Database Foundations for Business Analytics (BUAN 6320.502)
Project 1 (GROUP 4)

Step 1: Choose a Dataset

Our dataset (called Bay Area Bike Share) concerning bike travels around the San Francisco Bay Area was selected from the website kaggle.com. The information permits convenient, inexpensive, and rapid bike journeys throughout the San Francisco Bay Area. They create data releases that include details on the local stations, the available bikes and docks, and the trips taken by users and service subscribers.

The goal is to create a database and use SQL to extract data from database files and demonstrate the ability to look through and evaluate these values to provide meaningful business strategies.

OTHER DETAILS ABOUT DATASET

- Size of the dataset is 6GB.
- The dataset has structured data only.
- The data has missing values in one column called zip code in the trip.csv file.
- This data set explains the trends of the number of customers renting bikes from one station to another and number of trips.
- **File Description:**
  It is a collection of data samples of the years 2013 and 2014.
  The dataset comprises of three files and each of the files has the following information:
  Trip.csv
  It has the data about individual bike trips
  Status.csv
  It has the data about the number of bikes and docs available for a given station and minute.
  Station.csv
  It has the data that represents a station where users can pick up or return bikes.

Step 2: Business Understanding

2.1 Why did we choose to collect the dataset?

To understand the trends of bike rental patterns of every station and how they vary by time of day and the day of the week.

2.2 What are the goals?

- Identifying most time-consuming journeys.
- Quantifying how many times the rides last more than a day.
- Counting the number of customers who have signed up for the service.

- Determining whether the frequency of subscribed users availing the service is higher than the customers.
- Extracting the typical length of the journeys taken by subscribers.
- Determining the most frequently visited start and end stations.
- Determining the busiest routes.
- Quantifying how many stations are added each year in all cities.

2.3 How did we achieve it?

Found join conditions and used join command to execute joining of tables.

2.4 Insightful information gathered

Worked on the following questions and ran queries for the same.

2.4.1 What was the trip with the longest duration?

Please refer APPENDIX 2.4.1

2.4.2 How common is it for a ride to go over 24 hours?

Please refer APPENDIX 2.4.2

2.4.3 Do unregistered(Customer) users take longer or shorter trips?

Please refer APPENDIX 2.4.3

2.4.4 Number of station installed each year in different cities

Please refer APPENDIX 2.4.8

2.5 Improvements suggested to improve the business

If we plot a bar graph for
Station ids v/s Docks available and **Station ids v/s Bikes available.**
We can observe that **Station id 6** has less number of Docks and Bikes available. Hence they have to improve their business at Station 6 by increasing the number of Bikes and Docks count.

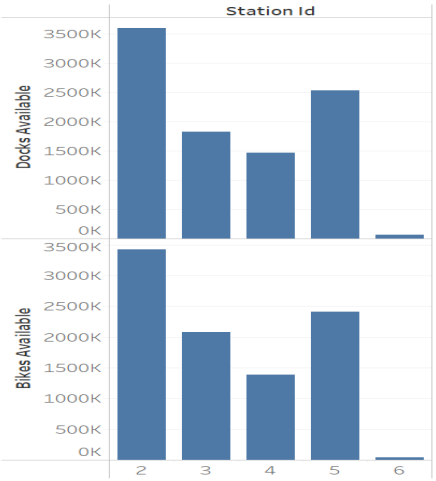


Image:1

Step 3: Data Understanding

Understanding each column of the above mentioned tables in Step 1.

Refer the below tables for points 3.1, 3.2
Following has the information of name, type and description of each column of each table used in the dataset.

Trip Table :
There are eleven columns in the Trip table

COLUMN NAME	TYPE	DESCRIPTION
ID	INT	It is the unique id for each trip.
DURATION	INT	It gives the information about time taken for each trip.
START_DATE	VARCHAR(26)	It gives the information about the start date of each trip.
START_STATION_NAME	VARCHAR(128)	It gives information about the branches of the start location(city) of each trip.
START_STATION_ID	INT	It indicates the id of the start station of each trip.
END_DATE	VARCHAR(26)	It gives information about the end date of each trip.
END_STATION_NAME	VARCHAR(128)	It gives information about the branches of the end location(city) of each trip.
END_STATION_ID	INT	It indicates the id of the end station of each trip.
BIKE_ID	INT	It is the unique id for each bike.
SUBSCRIPTION_TYPE	VARCHAR(26)	It describes the type of customer. It has two categories CASUAL(not a subscriber) and SUBSCRIBER
ZIP_CODE	INT	It is the unique id for each location.

Status Table:
There are five columns in the table

COLUMN NAME	TYPE	DESCRIPTION
STATION_ID	INT	It is a unique id for each station.
BIKES_AVAILABLE	INT	It gives information about the number of bikes available at each station.
DOCKS_AVAILABLE	INT	It gives information about the number of docks available at each station.
TIME	DATE	It gives information of the years sample 2013 and 2014
status_id	INT	It is a unique id for each record in status table, this column is added by us because there was not any unique key in status table

Station Table :
There are seven columns in the table.

COLUMN NAME	TYPE	DESCRIPTION
ID	INT	It is a unique id for each station
NAME	VARCHAR(128)	It gives the information about the name of each station.
LATITUDE	FLOAT	It gives the information about the latitude the station is located at
LONGITUDE	FLOAT	It gives the information about the longitude the station is located at
DOCK_COUNT	INT	It gives information about the number of docks available in a particular station.
CITY	VARCHAR(26)	It gives information about the city the station is located at
INSTALLATION_DATE	DATE	It gives information about the date a particular station was installed in.

3.3 What are some of the values each column contains?

Following are samples of some of the values of each column in each table of our dataset:

Status Table:

```
select * from status;
```

Query Result — Fetched 50 rows in 0.016 seconds

	STATION_ID	BIKES_AVAILABLE	DOCKS_AVAILABLE	TIME
1	2	2	25	29-AUG-13 12:06:01
2	2	2	25	29-AUG-13 12:07:01
3	2	2	25	29-AUG-13 12:08:01
4	2	2	25	29-AUG-13 12:09:01
5	2	2	25	29-AUG-13 12:10:01
6	2	2	25	29-AUG-13 12:11:01
7	2	2	25	29-AUG-13 12:12:01
8	2	2	25	29-AUG-13 12:13:01
9	2	2	25	29-AUG-13 12:15:01
10	2	2	25	29-AUG-13 12:16:02

Image:2

Trip Table:

```
select * from trip;
```

Script Output — Query Result — Fetched 50 rows in 0.003 seconds

	ID	DURATION	START_DATE	START_STATION_NAME	START_STATION_ID	END_STATION_ID	END_STATION_NAME	BIKE_ID	SUBSCRIPTION_TYPE	ZIP_CODE
1	4576	63	29-Aug-2013	South Van Ness at Market	66	66	South Van Ness at Market	520	Subscriber	94127
2	4607	70	29-Aug-2013	San Jose City Hall	10	10	San Jose City Hall	661	Subscriber	95138
3	4130	71	29-Aug-2013	Mountain View City Hall	27	27	Mountain View City Hall	48	Subscriber	97214
4	4251	77	29-Aug-2013	San Jose City Hall	10	10	San Jose City Hall	26	Subscriber	95060
5	4299	83	29-Aug-2013	South Van Ness at Market	66	67	Market at 10th	319	Subscriber	94103
6	4927	103	29-Aug-2013	Golden Gate at Polk	59	59	Golden Gate at Polk	527	Subscriber	94109
7	4500	109	29-Aug-2013	Santa Clara at Almaden	4	5	Adobe on Almaden	679	Subscriber	95112
8	4563	111	29-Aug-2013	San Salvador at 1st	8	8	San Salvador at 1st	687	Subscriber	95112
9	4760	113	29-Aug-2013	South Van Ness at Market	66	66	South Van Ness at Market	553	Subscriber	94103
10	4258	114	29-Aug-2013	San Jose City Hall	10	11	MLK Library	107	Subscriber	95060

Image:3

Station Table:

```
select * from Station;
```

Script Output — Query Result — Fetched 50 rows in 0.002 seconds

	ID	NAME	LAT	LONGITUDE	DOCK_COUNT	CITY	INSTALLATION_DATE
1	2	San Jose Diridon Caltrain Station	37.329	-121.901	27	San Jose	06-AUG-13 00:00:00
2	3	San Jose Civic Center	37.33	-121.888	15	San Jose	05-AUG-13 00:00:00
3	4	Santa Clara at Almaden	37.333	-121.894	11	San Jose	06-AUG-13 00:00:00
4	5	Adobe on Almaden	37.331	-121.893	19	San Jose	05-AUG-13 00:00:00
5	6	San Pedro Square	37.336	-121.894	15	San Jose	07-AUG-13 00:00:00
6	7	Paseo de San Antonio	37.333	-121.886	15	San Jose	07-AUG-13 00:00:00
7	8	San Salvador at 1st	37.33	-121.885	15	San Jose	05-AUG-13 00:00:00
8	9	Japantown	37.348	-121.894	15	San Jose	05-AUG-13 00:00:00
9	10	San Jose City Hall	37.337	-121.886	15	San Jose	06-AUG-13 00:00:00
10	11	MLK Library	37.335	-121.885	19	San Jose	06-AUG-13 00:00:00

Image:4

And the following data shows minimum, maximum values of the some numeric columns in each table:

In Status table: considering DOCKS_AVAILABLE column we found minimum(DOCKS_AVAILABLE) = 0, maximum(DOCKS_AVAILABLE) = 27

In Trip table: considering DURATION(in seconds) column we found minimum(DURATION) = 60 seconds, maximum(DURATION) = 722236 seconds

In Station table: considering DOCK_COUNT column we found
minimum(DOCK_COUNT) = 11, maximum(DOCK_COUNT) = 27

3.4 Verify the data quality?

3.4.1

3.4.1.1.Verify the quality of the name of the columns?

As the names of all the columns in tables are meaningful and indicate the actual purpose of that column we did not change the column names.

3.4.1.2. Data Standardization

- In the data the Trip table was having two date columns, Start_date and End_date, but the format of the date was not proper as per oracle sql.
  So we standardised the data by using excel functions like the Concatenate date function. We created update queries for all the records to update our Start_date and End_date in the required convenient format.

Below is the screenshot of the same for reference:

```
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4576;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4607;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4130;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4251;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4299;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4927;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4500;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4563;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4760;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4258;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4549;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4498;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4965;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4557;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4386;
Update trip set start_date ='29-Aug-13' , End_date = '29-Aug-13' where id =4749;
```

Image:5

- In the given data set, there is no unique id for the Status table, so we have created a column called status_id for the status table that has the unique data for each record in the table.

3.4.2 Missing values

There were 5810 rows with null values in the Trip Table and it is 3.1% of the total number of rows of trip table data.
Using the below query as shown in images the missing values were removed.

```
DELETE  from trip where zip_code is null;
```

Query Result | Script Output

Task completed in 0.121 seconds

```
5,810 rows deleted.
```

Image:6

```
DELETE from trip where zip_code is null;
```

Row Count

5,810 Rows

OK

Fetched 50 rows in 0.014 seconds

START_STATION_NAME	START_STATION_ID	END_DATE	END_STATION_NAME	END_STATION_ID	BIKE_ID	SUBSCRIPTION_TYPE	ZIP_CODE
5:59 Market at Sansome	77	8/29/2013 16:03	Temporary Transbay Terminal (Howard at Beale)	55	624	Customer	(null)
5:29 Grant Avenue at Columbus Avenue	73	8/29/2013 15:36	Washington at Kearney	46	474	Customer	(null)
7:42 Spear at Folsom	49	8/29/2013 17:49	Yerba Buena Center of the Arts (3rd @ Howard)	68	334	Customer	(null)
9:36 Grant Avenue at Columbus Avenue	73	8/29/2013 19:44	Temporary Transbay Terminal (Howard at Beale)	55	353	Customer	(null)
7:22 Beale at Market	56	8/29/2013 17:32	Market at 4th	76	626	Customer	(null)
3:09 Market at Sansome	77	8/29/2013 18:21	San Francisco City Hall	58	578	Customer	(null)
4:17 San Francisco Caltrain 2 (330 Townsend)	69	8/29/2013 14:30	Powell Street BART	39	541	Customer	(null)
3:07 Temporary Transbay Terminal (Howard at Beale)	55	8/29/2013 18:20	Grant Avenue at Columbus Avenue	73	353	Customer	(null)
3:56 Powell at Post (Union Square)	71	8/29/2013 19:10	Yerba Buena Center of the Arts (3rd @ Howard)	68	466	Customer	(null)
3:19 Commercial at Montgomery	45	8/29/2013 13:34	Market at 10th	67	506	Customer	(null)
3:01 Embarcadero at Sansome	60	8/29/2013 13:16	Embarcadero at Vallejo	48	579	Customer	(null)
7:50 Powell Street BART	39	8/29/2013 18:05	San Francisco Caltrain (Townsend at 4th)	70	574	Customer	(null)

Image:7

After Cleaning Data we can see in below image there are no null values in trip table:

```
desc trip;

select * 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 ;
```

Script Output | Query Result | Query Result 1

All Rows Fetched: 0 in 0.036 seconds

ID	DURATION	START_D...	START_S...	START_S...	END_DATE	END_STA...	END_STA...	BIKE_ID	SUBSCRIP...	ZIP_CODE

Image:8

3.4.3 Duplicate data

The selected dataset was free of duplicate data.

3.5 Statistics of the data for each column:
(This part will be done in Step 5 using MySql queries)

3.6 Relationships between the columns of the data

- start_station_id and End_station_id in trip table is related to id in Station table
- select count(*),duration,start_station_name,end_station_name from temp_trip group by duration,start_station_name,end_station_name;

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

Query Result

Fetched 50 rows in 0.129 seconds

	COUN...	DURATION	START_STATION_NAME	END_STATION_NAME
1	28	247	San Francisco Caltrain 2 (330 Townsend)	Townsend at 7th
2	26	233	San Francisco Caltrain 2 (330 Townsend)	Townsend at 7th
3	25	214	San Francisco Caltrain 2 (330 Townsend)	Townsend at 7th
4	25	216	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
5	24	232	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
6	24	239	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
7	23	221	San Francisco Caltrain 2 (330 Townsend)	Townsend at 7th
8	23	221	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
9	23	213	San Francisco Caltrain 2 (330 Townsend)	Townsend at 7th
10	23	215	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
11	23	214	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
12	23	212	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
13	23	199	San Francisco Caltrain 2 (330 Townsend)	Townsend at 7th
14	22	237	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
15	22	222	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
16	21	245	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)

Image:9

From the attached screenshot, we can conclude that for a displayed sample of 79 trips, the duration between the start and end station {San Francisco Caltrain 2 (330 Townsend), Townsend at 7th} is almost similar to each other. This shows that the duration column is related to the start and end station names' column.

Step 4: Design a Database

4.1 Requirement Analysis

This is already done in step 2, where we gathered and analyzed the requirements of our project.

4.2 Data Understanding

This is already done in step 3, where we focussed on each table of our dataset to analyse relationships among columns and tables, and also identified missing & duplicate values.

4.3 Schema Design

4.3.a Find entities, their attributes, their primary keys, and relationships between them

Entities: 3 entities in our table are: Trip, Status and Station

Attributes: Following are the attributes in each table:

Trip: ID, DURATION, START_DATE, START_STATION_NAME, START_STATION_ID, END_DATE, END_STATION_NAME, END_STATION_ID, BIKE_ID, SUBSRIPTION_TYPE, ZIP_CODE

Status: STATION_ID, BIKES_AVAILABLE, DOCKS_AVAILABLE, TIME, STATUS_ID

Station: ID, NAME, LAT, LONGITUDE, DOCK_COUNT, CITY, INSTALLATION_DATE

Primary keys: In trip table ('ID' is the primary key),
In Station table('ID' is the primary key),
In Status table('status_id' is the primary key, this column is added after downloading the dataset as there was no unique key to this table)

Relationships: Each 'Station' can be start station for one or many trips and can also be end station for one or many trips(i.e., one-to-many) and,
Each 'Station' HAS 'one or many statuses'(i.e., one-to-many)
The above mentioned relationships are binary relationships.

4.3.b Model all the constraints you believe should be there in your schema

In our dataset, we have the following primary keys(ID in Trip table, ID in Station table and status_id in Status table) and foreign keys(STATION_ID in Status table and START_STATION_ID, END_STATION_ID in Trip table).
All the above mentioned primary keys and foreign keys satisfy the key constraint(i.e., they are unique values) and referential integrity constraint. Apart from this, these columns don't have null values, so entity integrity constraint is also satisfied.

4.3.c Draw and ER diagram of your dataset

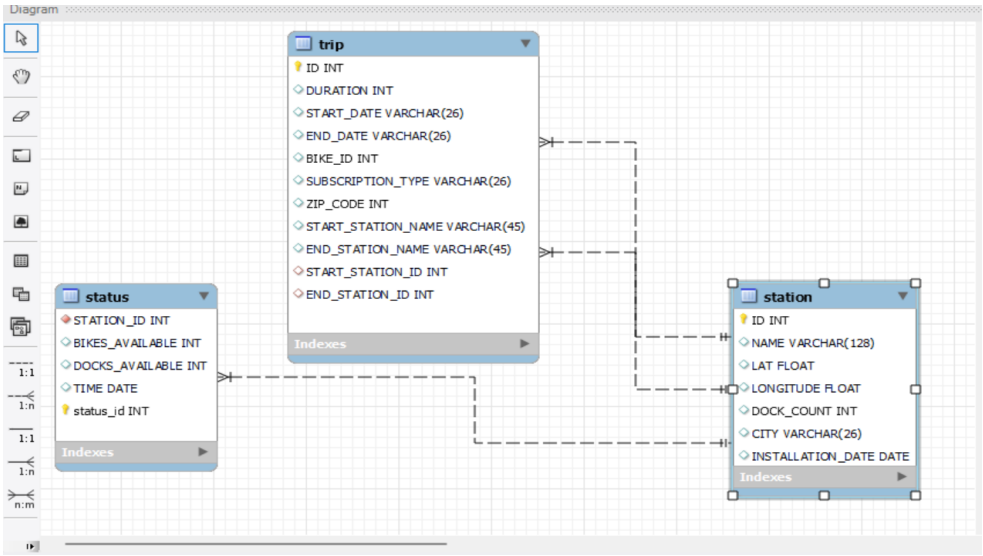


Image:10

4.3.d Translate your ER diagram into relations

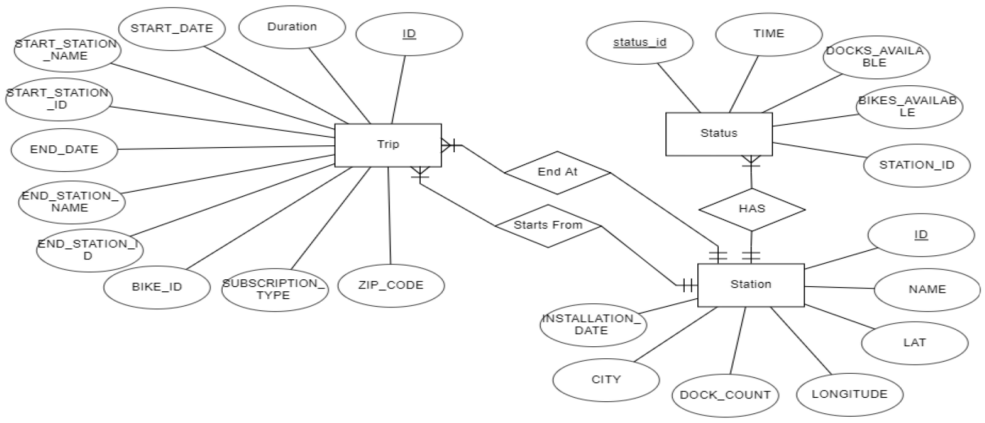


Image:11

4.4 Schema Normalization

4.4.a Find all the functional dependencies you can from your schema

Find all the functional dependencies you can from your schema

Station Table:
{id} -> {name, latitude, longitude, dock_count, city, installation_date}
{id, longitude, latitude} -> {name, dock_count, city, installation_date}
{id, name} -> {dock_count,city, installation_date, longitude, latitude}

Status Table:
{status_id} -> {time, station_id, bikes_available, docks_available}

Trip Table:
{id} -> {duration, start_date, end_date, bike_id, subscription_type, zip_code, start_station_name, end_station_name, zip_code}

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

CLOSURE :
For Status table, set of all attributes

A = {status_id, time, station_id, bikes_available, docks_available}
and
Functional dependency
F = {status_id} -> {time, station_id, bikes_available, docks_available}
and
X = status_id
X+ = {status_id}
Using F, X+ = {status_id, time, station_id, bikes_available, docks_available}

For Trip table, set of all attributes
A = {duration, start_date, end_date, bike_id, subscription_type }
and
Functional dependency
F = {id} -> { duration, start_date, end_date, bike_id, subscription_type}
and
X = id
X+ = {id}
Using F, X+ = {duration, start_date, end_date, bike_id, subscription_type }

For the Station table set of all attributes
A = {name, latitude, longitude, dock_count, city, installation_date}
and
Functional dependency
F = {id} -> {name, latitude, longitude, dock_count, city, installation_date}
and
X = id
X+ = {id}
Using F X+ = {name, latitude, longitude, dock_count, city, installation_date }

4.4.c Check if your schema is in BCNF (Boyce-Codd Normal Form)

{id} -> {name, latitude, longitude, dock_count, city, installation_date}
All are in same table and id is the key

{status_id} -> {time, station_id, bikes_available, docks_available}
All are in same table and status_id is the key

{id} -> {start_date, end_date, duration, start_station_name, end_station_name, start_station_id, end_station_id, duration, bike_id, subscription_type, zip_code}
All are in same table and id is the key

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

Violates BCNF hence needs to be decomposed

{trip_id} -> {start_station_id, end_station_id}
All are in the same table.
The start_station_id and end_station_id is a subset of trip_id

4.4.d If your schema violates BCNF, bring it to BCNF by decomposing it

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

F -> { {start_station_id, end_station_id} - {id} // Violates BCNF}}}

Hence decomposing trip table into a new Route table having a trip_id

{trip_id, start_station_id, end_station_id} {id, start_date, end_date, subscription_type, duration, zip_code, bike_id}

4.4.e Update your ER diagram with the latest scheme

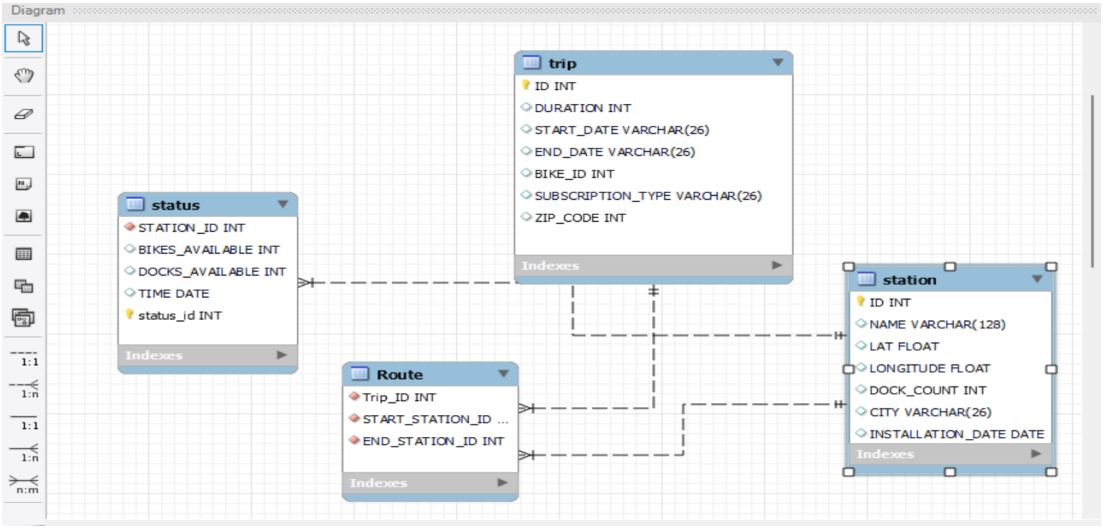


Image:12

4.5 Create your database in MySQL using the latest version of your schema

Created new database as show below:
Created a new table (Route) with column Trip_Id ,Start_Station_Id,End_Station_Id.

```
CREATE TABLE route
  AS (SELECT id,start_STATION_ID,START_STATION_NAME,end_station_id,END_STATION_NAME  FROM trip);

SELECT * FROM ROUTE;
```

Script Output × | Query Result × | Query Result 1 ×

SQL | Fetched 50 rows in 0.002 seconds

	ID	START_STATION_ID	START_STATION_NAME	END_STATION_ID	END_STATION_NAME
1	4576	66	South Van Ness at Market	66	South Van Ness at Market
2	4607	10	San Jose City Hall	10	San Jose City Hall
3	4130	27	Mountain View City Hall	27	Mountain View City Hall
4	4251	10	San Jose City Hall	10	San Jose City Hall
5	4299	66	South Van Ness at Market	67	Market at 10th
6	4927	59	Golden Gate at Polk	59	Golden Gate at Polk
7	4500	4	Santa Clara at Almaden	5	Adobe on Almaden
8	4563	8	San Salvador at 1st	8	San Salvador at 1st
9	4760	66	South Van Ness at Market	66	South Van Ness at Market
10	4258	10	San Jose City Hall	11	MLK Library
11	4549	49	Spear at Folsom	54	Embarcadero at Bryant
12	4498	6	San Pedro Square	4	Santa Clara at Almaden
13	4965	28	Mountain View Caltrain Station	28	Mountain View Caltrain Station

Image:13

4.6 Import the data into your database

4.6.a If there are errors while importing, document these errors in your report and mention how you dealt with them

We did not face any problem while creating and importing our database.

Step 5: Data Cleaning and Database Testing

5.1 For each table in your database, check all the columns and the values they contain

Checked all the columns, each column contains varchar data type and numeric data,Date type and there are no null values

5.2 For numeric columns, check for the statistics, and see what you find

5.2.1 For example: range, mode, mean, median, variance, counts (frequency) Describe what these values mean especially if you found something Interesting

Following images show queries that display Average, Median, Mode, Standard deviation and Range of numeric columns in each table.

For Trip table:

```
select avg(duration),median(duration),stats_mode(duration),stddev(duration),min(duration) || ' to ' ||max(duration) as range from trip;
```

Query Result ×

SQL | All Rows Fetched: 1 in 0.205 seconds

	AVG(DURATION)	MEDIAN(DURATION)	STATS_MODE(DURATION)	STDDEV(DURATION)	RANGE
1	1001.9930100527180756835207990950419396 84	511	376	5905.458821525406855627405036241523938958	60 to 722236

Image:14

Here Duration ranges from 60 to 722236 seconds. From this information we can say that duration varies a lot. And it also shows that some trips last for really small durations.

For Status table:

```
select avg(docks_available),median(docks_available),stats_mode(docks_available),stddev(docks_available),min(docks_available) || ' to ' ||max(docks_available) as range from status;
```

Query Result

All Rows Fetched: 1 in 6.585 seconds

	AVG(DOCKS_AVAILABLE)	MEDIAN(DOCKS_AVAILABLE)	STATS_MODE(DOCKS_AVAILABLE)	STDDEV(DOCKS_AVAILABLE)	RANGE
1	8.264554805245017514460587655399844063 85	8	7	3.1272091186009120634551266965627472371	0 to 27

Image:15

For Station table:

```
select avg(dock_count),median(dock_count),stats_mode(dock_count),stddev(dock_count),min(dock_count) || ' to ' ||max(dock_count) as range  from station;
```

Query Result

All Rows Fetched: 1 in 0 seconds

	AVG(DOCK_COUNT)	MEDIAN(DOCK_COUNT)	STATS_MODE(DOCK_COUNT)	STDDEV(DOCK_COUNT)	RANGE
1	17.657142857142857142857142857142857142 86	15	15	4.0104418574939525625210389132686414266 7	11 to 27

Image:16

5.2.2 You should be looking for missing values, values that seem to be outliers (typically far away from the mean), or data errors or any values that does not seem to be valid (like a typo)

Here Max value(722236) in duration seems to outlier which is very large from mean(1001.99) but we have checked top 50 trips which have duration from 127569 to 722236 so it is not an outlier

5.2.3 Make sure all the values of these columns are from the same type (all numeric)

Values are of same type

5.2.4 Document the problems you find; fix them and explain how you dealt with Them

We did not faced any problem

5.3 For character columns, check for all the values they contain

5.3.1 You should be looking 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)

Data does not contain missing values in dataset for character columns and

Data contains valid data except for the city column in the Station table.

```
select * from station where regexp_like( City, '$+[[:alnum:]]+[[:blank:]]+$');
select * from station where regexp_like(city, '^[[:blank:]]+[[:alnum:]]+');
```

Used the above queries for each type of varchar columns.

5.3.2 Make sure all the values are from the same type and domain

Type and domain of the data is the same in every column.

5.3.3 Document the problems you find, fix them, and explain how you dealt with them

Had trailing and leading space in City COLUMN OF STATION table
Used to remove space

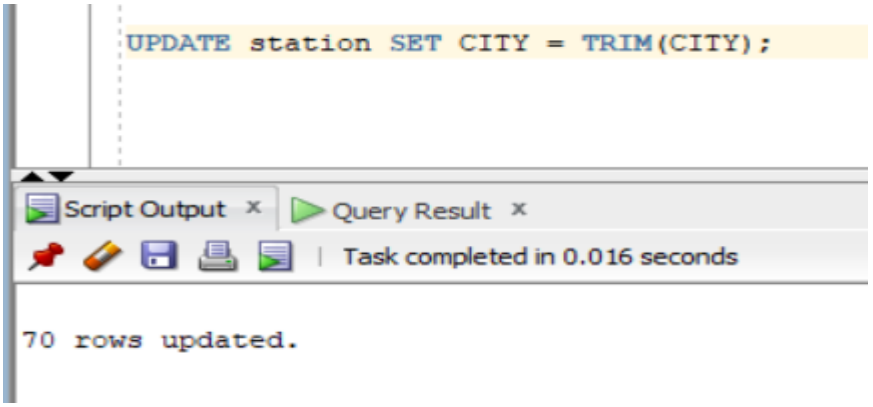


Image:17

5.4 Try to query your database especially from more than one table (by joining them) to see if the results make sense or not

5.4.1 Check if the results of these queries match what you expect

5.4.1.1 Most start popular station

Please refer APPENDIX **5.4.1.1**

5.4.1.2 Most end popular station

Please refer APPENDIX **5.4.1.2**

5.4.1.3 Most popular routes

Please refer APPENDIX **5.4.1.3**

5.4.1.4 Most popular route in each city

Please refer APPENDIX **5.4.1.4**

5.4.2 Check if the constraints are working properly

Checking Foreign Key constraints as shown below:

A.)

As Id from station table is foreign key in route table and status table so we get constraints error as shown in below image:18

```
DELETE FROM STATION WHERE ID =2;
```

Script Output | Query Result

Task completed in 18.908 seconds

```
Error starting at line : 3 in command -
DELETE FROM STATION WHERE ID =2
Error report -
ORA-02292: integrity constraint (UAT.ROUTE_FK1) violated - child record found
```

Image:18

B.)

As ID in trip table is foreign key in route table so we get constraint error as shown in below image:19

```
DELETE FROM trip WHERE ID =4576;
```

Script Output | Query Result

Task completed in 0.015 seconds

```
Error starting at line : 3 in command -
DELETE FROM trip WHERE ID =4576
Error report -
ORA-02292: integrity constraint (UAT.ROUTE_FK3) violated - child record found
```

Image:19

C.)

In the status table we have START_STATION_ID and END_STATION_ID as foreign keys, so when we tried to insert new values in it we got an Integrity Constraint Error as shown in below image:20.

```
insert into status values(120,12,12,'29-AUG-13 12:06:01');
```

Query Result | Script Output

Task completed in 0.027 seconds

```
Error starting at line : 3 in command -
insert into status values(120,12,12,'29-AUG-13 12:06:01')
Error report -
ORA-02291: integrity constraint (UAT.STATUS_FK1) violated - parent key not found
```

Image:20

APPENDIX

2.4.1 Trip with longest duration

SELECT duration from (SELECT * FROM trip ORDER BY duration DESC)
WHERE rownum <= 1;

We find that the longest ride is more than 200 hours which sounds like an error.

```
Select duration from (
SELECT *
FROM trip
ORDER BY duration DESC)
where rownum <= 1
;
```

Query Result

All Rows Fetched: 1 in 0.034 seconds

	DURATION
1	722236

Shown below is the result by considering only the top 10 highest time consumed rides.
SELECT* from (SELECT * FROM trip ORDER BY duration DESC)
WHERE rownum <= 10;

```
SELECT* from (
SELECT *
FROM trip
ORDER BY duration DESC)
WHERE rownum <= 10
;
```

Script Output | Query Result

All Rows Fetched: 10 in 0.025 seconds

	ID	DURATION	START_DATE	END_DATE	BIKE_ID	SUBSCRIPTION_TYPE	ZIP_CODE
1	111309	722236	30-Nov-2013	Dec-08-2013	247	Customer	94301
2	522337	720454	30-Oct-2014	Nov-07-2014	692	Customer	94010
3	129504	619322	18-Dec-2013	Dec-25-2013	653	Subscriber	94041
4	32121	597517	23-Sep-2013	Sep-30-2013	168	Customer	95051
5	812544	552697	18-Jun-2015	Jun-24-2015	70	Customer	90024
6	62246	429384	17-Oct-2013	Oct-22-2013	465	Customer	43113
7	815111	348211	20-Jun-2015	Jun-24-2015	312	Customer	94102
8	36617	329456	26-Sep-2013	Sep-30-2013	162	Customer	94306
9	21760	309479	14-Sep-2013	Sep-18-2013	242	Customer	95112
10	150269	303271	11-Jan-2014	Jan-15-2014	538	Customer	94110

This proves this is not an error.

2.4.2

Quantifying how many times the rides last more than a day.:

SELECT count(*) AS "Long Trips" FROM trip
WHERE duration >= 60*60*24;

```
SELECT count(*)
AS "Long Trips"
FROM trip
WHERE
duration >= 60*60*24;
```

Query Result

SQL | All Rows Fetched: 1 in 0.022 seconds

	Long Trips
1	77

2.4.3

Counting the number of customers who have signed up for the service.

Determining whether the frequency of subscribed users availing the service is higher than the customers.

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

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

uery Result

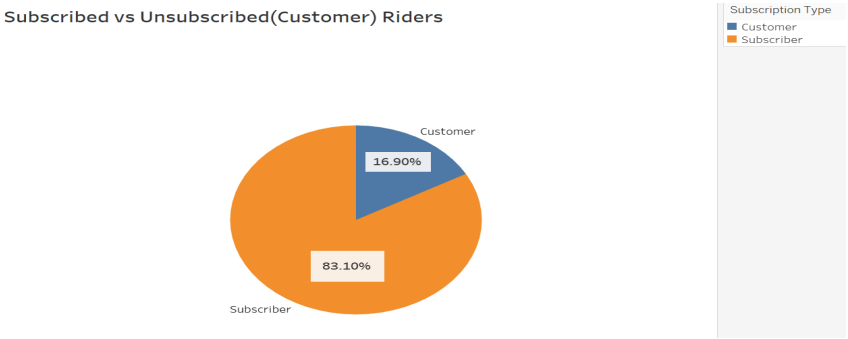
SQL | All Rows Fetched: 2 in 0.041 seconds

	SUBSCRIPTION_TYPE	Average Duration
1	Customer	57.7706695236320725420781317930563319048 5
2	Subscriber	9.836294425727221208516786339264029095 08

From this, we can say that on an average unregistered users take shorter trips when compared to registered users. Therefore, we can infer that the firm is doing well since the company has more subscribers because subscribers travel farther than normal customers.

Apart from the above data we can also see the count of registered(Subscriber) and unregistered(customer) users as shown below:

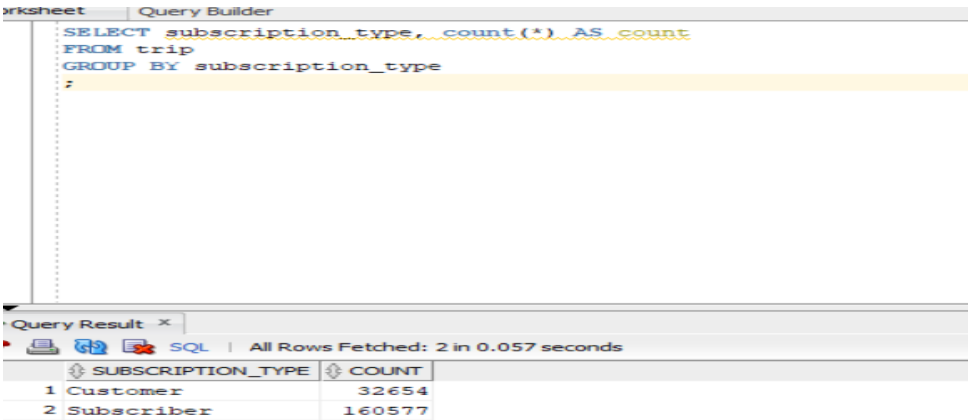
Firstly, let's plot a pie chart to determine the proportion



We can also check the count of of registered(Subscriber) and unregistered(Customer) users using below query:

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

SQL Query :



2.4.4

Quantifying how many stations are added each year in all cities:

SELECT city ,count(*),year FROM (SELECT city , to_char(installation_date , 'yyyy') AS year FROM station)
GROUP BY city , year;

```
select city ,count(*),year from (select city , to_char(installation_date , 'yyyy' ) as year from station)
group by city , year ;
```

Query Result

SQL | All Rows Fetched: 8 in 0.002 seconds

	CITY	COUNT(*)	YEAR
1	Mountain View	7	2013
2	Redwood City	1	2014
3	San Jose	15	2013
4	Palo Alto	5	2013
5	San Jose	1	2014
6	Redwood City	6	2013
7	San Francisco	34	2013
8	San Francisco	1	2014

5.4.1.1

select * from(
select count(*) num,
s0.name as stat_station,
s1.name as end_station
from route inner join station s0 on route.start_station_id = s0.id inner join station s1 on s1.id= route.end_station_id
group by s0.name,s1.name
order by num desc) where rownum <= 5;

```
select * from
(select station.name, count(trip.id) as num from trip inner join route on trip.id=route.id
inner join station on station.id = route.start_station_id
group by station.name
ORDER BY num DESC)
where rownum <= 5;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.122 seconds

	NAME	NUM
1	San Francisco Caltrain (Townsend at 4th)	14270
2	Harry Bridges Plaza (Ferry Building)	9097
3	San Francisco Caltrain 2 (330 Townsend)	8361
4	Temporary Transbay Terminal (Howard at Beale)	7645
5	Embarcadero at Sansome	7398

5.4.1.2

select * from
(select station.name, count(trip.id) as num from trip inner join route on trip.id=route.id inner join station on station.id = route.end_station_id
group by station.name ORDER BY num DESC) where rownum <= 5;

```
select * from
(select station.name, count(trip.id) as num from trip inner join route on trip.id=route.id
inner join station  on station.id = route.end_station_id
group by station.name
ORDER BY num DESC)
where rownum <= 5;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.121 seconds

	NAME	NUM
1	San Francisco Caltrain (Townsend at 4th)	17388
2	Harry Bridges Plaza (Ferry Building)	9004
3	Market at Sansome	8302
4	Embarcadero at Sansome	8223
5	San Francisco Caltrain 2 (330 Townsend)	8189

5.4.1.3

select * from
(select count(*) num,
s0.name as stat_station,
s1.name as end_station
from route inner join station s0 on route.start_station_id = s0.id inner join station s1 on s1.id= route.end_station_id group by s0.name,s1.name
order by num desc) where rownum <= 5;

```
select * from(
select
count(*) num,
s0.name as stat_station,
s1.name as end_station
from  route inner join station s0 on route.start_station_id = s0.id inner join station  s1 on s1.id= route.end_station_id
group by s0.name,s1.name
order by num desc) where rownum <= 5;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.085 seconds

	NUM	STAT_STATION	END_STATION
1	1698	Townsend at 7th	San Francisco Caltrain (Townsend at 4th)
2	1679	San Francisco Caltrain 2 (330 Townsend)	Townsend at 7th
3	1477	Harry Bridges Plaza (Ferry Building)	Embarcadero at Sansome
4	1152	Harry Bridges Plaza (Ferry Building)	2nd at Townsend
5	1117	2nd at Townsend	Harry Bridges Plaza (Ferry Building)

5.4.1.4

SQL Query to find the most popular route in each city :
SELECT DISTINCT * FROM (select MAX(NUM) OVER (PARTITION BY CITY),CITY from (select s1.city, count(*) num ,
s0.name as stat_station,
s1.name as end_station
from route inner join station s0 on route.start_station_id = s0.id inner join station s1 on s1.id= route.end_station_id
group by s1.city, s0.name,s1.name
order by num desc) ORDER BY NUM DESC);

	MAX(NUM)OVER(PARTITIONBYCITY)	CITY
1	1698	San Francisco
2	811	Mountain View
3	172	Redwood City
4	228	Palo Alto
5	710	San Jose