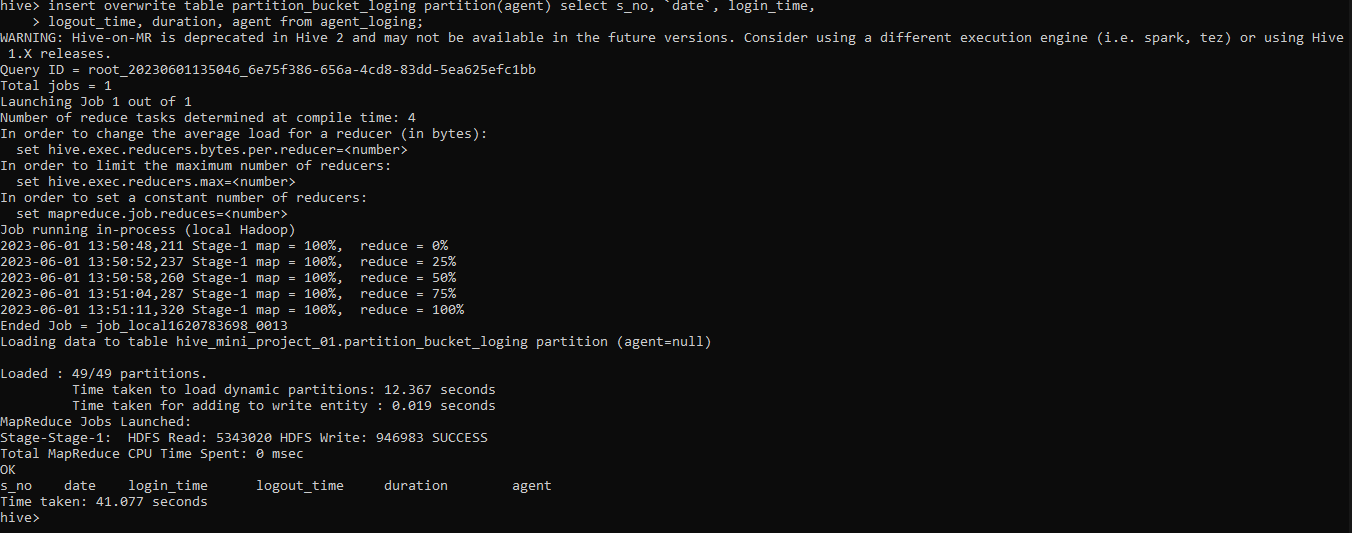
17. Perform partitioning on top of the agent column and then on top of that perform bucketing for each partitioning.

* set the below mentioned properties to be true.
* hive> set hive.exec.dynamic.partition=true;
* hive> set hive.exec.dynamic.patition.mode=nonstrict;
* Create partition bucket table with same columns in agent\_loging table



Load the data from agent\_loging to packet\_bucket\_loging table using following command.

* insert overwrite table partition\_bucket\_loging partition(agent) select s\_no, `date`, login\_time, logout\_time, duration, agent from agent\_loging;



=====================================================================================