Contents

[1. *Create* a *schema* based on the given dataset 2](#_Toc117109534)

[2. Dump the data inside the hdfs in the given schema location. 2](#_Toc117109535)

[3. List of all agents' names. 3](#_Toc117109536)

[4. Find out agent average rating. 3](#_Toc117109537)

[5. Total working days for each agents 3](#_Toc117109538)

[6. Total query that each agent have taken 3](#_Toc117109539)

[7. Total Feedback that each agent have received 3](#_Toc117109540)

[8. Agent name who have average rating between 3.5 to 4 3](#_Toc117109541)

[9. Agent name who have rating less than 3.5 3](#_Toc117109542)

[10. Agent name who have rating more than 4.5 3](#_Toc117109543)

[11. How many feedback agents have received more than 4.5 average 3](#_Toc117109544)

[12. average weekly response time for each agent 4](#_Toc117109545)

[13. average weekly resolution time for each agents 4](#_Toc117109546)

[14. Find the number of chat on which they have received a feedback 4](#_Toc117109547)

[15. Total contribution hour for each and every agents weekly basis 4](#_Toc117109548)

[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. 4](#_Toc117109549)

[17. Perform partitioning on top of the agent column and then on top of that perform bucketing for each partitioning. 5](#_Toc117109550)

# 1. Create a schema based on the given dataset

Created database mini\_project\_1 with command:

Create database mini\_project\_1;

Accessing database with command:

use mini\_project\_1;

Create table schema according to datasets provided, starting with agent login report as ‘agent\_login’

create table agent\_login

(

sl\_no int,

agent string,

date string,

login\_time string,

logout\_time string,

duration string

)

row format delimited

fields terminated by ',';

# 2. Dump the data inside the hdfs in the given schema location.

Load data into the table use command:

load data local inpath ‘file:///home/cloudera/mini\_Project\_1/ AgentLogingReport.csv’ into table agent\_login

Create table schema for second dataset e.i. AgentPerformance using command:

Create table agent\_performance

(

sl\_no int,

date string,

agent string,

total\_chats int,

average\_response\_time string,

average\_resolution\_time string,

average\_rating float,

total\_feedback int

)

row format delimited

fields terminated by ‘,’;

# 3. List of all agents' names.

Select distinct agent as agent from agent\_login;

# 4. Find out agent average rating.

Select agent, avg(average\_rating) from agent\_performance group by agent limit 5;

# 5. Total working days for each agents

Select agent, count(distinct date) from agent\_login group by agent;

# 6. Total query that each agent have taken

Select agent, sum(total\_chats) from agent\_performance group by agent limit 5;

# 7. Total Feedback that each agent have received

Select agent, sum(total\_feedback) form agent\_performance group by agent limit 5;

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

Select agent, average\_rating from agent\_performance where average\_rating between 3.5 and 4 limit 5;

# 9. Agent name who have rating less than 3.5

Select agent, average\_rating from agent\_performance where average\_rating < 3.5 order by average\_rating desc limit 5;

# 10. Agent name who have rating more than 4.5

Select agent, average\_rating from agent\_performance where average\_rating > 4.5 order by average\_rating desc limit 5;

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

Select agent, count(total\_feedback), average\_rating from agent\_performance where average\_rating > 4.5 group by agent, average\_rating limit 5;

# 12. average weekly response time for each agent

select s.agent, avg(col1[0]\*3600+col1[1]\*60+col1[2])/3600 from (select agent, split(average\_response\_time,':') as col1 from agent\_performance)s group by s.agent;

# 13. average weekly resolution time for each agents

select s.agent, avg(col1[0]\*3600+col1[1]\*60+col1[2])/3600 from (select agent, split(average\_resolution\_time,':') as col1 from agent\_performance)s group by s.agent;

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

Select agent, sum(total\_chats), sum(total\_feedback) from agent\_performance where total\_feedback >0 group by agent, total\_chats , total\_feedback limit 10;

Select agent, total\_chats, total\_feedback from agent\_performance select (where total\_feedback >0 group by agent, total\_chats , total\_feedback) limit 10;

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

Select s.agent, sum(col1[0]\*3600+col1[1]\*60+col1[2])/3600, s.weekly from(select agent, split(duration,':') as col1, weekofyear(Date) as weekly from agent\_logging)s group by s.agent, s.weekly limit 2;

# 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.

**> Inner join**

Select a.agent, a.date, a.duration, b.total\_chats, b.total\_feedback From agent\_login a Join agent\_performance b on a.agent=b.agent limit 10;

hive - e‘Select a.agent, a.date, a.duration, b.total\_chats, b.total\_feedback From agent\_login a Join agent\_performance b on a.agent=b.agent’ > /home/cloudera/Desktop/hanif\_project/innerjoin.csv;

**>Left Join**

Select a.agent, a.date, b.total\_chats, b.total\_feedback, a.duration from agent\_login a Left Join agent\_performance b on a.agent=b.agent limit 10;

hive - e ‘Select a.agent, a.date, b.total\_chats, b.total\_feedback, a.duration from agent\_login a Left Join agent\_performance b on a.agent=b.agent’ > /home/cloudera/Desktop/hanif\_project/leftjoin.csv;

**>Right Join**

Select a.agent, a.date, b.total\_chats, b.total\_feedback, a.duration from agent\_login a Right Join agent\_performance b on a.agent=b.agent limit 10;

Hive -e ‘Select a.agent, a.date, b.total\_chats, b.total\_feedback, a.duration from agent\_login a Right Join agent\_performance b on a.agent=b.agent’ > /home/cloudera/Desktop/hanif\_project/rightjoin.csv;

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

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

create table agent\_login\_partitioned

(

sl\_no int,

date string,

login\_time string,

logout\_time string,

duration string

)

Partitioned by (agent string)

CLUSTERED by (date) sorted by (date) INTO 4 BUCKETS

row format delimited

fields terminated by ',';

insert into table agent\_login\_partitioned partition (agent) select sl\_no, date, login\_time, logout\_time, duration, agent from agent\_login;

Create table agent\_performance\_partitioned

(

sl\_no int,

date string,

total\_chats int,

average\_response\_time string,

average\_resolution\_time string,

average\_rating float,

total\_feedback int

) Partitioned by (agent string)

CLUSTERED by (date) sorted by (date) INTO 8 BUCKETS

row format delimited

fields terminated by ',';

insert into table agent\_performance\_partitioned partition(agent) select agent, sl\_no, date, total\_chats, average\_response\_time, average\_resolution\_time, average\_rating, total\_feedback from agent\_performance;