Contents

Mohammed Hanif Shaikh

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;
```