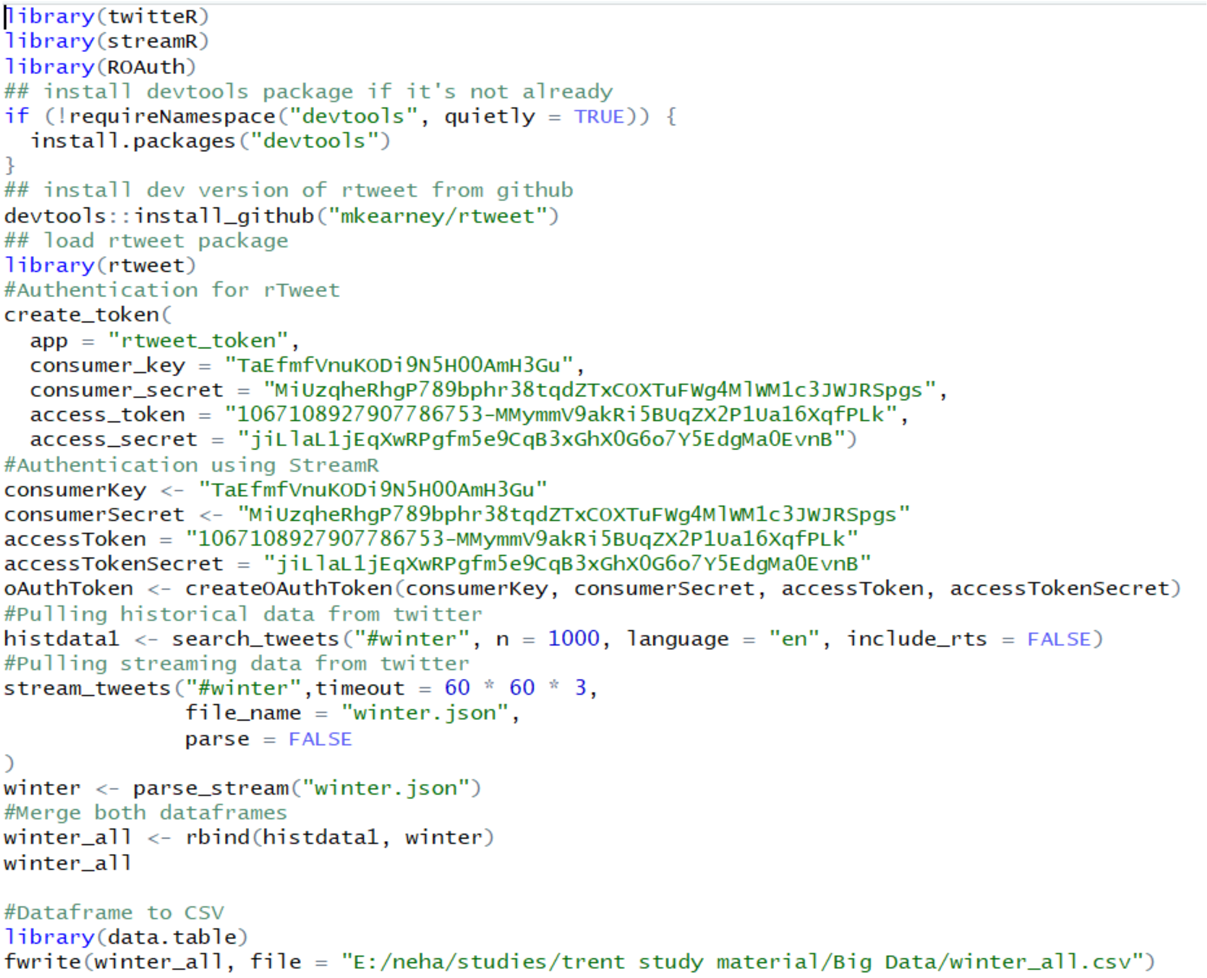
SQL Queries on Winter weather data and twitter data for #Winter

For the purpose of this assignment, I have collected two datasets; one is taken from the Twitter API and the other is taken from the Openweathermap API. From the Twitter API, all the tweets having the hashtag #winter is collected and stored in a .csv file. From the openweathermap API, a 16-day forecast data is collected from 10 different cities and stored in a .csv file.

Data from the twitter API:

I have collected the tweets having the hashtag winter using the twitter API. For that, it is required to get a key from the Twitter API and using that key we can collect live tweet data for the desired duration and store it in a .csv format. I have done this using R language and R studio. The code for getting the tweet data for the hashtag winter is as follows:

The code for collecting the tweet data by using the twitter API is as follows:

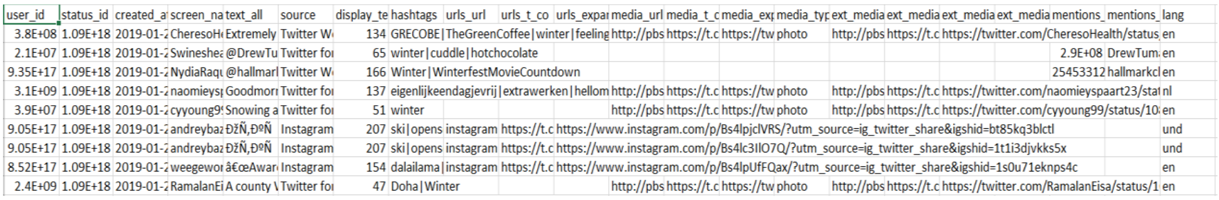


```
library(twitteR)
library(streamR)
library(ROAuth)
## install devtools package if it's not already
if (!requireNamespace("devtools", quietly = TRUE)) {
  install.packages("devtools")
}
## install dev version of rtweet from github
devtools::install_github("mkearney/rtweet")
## load rtweet package
library(rtweet)
#Authentication for rTweet
create_token(
  app = "rtweet_token",
  consumer_key = "TaEfmfVnuKODi9N5H00AmH3Gu",
  consumer_secret = "MiUzqheRhgP789bphr38tqdZTxCOXTuFWg4MlWM1c3JWJRSpgs",
  access_token = "1067108927907786753-MMymmV9akRi5BUqZX2P1Ua16XqfPLk",
  access_secret = "jiLlaL1jEqXwRPgfm5e9CqB3xGhX0G6o7Y5EdgMa0EvnB")
#Authentication using StreamR
consumerKey <- "TaEfmfVnuKODi9N5H00AmH3Gu"
consumerSecret <- "MiUzqheRhgP789bphr38tqdZTxCOXTuFWg4MlWM1c3JWJRSpgs"
accessToken = "1067108927907786753-MMymmV9akRi5BUqZX2P1Ua16XqfPLk"
accessTokenSecret = "jiLlaL1jEqXwRPgfm5e9CqB3xGhX0G6o7Y5EdgMa0EvnB"
oAuthToken <- createOAuthToken(consumerKey, consumerSecret, accessToken, accessTokenSecret)
#Pulling historical data from twitter
histdata1 <- search_tweets("#winter", n = 1000, language = "en", include_rts = FALSE)
#Pulling streaming data from twitter
stream_tweets("#winter",timeout = 60 * 60 * 3,
              file_name = "winter.json",
              parse = FALSE
)
winter <- parse_stream("winter.json")
#Merge both dataframes
winter_all <- rbind(histdata1, winter)
winter_all

#Dataframe to CSV
library(data.table)
fwrite(winter_all, file = "E:/neha/studies/trent study material/Big Data/winter_all.csv")
```

In the above code, the required packages are twitter, streamR, rTweet and ROAuth. We first authenticate by using the consumer key and access token provided by the twitter API. This authentication is for rTweet. Next, we provide authentication for StreamR in the form of consumer keys and access tokens. Then, we pull 1000 historical tweets that contain the #winter and which is in English language and store it in a dataframe. Next, we pull live streaming data for a duration of 3 hours and store this data in another dataframe. Now, we merge both the dataframes using the rbind() function and then convert these into .csv file by using the data.table library in R.

SQL Queries on Winter weather data and twitter data for #Winter

Thus, the collected data in a .csv file looks like this:



user_id	status_id	created_a	screen_na	text_all	source	display_te	hashtags	urls_url	urls_t_co	urls_expa	media_url	media_t_c	media_exp	media_typ	ext_media	ext_media	ext_media	ext_media	mentions_	mentions_	lang
3.8E+08	1.09E+18	2019-01-2	CheresoH	Extremely	Twitter W	134	GRECOBE|TheGreenCoffee|winter|feeling				http://pbs	https://t.c	https://tw	photo	http://pbs	https://t.c	https://twitter.com/CheresoHealth/status				en
2.1E+07	1.09E+18	2019-01-2	Swineshe	@DrewTu	Twitter fo	65	winter|cuddle|hotchocolate												2.9E+08	DrewTum	en
9.35E+17	1.09E+18	2019-01-2	NydiaRaq	@hallmar	Twitter W	166	Winter|WinterfestMovieCountdown												25453312	hallmarkcl	en
3.1E+09	1.09E+18	2019-01-2	naomieys	Goodmor	Twitter fo	137	eigenlijkeendagjevrij|extrawerken|hellom				http://pbs	https://t.c	https://tw	photo	http://pbs	https://t.c	https://twitter.com/naomieyspaart23/stat				nl
3.9E+07	1.09E+18	2019-01-2	cyyoung9	Snowing a	Twitter fo	51	winter				http://pbs	https://t.c	https://tw	photo	http://pbs	https://t.c	https://twitter.com/cyyoung99/status/10				en
9.05E+17	1.09E+18	2019-01-2	andreybaz	ĐžÑ,ĐºÑ	Instagram	207	ski|opens	instagram	https://t.c	https://www.instagram.com/p/Bs4lpjclVRS/?utm_source=ig_twitter_share&igshid=bt85kq3blctl											und
9.05E+17	1.09E+18	2019-01-2	andreybaz	ĐžÑ,ĐºÑ	Instagram	207	ski|opens	instagram	https://t.c	https://www.instagram.com/p/Bs4lc3IlO7Q/?utm_source=ig_twitter_share&igshid=1t1i3djvkks5x											und
8.52E+17	1.09E+18	2019-01-2	weegewor	â€œAwar	Instagram	154	dalailama|instagram		https://t.c	https://www.instagram.com/p/Bs4lpUfFQax/?utm_source=ig_twitter_share&igshid=1s0u71eknps4c											en
2.4E+09	1.09E+18	2019-01-2	RamalanEi	A county \	Twitter fo	47	Doha|Winter				http://pbs	https://t.c	https://tw	photo	http://pbs	https://t.c	https://twitter.com/RamalanEisa/status/1				en

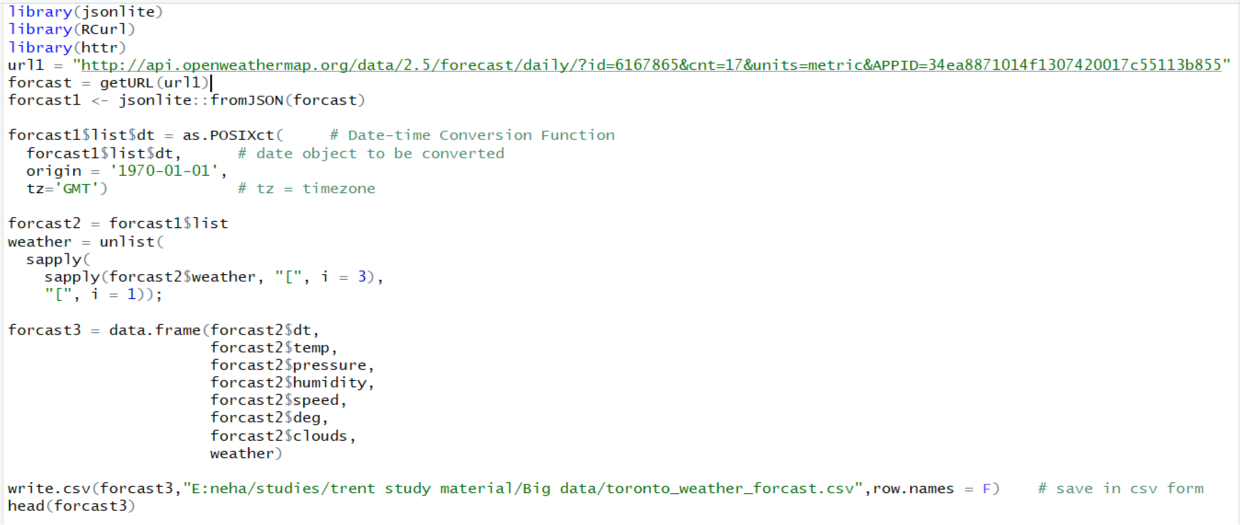
It contains columns like user_id, status_id, the actual tweet, the source of the tweet, the hashtags associated with it, media and url information and so on.

Data from Openweathermap API:

Like the twitter API, to collect weather data from openweathermap, we need a key for authentication. After getting the key, we can get a 16-day forecast for any city that we need. An example of using this API is:

"http://api.openweathermap.org/data/2.5/forecast/daily/?id=6167865&cnt=17&units=metric&APPID=34ea8871014f1307420017c55113b855

In the above call, we get a 16-day forecast for Toronto which has the city id 6167865. We also specify the units as metrics to get the temperatures in degree Celsius. I then used this API call to get weather data about 10 different cities and then merged them in a single .csv file. I have used R to do this. The code for getting the data for Toronto and storing it in a file is as follows:

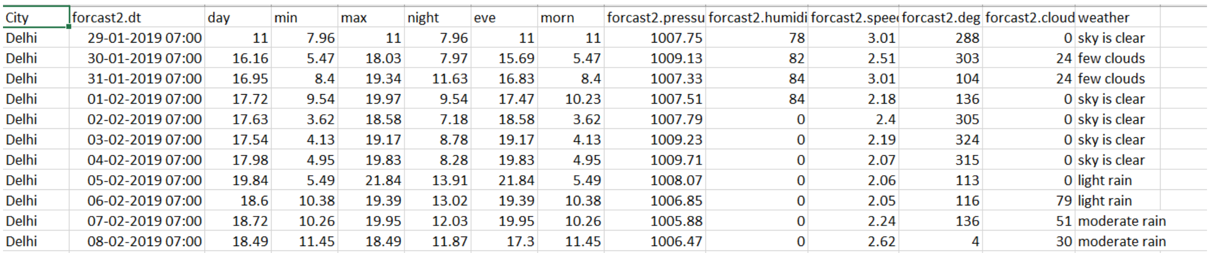


```
library(jsonlite)
library(RCurl)
library(httr)
url1 = "http://api.openweathermap.org/data/2.5/forecast/daily/?id=6167865&cnt=17&units=metric&APPID=34ea8871014f1307420017c55113b855"
forcast = getURL(url1)
forcast1 <- jsonlite::fromJSON(forcast)

forcast1$list$dt = as.POSIXct(     # Date-time Conversion Function
  forcast1$list$dt,      # date object to be converted
  origin = '1970-01-01',
  tz='GMT')              # tz = timezone

forcast2 = forcast1$list
weather = unlist(
  sapply(
    sapply(forcast2$weather, "[", i = 3),
    "[", i = 1));

forcast3 = data.frame(forcast2$dt,
                      forcast2$temp,
                      forcast2$pressure,
                      forcast2$humidity,
                      forcast2$speed,
                      forcast2$deg,
                      forcast2$clouds,
                      weather)

write.csv(forcast3,"E:neha/studies/trent study material/Big data/toronto_weather_forcast.csv",row.names = F)    # save in csv form
head(forcast3)
```

I have used the libraries jsonlite, RCurl and httr for this purpose. We then convert the date and time in proper format and store it appropriately. We then get the required columns for which we need the weather data and then write this data in a .csv file. We do this for different cities by just changing the city-id.

The final csv file looks somewhat like this:

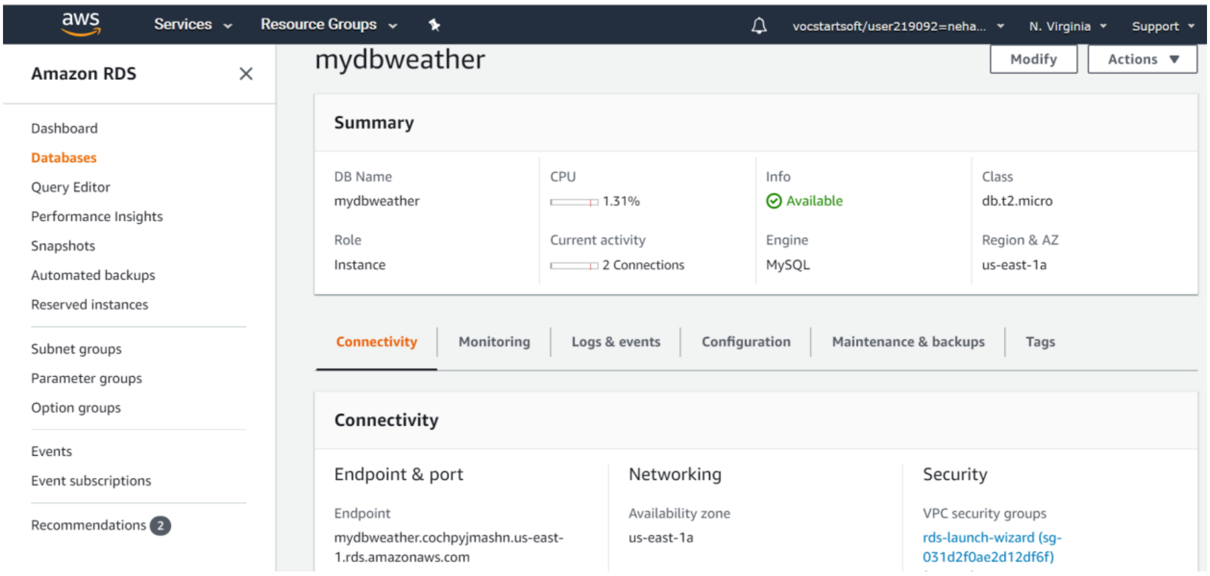
SQL Queries on Winter weather data and twitter data for #Winter



City	forcast2.dt	day	min	max	night	eve	morn	forcast2.pressu	forcast2.humidi	forcast2.spee	forcast2.deg	forcast2.cloud	weather
Delhi	29-01-2019 07:00	11	7.96	11	7.96	11	11	1007.75	78	3.01	288	0	sky is clear
Delhi	30-01-2019 07:00	16.16	5.47	18.03	7.97	15.69	5.47	1009.13	82	2.51	303	24	few clouds
Delhi	31-01-2019 07:00	16.95	8.4	19.34	11.63	16.83	8.4	1007.33	84	3.01	104	24	few clouds
Delhi	01-02-2019 07:00	17.72	9.54	19.97	9.54	17.47	10.23	1007.51	84	2.18	136	0	sky is clear
Delhi	02-02-2019 07:00	17.63	3.62	18.58	7.18	18.58	3.62	1007.79	0	2.4	305	0	sky is clear
Delhi	03-02-2019 07:00	17.54	4.13	19.17	8.78	19.17	4.13	1009.23	0	2.19	324	0	sky is clear
Delhi	04-02-2019 07:00	17.98	4.95	19.83	8.28	19.83	4.95	1009.71	0	2.07	315	0	sky is clear
Delhi	05-02-2019 07:00	19.84	5.49	21.84	13.91	21.84	5.49	1008.07	0	2.06	113	0	light rain
Delhi	06-02-2019 07:00	18.6	10.38	19.39	13.02	19.39	10.38	1006.85	0	2.05	116	79	light rain
Delhi	07-02-2019 07:00	18.72	10.26	19.95	12.03	19.95	10.26	1005.88	0	2.24	136	51	moderate rain
Delhi	08-02-2019 07:00	18.49	11.45	18.49	11.87	17.3	11.45	1006.47	0	2.62	4	30	moderate rain

Using AWS and MySQL Workbench to run basic queries:

Now we use AWS and MySQL Workbench to load these datasets and run basic queries on it. After creating the database mydbweather in AWS, we can get the endpoint and port.



mydbweather

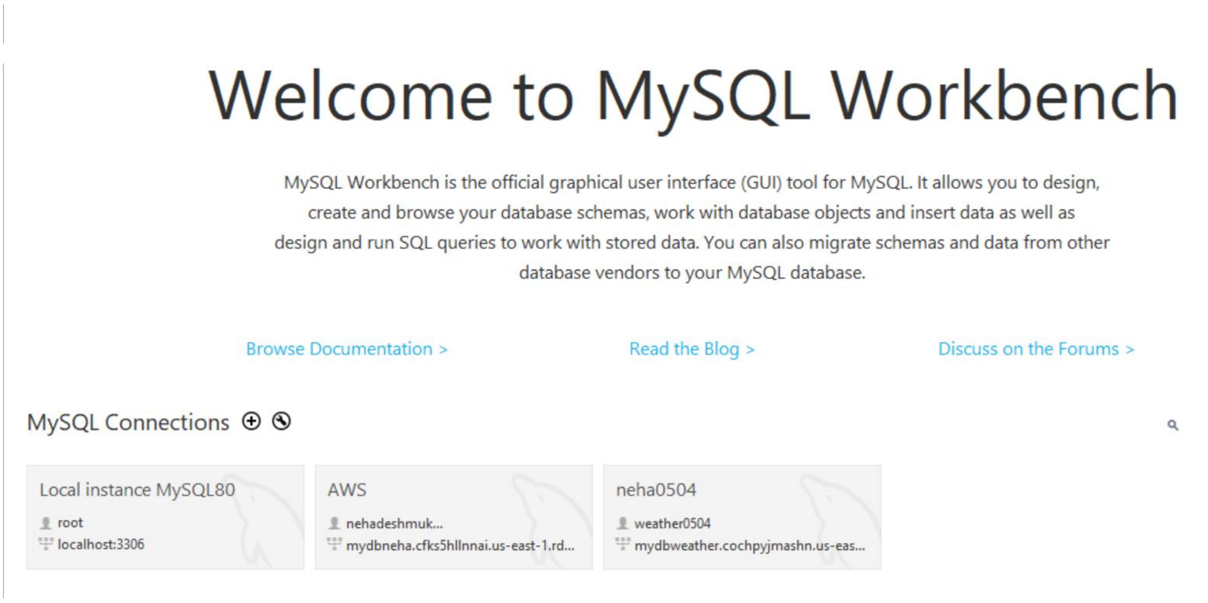
Summary

DB Name	CPU	Info	Class
mydbweather	1.31%	Available	db.t2.micro
Role	**Current activity**	**Engine**	**Region & AZ**
Instance	2 Connections	MySQL	us-east-1a

Connectivity

Endpoint & port	Networking	Security
Endpoint	Availability zone	VPC security groups
mydbweather.cochpyjmashn.us-east-1.rds.amazonaws.com	us-east-1a	rds-launch-wizard (sg-031d2f0ae2d12df6f)

We use this endpoint to set as the host in our MySQL Workbench. We can then connect to AWS through MySQL Workbench. After we connect successfully, we get the following:



Welcome to MySQL Workbench

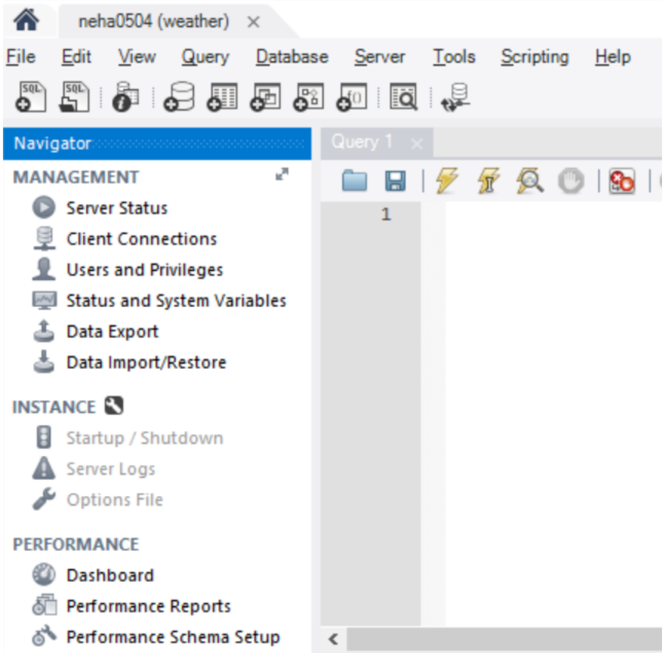
MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Local instance MySQL80
root
localhost:3306

AWS
nehadeshmuk...
mydbneha.cfks5hllnnai.us-east-1.rd...

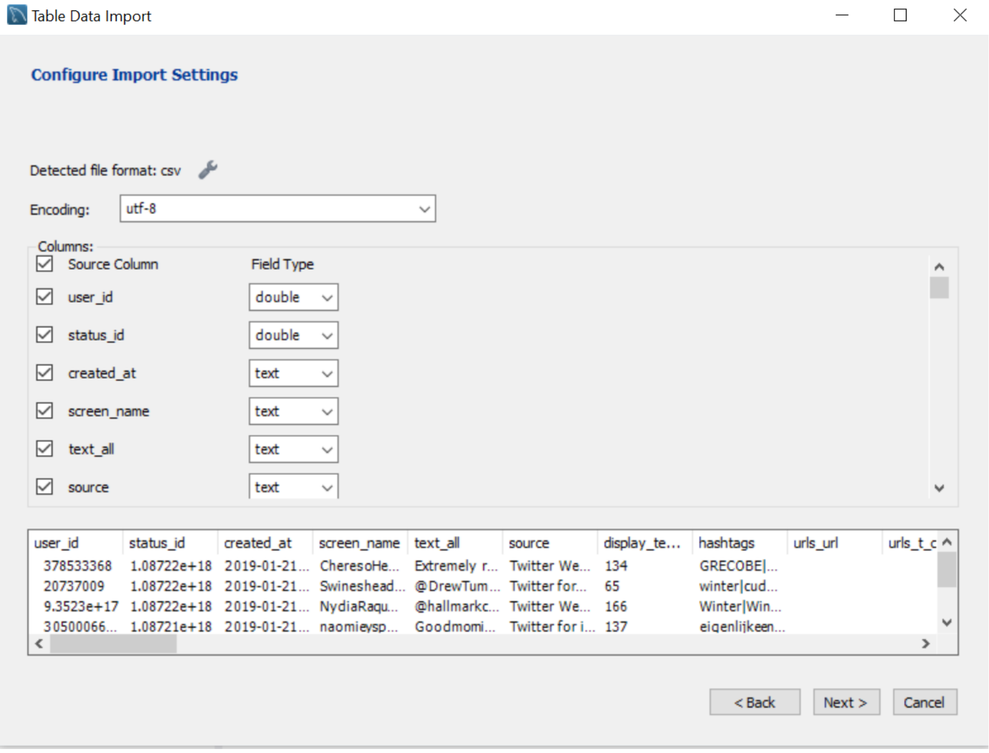
neha0504
weather0504
mydbweather.cochpyjmashn.us-eas...

SQL Queries on Winter weather data and twitter data for #Winter



From the above images, we see that the third connection neha0504 is active and running on AWS.

We can now import the two files in the workbench. For that, we first create the schema called weather and in them, we can import the files.



Detected file format: csv

Encoding: utf-8

Columns:

☑ Source Column	Field Type
☑ user_id	double
☑ status_id	double
☑ created_at	text
☑ screen_name	text
☑ text_all	text
☑ source	text

user_id	status_id	created_at	screen_name	text_all	source	display_te...	hashtags	urls_url	urls_t_c
378533368	1.08722e+18	2019-01-21...	CheresoHe...	Extremely r...	Twitter We...	134	GRECOBE|...		
20737009	1.08722e+18	2019-01-21...	Swineshead...	@DrewTum...	Twitter for...	65	winter|cud...		
9.3523e+17	1.08722e+18	2019-01-21...	NydiaRaqu...	@hallmarkc...	Twitter We...	166	Winter|Win...		
30500066...	1.08721e+18	2019-01-21...	naomieysp...	Goodmomi...	Twitter for i...	137	eigenlijkeen...		

Thus, the weather data file and the tweets data file is imported.

SQL Queries on Winter weather data and twitter data for #Winter

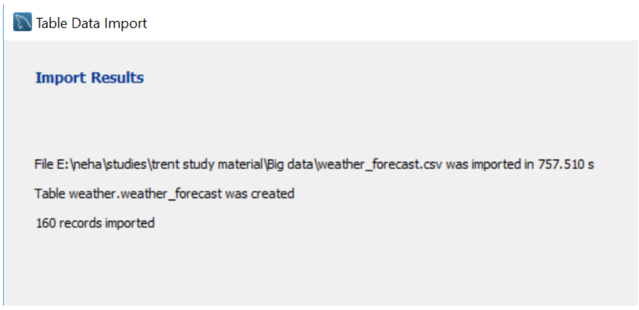


Table Data Import

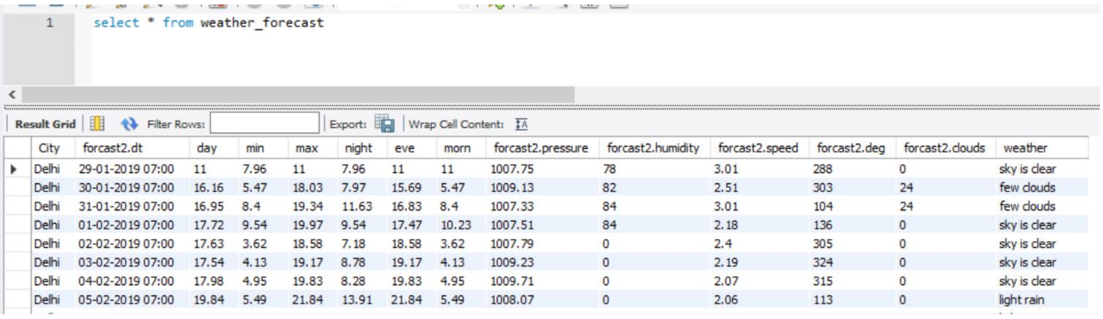
Import Results

File E:\neha\studies\trent study material\Big data\weather_forecast.csv was imported in 757.510 s

Table weather.weather_forecast was created

160 records imported

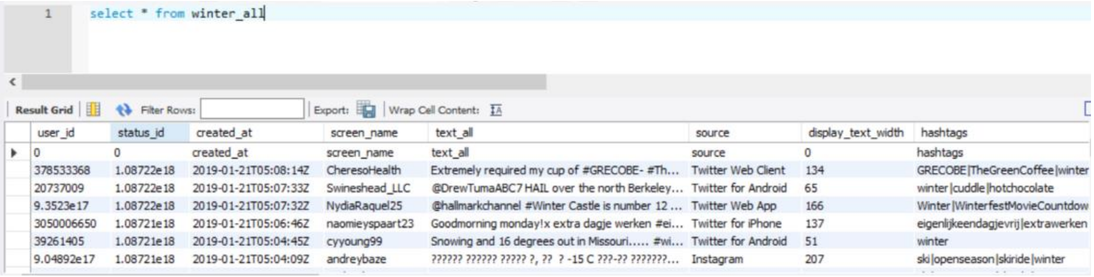
We can see the weather_forecast file as follows:



```
select * from weather_forecast
```

City	forcast2.dt	day	min	max	night	eve	morn	forcast2.pressure	forcast2.humidity	forcast2.speed	forcast2.deg	forcast2.clouds	weather
Delhi	29-01-2019 07:00	11	7.96	11	7.96	11	11	1007.75	78	3.01	288	0	sky is clear
Delhi	30-01-2019 07:00	16.16	5.47	18.03	7.97	15.69	5.47	1009.13	82	2.51	303	24	few clouds
Delhi	31-01-2019 07:00	16.95	8.4	19.34	11.63	16.83	8.4	1007.33	84	3.01	104	24	few clouds
Delhi	01-02-2019 07:00	17.72	9.54	19.97	9.54	17.47	10.23	1007.51	84	2.18	136	0	sky is clear
Delhi	02-02-2019 07:00	17.63	3.62	18.58	7.18	18.58	3.62	1007.79	0	2.4	305	0	sky is clear
Delhi	03-02-2019 07:00	17.54	4.13	19.17	8.78	19.17	4.13	1009.23	0	2.19	324	0	sky is clear
Delhi	04-02-2019 07:00	17.98	4.95	19.83	8.28	19.83	4.95	1009.71	0	2.07	315	0	sky is clear
Delhi	05-02-2019 07:00	19.84	5.49	21.84	13.91	21.84	5.49	1008.07	0	2.06	113	0	light rain

And the winter hashtags database as follows:

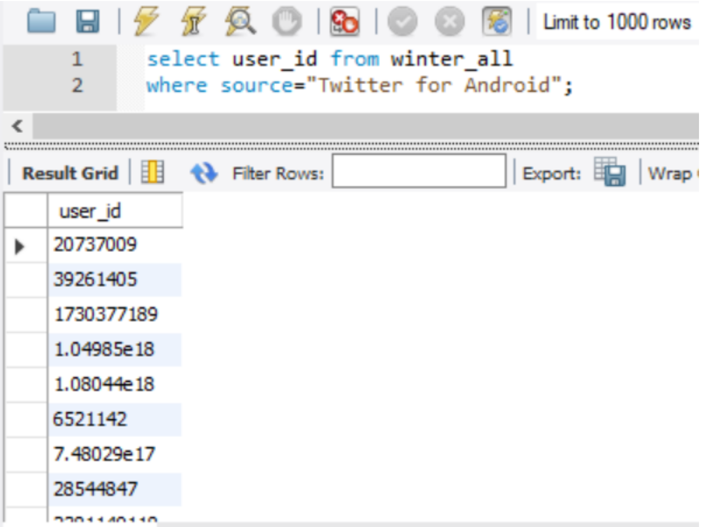


```
select * from winter_all
```

user_id	status_id	created_at	screen_name	text_all	source	display_text_width	hashtags
0	0	created_at	screen_name	text_all	source	0	hashtags
378533368	1.08722e18	2019-01-21T05:08:14Z	CheresoHealth	Extremely required my cup of #GRECOBE- #Th...	Twitter Web Client	134	GRECOBE|TheGreenCoffee|winter
20737009	1.08722e18	2019-01-21T05:07:33Z	Swineshead_LLC	@DrewTumaABC7 HAIL over the north Berkeley...	Twitter for Android	65	winter|cuddle|hotchocolate
9.3523e17	1.08722e18	2019-01-21T05:07:32Z	NydiaRaquel25	@hallmarkchannel #Winter Castle is number 12 ...	Twitter Web App	166	Winter|WinterfestMovieCountdow
3050006650	1.08721e18	2019-01-21T05:06:46Z	naomieyspaart23	Goodmorning monday!x extra dagje werken #ei...	Twitter for iPhone	137	eigenlijkeendagjevrij|extrawerken
39261405	1.08721e18	2019-01-21T05:04:45Z	cyyoung99	Snowing and 16 degrees out in Missouri..... #wi...	Twitter for Android	51	winter
9.04892e17	1.08721e18	2019-01-21T05:04:09Z	andreybaze	?????? ?????? ????? ?, ?? ? -15 C ???-?? ???????...	Instagram	207	ski|openseason|skiride|winter

Now, we can run basic queries on these databases.

Query 1: We select user_id with tweets data having source as twitter for android.



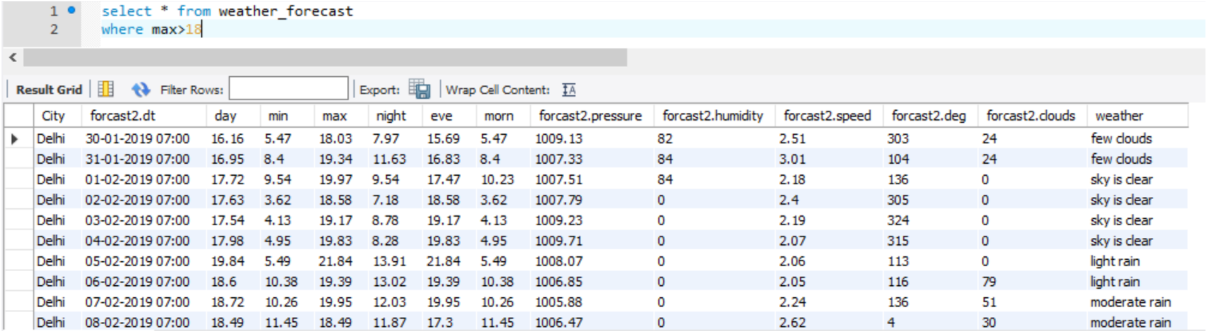
```
select user_id from winter_all
where source="Twitter for Android";
```

user_id
20737009
39261405
1730377189
1.04985e18
1.08044e18
6521142
7.48029e17
28544847

Thus, in the result we see that only user_id having source as twitter for android are displayed.

SQL Queries on Winter weather data and twitter data for #Winter

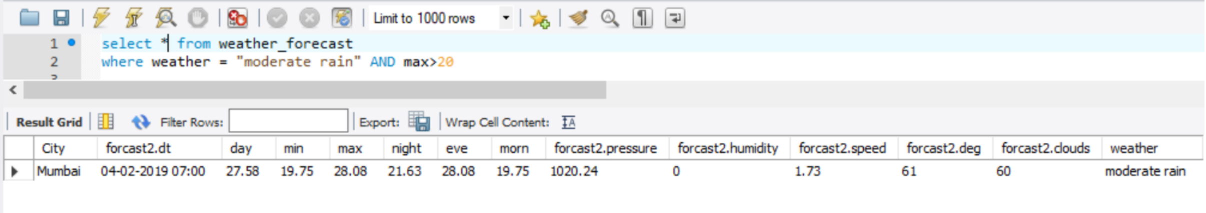
Query 2: From the weather_forecast database, we select the data having the maximum temperature greater than 18 degree Celsius.



```
select * from weather_forecast
where max>18
```

City	forcast2.dt	day	min	max	night	eve	morn	forcast2.pressure	forcast2.humidity	forcast2.speed	forcast2.deg	forcast2.clouds	weather
Delhi	30-01-2019 07:00	16.16	5.47	18.03	7.97	15.69	5.47	1009.13	82	2.51	303	24	few clouds
Delhi	31-01-2019 07:00	16.95	8.4	19.34	11.63	16.83	8.4	1007.33	84	3.01	104	24	few clouds
Delhi	01-02-2019 07:00	17.72	9.54	19.97	9.54	17.47	10.23	1007.51	84	2.18	136	0	sky is clear
Delhi	02-02-2019 07:00	17.63	3.62	18.58	7.18	18.58	3.62	1007.79	0	2.4	305	0	sky is clear
Delhi	03-02-2019 07:00	17.54	4.13	19.17	8.78	19.17	4.13	1009.23	0	2.19	324	0	sky is clear
Delhi	04-02-2019 07:00	17.98	4.95	19.83	8.28	19.83	4.95	1009.71	0	2.07	315	0	sky is clear
Delhi	05-02-2019 07:00	19.84	5.49	21.84	13.91	21.84	5.49	1008.07	0	2.06	113	0	light rain
Delhi	06-02-2019 07:00	18.6	10.38	19.39	13.02	19.39	10.38	1006.85	0	2.05	116	79	light rain
Delhi	07-02-2019 07:00	18.72	10.26	19.95	12.03	19.95	10.26	1005.88	0	2.24	136	51	moderate rain
Delhi	08-02-2019 07:00	18.49	11.45	18.49	11.87	17.3	11.45	1006.47	0	2.62	4	30	moderate rain

Query 3: From the weather_forecast data, we display only the information of cities having moderate rain and maximum temperature greater than 20.

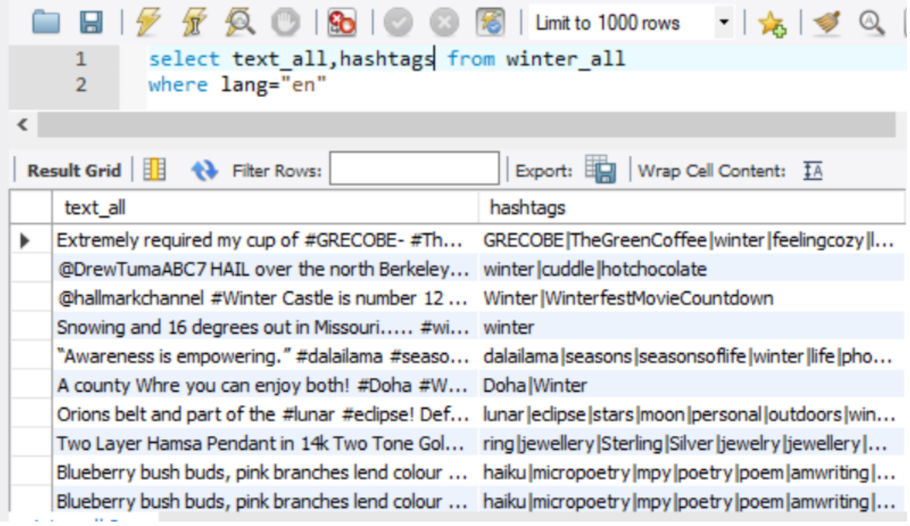


```
select * from weather_forecast
where weather = "moderate rain" AND max>20
```

City	forcast2.dt	day	min	max	night	eve	morn	forcast2.pressure	forcast2.humidity	forcast2.speed	forcast2.deg	forcast2.clouds	weather
Mumbai	04-02-2019 07:00	27.58	19.75	28.08	21.63	28.08	19.75	1020.24	0	1.73	61	60	moderate rain

Thus, only Mumbai has such conditions so only one result is displayed.

Query 4: From the winter hashtags data, we display the text and the hashtags on tweets that are tweeted in English.

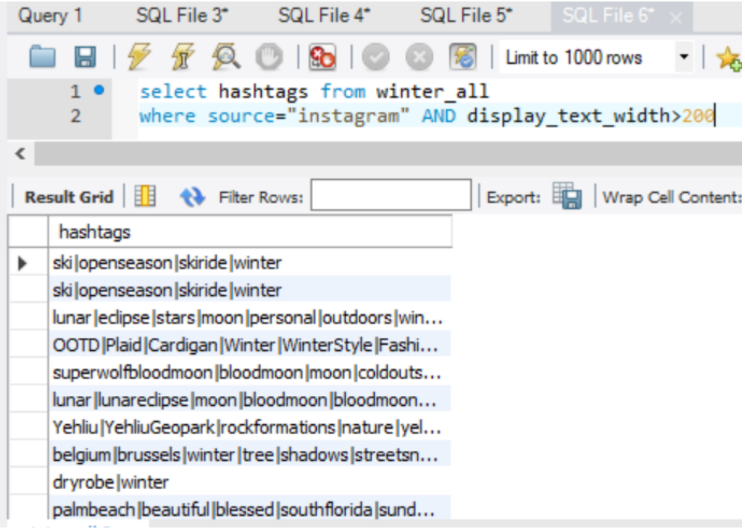


```
select text_all,hashtags from winter_all
where lang="en"
```

text_all	hashtags
Extremely required my cup of #GRECOBE- #Th...	GRECOBE|TheGreenCoffee|winter|feelingcozy|l...
@DrewTumaABC7 HAIL over the north Berkeley...	winter|cuddle|hotchocolate
@hallmarkchannel #Winter Castle is number 12 ...	Winter|WinterfestMovieCountdown
Snowing and 16 degrees out in Missouri..... #wi...	winter
"Awareness is empowering." #dalailama #seaso...	dalailama|seasons|seasonsoflife|winter|life|pho...
A county Whre you can enjoy both! #Doha #W...	Doha|Winter
Orions belt and part of the #lunar #eclipse! Def...	lunar|eclipse|stars|moon|personal|outdoors|win...
Two Layer Hamsa Pendant in 14k Two Tone Gol...	ring|jewellery|Sterling|Silver|jewelry|jewellery|...
Blueberry bush buds, pink branches lend colour ...	haiku|micropoetry|mpy|poetry|poem|amwriting|...
Blueberry bush buds, pink branches lend colour ...	haiku|micropoetry|mpy|poetry|poem|amwriting|...

SQL Queries on Winter weather data and twitter data for #Winter

Query 5: From the winter hashtags data, we display the display text width and hashtags when the source is Instagram.

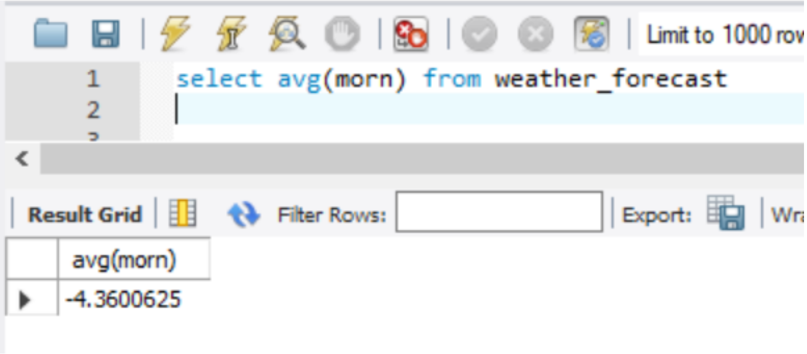


```sql
select hashtags from winter_all
where source="instagram" AND display_text_width>200
```

hashtags
ski|openseason|skiride|winter
ski|openseason|skiride|winter
lunar|eclipse|stars|moon|personal|outdoors|win...
OOTD|Plaid|Cardigan|Winter|WinterStyle|Fashi...
superwolfbloodmoon|bloodmoon|moon|coldouts...
lunar|lunareclipse|moon|bloodmoon|bloodmoon...
Yehliu|YehliuGeopark|rockformations|nature|yel...
belgium|brussels|winter|tree|shadows|streetsn...
dryrobe|winter
palmbeach|beautiful|blessed|southflorida|sund...

Some Queries consisting of statistical functions are:

Query 1: From the weather forecast database, we find the average morning temperature.

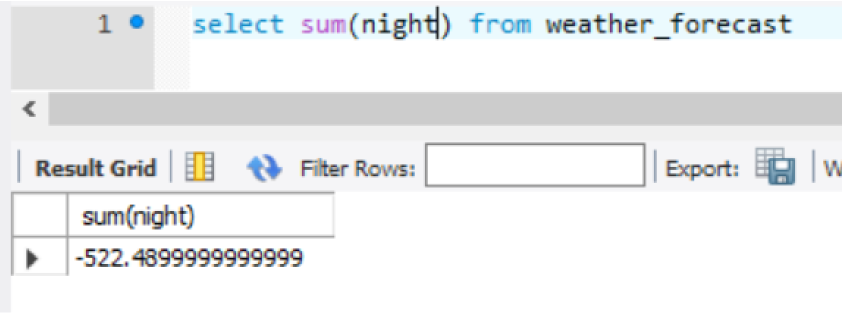


```sql
select avg(morn) from weather_forecast
```

avg(morn)
-4.3600625

Hence, the average morning temperature is -4.36.

Query 2: From the weather forecast database, we find the sum of night temperature.

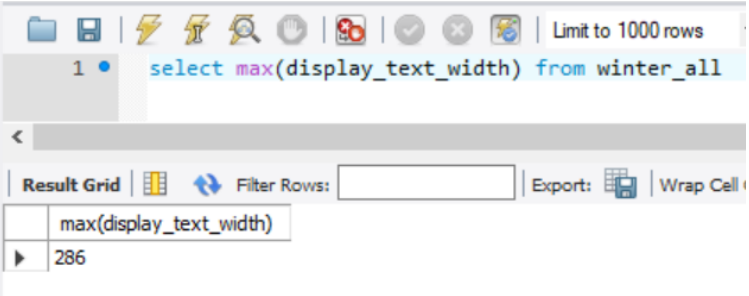


```sql
select sum(night) from weather_forecast
```

sum(night)
-522.4899999999999

SQL Queries on Winter weather data and twitter data for #Winter

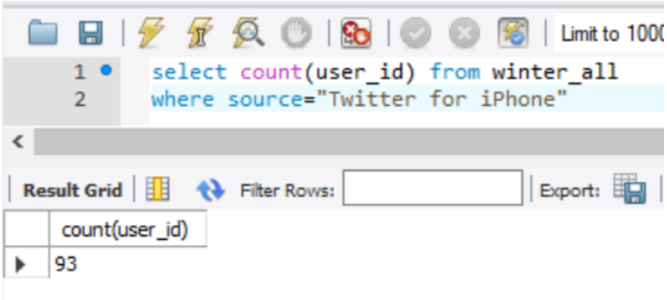
Query 3: From the twitter database, we can find the maximum text display width.



```
select max(display_text_width) from winter_all
```

max(display_text_width)
286

Query 4: From the twitter winter hashtags, find the number of users that have tweeted from iPhones.



```
select count(user_id) from winter_all
where source="Twitter for iPhone"
```

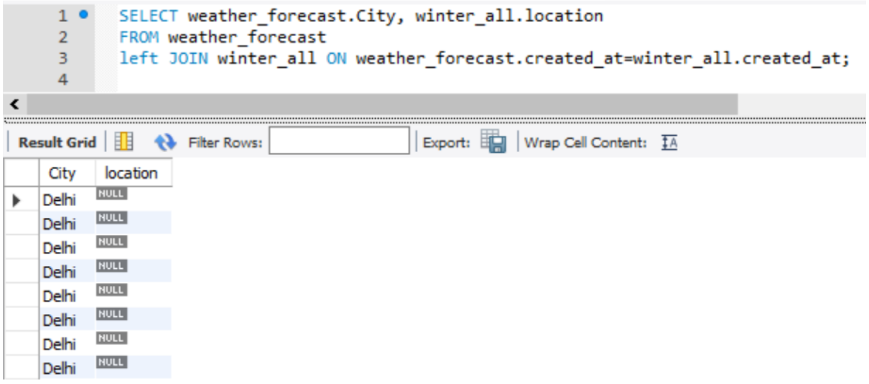
count(user_id)
93

Demonstrating Joins for the databases:

Query 1: Left Join

Left join returns all the records from table 1 and only the matched records from table 2. It returns NULL in table 2 if there is no match.

For our demonstration, I have joined the two databases with created_at attribute as the common attribute. I have displayed the city names from weather_forecast database and location from Winter_all database.



```
SELECT weather_forecast.City, winter_all.location
FROM weather_forecast
left JOIN winter_all ON weather_forecast.created_at=winter_all.created_at;
```

City	location
Delhi	NULL
Delhi	NULL
Delhi	NULL
Delhi	NULL
Delhi	NULL
Delhi	NULL
Delhi	NULL
Delhi	NULL

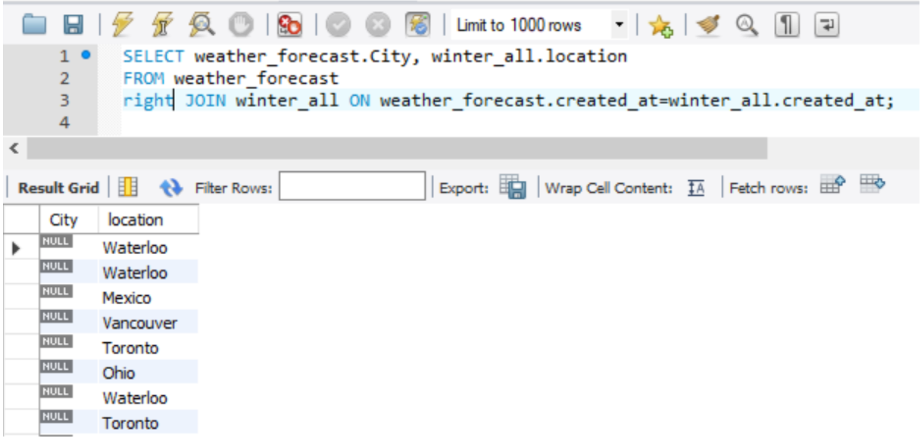
In the above result, it is seen that it returns all the records from table 1 (weather_forecast) and matched records (in this case Null) from table 2 (winter_all).

Query 2: Right Join

Right join returns the matched records from table 1 and all the records from table 2. It returns NULL in table 1 if there is no match.

SQL Queries on Winter weather data and twitter data for #Winter

For our demonstration, I have joined the two databases with created_at attribute as the common attribute. I have displayed the city names from weather_forecast database and location from Winter_all database.



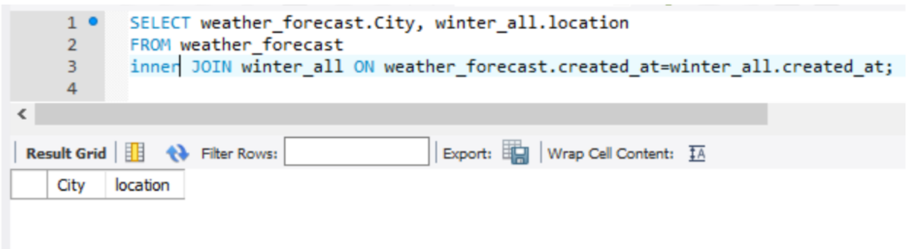
```
SELECT weather_forecast.City, winter_all.location
FROM weather_forecast
right JOIN winter_all ON weather_forecast.created_at=winter_all.created_at;
```

City	location
NULL	Waterloo
NULL	Waterloo
NULL	Mexico
NULL	Vancouver
NULL	Toronto
NULL	Ohio
NULL	Waterloo
NULL	Toronto

In the above result, it is seen that it returns the matched records (in this case Null) from table 1 (weather_forecast) and all the records from table 2 (winter_all).

Query 3: Inner Join

Inner Joins return records with matching values from both the tables.



```
SELECT weather_forecast.City, winter_all.location
FROM weather_forecast
inner JOIN winter_all ON weather_forecast.created_at=winter_all.created_at;
```

City	location

In our demonstration, we see that the result does not return any rows. This means that there are no matching values in both the databases.

Conclusion:

In this report, I have explained how the data was collected and what it looks like. I have connected the databases to AWS and then run it using Workbench. I have performed some basic sql functions on the databases and also executed some statistical functions on it. Along with it, I have also performed some joins on the databases.

References:

1. https://rstudio-pubs-static.s3.amazonaws.com/152810_90f63f8a10fa40578627bf7d8ff3ad06.html
2. https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-table.html
3. https://www.dofactory.com/sql/select-count-sum-avg