create database miniprojects;

use miniprojects;

hadoop fs -cp file:///home/cloudera/Desktop/AgentPerformance.csv /tmp/data

hadoop fs -cp file:///home/cloudera/Desktop/AgentLogingReport.csv /tmp/data

# create table agent\_performance

1.) and 2.) ----------

create table agent\_performance

(

sl\_no int,

date string,

agent\_name string,

total\_chats int,

avg\_resp\_time array<int>,

avg\_resol\_time array<int>,

avg\_rating double,

total\_feedback int

)

row format delimited

fields terminated by ','

collection items terminated by ':'

tblproperties("skip.header.line.count"="1");

# load data from hdfs into table

load data inpath '/tmp/data/AgentPerformance.csv' into table agent\_performance;

# create agent\_login\_report

create table agent\_login\_report

(

sl\_no int,

agent\_name string,

date string,

login\_time array<int>,

logout\_time array<int>,

duration array<int>

)

row format delimited

fields terminated by ','

collection items terminated by ':'

tblproperties("skip.header.line.count"="1");

# load data from hdfs into agent\_login\_report

load data inpath '/tmp/data/AgentLogingReport.csv' into table agent\_login\_report;

3.) --------

select distinct(agent\_name) from agent\_performance;



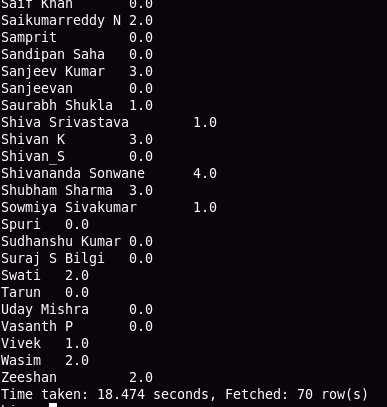
4.) ------

# since each agent has 30 average ratings(data is for the month of july), the query finds avg for all of them for each agent.

select agent\_name, round(avg(avg\_rating))

from agent\_performance

group by agent\_name;



5.)--------

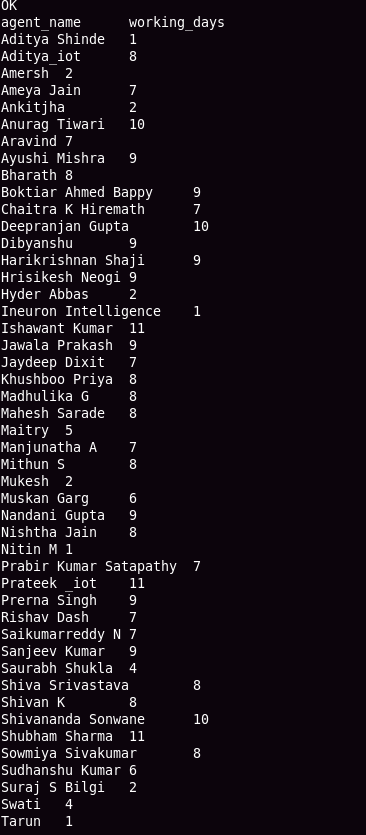
# query is executed on agent\_login\_report table

# Since an agent was logged in multiple times in a day. we have to count the distinct days for each agent

select agent\_name, count(distinct(date)) as working\_days

from agent\_login\_report

group by agent\_name;



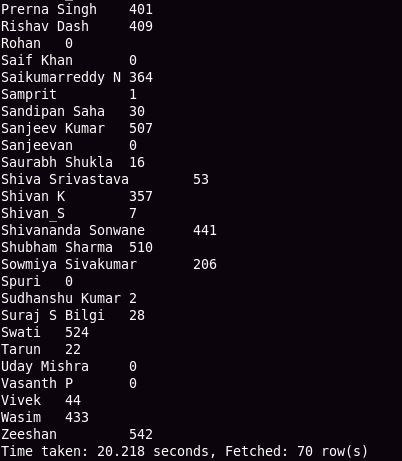
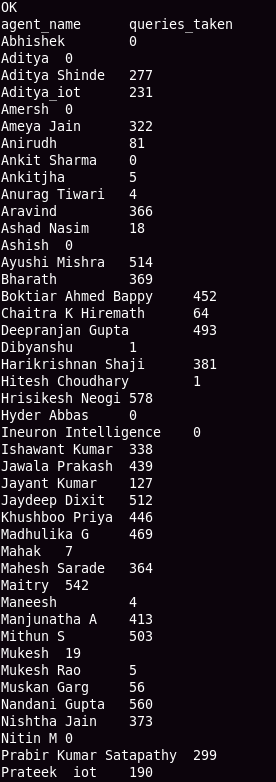
6.)------

# total\_chats field conveys that count that agent had chats with customer to resolve queries on daliy basis. So we can consider this column to query

select agent\_name, sum(total\_chats) as queries\_taken

from agent\_performance

group by agent\_name;

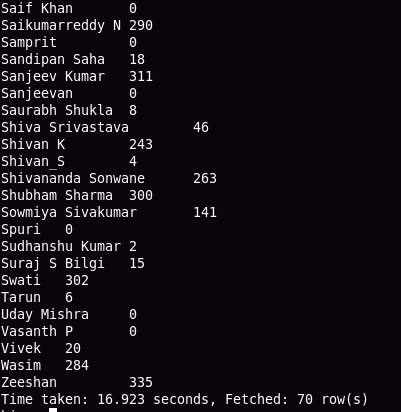
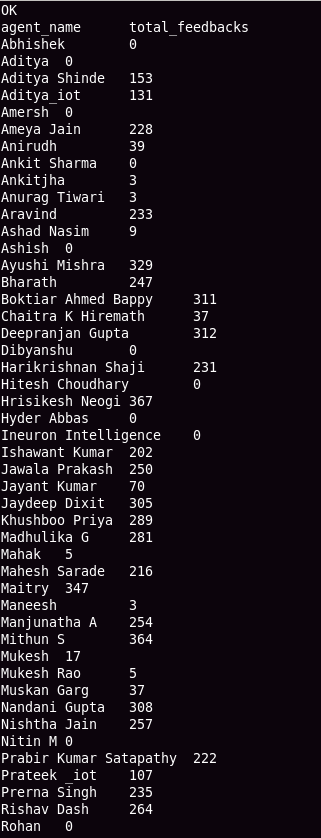


7.) ------

select agent\_name, sum(total\_feedback) as total\_feedbacks

from agent\_performance

group by agent\_name;



8.) -------

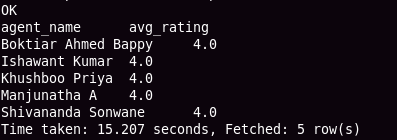
# since agent has ratings everyday. First avg all the ratings for each agent and find those who has ratings >=35 and <=4.

select agent\_name, round(avg(avg\_rating)) as avg\_rating

from agent\_performance

group by agent\_name

having round(avg(avg\_rating)) >= 3.5 and round(avg(avg\_rating)) <=4;



9.) ---------

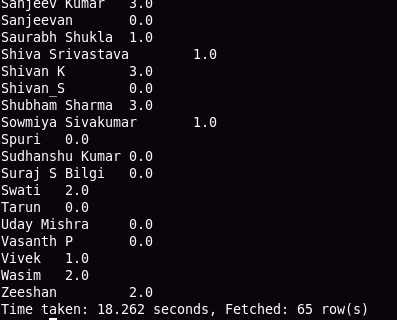
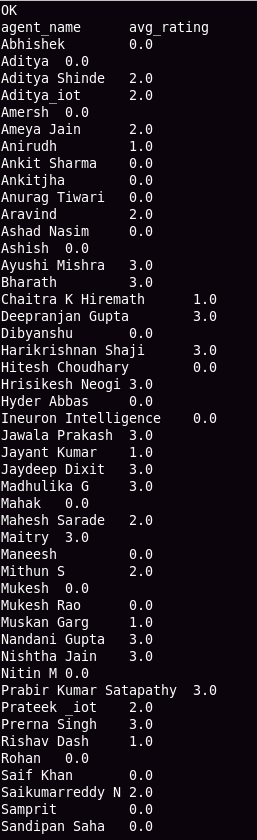
# considering rating as average rating for whole month for each agent

select agent\_name, round(avg(avg\_rating)) as avg\_rating

from agent\_performance

group by agent\_name

having round(avg(avg\_rating)) < 3.5;



# considering rating as daily rating

select agent\_name, avg\_rating

from agent\_performance

where avg\_rating < 3.5;

10.) -------

# considering rating as daily rating

select agent\_name, avg\_rating

from agent\_performance

where avg\_rating > 4.5;

11.)

# take average ratings for each agent and round it to two decimal values. then fetch those are having avg > 4.5

select agent\_name, round(avg(total\_feedback)) as avg\_feedback

from agent\_performance

group by agent\_name

having round(avg(total\_feedback)) > 4.5;



12.) -------

# first calclate the sum of daily response time for each agent. Then find the weekly average.

with agent\_daily\_resp\_time\_table as

(

select from\_unixtime(unix\_timestamp(date , 'MM/dd/yyyy'), 'yyyy-MM-dd') as date, agent\_name,

sum(avg\_resp\_time[0] \* 3600 + avg\_resp\_time[1] \* 60 + avg\_resp\_time[2]) as total\_daily\_resp\_time

from agent\_performance

group by date,agent\_name

),

weekly\_avg\_table as

(

select weekofyear(date) as week\_of\_year,agent\_name,round(avg(total\_daily\_resp\_time)) as weekly\_avg\_resp\_time\_in\_sec from agent\_daily\_resp\_time\_table

group by weekofyear(date),agent\_name

)

select agent\_name,

sum(case when week\_of\_year = 26 then weekly\_avg\_resp\_time\_in\_sec else 0 end) as week\_1,

sum(case when week\_of\_year = 27 then weekly\_avg\_resp\_time\_in\_sec else 0 end) as week\_2,

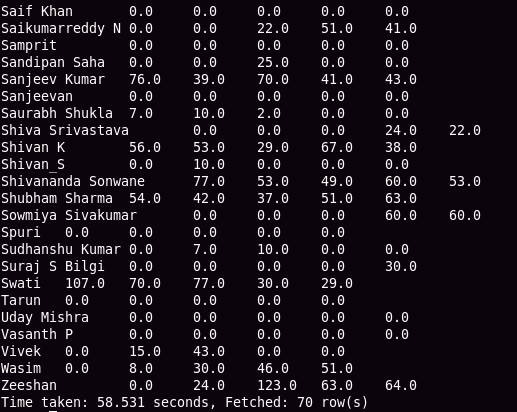
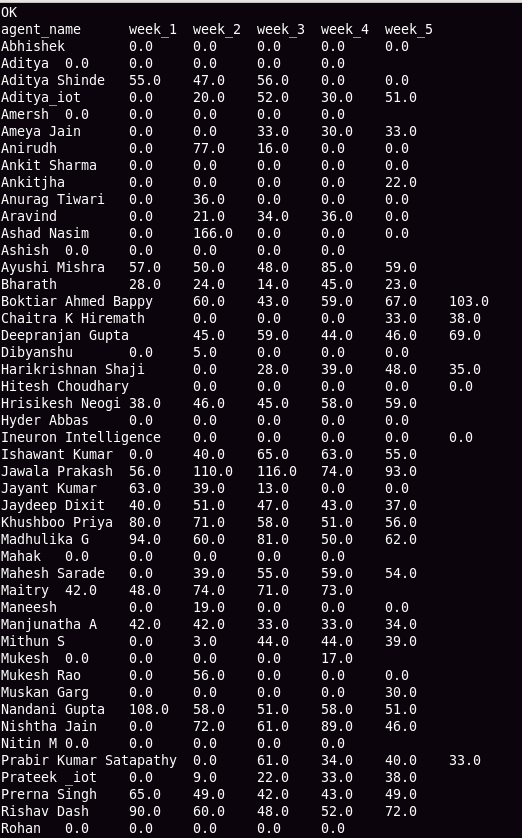
sum(case when week\_of\_year = 28 then weekly\_avg\_resp\_time\_in\_sec else 0 end) as week\_3,

sum(case when week\_of\_year = 29 then weekly\_avg\_resp\_time\_in\_sec else 0 end) as week\_4,

sum(case when week\_of\_year = 30 then weekly\_avg\_resp\_time\_in\_sec else 0 end) as week\_5

from weekly\_avg\_table

group by agent\_name;



13.) --------

# first calclate the sum of daily resolution time for each agent. Then find the weekly average

with agent\_daily\_resol\_time\_table as

(

select from\_unixtime(unix\_timestamp(date , 'MM/dd/yyyy'), 'yyyy-MM-dd') as date, agent\_name,

sum(avg\_resol\_time[0] \* 3600 + avg\_resol\_time[1] \* 60 + avg\_resol\_time[2]) as total\_daily\_resol\_time

from agent\_performance

group by date,agent\_name

),

weekly\_avg\_table as

(

select weekofyear(date) as week\_of\_year,agent\_name,round(avg(total\_daily\_resol\_time)) as weekly\_avg\_resol\_time\_in\_sec from agent\_daily\_resol\_time\_table

group by weekofyear(date),agent\_name

)

select agent\_name,

sum(case when week\_of\_year = 26 then weekly\_avg\_resol\_time\_in\_sec else 0 end) as week\_1,

sum(case when week\_of\_year = 27 then weekly\_avg\_resol\_time\_in\_sec else 0 end) as week\_2,

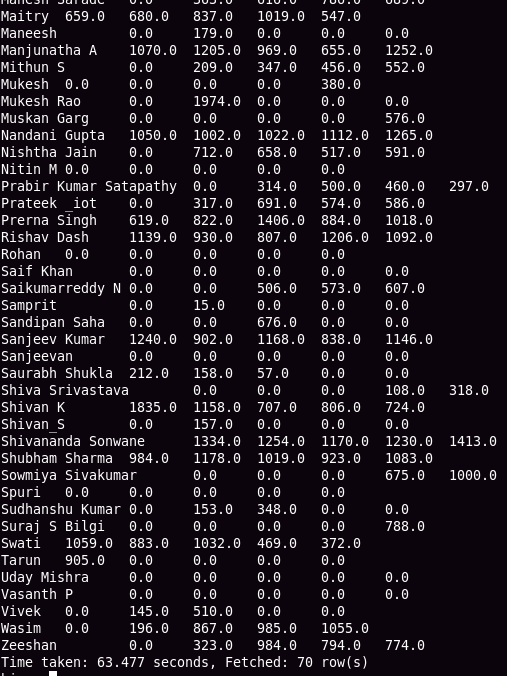
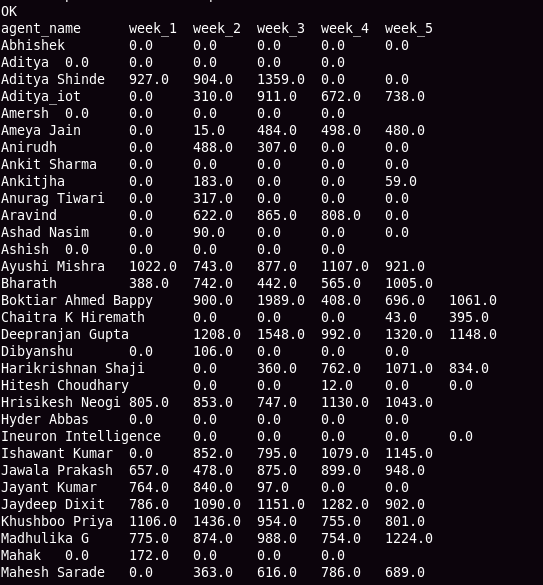
sum(case when week\_of\_year = 28 then weekly\_avg\_resol\_time\_in\_sec else 0 end) as week\_3,

sum(case when week\_of\_year = 29 then weekly\_avg\_resol\_time\_in\_sec else 0 end) as week\_4,

sum(case when week\_of\_year = 30 then weekly\_avg\_resol\_time\_in\_sec else 0 end) as week\_5

from weekly\_avg\_table

group by agent\_name;

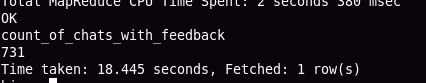


14.) --------

select count(total\_chats) as count\_of\_chats\_with\_feedback

from agent\_performance

where total\_feedback != 0;



15.)---------

change\_date\_udf.py

-----------------

import sys

for line in sys.stdin:

line = line.strip("\n\r")

date, agent\_name, duration = line.split("\t")

date = date.replace("-","/")

date = date.replace("Jul/22","07/2022")

hour,minutes,sec = duration.split(":")

duration = str(round((float(hour) + float(minutes)/60 + float(sec)/3600),2))

result = "\t".join([date,agent\_name,duration])

print(result)

# add .py file to hive

add files /home/cloudera/Desktop/change\_date\_udf.py

# find daily working hours. Then find weekly average . Duration is in hours .

with cte as

(

select transform (date,agent\_name,duration)

using 'python change\_date\_udf.py' as (date string, agent\_name string, duration double)

from agent\_login\_report

),

daily\_working\_hrs\_table as

(

select from\_unixtime(unix\_timestamp(date , 'dd/MM/yyyy'), 'yyyy-MM-dd') as date, agent\_name,

round(sum(duration)) as duration from cte

group by from\_unixtime(unix\_timestamp(date , 'dd/MM/yyyy'), 'yyyy-MM-dd') , agent\_name

),

weekly\_working\_hrs\_table as

(

select weekofyear(date) as week\_of\_year, agent\_name,

round(sum(duration)/4) as weekly\_avg from daily\_working\_hrs\_table

group by weekofyear(date) , agent\_name

)

select agent\_name,

sum(case when week\_of\_year = 26 then weekly\_avg else 0 end) as week\_1,

sum(case when week\_of\_year = 27 then weekly\_avg else 0 end) as week\_2,

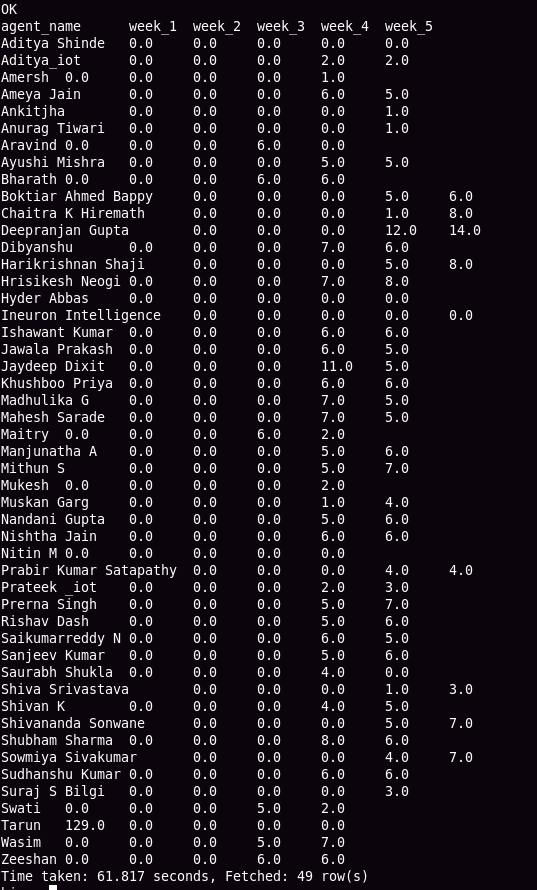
sum(case when week\_of\_year = 28 then weekly\_avg else 0 end) as week\_3,

sum(case when week\_of\_year = 29 then weekly\_avg else 0 end) as week\_4,

sum(case when week\_of\_year = 30 then weekly\_avg else 0 end) as week\_5

from weekly\_working\_hrs\_table

group by agent\_name;



16) -------

a) INNER JOIN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# create a table to store the left joined data

create table inner\_join\_data

row format delimited

fields terminated by ','

lines terminated by '\n'

stored as textfile

as

select

'a.sl\_no' as a\_sl\_no,

'a.date' as a\_date,

'a.agent\_name' as a\_agent\_name,

'a.total\_chats' as a\_total\_chats,

'a.avg\_resp\_time' as a\_avg\_resp\_time,

'a.avg\_resol\_time' as a\_avg\_resol\_time,

'a.avg\_rating' as a\_avg\_rating,

'a.total\_feedback' as a\_total\_feedback,

'b.sl\_no' as b\_sl\_no,

'b.agent\_name' as b\_agent\_name,

'b.date' as b\_date,

'b.login\_time' as b\_login\_time,

'b.logout\_time' as b\_logout\_time,

'b.duration' as b\_duration;

#insert into that table after join operation

insert into inner\_join\_data

select \* from agent\_performance a

inner join agent\_login\_report b

on a.agent\_name = b.agent\_name;

# find the location of table

describe formatted inner\_join\_data;

hdfs://quickstart.cloudera:8020/user/hive/warehouse/miniprojects.db/inner\_join\_data

# using below line of code move data from hdfs to local

hadoop fs -cat hdfs://quickstart.cloudera:8020/user/hive/warehouse/miniprojects.db/inner\_join\_data/\* > ~/Desktop/csv\_files/inner\_join\_data.csv

LEFT JOIN

\*\*\*\*\*\*\*\*\*\*

# we have already a temporary table to store the joined data. Just rename it for our use.

alter table inner\_join\_data rename to left\_join\_data;

# insert data into table after left join operation.

insert into left\_join\_data

select \* from agent\_performance a

left join agent\_login\_report b

on a.agent\_name = b.agent\_name;

# move data from hdfs to local

hadoop fs -cat hdfs://quickstart.cloudera:8020/user/hive/warehouse/miniprojects.db/left\_join\_data/\* > ~/Desktop/csv\_files/left\_join\_data.csv

RIGHT JOIN

\*\*\*\*\*\*\*\*\*\*

# we have already a temporary table to store the joined data. Just rename it for our use.

alter table left\_join\_data rename to right\_join\_data;

# insert data into table after right join operation.

insert into right\_join\_data

select \* from agent\_performance a

right join agent\_login\_report b

on a.agent\_name = b.agent\_name;

# move data from hdfs to local

hadoop fs -cat hdfs://quickstart.cloudera:8020/user/hive/warehouse/miniprojects.db/right\_join\_data/\* > ~/Desktop/csv\_files/right\_join\_data.csv

FULL JOIN

\*\*\*\*\*\*\*\*\*\*

# we have already a temporary table to store the joined data. Just rename it for our use.

alter table right\_join\_data rename to full\_join\_data;

# insert data into table after full join operation.

insert into full\_join\_data

select \* from agent\_performance a

full join agent\_login\_report b

on a.agent\_name = b.agent\_name;

# move data from hdfs to local

hadoop fs -cat hdfs://quickstart.cloudera:8020/user/hive/warehouse/miniprojects.db/full\_join\_data/\* > ~/Desktop/csv\_files/full\_join\_data.csv

17) --------

# for partitioning we use dynamic partitioning .set this property for dynamic partitioning

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

# create a table to store the partitioned data

create table agent\_performance\_dynamic\_part

(

sl\_no int,

date string,

total\_chats int,

avg\_resp\_time string,

avg\_resol\_time string,

avg\_rating double,

total\_feedback int

)

partitioned by (agent\_name string);

# since we have many agents. partition data based on agent\_name

# ------> Note: The partition column/s should be used at last in select statement <--------

insert overwrite table agent\_performance\_dynamic\_part partition(agent\_name) select sl\_no, date,total\_chats, avg\_resp\_time, avg\_resol\_time, avg\_resol\_time, avg\_rating total\_feedback, agent\_name

from agent\_performance;

# you can check the partitions in hdfs

hadoop fs -ls /user/hive/warehouse/miniprojects.db/agent\_performance\_dynamic\_part/\*

Bucketing

\*\*\*\*\*\*\*\*\*

# first set bucketing to true

set hive.enforce.bucketing=true;

create table bucket\_agent\_performance

(

sl\_no int,

agent\_name string,

date string,

total\_chats int,

avg\_resp\_time string,

avg\_resol\_time string,

avg\_rating double,

total\_feedback int

)

clustered by (agent\_name)

sorted by (sl\_no)

into 4 buckets;

# create buckets

insert overwrite table bucket\_agent\_performance select \* from agent\_performance\_dynamic\_part;

# you can check the buckets in the hdfs

hadoop fs -ls /user/hive/warehouse/miniprojects.db/bucket\_agent\_performance/\*