1. Create a schema based on the given dataset

```
create table agent_performance_bkp
   (
   sl_no int,
   ag_date string,
   agent_name string,
   total_chats int,
   average_response_time string,
   average_resolution_time string,
   average_rating float,
   total_feedback int)
   row format delimited
   fields terminated by ','
   tblproperties("skip.header.line.count"="1");

create table agent_performance
   (
   sl_no int,
   ag_date date,
   agent_name string,
   total_chats int,
   average_response_time bigint,
   average_resolution_time bigint,
   average_rating float,
   total_feedback int)
   row format delimited
   fields terminated by ',';


   create table agent_loging_report
   (
   sl_no int,
   agent_name string,
   ag_date date,
   login_time bigint,
   logout_time bigint,
   duration bigint)
   row format delimited
   fields terminated by ',';


create table agent_loging_report_bkp
   (
```

```
    sl_no int,
    agent_name string,
    ag_date string,
    login_time string,
    logout_time string,
    duration string)
    row format delimited
    fields terminated by ','
    tblproperties("skip.header.line.count"="1");
```

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

```
    load data local inpath 'file:///config/workspace/AgentPerformance.csv' into table
agent_performance_bkp;


    insert into table agent_performance select sl_no,
    from_unixtime(unix_timestamp(ag_date,'mm/dd/yyyy'), 'yyyy-mm-dd'), agent_name, total_chats,
    case when average_response_time like '%:%:%' then
unix_timestamp(average_response_time,'HH:mm:ss')
                when average_response_time like '%:%'   then
unix_timestamp(average_response_time,'mm:ss')
                else average_response_time
            end as average_response_time,
    case when average_resolution_time like '%:%:%' then
unix_timestamp(average_resolution_time,'HH:mm:ss')
                when average_resolution_time like '%:%'   then
unix_timestamp(average_resolution_time,'mm:ss')
                else average_resolution_time
            end as average_resolution_time,
    average_rating, total_feedback
    from agent_performance_bkp;


    load data local inpath'file:///config/workspace/AgentLogingReport.csv' into table
agent_loging_report_bkp;


    insert into table agent_loging_report select sl_no, agent_name,
    from_unixtime(unix_timestamp(substr(ag_date,0,11),'dd-MMM-yy')),
    case when login_time like '%:%:%' then unix_timestamp(login_time,'HH:mm:ss')
                when login_time like '%:%'   then unix_timestamp(login_time,'mm:ss')
```

```
            else login_time
        end as login_time,
    case when logout_time like '%:%:%' then unix_timestamp(logout_time,'HH:mm:ss')
            when logout_time like '%:%'   then unix_timestamp(logout_time,'mm:ss')
            else logout_time
        end as logout_time,
    case when duration like '%:%:%' then unix_timestamp(duration,'HH:mm:ss')
            when duration like '%:%'   then unix_timestamp(duration,'mm:ss')
            else duration
        end as duration
    from agent_loging_report_bkp;
```

3. List of all agents names.

```
    select "agent_loging_report", count(DISTINCT(agent_name)) as in_log_report from
agent_loging_report
```

4. Find out agent average rating.

```
    select agent_name, ROUND(AVG(average_rating), 2) as Average_rating
    from agent_performance
    group by agent_name
    order by Average_rating desc;
```

5. Total working days for each agents

```
    select "agent_performance", agent_name, count(distinct ag_date) as working_days
    from agent_performance
    group by agent_name;
```

6. Total query that each agent have taken

```
    select agent_name, SUM(total_chats) as total_queries
    from agent_performance
    group by agent_name;
```

7. Total Feedback that each agent have received

```
    select agent_name, SUM(total_feedback) as total_feedbacks
    from agent_performance
```

```
group by agent_name
order by total_feedbacks;
```

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

```
select agent_name, AVG(average_rating)
from agent_performance
group by agent_name
having ROUND(AVG(average_rating),3) between 3.5 and 4
order by ROUND(AVG(average_rating),1) desc;

select agent_name, AVG(average_rating)
from agent_performance
group by agent_name
having AVG(average_rating) between 3.5 and 4;
```

9. Agent name who have rating less than 3.5

```
select agent_name, AVG(average_rating)
from agent_performance
group by agent_name
having AVG(average_rating) < 3.5;
```

10. Agent name who have rating more than 4.5

```
select agent_name, AVG(average_rating)
from agent_performance
group by agent_name
having ROUND(AVG(average_rating), 3) > 4.5;
```

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

```
select agent_name, COUNT(average_rating) AS frequency
from agent_performance
WHERE average_rating > 4.5
group by agent_name;
```

12. average weekly response time for each agent

```
select agent_name, weekofyear(ag_date) as week_no, AVG(average_response_time)
from agent_performance
group by agent_name, weekofyear(ag_date);
```

13. average weekly resolution time for each agents

```
select agent_name, weekofyear(ag_date) as week_no, AVG(average_resolution_time)
from agent_performance
group by agent_name, weekofyear(ag_date);
```

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

```
select agent_name, count(total_chats)
from agent_performance
where total_chats <> 0
group by agent_name;
```

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

```
select agent_name, weekofyear(ag_date) as week_no, (sum(duration)/60)/60 as hours_worked
from agent_loging_report
group by agent_name, weekofyear(ag_date);
```

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.

```
set hive.variable.substitute=true;
```

Inner Join:

```
with data1 as (select agent_name, (sum(duration)/60)/60 as hours_worked from
agent_loging_report where agent_name like 'Ayushi Mishra' group by agent_name)

select ap.agent_name, sum(ap.total_chats) as total_chats, sum(ap.average_response_time/60)/60
as response_time, AVG(ap.average_rating) as avg_rating,
sum(ap.average_resolution_time/60)/60 as resolution_time, sum(ap.total_feedback) as
total_feedback, d.hours_worked
from data1 d
inner join agent_performance ap
```

```
on d.agent_name = ap.agent_name
group by ap.agent_name, d.hours_worked;
```

Loading data from query into Local File system

```
hive -e 'set hive.cli.print.header=true;
select ap.agent_name, sum(ap.total_chats) as total_chats,
Round(sum(ap.average_response_time/60)/60, 3) as response_time,
Round(sum(ap.average_resolution_time/60)/60, 3) as resolution_time,
Round(avg(average_rating), 3) as avg_rating, sum(ap.total_feedback) as total_feedback,
Round(d.hours_worked, 3) as hours_served
from hive_class_b1.agent_performance ap
inner join (select agent_name, (sum(duration)/60)/60 as hours_worked from
hive_class_b1.agent_loging_report group by agent_name) d
on d.agent_name = ap.agent_name
group by ap.agent_name, d.hours_worked' | sed 's/[\t]/,/g'  >
file:///config/workspace/Downloads/InnerJoinOutputfile.csv;
```

Left Join:

```
select ap.agent_name, sum(ap.total_chats) as total_chats,
Round(sum(ap.average_response_time/60)/60, 3) as response_time,
Round(sum(ap.average_resolution_time/60)/60, 3) as resolution_time, Round(avg(average_rating),
3) as avg_rating, sum(ap.total_feedback) as total_feedback, Round(d.hours_worked, 3) as hours_served
from agent_performance ap
left join (select agent_name, (sum(duration)/60)/60 as hours_worked from agent_loging_report
group by agent_name) d
on d.agent_name = ap.agent_name
group by ap.agent_name, d.hours_worked;
```

Loading data from query into Local File system

```
hive -e 'set hive.cli.print.header=true;
select ap.agent_name, sum(ap.total_chats) as total_chats,
Round(sum(ap.average_response_time/60)/60, 3) as response_time,
```

```
        Round(sum(ap.average_resolution_time/60)/60, 3) as resolution_time,
Round(avg(average_rating), 3) as avg_rating, sum(ap.total_feedback) as total_feedback,
Round(d.hours_worked, 3) as hours_served
        from hive_class_b1.agent_performance ap
        left join (select agent_name, (sum(duration)/60)/60 as hours_worked from
hive_class_b1.agent_loging_report group by agent_name) d
        on d.agent_name = ap.agent_name
        group by ap.agent_name, d.hours_worked' | sed 's/[\t]/,/g'
>file:///config/workspace/Downloads/LeftJoinOutputfile.csv;
```

Right Join:

```
    select ap.agent_name, sum(ap.total_chats) as total_chats,
Round(sum(ap.average_response_time/60)/60, 3) as response_time,
    Round(sum(ap.average_resolution_time/60)/60, 3) as resolution_time, Round(avg(average_rating),
3) as avg_rating, sum(ap.total_feedback) as total_feedback, Round(d.hours_worked, 3) as hours_served
    from agent_performance ap
    Right join (select agent_name, (sum(duration)/60)/60 as hours_worked from agent_loging_report
group by agent_name) d
    on d.agent_name = ap.agent_name
    group by ap.agent_name, d.hours_worked;
```

Loading data from query into Local File system

```
        hive -e 'set hive.cli.print.header=true; select ap.agent_name, sum(ap.total_chats) as total_chats,
Round(sum(ap.average_response_time/60)/60, 3) as response_time,
        Round(sum(ap.average_resolution_time/60)/60, 3) as resolution_time,
Round(avg(average_rating), 3) as avg_rating, sum(ap.total_feedback) as total_feedback,
Round(d.hours_worked, 3) as hours_served
        from hive_class_b1.agent_performance ap
        Right join (select agent_name, (sum(duration)/60)/60 as hours_worked from
hive_class_b1.agent_loging_report group by agent_name) d
        on d.agent_name = ap.agent_name
        group by ap.agent_name, d.hours_worked' | sed 's/[\t]/,/g'  >
file:///config/workspace/Downloads/LeftJoinOutputfile.csv;
```

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

First set the below properties

```
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.patition.mode=nonstrict;
```

Create Partition_bucketed table:

```
create table partition_bucketed_loging
(
   s_no int,
   date date,
   login_time string,
   logout_time string,
   duration string
)
partitioned by (agent string)
clustered by(s_no)
into 4 buckets
row format delimited
fields terminated by ','
stored as textfile;
```

Load data into Partition_bucketed table:

```
insert overwrite table partition_bucketed_loging partition(agent) select s_no, date, login_time, logout_time, duration, agent from agent_loging;
```