DATABASE FOUNDATIONS FOR BUSINESS ANALYTICS
(BUAN 6320)

GROUP 14

PROJECT REPORT

Submitted By: -

Sahejbir Singh Kumar
Indrajeet S Thakare
Bala Krishna Bobbili
Mano Snigdha Devara
Shreya Shamarthi
Harshitha Reddy Nandikonda

ABOUT THE DATASET

We have chosen our dataset (named Bay Area Bike Share) from Kaggle about the bike trips around the San Francisco Bay Area. The data enables quick, easy, and affordable bike trips around the San Francisco Bay Area. They produce data releases with specifics about the stations nearby, available bikes and docks, and trips taken by customers and subscribers of the service. There are three data tables named – station, status, and trip. Station table contains the bike station data such as the id, name, city, dock count etc. The trip table has data about the availed bike trips like start and end time, station duration etc. The status table contains the number of bikes and docks that are available for a given station at a given time.

AIM OF THE PROJECT

The aim of this project is to establish a database and exhibit skill in extracting data from database files using SQL as well as the ability to examine and analyze these values.

BUSINESS UNDERSTANDING

The bay area's bike share data, given by a bike rental company named Bike Share, is used to extract meaningful insights. On examining and analyzing the data, the purpose is to verify the statistics and draw actionable conclusions so that the business can run smoothly. From the gathered data, we can mainly identify the most and least preferred bike stations, bike models and routes. We want to determine how different variables can affect the number of bikes rented, along with their duration. Using these insights, we can optimize our resources by allocating more bikes in the most preferred areas, similarly, by identifying the most preferred bike models we can optimize the ratio of different bikes present in a bike station.

The business goals are: -

- Identifying trips with longest duration.
- Identifying how frequently rides are over 24 hours.
- Identifying how many users have subscribed to the service.
- Identifying whether unsubscribed users are taking longer trips or shorter trips.
- Extracting the average duration for trips availed by subscribed users.
- Identifying the most popular start and end stations.
- Identifying most popular routes.
- Identifying how many stations are installed each year in each city.

On analyzing the aforementioned information, we can suggest strategies to improve the efficiency of the business, for example, identifying which why some routes are more

popular, which is because of more available stations in those routes. This suggests that more stations need to be added in the least popular routes.

DATA UNDERSTANDING

- **Identifying data type, information, values, scales and range of data in each column.**

Station Table:

Column Name	Data Type	Description
station_id	INT	Unique id for each station
name	VARCHAR	Name of each station
latitude	DOUBLE	Provides information about the latitude the station is located at
longitude	DOUBLE	Provides information about the longitude the station is located at
dock_count	INT	Provides information about the number of docks at each station
city	VARCHAR	Tells the city where the station is located
installation_date	DATE	Provides the date of installation of the station

Status Table:

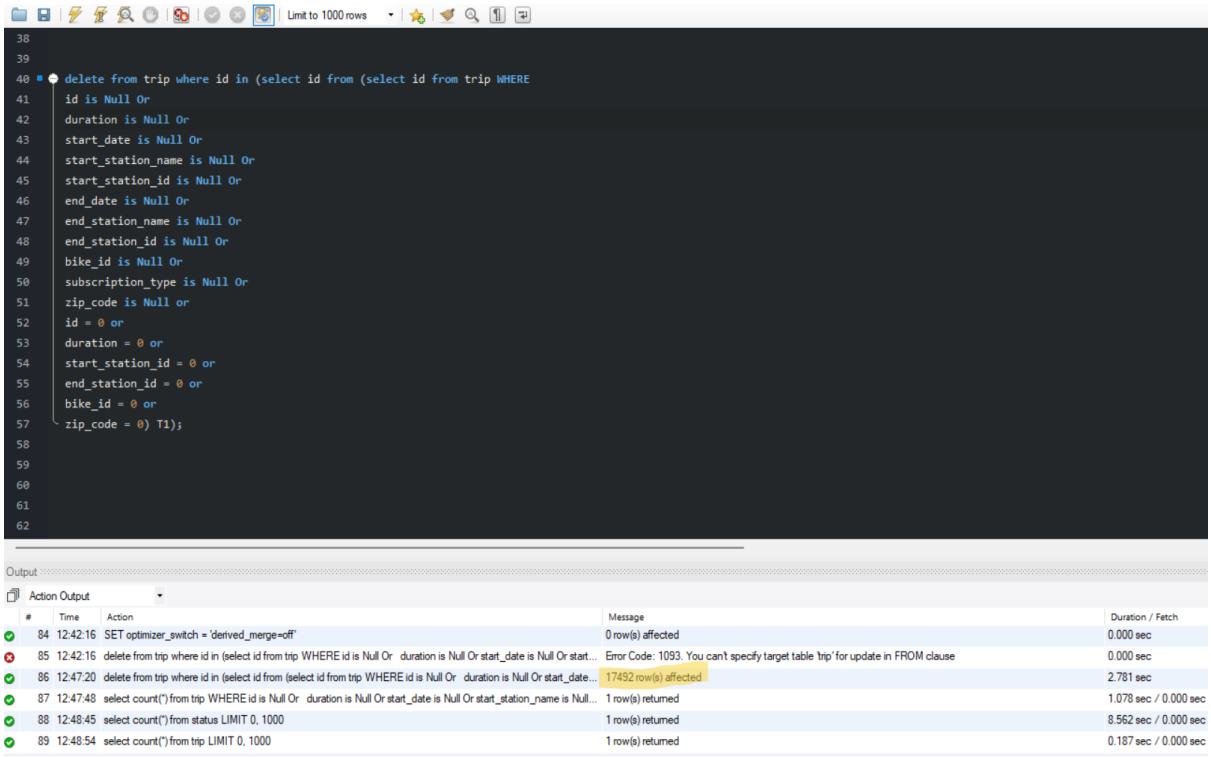
Column Name	Data Type	Description
station_id	INT	Provides unique id for each station
bikes_available	INT	Provides the information about the number of bikes available at each station
docks_available	INT	Provides the information about the number of docks available at each station
time	DATE	It gives the information of the years (range 2013-14)
status_id	INT	This has been created to give a unique id to each station id

Trip Table:

Column Name	Data Type	Description
trip_id	INT	Unique id for each trip
duration	INT	Gives information about the time taken for each trip
start_date	VARCHAR	Gives information about the start date of each trip
start_station_name	VARCHAR	Gives information about the station name at which the trip started for each trip
start_station_id	INT	Gives the id of the start station of each trip
end_date	VARCHAR	Gives the information of the end date of each trip
end_station_name	VARCHAR	Gives information of the location where the trip ended
end_station_id	INT	Gives the id of the end station for each trip
bike_id	INT	Gives unique id for each bike
subscription_type	VARCHAR	Gives the information of whether the trip has been taken by a subscribed user or an unsubscribed user
zip_code	INT	It gives the zip codes of each station

- **Verify data quality**

We checked for missing data (NULL values) and removed those values to maintain the data quality.



```
delete from trip where id in (select id from (select id from trip WHERE
id is Null Or
duration is Null Or
start_date is Null Or
start_station_name is Null Or
start_station_id is Null Or
end_date is Null Or
end_station_name is Null Or
end_station_id is Null Or
bike_id is Null Or
subscription_type is Null Or
zip_code is Null or
id = 0 or
duration = 0 or
start_station_id = 0 or
end_station_id = 0 or
bike_id = 0 or
zip_code = 0) T1);
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
84	12:42:16	SET optimizer_switch = 'derived_merge=off'	0 row(s) affected	0.000 sec
85	12:42:16	delete from trip where id in (select id from trip WHERE id is Null Or duration is Null Or start_date is Null Or start...	Error Code: 1093. You can't specify target table 'trip' for update in FROM clause	0.000 sec
86	12:47:20	delete from trip where id in (select id from (select id from trip WHERE id is Null Or duration is Null Or start_date...	17492 row(s) affected	2.781 sec
87	12:47:48	select count(*) from trip WHERE id is Null Or duration is Null Or start_date is Null Or start_station_name is Null...	1 row(s) returned	1.078 sec / 0.000 sec
88	12:48:45	select count(*) from status LIMIT 0, 1000	1 row(s) returned	8.562 sec / 0.000 sec
89	12:48:54	select count(*) from trip LIMIT 0, 1000	1 row(s) returned	0.187 sec / 0.000 sec

We have changed column names of ‘id’ to ‘station_id’, ‘trip_id’ in station and trip table respectively. Column ‘name’ in station table has been changed to ‘station_name’. Ther queries used to perform this are:

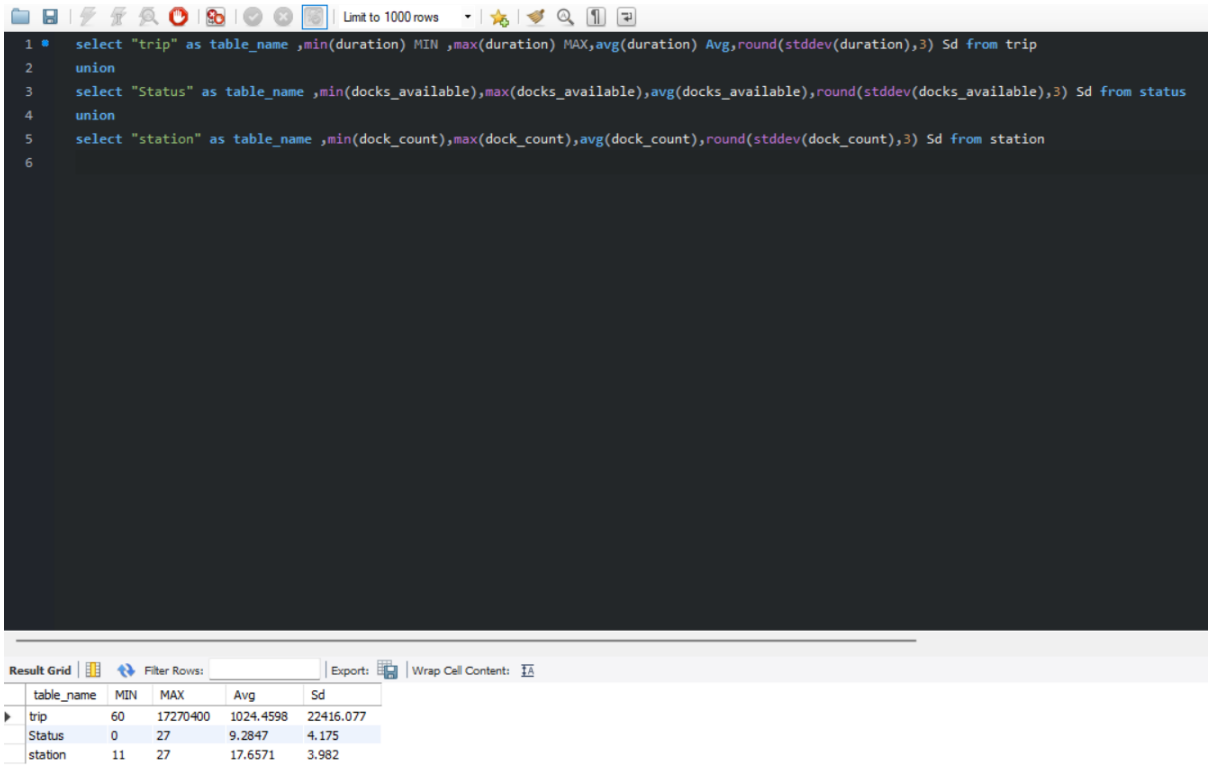
alter table station rename column column id to station_id; alter table station rename column name to station_name

alter table trip rename column id to trip_id

Since there is no Primary Key in status table, we have added a column with incremental value and assigned that as a Primary Key. The name of the column is ‘status_id’. The query is mentioned below:

alter table status add status_id int unsigned not null auto_increment, add primary key (status_id)

- **Provide a simple statistic for the columns**

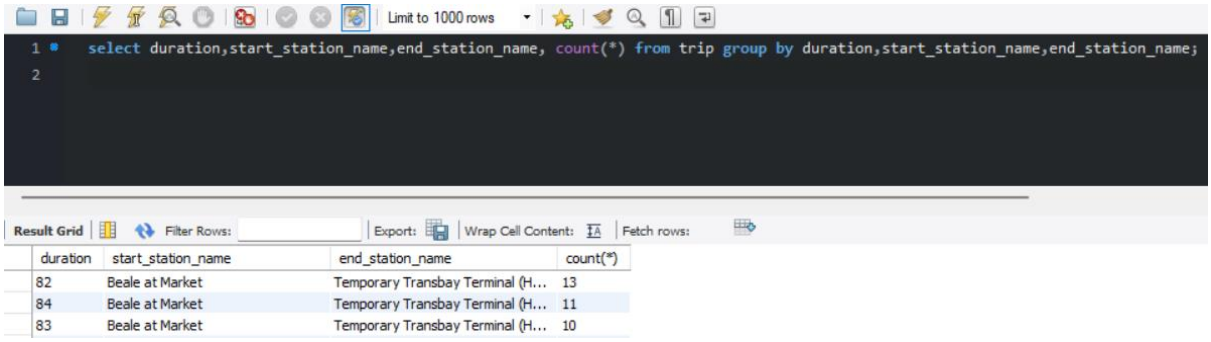


```
select "trip" as table_name ,min(duration) MIN ,max(duration) MAX,avg(duration) Avg,round(stddev(duration),3) Sd from trip
union
select "Status" as table_name ,min(docks_available),max(docks_available),avg(docks_available),round(stddev(docks_available),3) Sd from status
union
select "station" as table_name ,min(dock_count),max(dock_count),avg(dock_count),round(stddev(dock_count),3) Sd from station
```

Result Grid

table_name	MIN	MAX	Avg	Sd
trip	60	17270400	1024.4598	22416.077
Status	0	27	9.2847	4.175
station	11	27	17.6571	3.982

- **Finding relationships between columns:**



```
select duration,start_station_name,end_station_name, count(*) from trip group by duration,start_station_name,end_station_name;
```

Result Grid

duration	start_station_name	end_station_name	count(*)
82	Beale at Market	Temporary Transbay Terminal (H...	13
84	Beale at Market	Temporary Transbay Terminal (H...	11
83	Beale at Market	Temporary Transbay Terminal (H...	10

As seen in the above snapshot, we find that the time taken between start station “Harry Bridges Plaza (Ferry Building)” and end station “2nd at Folsom” is **quite approximate** to each other. Hence, the column ‘duration’ is **related** on the two columns ‘start_station_name’ and ‘end_station_station’.

DESIGNING THE DATABASE

- **Schema Design**

  The three entities/relations in the dataset are – station, status and trip.

Entity	Attributes	Keys
	station_id	**Primary Key**
	name	
	latitude	
station	longitude	
	dock_count	
	city	
	installation_date	

Entity	Attributes	Keys
	trip_id	**Primary Key**
	duration	
	start_date	
trip	start_station_name	
	start_station_id	
	end_date	
	end_station_name	
	end_station_id	
	bike_id	
	subscription_type	
	zip_code	

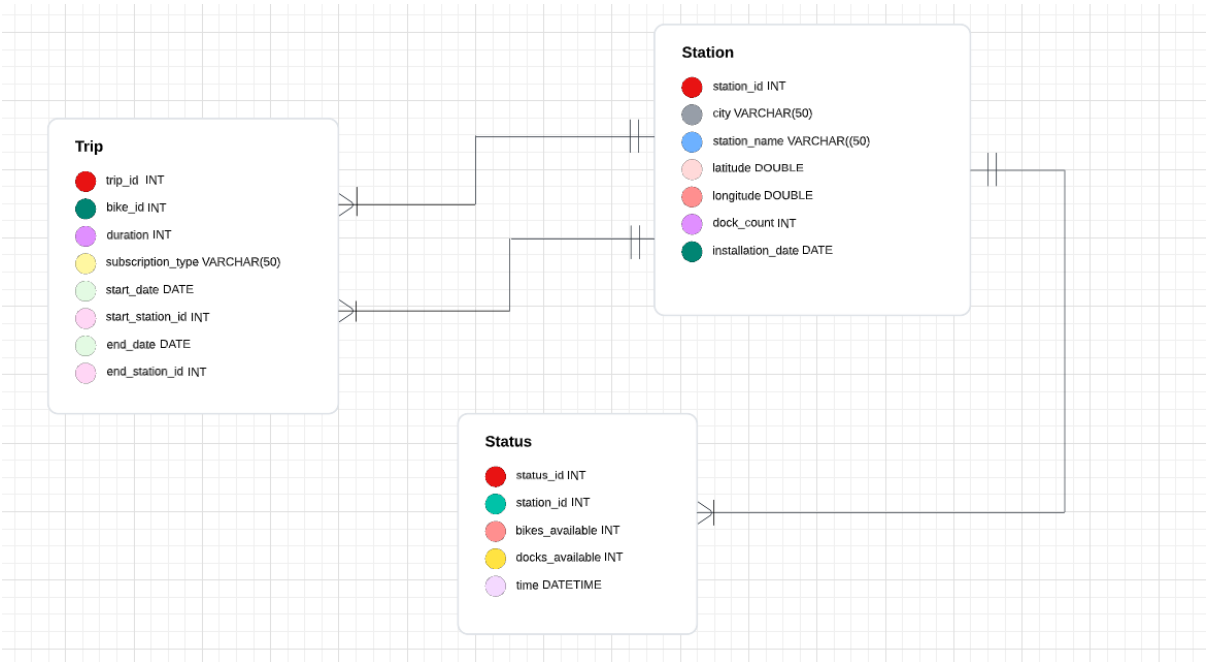
Entity	Attributes	Keys
	station_id	
	bikes_available	
	docks_available	
status	time	
	status_id	**Primary Key**

The trip table contains start and end station id from station table.
The status table contains availability of bikes and docks at a station from station table.
The station table has unique id and name for each station.

- **Model all the constraints you believe should be there in your schema**

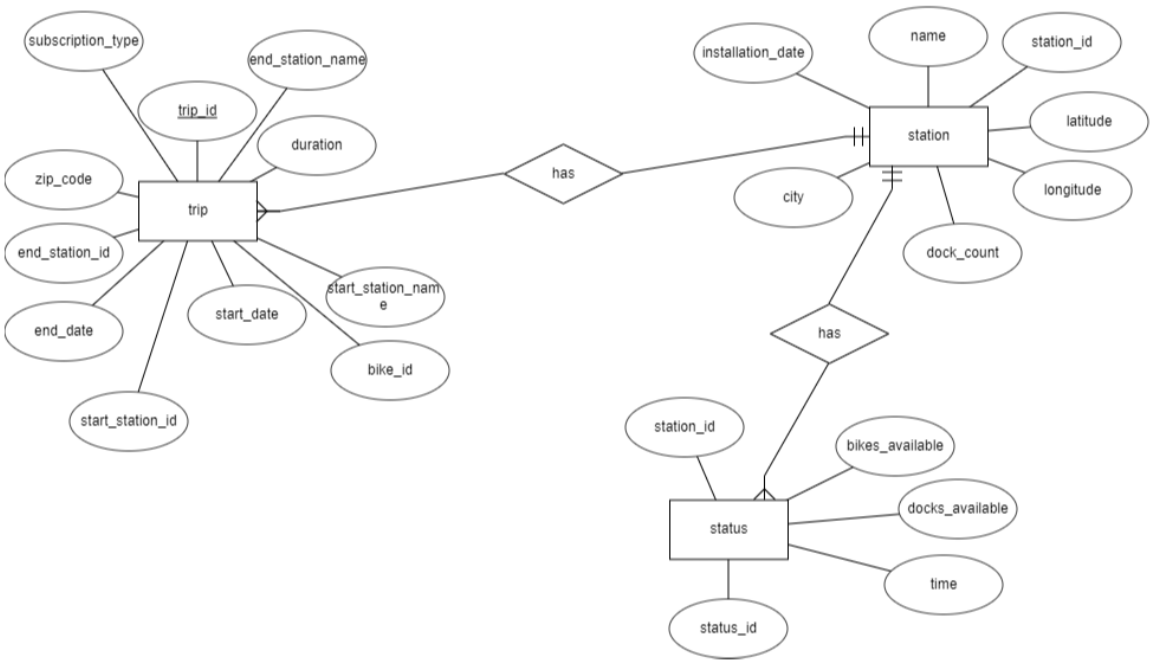
The trip table and station table have a **many-to-many** relationship.
The station table has a **Primary Key** named station_id which is a **Foreign Key** to the trip table as start_station_id and end_station_id.

- **ER Diagram**



Red dot indicates the **Primary Key(s)**.

- **Translate your ER diagram into relations**



- **Normalization of the Schema**

  Functional dependencies of the dataset are given below:

  trip_id - > {start_date, end_date, start_station_id, end_station_id, duration, bike_id, subscription_type}
  station_id - > {station_name, latitude, longitude, dock_count, city, installation_date}
  station_id, station_name - > {latitude, longitude, dock_count, city, installation_date}
  station_id, longitude, latitude - > { station_name, dock_count, city, installation_date}
  status_id - > {time, station_id, bikes_available, docks_available}

  Check if the keys you have chosen for your relations are minimal

  In the table Status, we have defined the set of attributes, A, functional dependency (F: X ->Y).
  Set of attributes, A = {status_id, time, station_id, bikes_available, docks_available}
  X -> status_id
  Functional dependency, F: status_id - > {time, station_id, bikes_available, docks_available}
  Now we need the closure of the attribute. Finding X^+:
  X^+ = {status_id}
  Comparing the functional dependency F with the closure X^+ ,we get the below details:
  X^+ = {status_id, time, station_id, bikes_available, docks_available}
  This suggests that the Primary Key is status_id.
  Using the above method, we can conclude that the Primary Keys for trip table and station table are trip_id and station_id respectively.

  Check if your schema is in Boyce-Codd Normal Form

  For station table, ‘station_id’ is the primary key and all attributes are in the relation as shown below

  {station_id} -> {station_name, longitude, latitude, dock_count, city, installation_date}

  Similarly, ‘status_id’ is the primary key in relation status and the attributes are present as shown:

  {status_id} -> {station_id, bikes_available, docks_available, time}

  Trip table has ‘trip_id’ as the primary key having all other attributes:

  {trip_id} -> {duration, start_date, start_station_name, start_station_id, end_station_id, end_date, end_station_name, subscription_type, bike_id, zip_code}

This concludes that the above table is in BCNF as there is no violation in the above-mentioned functional dependencies. There is no need to update the E-R diagram as the dataset is already in BCNF form.

- **Create your database using latest version of schema and import the data.**

  Since the data size is too large, we couldn't import the data manually using data import wizard. We used the below query to import the data. Infile access was given to execute this query.

  set global local_infile = 1;

  use project;

  create table status(station_id int,bike_available int,docks_available int,times date);

  load data local infile "E:/DB Project/Project/data/status.csv"

  into table status

  fields terminated by ","

  Enclosed by ""

  lines terminated by "\n"

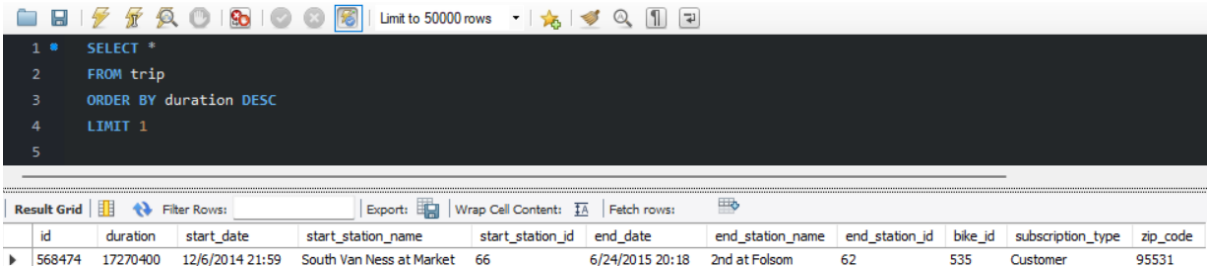
  ignore 1 rows;

  This was done for all the three tables respectively.

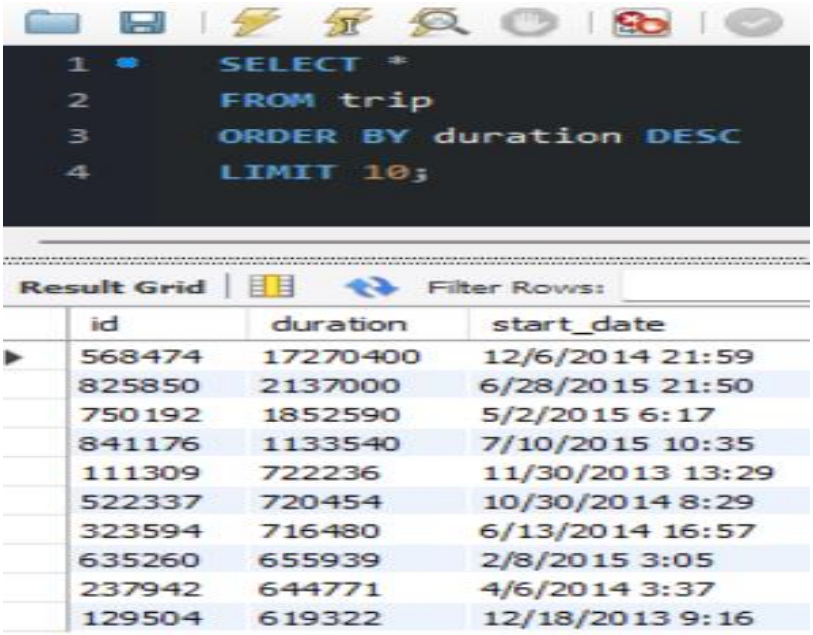
DATA CLEANING AND DATABASE TESTING

We have checked all the columns and the values they contain for each table (shown above) and the numerical statistics has also been checked as above.

- **Identifying trips with longest duration.**



It appears that the longest ride recorded is over six months long. It is very likely that this could have been a glitch. Pulling up the top 10 longest rides will hopefully provide some context as to whether this datapoint is a fluke.



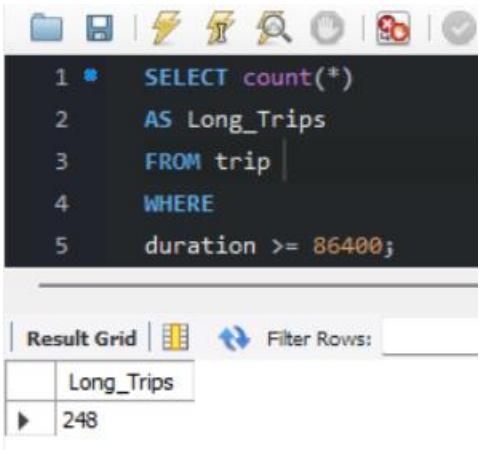
```
SELECT *
FROM trip
ORDER BY duration DESC
LIMIT 10;
```

Result Grid | Filter Rows:

id	duration	start_date
568474	17270400	12/6/2014 21:59
825850	2137000	6/28/2015 21:50
750192	1852590	5/2/2015 6:17
841176	1133540	7/10/2015 10:35
111309	722236	11/30/2013 13:29
522337	720454	10/30/2014 8:29
323594	716480	6/13/2014 16:57
635260	655939	2/8/2015 3:05
237942	644771	4/6/2014 3:37
129504	619322	12/18/2013 9:16

The longest ride of 6 months is a fluke comparing with the other longest rides.

- **Identifying how frequently rides are over 24 hours.**

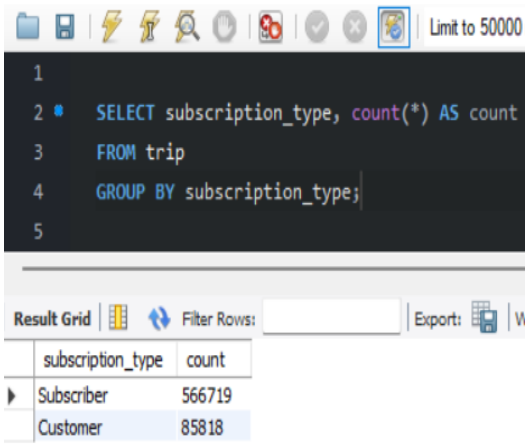


```
SELECT count(*)
AS Long_Trips
FROM trip
WHERE
duration >= 86400;
```

Result Grid | Filter Rows:

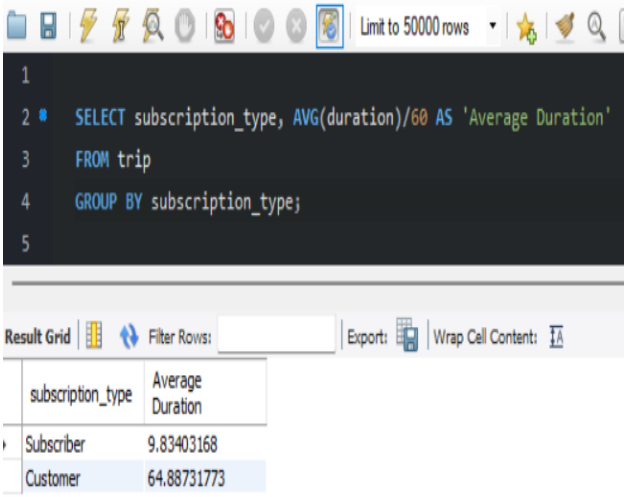
Long_Trips
248

- **Identifying how many users have subscribed to the service and whether unsubscribed users are taking longer trips or shorter trips**



```
SELECT subscription_type, count(*) AS count
FROM trip
GROUP BY subscription_type;
```

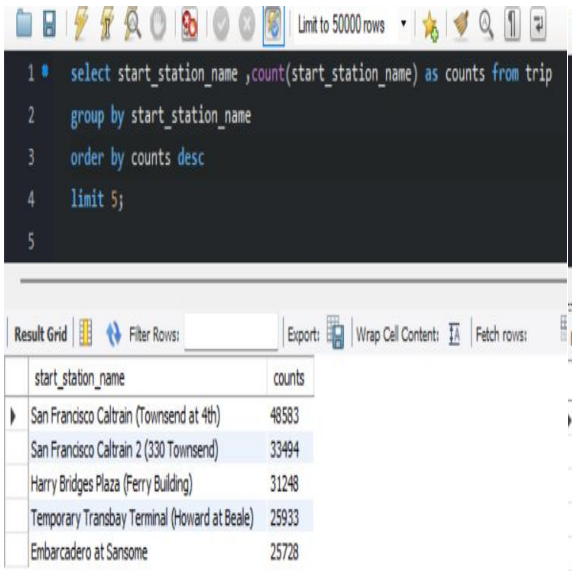
subscription_type	count
Subscriber	566719
Customer	85818



```
SELECT subscription_type, AVG(duration)/60 AS 'Average Duration'
FROM trip
GROUP BY subscription_type;
```

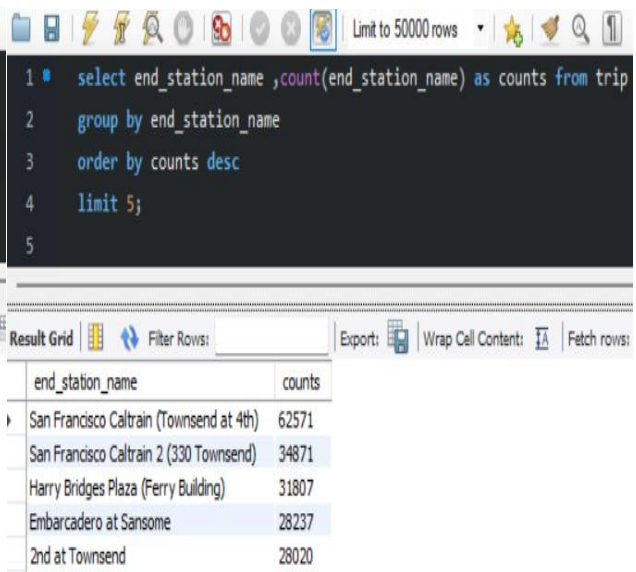
subscription_type	Average Duration
Subscriber	9.83403168
Customer	64.88731773

- **Identifying the most popular start and end stations**



```
select start_station_name ,count(start_station_name) as counts from trip
group by start_station_name
order by counts desc
limit 5;
```

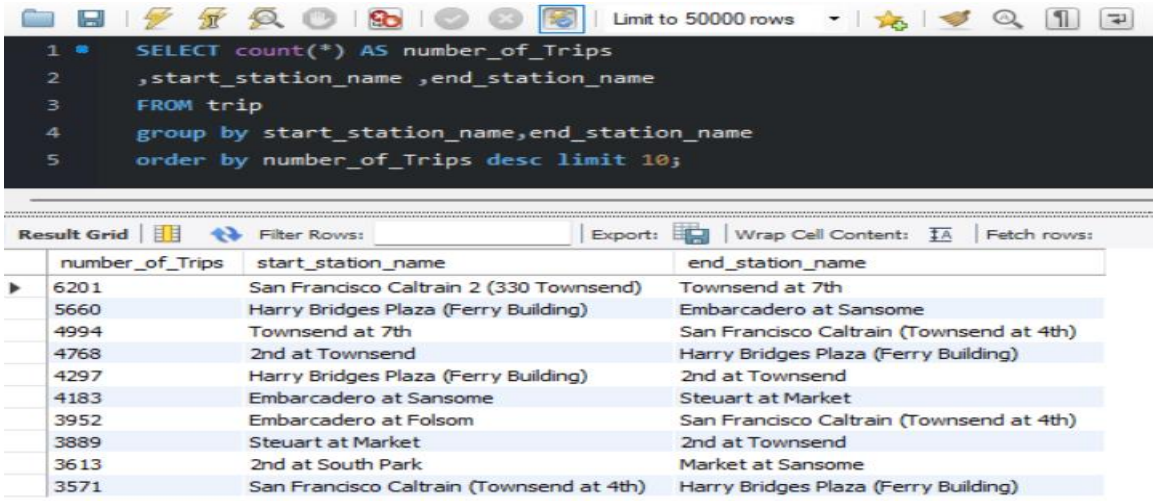
start_station_name	counts
San Francisco Caltrain (Townsend at 4th)	48583
San Francisco Caltrain 2 (330 Townsend)	33494
Harry Bridges Plaza (Ferry Building)	31248
Temporary Transbay Terminal (Howard at Beale)	25933
Embarcadero at Sansome	25728



```
select end_station_name ,count(end_station_name) as counts from trip
group by end_station_name
order by counts desc
limit 5;
```

end_station_name	counts
San Francisco Caltrain (Townsend at 4th)	62571
San Francisco Caltrain 2 (330 Townsend)	34871
Harry Bridges Plaza (Ferry Building)	31807
Embarcadero at Sansome	28237
2nd at Townsend	28020

- **Identifying most popular routes.**

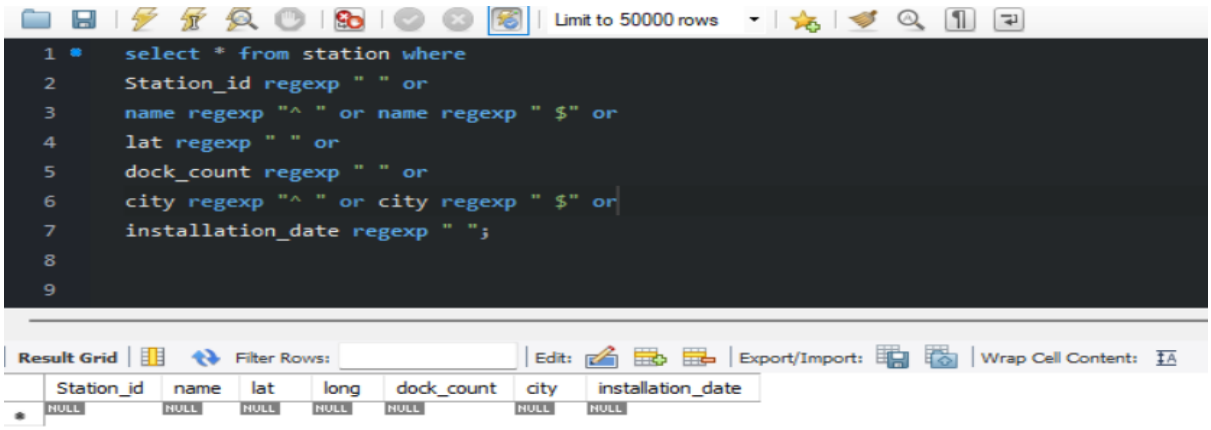


```
SELECT count(*) AS number_of_Trips
,start_station_name ,end_station_name
FROM trip
group by start_station_name,end_station_name
order by number_of_Trips desc limit 10;
```

number_of_Trips	start_station_name	end_station_name
6201	San Francisco Caltrain 2 (330 Townsend)	Townsend at 7th
5660	Harry Bridges Plaza (Ferry Building)	Embarcadero at Sansome
4994	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
4768	2nd at Townsend	Harry Bridges Plaza (Ferry Building)
4297	Harry Bridges Plaza (Ferry Building)	2nd at Townsend
4183	Embarcadero at Sansome	Steuart at Market
3952	Embarcadero at Folsom	San Francisco Caltrain (Townsend at 4th)
3889	Steuart at Market	2nd at Townsend
3613	2nd at South Park	Market at Sansome
3571	San Francisco Caltrain (Townsend at 4th)	Harry Bridges Plaza (Ferry Building)

Check for missing values or data errors or values that does not seem to be valid (e.g., sometimes there are white spaces in some of the cells either before or after the value)

As shown below, there are no white spaces present in any of the columns in the three tables. Snapshots for station, status and trip table has been shown below.



```
select * from station where
Station_id regexp " " or
name regexp "^ " or name regexp " $" or
lat regexp " " or
dock_count regexp " " or
city regexp "^ " or city regexp " $" or
installation_date regexp " ";
```

Station_id	name	lat	long	dock_count	city	installation_date
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
select * from trip where
Trip_id regexp "^ " or Trip_id regexp " $" or
duration regexp "^ " or duration regexp " $" or
start_date regexp "^ " or start_date regexp " $" or
start_station_name regexp "^ " or start_station_name regexp " $" or
start_station_id regexp "^ " or start_station_id regexp " $" or
end_date regexp "^ " or end_date regexp " $" or
end_station_name regexp "^ " or end_station_name regexp " $" or
end_station_id regexp "^ " or end_station_id regexp " $" or
bike_id regexp "^ " or bike_id regexp " $" or
subscription_type regexp "^ " or subscription_type regexp " $" or
zip_code regexp "^ " or zip_code regexp " $";
```

Trip_id	duration	start_date	start_station_name	start_station_id	end_date	end_station_name	end_station_id	bike_id	subscription_type	zip_code
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
select * from status where station_id regexp "^ " or station_id regexp " $" or
bikes_available regexp "^ " or bikes_available regexp " $" or
docks_available regexp "^ " or docks_available regexp " $" or
times regexp "^ " or times regexp " $" or
status_id regexp "^ " or status_id regexp " $";
```

station_id	bikes_available	docks_available	times	status_id
NULL	NULL	NULL	NULL	NULL

Try to query your database especially from more than one table (by joining them) to see if the results make sense or not
Check if the results of these queries match what you expect
Check if the constraints are working properly

- **Identifying the most preferred stations**

```
select s.station_name,count(*) as count from station s inner join trip t on s.station_id = t.start_station_id
group by s.station_name order by count desc;
```

station_name	count
San Francisco Caltrain (Townsend at 4th)	48583
San Francisco Caltrain 2 (330 Townsend)	33494
Harry Bridges Plaza (Ferry Building)	31248
Temporary Transbay Terminal (Howard at Beale)	25933
Embarcadero at Sansome	25728
2nd at Townsend	25350
Steuart at Market	24220
Market at Sansome	23643
Townsend at 7th	23500
Market at 10th	19815

- **Identifying the least preferred stations**

```
select s.station_name,count(*) as count from station s inner join trip t on s.station_id = t.start_station_id
group by s.station_name order by count;
```

station_name	count
Redwood City Public Library	198
Franklin at Maple	219
San Mateo County Center	286
Redwood City Medical Center	309
Mezes Park	337
Stanford in Redwood City	482
Park at Olive	710
Santa Clara County Civic Center	848
California Ave Caltrain Station	970
Rengstorff Avenue / California Street	1084

We have added foreign keys in the tables as follows:

```
ALTER TABLE status
ADD FOREIGN KEY (Station_id) REFERENCES station(station_id);

ALTER TABLE Trip
ADD FOREIGN KEY (Start_station_id) REFERENCES station(station_id);

ALTER TABLE Trip
ADD FOREIGN KEY (end_station_id) REFERENCES station(station_id);
```

Action Output

#	Time	Action	Message
1	17:23:39	Apply changes to station	Changes applied
2	17:24:25	Apply changes to trip	Changes applied
3	17:27:19	ALTER TABLE status ADD FOREIGN KEY (Station_id) REFERENCES station(station_id)	71984434 row(s) affected Records: 71984434 Duplicates: 0 Warnings: 0
4	17:41:16	ALTER TABLE Trips ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)	Error Code: 1146. Table 'sqlproject.trips' doesn't exist
5	17:41:29	ALTER TABLE Trip ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)	Error Code: 1072. Key column 'PersonID' doesn't exist in table
6	17:44:05	select * from status LIMIT 0, 50000	50000 row(s) returned
7	17:44:23	select * from trip LIMIT 0, 50000	50000 row(s) returned
8	17:45:45	ALTER TABLE Trip ADD FOREIGN KEY (Start_station_id) REFERENCES station(station_id)	652537 row(s) affected Records: 652537 Duplicates: 0 Warnings: 0
9	17:45:51	ALTER TABLE Trip ADD FOREIGN KEY (end_station_id) REFERENCES station(station_id)	652537 row(s) affected Records: 652537 Duplicates: 0 Warnings: 0

The **station** table has a **Primary Key** named '**station_id**' which is a **Foreign Key** in **status** table.

The **station** table has a **Primary Key** named '**station_id**' which is a **Foreign Key** to the **trip** table as '**start_station_id**' and '**end_station_id**'.