

Project Title - Northeastern Transportation System

Project Scope:

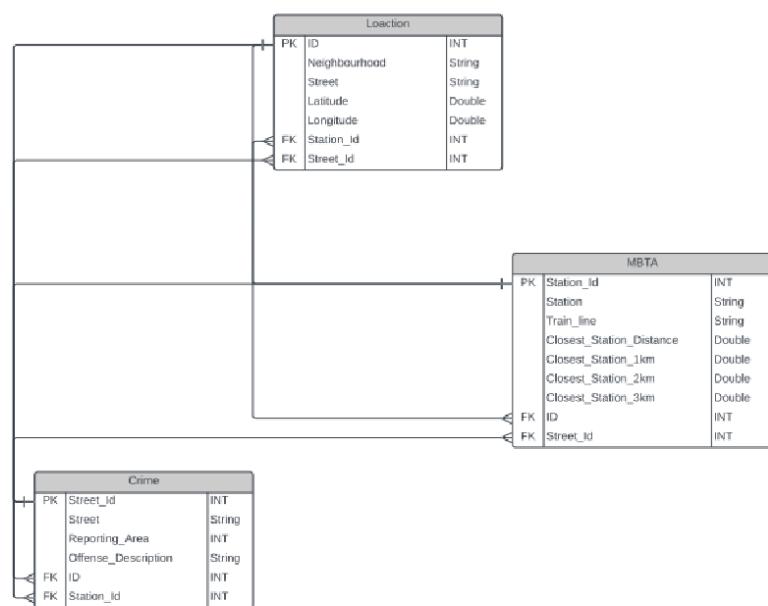
The project intends to serve the Northeastern community by assisting students, faculty, and other community members to reach their destinations promptly and safely regardless of the time using different modes of transport viz. cabs, buses, trams, driving.

Project Description:

We are developing a transportation database system, specifically for the Northeastern community, where the source will be Boston city for the traveller to go to his desired location which will comprise (Station Names, Street Names, Crime Rate and Offense Description). Once the location is determined, he will be able to see the mode of transportation, closest station of T-line (He will also get to know A particular station has how many lines and also get to know the type of lines eg-Red , Orange , Green). According to neighbourhood he will be also able to get to know the station within the range of 3miles in the database which will also include the street name and the percentage of crime rate on that street and also the type of offense that has been occurred. This will be very helpful for commuters who are uncertain of which streets are safe or not, to reach their destination so the safety parameter is distinct and the crime rate of a particular street. The user then can choose his or her route to the destination in accordance with it.

ER Diagram:

ER Diagram



Abbreviation:

PK-Primary Key

FK-Foreign Key

ER Diagram SOL Code:

```
CREATE TABLE `MBTA` (
    `Station_Id` INT,
    `Station` String,
    `Train_line` String,
    `Closest_Station_Distance` Double,
    `Closest_Station_1km` Double,
    `Closest_Station_2km` Double,
    `Closest_Station_3km` Double,
    `ID` INT,
    `Street_Id` INT,
    PRIMARY KEY (`Station_Id`),
    FOREIGN KEY (`Street_Id`) REFERENCES `Crime`(`Street_Id`),
    FOREIGN KEY (`ID`) REFERENCES `Loaction`(`ID`)
);
```

```
CREATE TABLE `Crime` (
    `Street_Id` INT,
    `Street` String,
    `Reporting_Area` INT,
    `Offense_Description` String,
    `ID` INT,
    `Station_Id` INT,
    PRIMARY KEY (`Street_Id`),
    FOREIGN KEY (`Station_Id`) REFERENCES `MBTA`(`Station_Id`),
    FOREIGN KEY (`ID`) REFERENCES `Loaction`(`ID`)
);
```

```
CREATE TABLE `Loaction` (
```

```

`ID` INT,
`Neighbourhood` String,
`Street` String,
`Latitude` Double,
`Longitude` Double,
`Station_Id` INT,
`Street_Id` INT,
PRIMARY KEY (`ID`),
FOREIGN KEY (`Street_Id`) REFERENCES `Crime`(`Street_Id`),
FOREIGN KEY (`Station_Id`) REFERENCES `MBTA`(`Station_Id`)
);

```

Sample Table:

Table 1: Location

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Table:** tab1
- Columns:**
 - STREET_ID
 - STREET
 - REPORTING_AREA
 - OFFENSE_DESCRIPTION
 - MyUnknownColumn
 - MyUnknownColumn_0
 - Station_ID
 - line_001
 - closest_station_distance
 - closest_station_0m
 - closest_station_20m
 - closest_station_30m
 - ID
- Data (Result Grid):**

ID	neighbourhood	STREET	latitude	longitude	Station_ID
1	Back Bay	HUNTINGTON AVE	42.35262417	-71.08051384	500
2	Jamaica Plain	MASSACHUSETTS AVE	42.31417399	-71.154147	501
3	Dorchester	MASSACHUSETTS AVE	42.30088321	-71.08189711	502
4	Back Bay	WASHINGTON ST	42.34526225	-71.080893	503
5	Jamaica Plain	HUNTINGTON AVE	42.30957363	-71.10629241	504
6	Jamaica Plain	HUNTINGTON AVE	42.303763	-71.11402683	505
7	Jamaica Plain	HUNTINGTON AVE	42.3045593	-71.11364252	506
8	Jamaica Plain	HUNTINGTON AVE	42.3169122	-71.11302542	507
9	Back Bay	WASHINGTON ST	42.3516337	-71.08746027	508
10	Jamaica Plain	HUNTINGTON AVE	42.29105052	-71.11593555	509
11	South End	TREMONT ST	42.34643651	-71.07480756	510
12	Beacon Hill	TREMONT ST	42.36075556	-71.05399474	511
13	Jamaica Plain	BEACON ST	42.31217991	-71.10473244	512
14	Fenway	WASHINGTON ST	42.34392163	-71.09613946	513
15	Roxbury	BLUE HILL AVE	42.3311251	-71.07493728	514
16	Beacon Hill	BOYLSTON ST	42.3610482	-71.05593422	515
17	Back Bay	DORCHESTER AVE	42.34853314	-71.0548558	516
18	Roxbury	BOYLSTON ST	42.32642042	-71.09013271	517

Table 2: Crime

The screenshot shows the MySQL Workbench interface with the 'Crime' database selected. The left sidebar displays the schema structure, including tables 'tab2' and 'tab3'. The main area shows the results of a query: 'SELECT * from tab2'. The results grid contains 16 rows of data, including columns: STREETID, STREET, REPORTING_AREA, OFFENSE_DESCRIPTION, and Station_ID. The data includes various offense types like Larceny, Harassment, Assault, and Trespassing across different streets and reporting areas.

STREETID	STREET	REPORTING_AREA	OFFENSE_DESCRIPTION	Station_ID
1	HUNTINGTON AVE	146	LARCENY SHOPLIFTING	500
2	MASSACHUSETTS AVE	185	VERBAL DISPUTE	501
3	MASSACHUSETTS AVE	185	LARCENY ALL OTHERS	502
4	WASHINGTON ST	450	HARASSMENT	503
5	HUNTINGTON AVE	600	ASSAULT - AGGRAVATED	504
6	HUNTINGTON AVE	600	ASSAULT SIMPLE - BATTERY	505
7	HUNTINGTON AVE	600	DISORDERLY CONDUCT	506
8	HUNTINGTON AVE	600	TRESPASSING	507
9	WASHINGTON ST	562	LARCENY THEFT FROM MV - NON-ACCESSORY	508
10	HUNTINGTON AVE	594	VERBAL DISPUTE	509
11	TREMONT ST	111	ASSAULT - AGGRAVATED - BATTERY	510
12	TREMONT ST	122	MV - LEAVING SCENE - PROPERTY DAMAGE	511
13	BEACON ST	512	INVESTIGATE PERSON	512
14	WASHINGTON ST	752	ASSAULT - SIMPLE	513
15	BLUE HILL AVE	425	INVESTIGATE PROPERTY	514
16	BOYLSTON ST	117	TRESPASSING	515

Table 3: MBTA

The screenshot shows the MySQL Workbench interface with the 'MBTA' database selected. The left sidebar displays the schema structure, including tables 'tab2' and 'tab3'. The main area shows the results of a query: 'SELECT * from tab3'. The results grid contains 11 rows of data, including columns: Station_ID, station, line_001, closest_station_distance, closest_station_3km, closest_station_2km, closest_station_1km, and ID. The data lists various MBTA stations and their proximity to each other.

Station_ID	station	line_001	closest_station_distance	closest_station_3km	closest_station_2km	closest_station_1km	ID
500	Copley Station	green	0.388574513	Copley Station	Copley Station	Copley Station	1
501	Green Street Station	orange	0.756237262	Green Street Station	Green Street Station	Green Street Station	2
502	Shawmut Station	red	1.57617878	Shawmut Station	Shawmut Station	Shawmut Station	3
503	Prudential Station	green	0.091817796	Prudential Station	Prudential Station	Prudential Station	4
504	Green Street Station	orange	0.15170446	Green Street Station	Green Street Station	Green Street Station	5
505	Forest Hills Station	orange	0.378969426	Forest Hills Station	Forest Hills Station	Forest Hills Station	6
506	Forest Hills Station	orange	0.471992524	Forest Hills Station	Forest Hills Station	Forest Hills Station	7
507	Stony Brook Station	orange	0.68090611	Stony Brook Station	Stony Brook Station	Stony Brook Station	8
508	Hynes Convention Center	green	0.412716007	Hynes Convention Center	Hynes Convention Center	Hynes Convention Center	9
509	Forest Hills Station	orange	1.065981674	Forest Hills Station	Forest Hills Station	Forest Hills Station	10
510	Rack Rav Station	orange	0.147209271	Rack Rav Station	Rack Rav Station	Rack Rav Station	11

Main Database:

SQL File 6* | Limit to 1000 rows |

```
1 select * from maindatabase
```

Result Grid | Filter Rows | | | | | | | |

ID	neighbourhood	STREET	latitude	longitude	Station_ID	station	line_001	closest_station_distance	closest_station_1km	closest_station_2km	close
1	Back Bay	HUNTINGTON AVE	42.35262417	-71.08051384	500	Copley Station	green	0.388674513	Copley Station	Copley Station	Copley
2	Jamaica Plain	MASSACHUSETTS AVE	42.31417399	-71.1151417	501	Green Street Station	orange	0.754237262	Green Street Station	Green Street Station	Greer
3	Dorchester	MASSACHUSETTS AVE	42.30088321	-71.08189711	502	Shawmut Station	red	1.576817678	Shawmut Station	Shawmut Station	Shaw
4	Back Bay	WASHINGTON ST	42.34526225	-71.0807893	503	Prudential Station	green	0.091817796	Prudential Station	Prudential Station	Prude
5	Jamaica Plain	HUNTINGTON AVE	42.30957368	-71.1062941	504	Green Street Station	orange	0.151704646	Green Street Station	Green Street Station	Greer
6	Jamaica Plain	HUNTINGTON AVE	42.3037631	-71.1140263	505	Forest Hills Station	orange	0.378969426	Forest Hills Station	Forest Hills Station	Fores
7	Jamaica Plain	HUNTINGTON AVE	42.3041559	-71.11364252	506	Forest Hills Station	orange	0.471992524	Forest Hills Station	Forest Hills Station	Forces
8	Jamaica Plain	HUNTINGTON AVE	42.31691221	-71.11302542	507	Stony Brook Station	orange	0.68890511	Stony Brook Station	Stony Brook Station	Ston
9	Back Bay	WASHINGTON ST	42.35163374	-71.08746027	508	Hynes Convention Center	green	0.412716007	Hynes Convention Center	Hynes Convention Center	Hyne
10	Jamaica Plain	HUNTINGTON AVE	42.29106052	-71.11935355	509	Forest Hills Station	orange	1.065981674	Forest Hills Station	Forest Hills Station	Fores
11	South End	TREMONT ST	42.34643651	-71.07480756	510	Back Bay Station	orange	0.1420921	Back Bay Station	Back Bay Station	Back
12	Beacon Hill	TREMONT ST	42.36075556	-71.06399474	511	Bowdoin Station	blue	0.172017855	Bowdoin Station	Bowd	Bowd
13	Jamaica Plain	BEACON ST	42.31217591	-71.10473244	512	Green Street Station	orange	0.323239118	Green Street Station	Green Street Station	Greer
14	Fenway	WASHINGTON ST	42.43352163	-71.09613946	513	Kenmore Station	green	0.605215481	Kenmore Station	Kenmore Station	Kenn
15	Roxbury	BLUE HILL AVE	42.33311251	-71.07493728	514	Symphony Station	green	1.35097293	Symphony Station	Symp	Symp
16	Beacon Hill	BOYLSTON ST	42.3610482	-71.05593422	515	Bowdoin Station	blue	0.316510952	Bowdoin Station	Bowd	Bowd
17	Back Bay	DORCHESTER AVE	42.34853314	-71.08548558	516	Hynes Convention Center	green	0.2117009	Hynes Convention Center	Hynes Convention Center	Hyne
18	Roxbury	BOYLSTON ST	42.32642042	-71.09032371	517	Roxbury Crossing Station	orange	0.716243614	Roxbury Crossing Station	Roxbury Crossing Station	Roxb
19	North End	CENTRE ST	42.36462415	-71.10370724	518	Haymarket Station	orange	0.432998822	Haymarket Station	Haymarket Station	Haym
20	Mission Hill	WASHINGTON ST	42.33393473	-71.10313501	519	Brigham Circle Station	green	0.12895494	Brigham Circle Station	Brigham Circle Station	Brigh
21	Kenmore	WILBURTON CT	42.3301024	-71.084262	520	Kenmore Station	red	0.42007419	Kenmore Station	Kenmore Station	Kenmo

closest_station_3km	REPORTING_AREA	OFFENSE_DESCRIPTION
Copley Station	146	LARCENY SHOPLIFTING
Green Street Station	185	VERBAL DISPUTE
Shawmut Station	185	LARCENY ALL OTHERS
Prudential Station	450	HARASSMENT
Green Street Station	600	ASSAULT - AGGRAVATED
Forest Hills Station	600	ASSAULT SIMPLE - BATTERY
Forest Hills Station	600	DISORDERLY CONDUCT
Stony Brook Station	600	TRESPASSING
Hynes Convention Center	562	LARCENY THEFT FROM MV - NON-ACCESSORY
Forest Hills Station	594	VERBAL DISPUTE
Back Bay Station	111	ASSAULT - AGGRAVATED - BATTERY
Bowdoin Station	122	M/V - LEAVING SCENE - PROPERTY DAMAGE
Green Street Station	512	INVESTIGATE PERSON
Kenmore Station	752	ASSAULT - SIMPLE
Symphony Station	425	INVESTIGATE PROPERTY
Bowdoin Station	117	TRESPASSING
Hynes Convention Center	348	ASSAULT - AGGRAVATED - BATTERY
Roxbury Crossing Station	138	INVESTIGATE PERSON
Haymarket Station	291	ASSAULT SIMPLE - BATTERY
Brigham Circle Station	778	PROPERTY - FOUND
Kenmore Station	261	ASSAULT SIMPLE - BATTERY

Output of Views:

Q.1 To display street name where Reporting crime count more than the average reporting count.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is 'neut'.
- Tables:** Contains 'maindatabase', 'tab1', 'tab2', 'tab3'.
- Views:** Contains 'namegiven', 'q', 'question_1', 'question_2', 'question_3', 'question_4', 'question_5', 'question_6'.
- SQL Editor:** The query is:create view question_6 as
select STREET , REPORTING_AREA from neut.maindatabase where REPORTING_AREA > (select avg(REPORTING_AREA) from neut.maindatabase)
select * from question_6
- Result Grid:** Shows the results of the query:| STREET | REPORTING_AREA |
| --- | --- |
| WASHINGTON ST | 450 |
| HUNTINGTON AVE | 600 |
| WASHINGTON ST | 562 |
| HUNTINGTON AVE | 594 |
| RAPORTING | 512 |
- Action Output:** Displays the execution log:| # | Time | Action | Message | Duration / Hatch |
| --- | --- | --- | --- | --- |
| 1 | 19:34:49 | use neut | 0 row(s) affected | 0.000 sec |
| 2 | 19:34:56 | create view question_6 as select distinct neighbourhood , STREET , REPORTING_AREA from neut.maindatabase where REPORTING_AREA > (select avg(REPORTING_AREA) from neut.maindatabase) | 0 row(s) affected | 0.015 sec |
| 3 | 19:35:14 | select * from question_6 LIMIT 0, 1000 | 6 rows(s) returned | 0.000 sec / 0.000 sec |
| 4 | 19:37:33 | create view question_5 as select tab1.neighbourhood , tab2.REPORTING_AREA , tab2.STREET from neut.maindatabase as tab1 inner join neut.maindatabase as tab2 on tab1.neighbourhood = tab2.neighbourhood | 0 row(s) affected | 0.000 sec |
| 5 | 19:37:46 | select * from question_5 LIMIT 0, 1000 | 22 rows(s) returned | 0.015 sec / 0.000 sec |
| 6 | 19:40:02 | create view question_6 as select STREET , REPORTING_AREA from neut.maindatabase where REPORTING_AREA > (select avg(REPORTING_AREA) from neut.maindatabase) | 0 row(s) affected | 0.000 sec |
| 7 | 19:41:13 | select * from question_6 LIMIT 0, 1000 | 113 row(s) returned | 0.000 sec / 0.000 sec |

Q.2 To find the neighbourhood name and street name who has reported crime less than 100

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is 'neut'.
- Tables:** Contains 'maindatabase', 'tab1', 'tab2', 'tab3'.
- Views:** Contains 'namegiven', 'q', 'question_1', 'question_2', 'question_3', 'question_4', 'question_5', 'question_6', 'question_10'.
- SQL Editor:** The query is:create view question_10 as
select distinct neighbourhood,STREET from tab1 where exists
(select REPORTING_AREA ,STREET from tab2 where tab1.STREET = tab2.STREET AND REPORTING_AREA < 100)
select * from question_10
- Result Grid:** Shows the results of the query:| neighbourhood | STREET |
| --- | --- |
| Dorchester | CAMBRIDGE ST |
| Dorchester | BOWDISHN ST |
| Mission Hill | CAMBRIDGE ST |
| Fenway | FANBECK HALL MARKETPLACE |
| Hyde Park | CAMBRIDGE ST |
| Beacon Hill | BENNINGTON ST |
| Hyde Park | BENNINGTON ST |
- Action Output:** Displays the execution log:| # | Time | Action | Message | Duration / Hatch |
| --- | --- | --- | --- | --- |
| 1 | 19:46:27 | use neut | 0 row(s) affected | 0.016 sec |
| 2 | 19:49:39 | create view question_8 as select tab2.STREET ,tab3.station.tab2.OFFENSE_TYPE ,tab3.station.tab2.REPORTING_AREA ,tab3.station.tab2.neighbourhood ,tab3.station.tab2.STREET from neut.maindatabase as tab1 inner join neut.maindatabase as tab2 on tab1.neighbourhood = tab2.neighbourhood inner join neut.maindatabase as tab3 on tab2.REPORTING_AREA = tab3.REPORTING_AREA and tab2.STREET = tab3.STREET and tab2.OFFENSE_TYPE = tab3.OFFENSE_TYPE | 0 row(s) affected | 0.016 sec |
| 3 | 19:49:55 | select * from question_8 LIMIT 0, 1000 | 8 row(s) returned | 0.000 sec / 0.000 sec |
| 4 | 19:52:32 | create view question_9 as select tab1.neighbourhood ,tab1.STREET ,min(tab2.REPORTING_AREA) as MIN_REPORTING_AREA from neut.maindatabase as tab1 inner join neut.maindatabase as tab2 on tab1.neighbourhood = tab2.neighbourhood group by tab1.neighbourhood ,tab1.STREET | 0 row(s) affected | 0.000 sec |
| 5 | 19:52:49 | select * from question_9 LIMIT 0, 1000 | 1 row(s) returned | 0.016 sec / 0.000 sec |
| 6 | 19:56:50 | create view question_10 as select distinct neighbourhood,STREET from tab1 where exists (select REPORTING_AREA ,STREET from tab2 where tab1.STREET = tab2.STREET AND REPORTING_AREA < 100) select * from question_10 | 0 row(s) affected | 0.000 sec |
| 7 | 19:57:11 | select * from question_10 LIMIT 0, 1000 | 10 row(s) returned | 0.000 sec / 0.000 sec |

Q.3 Check for the closest station from 1,2,3km using case expression

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `neut` with tables `maindatabase`, `tab1`, `tab2`, `tab3` and views `namegiven`, `q`, `question_1` through `question_6`.
- SQL Editor:** Contains the following SQL code:

```

1 create view question_12 as
2 select distinct Station , closest_station_1km , closest_station_2km , closest_station_3km
3 from neut.maindatabase
4 order by
5      (case
6      when closest_station_1km is null then closest_station_2km
7      when closest_station_2km is null then closest_station_3km
8      else closest_station_1km
9      END);
10 select * from question_12;
    
```
- Result Grid:** Displays the results of the query, showing columns `Station`, `closest_station_1km`, `closest_station_2km`, and `closest_station_3km` for various stations like Forest Hills Station, Boston College Station, etc.
- Action Output:** Shows the history of actions taken in the session, including the creation of the view and its execution.

Q.4 To find the neighbourhood's name and street name who has reported crime less than 300

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `neut` with tables `maindatabase`, `tab1`, `tab2`, `tab3` and views `namegiven`, `q`, `question_1` through `question_6`.
- SQL Editor:** Contains the following SQL code:

```

1 create view question_20 as
2 select distinct neighbourhood , STREET from tab1 where exists (select REPORTING_AREA , STREET FROM tab2
3 where tab1.STREET = tab2.STREET AND REPORTING_AREA < 300);
4 select * from question_20;
    
```
- Result Grid:** Displays the results of the query, showing columns `neighbourhood` and `STREET` for locations like Back Bay, Jamaica Plain, etc.
- Action Output:** Shows the history of actions taken in the session, including the creation of the view and its execution.

Q5. Which are the stations near to particular neighbourhood ?

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the **neut** database selected.
- SQL Editor:** Contains the following SQL code:


```
1
2  create view question_1 as
3  select distinct tab1.neighbourhood , tab3.closest_station_1km from tab1 left join tab3 on tab1.station_id = tab3.station_id;
4 •  select * from question_1
```
- Result Grid:** Displays the results of the query, showing the neighbourhood and the closest station within 1km. The data includes:

neighbourhood	closest_station_1km
Back Bay	Copley Station
Jamaica Plain	Green Street Station
Dorchester	
Back Bay	Prudential Station
Jamaica Plain	Forest Hills Station
Jamaica Plain	Stony Brook Station
Back Bay	Hynes Convention Center
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
1	19:19:44	use neut;	0 row(s) affected	0.000 sec
2	19:20:45	create view question_1 as select distinct tab1.neighbourhood , tab3.closest_st... 0 row(s) affected	0.000 sec	
3	19:21:34	select * from question_1 LIMIT 0, 1000 81 row(s) returned	0.016 sec / 0.000 sec	

Q6. To find the nearest station , distance , neighbourhood from the street name MASSACHUSETTS AVE

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the **neut** database selected.
- SQL Editor:** Contains the following SQL code:


```
1 •  create view question_2 as
2  select neighbourhood , STREET , min(closest_station_distance) , closest_station_1km from neut.maindatabase where STREET = "MASSACHUSETTS AVE"
3 •  select * from question_2
```
- Result Grid:** Displays the results of the query, showing the neighbourhood, street name, minimum distance, and closest station. The data is:

neighbourhood	STREET	min(closest_station_distance)	closest_station_1km
Jamaica Plain	MASSACHUSETTS AVE	0.106319184	Green Street Station
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
1	19:19:44	use neut;	0 row(s) affected	0.000 sec
2	19:20:45	create view question_1 as select distinct tab1.neighbourhood , tab3.closest_st... 0 row(s) affected	0.000 sec	
3	19:21:34	select * from question_1 LIMIT 0, 1000 81 row(s) returned	0.016 sec / 0.000 sec	
4	19:26:12	create view question_2 as select neighbourhood , STREET , min(closest_stati... Error Code: 1054. Unknown column 'closest_station_1km' in field list'	0.000 sec	
5	19:26:24	create view question_2 as select neighbourhood , STREET , min(closest_stati... 0 row(s) affected	0.000 sec	
6	19:26:50	select * from question_2 LIMIT 0, 1000 1 row(s) returned	0.000 sec / 0.000 sec	

Q7. To count the number of street in each neighbourhood

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "neut".
- SQL Editor:** Contains the following SQL code:


```
1 •  create view question_3 as
2 •  select distinct neighbourhood , count(STREET) from neut.maindatabase group by neighbourhood order by STREET;
3 •  select * from question_3
```
- Result Grid:** Displays the results of the query:

neighbourhood	count(STREET)
Roxbury	12
Hyde Park	4
North End	8
Roslindale	7
South Boston Waterfront	2
Charlestown	4
West End	3
Back Bay	21
- Action Output:** Shows the history of actions:

#	Time	Action	Message	Duration / Fetch
1	19:30:32	use neut	0 row(s) affected	0.000 sec
2	19:30:42	create view question_3 as select distinct neighbourhood , count(STREET) from... Error Code: 1050. Table 'question_3' already exists		0.000 sec
3	19:30:55	create view question_3 as select distinct neighbourhood , count(STREET) fro... 0 row(s) affected		0.000 sec
4	19:31:11	select * from question_3 LIMIT 0, 1000	22 row(s) returned	0.015 sec / 0.000 sec

Q8. To find the street, neighbourhood name where reporting crime counts are more than 800

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "neut".
- SQL Editor:** Contains the following SQL code:


```
1 •  create view question_4 as
2 •  select distinct neighbourhood , STREET , REPORTING_AREA from neut.maindatabase where REPORTING_AREA
3 •  group by STREET , REPORTING_AREA ORDER BY STREET;
4 •  select * from question_4
```
- Result Grid:** Displays the results of the query:

neighbourhood	STREET	REPORTING_AREA
South End	BRIGHTON AVE	802
South End	DORCHESTER AVE	817
Back Bay	FIDELIS WAY	938
Allston	HEATH ST	906
Jamaica Plain	RIVER ST	951
Brighton	WASHINGTON ST	933
- Action Output:** Shows the history of actions:

#	Time	Action	Message	Duration / Fetch
1	19:34:49	use neut	0 row(s) affected	0.000 sec
2	19:34:56	create view question_4 as select distinct neighbourhood , STREET , REPORT... 0 row(s) affected		0.015 sec
3	19:35:14	select * from question_4 LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Q9. What is the count of reporting crime of a particular neighbourhood?

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (connector, db, finalproject, info, neut). Under the neut schema, there are Tables (maindatabase, tab1, tab2, tab3) and Views (namegivin, q, question_1, question_2, question_3, question_5, question_3).
- SQL Editor:** SQL File 7* contains the following code:


```
1 • create view question_5 as
2 select tab1.neighbourhood , tab2.REPORTING_AREA from tab1 inner join tab2 on tab1.STREET = tab2.STREET group by neighbourhood;
3 • select * from question_5;
```
- Result Grid:** Displays the results of the query in SQL File 5*. The columns are neighbourhood and REPORTING_AREA. The data is:

neighbourhood	REPORTING_AREA
Back Bay	283
Jamaica Plain	172
Dorchester	172
South End	102
Beacon Hill	102
Fenway	282
Roxbury	444
North End	646
- Action Output:** Shows the history of actions taken during the session, including the creation of the view and its execution.
- System Status:** Shows the system status (1°C, Mostly clear), taskbar icons, and system tray information (ENG IN, 19:38, 13-12-2022).

Q10. To select the station and neighbourhood name which has more than one train connection

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (connector, db, finalproject, info, neut). Under the neut schema, there are Tables (maindatabase, tab1, tab2, tab3) and Views (namegivin, q, question_1, question_2, question_3, question_5, question_3).
- SQL Editor:** SQL File 7* contains the following code:


```
1 • create view question_7 as
2 select neighbourhood , station,line__001 from maindatabase where station = "Park Street Station";
3 • select * from question_7;
```
- Result Grid:** Displays the results of the query in SQL File 5*. The columns are neighbourhood, station, and line_001. The data is:

neighbourhood	station	line_001
Beacon Hill	Park Street Station	red
Downtown	Park Street Station	orange
Beacon Hill	Park Street Station	Park Street Station
Downtown	Park Street Station	red
- Action Output:** Shows the history of actions taken during the session, including the creation of the view and its execution.
- System Status:** Shows the system status (1°C, Mostly clear), taskbar icons, and system tray information (ENG IN, 19:44, 13-12-2022).

Q11. To select the street name and station who has committed the offence of Verbal Dispute

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 7* SQL File 4* SQL File 4* SQL File 5*
SCHEMAS
Filter objects
connector
db
finalproject
info
neut
Tables
maindatabase
tab1
tab2
tab3
Views
namegiven
q
question_1
question_2
question_3
question_4
question_5
question_6
Administration Schemas
Information
No object selected
Object Info Session
1°C Mostly clear
19:50 ENG IN 13-12-2022

```

```

1 create view question_8 as
2 select tab2.STREET , tab3.station,tab2.OFFENSE_DESCRIPTION
3 from tab2 inner join tab3 on tab2.Station_ID = tab3.Station_ID
4 where tab2.OFFENSE_DESCRIPTION = "VERBAL DISPUTE";
5 * select * from question_8

```

STREET	station	OFFENSE_DESCRIPTION
MASSACHUSETTS AVE	Green Street Station	VERBAL DISPUTE
HUNTINGTON AVE	Forest Hills Station	VERBAL DISPUTE
MASSACHUSETTS AVE	Charles/MGH Station	VERBAL DISPUTE
HARVARD ST	Stony Brook Station	VERBAL DISPUTE
RIVER ST	Green Street Station	VERBAL DISPUTE
RUE DE LA VIEILLE	Hanover St Station	VERBAL DISPUTE

Action Output

#	Time	Action	Message	Duration / Fetch
1	19:46:27	use neut	0 row(s) affected	0.016 sec
2	19:49:39	create view question_8 as select tab2.STREET , tab3.station,tab2.OFFENSE_DESCRIPTION from tab2 inner join tab3 on tab2.Station_ID = tab3.Station_ID where tab2.OFFENSE_DESCRIPTION = "VERBAL DISPUTE";	0 row(s) affected	0.016 sec
3	19:49:55	select * from question_8 LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

Q12. To select the neighbourhood, street and reported crime area which has the most minimum reported crime

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 7* SQL File 4* SQL File 5*
SCHEMAS
Filter objects
connector
db
finalproject
info
neut
Tables
maindatabase
tab1
tab2
tab3
Views
namegiven
q
question_1
question_2
question_3
question_4
question_5
question_6
Administration Schemas
Information
No object selected
Object Info Session
1°C Mostly clear
19:52 ENG IN 13-12-2022

```

```

1 create view question_9 as
2 select tab1.neighbourhood , tab1.STREET, min(tab2.REPORTING_AREA) as Minimum_Reported_Crime
3 from tab1 left join tab2 on tab1.STREET = tab2.STREET;
4 * select * from question_9

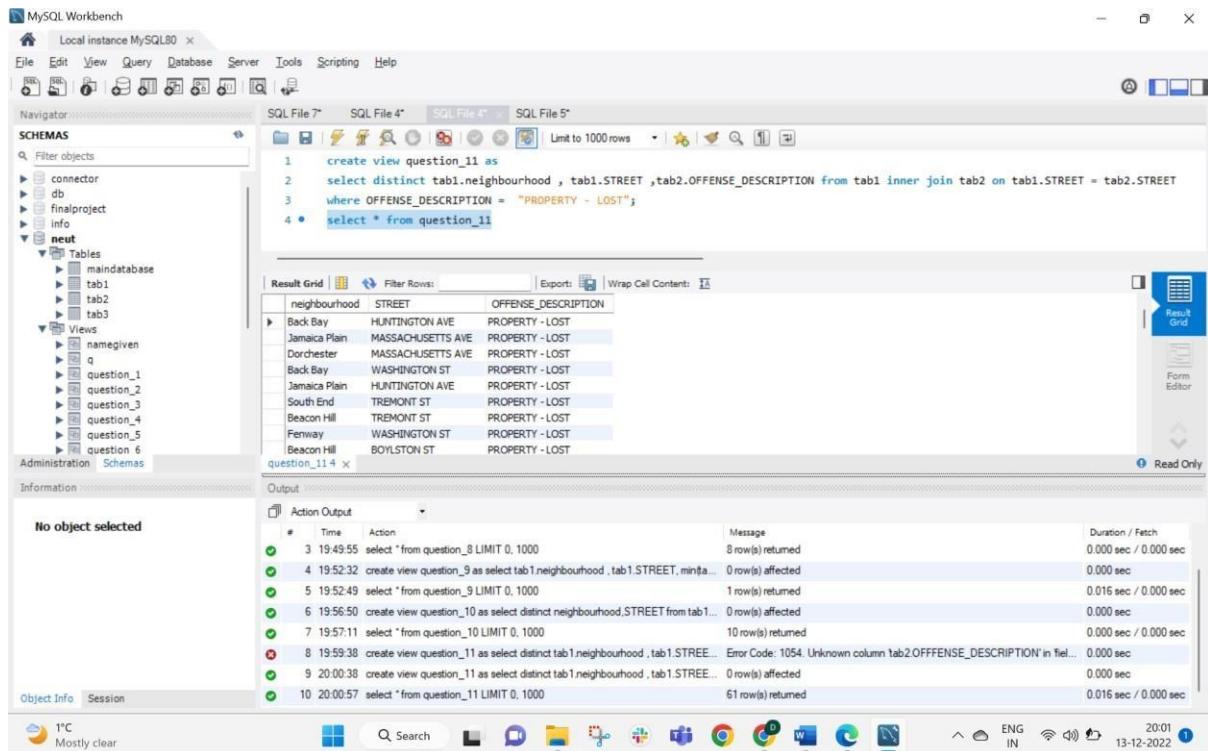
```

neighbourhood	STREET	Minimum_Reported_Crime
Back Bay	HUNTINGTON AVE	23

Action Output

#	Time	Action	Message	Duration / Fetch
1	19:46:27	use neut	0 row(s) affected	0.016 sec
2	19:49:39	create view question_9 as select tab1.neighbourhood , tab1.STREET, min(tab2.REPORTING_AREA) as Minimum_Reported_Crime from tab1 left join tab2 on tab1.STREET = tab2.STREET;	0 row(s) affected	0.016 sec
3	19:49:55	select * from question_9 LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
4	19:52:32	create view question_9 as select tab1.neighbourhood , tab1.STREET, min(tab2.REPORTING_AREA) as Minimum_Reported_Crime from tab1 left join tab2 on tab1.STREET = tab2.STREET;	0 row(s) affected	0.000 sec
5	19:52:49	select * from question_9 LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

Q13. To find the Neighbourhood name and street name and who has reported the offense of property lost



The screenshot shows the MySQL Workbench interface with the following details:

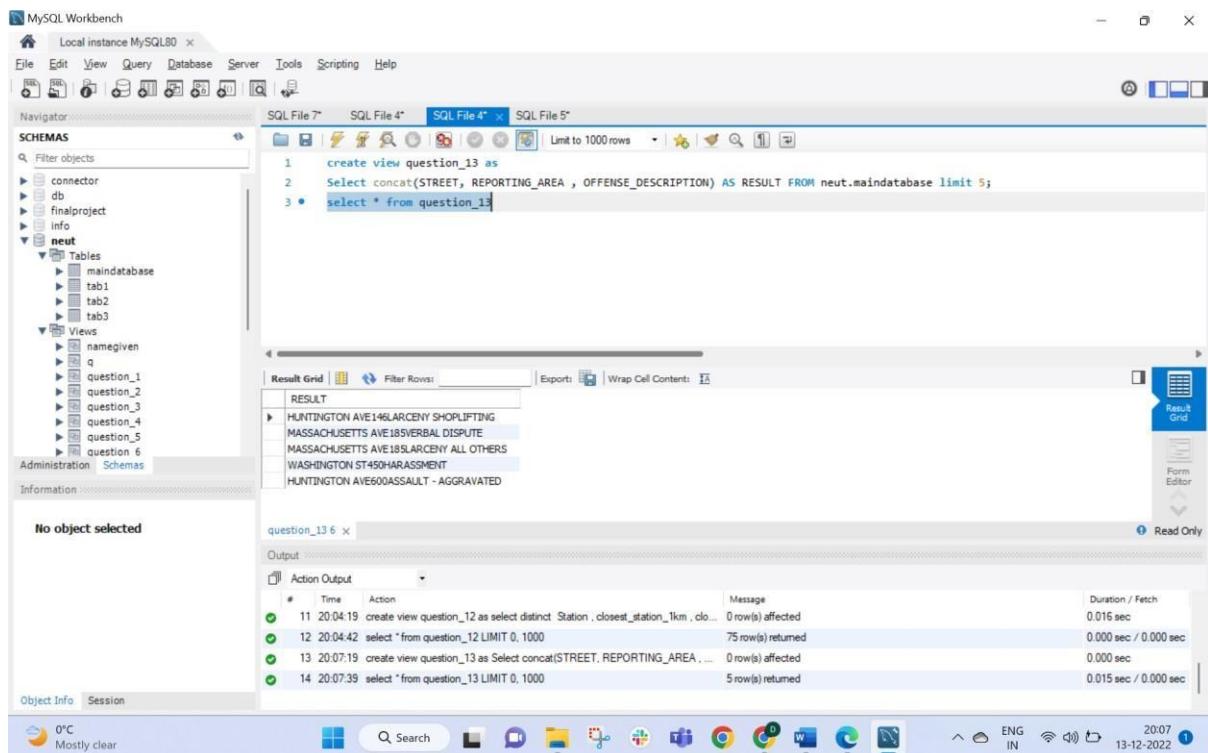
- Navigator:** Shows the database schema with the **neut** database selected.
- SQL Editor:** Contains the following SQL code:


```

1  create view question_11 as
2  select distinct tab1.neighbourhood , tab1.STREET ,tab2.OFFENSE_DESCRIPTION from tab1 inner join tab2 on tab1.STREET = tab2.STREET
3  where OFFENSE_DESCRIPTION = "PROPERTY - LOST"
4  *  select * from question_11
      
```
- Result Grid:** Displays the results of the query, showing neighborhood, street, and offense description. The data includes:

neighbourhood	STREET	OFFENSE_DESCRIPTION
Back Bay	HUNTINGTON AVE	PROPERTY - LOST
Jamaica Plain	MASSACHUSETTS AVE	PROPERTY - LOST
Dorchester	MASSACHUSETTS AVE	PROPERTY - LOST
Back Bay	WASHINGTON ST	PROPERTY - LOST
Jamaica Plain	HUNTINGTON AVE	PROPERTY - LOST
South End	TREMONT ST	PROPERTY - LOST
Beacon Hill	TREMONT ST	PROPERTY - LOST
Fenway	WASHINGTON ST	PROPERTY - LOST
Beacon Hill	BOYLSTON ST	PROPERTY - LOST
- Action Output:** Shows the execution history of the query, including the time, action, message, and duration.

Q14. To find the recent 5 offense, reported crime of the streets



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the **neut** database selected.
- SQL Editor:** Contains the following SQL code:


```

1  create view question_13 as
2  Select concat(STREET, REPORTING_AREA , OFFENSE_DESCRIPTION) AS RESULT FROM neut.maindatabase limit 5;
3  *  select * from question_13
      
```
- Result Grid:** Displays the results of the query, showing concatenated street, reporting area, and offense description. The data includes:

RESULT
HUNTINGTON AVE146LARCENY SHOPLIFTING
MASSACHUSETTS AVE185VERBAL DISPUTE
MASSACHUSETTS AVE185LARCENY ALL OTHERS
WASHINGTON ST450HARASSMENT
HUNTINGTON AVE600ASSAULT - AGGRAVATED
- Action Output:** Shows the execution history of the query, including the time, action, message, and duration.

Q15. To find the station, the train line of the street using “USING Clause”

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure under the 'neut' database, including tables like maindatabase, tab1, tab2, tab3, and views like namewin, q, question_1, question_2, question_3, question_4, question_5, and question_6.
- SQL Editor:** Contains the following SQL code:


```
1 create view question_14 as
2 SELECT DISTINCT e.STREET , d.station , d.line__001 from tab1 e JOIN tab3 d
3 using(Station_ID);
4 * select * from question_14
```
- Result Grid:** Displays the results of the query, showing columns STREET, station, and line__001. The data includes:

STREET	station	line__001
HUNTINGTON AVE	Copley Station	green
MASSACHUSETTS AVE	Green Street Station	orange
MASSACHUSETTS AVE	Shawmut Station	red
WASHINGTON ST	Prudential Station	green
HUNTINGTON AVE	Green Street Station	orange
HUNTINGTON AVE	Forest Hills Station	orange
HUNTINGTON AVE	Stony Brook Station	orange
WASHINGTON ST	Hynes Convention Center	green
TREMONT ST	Back Bay Station	orange
TREMONT ST	Bowdoin Station	blue
- Action Output:** Shows the execution log with entries for each step of the query execution.

Q16 To find the closest station, distance from a particular neighbourhood and street.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure under the 'neut' database, including tables like maindatabase, tab1, tab2, tab3, and views like namewin, q, question_1, question_2, question_3, question_4, question_5, and question_6.
- SQL Editor:** Contains the following SQL code:


```
1 create view question_15 as
2 select distinct tab1.neighbourhood,tab1.STREET , tab3.closest_station_distance, tab3.closest_station_1km
3 from tab1 inner join tab3 on tab1.ID = tab3.ID;
4 * select * from question_15;
```
- Result Grid:** Displays the results of the query, showing columns neighbourhood, STREET, closest_station_distance, and closest_station_1km. The data includes:

neighbourhood	STREET	closest_station_distance	closest_station_1km
Back Bay	HUNTINGTON AVE	0.388674513	Copley Station
Jamaica Plain	MASSACHUSETTS AVE	0.756237262	Green Street Station
Dorchester	MASSACHUSETTS AVE	1.576817878	
Back Bay	WASHINGTON ST	0.091817796	Prudential Station
Jamaica Plain	HUNTINGTON AVE	0.151704646	Green Street Station
Jamaica Plain	HUNTINGTON AVE	0.378969426	Forest Hills Station
Jamaica Plain	HUNTINGTON AVE	0.471992524	Forest Hills Station
Jamaica Plain	HUNTINGTON AVE	0.680906111	Stony Brook Station
	WASHINGTON ST	0.417716007	Hynes Convention Center
- Action Output:** Shows the execution log with entries for each step of the query execution.

Q17 . We have created a view and given the name (namegiven)

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'neut' schema, there is a 'Views' folder containing a view named 'namegiven'. The SQL Editor tab contains the following code:

```

1 create view question_16 as
2 SELECT neighbourhood , street , closest_station_distance, REPORTING_AREA,OFFENSE_DESCRIPTION,station from neut.maindatabase;
3 • select * from question_16;

```

The Result Grid shows the results of the query, listing various neighborhoods, streets, distances, reporting areas, offense descriptions, and stations. The Output pane shows the history of actions taken:

#	Time	Action	Message	Duration / Fetch
17	20:14:00	create view question_15 as select distinct tab1.neighbourhood,tab1.STREET...	0 row(s) affected	0.015 sec
18	20:14:24	select * from question_15 LIMIT 0, 1000	246 row(s) returned	0.016 sec / 0.000 sec
19	20:18:11	create view question_16 as SELECT neighbourhood , street , closest_statio...	Error Code: 1054. Unknown column 'neighbourhood' in field list'	0.000 sec
20	20:18:19	create view question_16 as SELECT neighbourhood , street , closest_statio...	0 row(s) affected	0.000 sec
21	20:18:48	select * from question_16 LIMIT 0, 1000	245 row(s) returned	0.000 sec / 0.000 sec

Q18. To find street name, neighbourhood and offense description on the basis on reporting crime

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'neut' schema, there is a 'Views' folder containing a view named 'question_17'. The SQL Editor tab contains the following code:

```

1 create view question_17 as
2 select distinct tab1.neighbourhood , tab1.STREET , tab2.REPORTING_AREA, tab2.OFFENSE_DESCRIPTION
3 from tab1 inner join tab2 on tab1.STREET = tab2.STREET
4 where REPORTING_AREA > 700
5 group by STREET , REPORTING_AREA ORDER BY STREET;
6 • select * from question_17;

```

The Result Grid shows the results of the query, listing neighborhoods, streets, reporting areas, and offense descriptions. The Output pane shows the history of actions taken:

#	Time	Action	Message	Duration / Fetch
18	20:14:24	select * from question_15 LIMIT 0, 1000	246 row(s) returned	0.016 sec / 0.000 sec
19	20:18:11	create view question_16 as SELECT neighbourhood , street , closest_statio...	Error Code: 1054. Unknown column 'neighbourhood' in field list'	0.000 sec
20	20:18:19	create view question_16 as SELECT neighbourhood , street , closest_statio...	0 row(s) affected	0.000 sec
21	20:18:48	select * from question_16 LIMIT 0, 1000	245 row(s) returned	0.000 sec / 0.000 sec
22	20:24:27	create view question_17 as select distinct tab1.neighbourhood , tab1.STREE...	0 row(s) affected	0.015 sec
23	20:25:03	select * from question_17 LIMIT 0, 1000	23 row(s) returned	0.016 sec / 0.000 sec

Q19. To find the station which is farest from the street along with the distance using “PARTITION BY” clause

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 7* SQL File 4* SQL File 4* SQL File 5*
Schemas Filter objects
connector db finalproject info neut Tables maindatabase tab1 tab2 tab3
Views namegiven q question_1 question_2 question_3 question_4 question_5 question_6
Administration Schemas
Information No object selected
Object Info Session
0°C Clear
Result Grid Filter Rows: Export: Wrap Cell Content: Result Grid
STREET STATION distance
HUNTINGTON AVE Copley Station 0.388674513
MASSACHUSETTS AVE Green Street Station 0.756237262
MASSACHUSETTS AVE Shawmut Station 1.576817878
WASHINGTON ST Prudential Station 0.091817796
HUNTINGTON AVE Green Street Station 0.151704646
HUNTINGTON AVE Forest Hills Station 0.378969426
HUNTINGTON AVE Forest Hills Station 0.471992524
question_18 11 ×
Output Action Output
# Time Action Message Duration / Fetch
18 20:14:24 select * from question_15 LIMIT 0, 1000 246 row(s) returned 0.016 sec / 0.000 sec
19 20:18:11 create view question_16 as SELECT neighbourhood , street , closest_station... Error Code: 1054. Unknown column 'neighbourhood' in field list' 0.000 sec
20 20:18:19 create view question_16 as SELECT neighbourhood , street , closest_station... 0 row(s) affected 0.000 sec
21 20:18:48 select * from question_16 LIMIT 0, 1000 245 row(s) returned 0.000 sec / 0.000 sec
22 20:24:27 create view question_17 as select distinct tab1.neighbourhood , tab1.STREE... 0 row(s) affected 0.015 sec
23 20:25:03 select * from question_17 LIMIT 0, 1000 23 row(s) returned 0.016 sec / 0.000 sec
24 20:28:11 create view question_18 as SELECT STREET, STATION , MAX(closest_st... 0 row(s) affected 0.016 sec
25 20:28:37 select * from question_18 LIMIT 0, 1000 245 row(s) returned 0.000 sec / 0.000 sec
20:28 13-12-2022

```

Q20. To find the Neighbourhood name and street name and who has reported the offense of TRESPASSING

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 7* SQL File 4* SQL File 4* SQL File 5*
Schemas Filter objects
connector db finalproject info neut Tables maindatabase tab1 tab2 tab3
Views namegiven q question_1 question_2 question_3 question_4 question_5 question_6
Administration Schemas
Information Table: tab2
Table: tab2
Columns:
STREETID int
STREET text
REPORTING_AREA int
OFFENSE_DESCRIPTION text
Station_ID int
Object Info Session
0°C Clear
Result Grid Filter Rows: Export: Wrap Cell Content: Result Grid
STREET station OFFENSE_DESCRIPTION
HUNTINGTON AVE Stony Brook Station TRESPASSING
BOYLSTON ST Bowdoin Station TRESPASSING
GERARD ST Andrew Station TRESPASSING
question_19 12 ×
Output Action Output
# Time Action Message Duration / Fetch
23 20:25:03 select * from question_17 LIMIT 0, 1000 23 row(s) returned 0.016 sec / 0.000 sec
24 20:28:11 create view question_18 as SELECT STREET, STATION , MAX(closest_st... 0 row(s) affected 0.016 sec
25 20:28:37 select * from question_18 LIMIT 0, 1000 245 row(s) returned 0.000 sec / 0.000 sec
26 20:32:22 create procedure top_STREETOP() begin select STREET , line____001 from n... 0 row(s) affected 0.016 sec
27 20:32:42 select * from question_19 LIMIT 0, 1000 Error Code: 1146. Table 'neut question_19' doesn't exist. 0.000 sec
28 20:33:47 select * from question_19 LIMIT 0, 1000 Error Code: 1146. Table 'neut question_19' doesn't exist. 0.000 sec
29 20:41:30 create view question_19 as select tab2.STREETID , tab2.station , tab2.OFFENSE_DESCRIPTION select tab2.STREETID , tab2.station , tab2.OFFENSE_DESCRIPTION FROM tab2 inner join tab3 on tab2.Station_ID = tab3.Station_ID where tab2.OFFENSE_DESCRIPTION = 'TRESPASSING' 0.016 sec
30 20:41:47 select * from question_19 LIMIT 0, 1000 0.000 sec
31 20:42:02 select * from question_19 LIMIT 0, 1000 0.000 sec / 0.000 sec
20:42 13-12-2022

```

Use Cases:

1. Created a store procedure for the crime reports more than 900

The screenshot shows a database interface with a left sidebar for 'SCHEMAS' containing 'maindatabase' and 'tab1'. Under 'tab1', there are 'Columns' (ID, neighbourhood, STREET, latitude, longitude, Station_ID) and 'Triggers'. Below 'tab1' is 'tab2' with columns (STREETID, STREET, REPORTING_AREA, OFFENSE_DESCRIPTION, MyUnknownColumn). A central pane displays a code editor with a stored procedure definition:

```
1 delimiter &
2 create procedure top_STREETW()
3 begin
4 select STREET, REPORTING_AREA
5 from neut.maindatabase where REPORTING_AREA >900;
6 End &
7 delimiter ;
8 call top_STREETW();
```

Below the code editor is a 'Result Grid' showing the output of the query:

STREET	REPORTING_AREA
RIVER ST	951
WASHINGTON ST	933
HEATH ST	906
FIDELIS WAY	938

2. Which are the stations near to particular neighbourhood ?

The screenshot shows a database interface with a left sidebar for 'SCHEMAS' containing 'maindatabase' and 'tab1'. Under 'tab1', there are 'Columns' (ID, neighbourhood, STREET, latitude, longitude, Station_ID) and 'Triggers'. Below 'tab1' is 'tab2' with columns (STREETID, STREET, REPORTING_AREA, OFFENSE_DESCRIPTION, MyUnknownColumn). A central pane displays a query in the code editor:

```
1 select distinct tab1.neighbourhood ,tab3.closest_station_1km from tab1 left join tab3 on tab1.station_id = tab3.station_id;
```

Below the code editor is a 'Result Grid' showing the output of the query:

neighbourhood	closest_station_1km
Back Bay	Copley Station
Jamaica Plain	Green Street Station
Dorchester	
Back Bay	Prudential Station
Jamaica Plain	Forest Hills Station
Jamaica Plain	Stony Brook Station
Back Bay	Hynes Convention Center
Jamaica Plain	
South End	Back Bay Station
Beacon Hill	Bowdoin Station
Fenway	Kenmore Station
Roxbury	
Roxbury	Roxbury Crossing Station
North End	Haymarket Station
Murphy Hill	Quincy Center Station

3. To find the nearest station , distance , neighbourhood from the street name MASSACHUSETTS AVE

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with Schemas and tables (tab1 and tab2). The right pane shows the results of a query:

```

SQL File 6* SQL File 3* SQL File 4* SQL File 5* SQL File 6*
1   SELECT neighbourhood ,STREET , min(closest_station_distance) , closest_station_1km
2   from neut.maindatabase where STREET = "MASSACHUSETTS AVE";

```

Result Grid

neighbourhood	STREET	min(closest_station_distance)	closest_station_1km
Jamaica Plain	MASSACHUSETTS AVE	0.106319184	Green Street Station

4. To count the number of street in each neighbourhood

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with Schemas and tables (tab1 and tab2). The right pane shows the results of a query:

```

SQL File 6* SQL File 3* SQL File 4* SQL File 5* SQL File 6*
1 • select distinct neighbourhood ,count(STREET) from neut.maindatabase group by neighbourhood order by STREET DESC;

```

Result Grid

neighbourhood	count(STREET)
Fenway	16
Mission Hill	6
Downtown	19
South Boston	12
South End	23
Beacon Hill	24
East Boston	6
West Roxbury	2
Jamaica Plain	30
Dorchester	20
Brighton	12
Allston	9
Mattapan	2
Back Bay	21
Others	?

5. To find the street, neighbourhood name where reporting crime counts are more than 800

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with Schemas and tables (tab1 and tab2). The right pane shows the results of a query:

```

SQL File 6* SQL File 3* SQL File 4* SQL File 5* SQL File 6*
1   SELECT DISTINCT neighbourhood ,STREET, REPORTING_AREA FROM neut.maindatabase WHERE REPORTING_AREA > 800
2
3   GROUP BY STREET,REPORTING_AREA ORDER BY STREET

```

Result Grid

neighbourhood	STREET	REPORTING_AREA
South End	BRIGHTON AVE	802
South End	DORCHESTER AVE	817
Back Bay	FIDELIS WAY	938
Allston	HEATH ST	906
Jamaica Plain	RIVER ST	951
Brighton	WASHINGTON ST	933

6. What is the count of reporting crime of a particular neighbourhood ?

The screenshot shows the Oracle SQL Developer interface. The left pane displays the Navigator with Schemas and tables tab1 and tab2 selected. The right pane shows the SQL Editor with the following query:

```
1 • select tab1.neighbourhood , tab2.REPORTING_AREA from tab1 inner join tab2 on tab1.STREET = tab2.STREET group by neighbourhood;
```

The Result Grid shows the output:

neighbourhood	REPORTING_AREA
Back Bay	283
Jamaica Plain	172
Dorchester	172
South End	102
Beacon Hill	102
Fenway	282
Roxbury	444
North End	646
Mission Hill	282
Charlestown	404
Roslindale	646
Downtown	282
South Boston	282
Chinatown	283
Braintree	177

7. To display street name where Reporting crime count more than the avg reporting count

The screenshot shows the Oracle SQL Developer interface. The left pane displays the Navigator with Schemas and tables tab1 and tab2 selected. The right pane shows the SQL Editor with the following query:

```
1 • select STREET, REPORTING_AREA
2   from neut.maindatabase where REPORTING_AREA > (select avg( REPORTING_AREA ) from neut.maindatabase)
```

The Result Grid shows the output:

STREET	REPORTING_AREA
WASHINGTON ST	450
HUNTINGTON AVE	600
WASHINGTON ST	562
HUNTINGTON AVE	594
BEACON ST	512
WASHINGTON ST	752
BLUE HILL AVE	425
WASHINGTON ST	778
TALBOT AVE	441
WASHINGTON ST	777
BOYLSTON ST	627
MANUFACTURING ST	416

8. To select the station and neighbourhood name which has more than one train connection

The screenshot shows the Oracle SQL Developer interface. The left pane displays the Navigator with Schemas and tables tab1 and tab2 selected. The right pane shows the SQL Editor with the following query:

```
1 • select neighbourhood,station,line_001 from maindatabase where station="Park Street Station"
```

The Result Grid shows the output:

neighbourhood	station	line_001
Beacon Hill	Park Street Station	red
Downtown	Park Street Station	orange
Beacon Hill	Park Street Station	green
Downtown	Park Street Station	red

9. To select the street name and station who has committed the offence of Verbal Dispute

```

SQL File 5* SQL File 3* SQL File 4* SQL File 5* SQL File 6*
1 • select tab2.STREET,tab3.station,tab2.OFFENSE_DESCRIPTION
2   from tab2 inner join tab3 on tab2.Station_ID=tab3.Station_ID
3   where tab2.OFFENSE_DESCRIPTION= "VERBAL DISPUTE"

```

STREET	station	OFFENSE_DESCRIPTION
MASSACHUSETTS AVE	Green Street Station	VERBAL DISPUTE
HUNTINGTON AVE	Forest Hills Station	VERBAL DISPUTE
MASSACHUSETTS AVE	Charles/MGH Station	VERBAL DISPUTE
HARVARD ST	Stony Brook Station	VERBAL DISPUTE
RIVER ST	Green Street Station	VERBAL DISPUTE
BLUE HILL AVE	Haymarket Station	VERBAL DISPUTE
WASHINGTON ST	Community College Station	VERBAL DISPUTE
WARREN ST	Copley Station	VERBAL DISPUTE

10. To select the neighbourhood, street and reported crime area which has the most minimum reported crime

```

SQL File 6* SQL File 3* SQL File 4* SQL File 5* SQL File 6*
1 • select tab1.neighbourhood, tab1.STREET,min(tab2.REPORTING_AREA) as Minimum_Reported_Crime
2   from tab1 left join tab2 on tab1.STREET=tab2.STREET
3

```

neighbourhood	STREET	Minimum_Reported_Crime
Back Bay	HUNTINGTON AVE	23

11. To find the neighbourhood name and street name who has reported crime less than 100

```

SQL File 6* SQL File 3* SQL File 4* SQL File 5* SQL File 6*
1 • select DISTINCT neighbourhood,STREET From tab1 where exists (select REPORTING_AREA,STREET
2   from tab2 where tab1.STREET = tab2.STREET AND REPORTING_AREA < 100)
3

```

neighbourhood	STREET
Dorchester	CAMBRIDGE ST
Dorchester	BOWDOIN ST
Mission Hill	CAMBRIDGE ST
Fenway	FAANEUIL HALL MARKETPLACE
Hyde Park	CAMBRIDGE ST
Beacon Hill	BENNINGTON ST
Hyde Park	BENNINGTON ST
South Boston	BOWDOIN ST

12. To find the Neighbourhood name and street name and who has reported the offense of property lost

```

SELECT distinct tab1.neighbourhood, tab1.STREET,tab2.OFFENSE_DESCRIPTION FROM
tab1 inner join tab2 on tab1.STREET=TAB2.STREET
WHERE OFFENSE_DESCRIPTION = "PROPERTY - LOST"

```

neighbourhood	STREET	OFFENSE_DESCRIPTION
Back Bay	HUNTINGTON AVE	PROPERTY - LOST
Jamaica Plain	MASSACHUSETTS AVE	PROPERTY - LOST
Dorchester	MASSACHUSETTS AVE	PROPERTY - LOST
Back Bay	WASHINGTON ST	PROPERTY - LOST
Jamaica Plain	HUNTINGTON AVE	PROPERTY - LOST
South End	TREMONT ST	PROPERTY - LOST
Beacon Hill	TREMONT ST	PROPERTY - LOST
Fenway	WASHINGTON ST	PROPERTY - LOST

13. Check for the closest station from 1,2,3km using case expression

```

SELECT distinct Station , closest_station_1km,closetest_station_2km,closetest_station_3km
FROM neut.maindatabase
ORDER BY
(CASE
WHEN closest_station_1km IS NULL THEN closest_station_2km
WHEN closest_station_2km IS NULL THEN closest_station_3km
ELSE closest_station_1km
END);

```

Station	closest_station_1km	closest_station_2km	closest_station_3km
Forest Hills Station			
Boston College Station	Boston College Station	Boston College Station	
Shawmut Station	Shawmut Station	Shawmut Station	
South Street Station	South Street Station	South Street Station	
Harvard Avenue Station	Harvard Avenue Station	Harvard Avenue Station	
Broadway Station	Broadway Station	Broadway Station	
Forest Hills Station	Forest Hills Station	Forest Hills Station	
Andrew Station	Andrew Station	Andrew Station	
Packards Corner Station	Packards Corner Station	Packards Corner Station	
Andrew Station	Andrew Station	Andrew Station	
Symphony Station	Symphony Station	Symphony Station	

14. To find the recent 5 offense, reported crime of the streets

```

SELECT CONCAT(STREET,REPORTING_AREA,OFFENSE_DESCRIPTION) AS
Result
FROM neut.maindatabase limit 5

```

Result
HUNTINGTON AVE146LARCENY SHOPLIFTING
MASSACHUSETTS AVE18VERBAL DISPUTE
MASSACHUSETTS AVE18LARCENY ALL OTHERS
WASHINGTON ST450HARASSMENT
HUNTINGTON AVE600ASSAULT - AGGRAVATED

15. To find the station, the train line of the street using “USING Clause”

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the schema structure, including tables tab1 and tab3. The table tab3 has columns: STREET, station, line_001, closest_station_distance, closest_station_1km, closest_station_2km, and closest_station_3km. The query window contains the following T-SQL code:

```

1 •  SELECT Distinct e.STREET,d.station, d.line_001
2   FROM tab1 e JOIN tab3 d
3     USING(Station_ID);

```

The results grid shows the output of the query:

STREET	station	line_001
HUNTINGTON AVE	Copley Station	green
MASSACHUSETTS AVE	Green Street Station	orange
MASSACHUSETTS AVE	Shawmut Station	red
WASHINGTON ST	Prudential Station	green
HUNTINGTON AVE	Green Street Station	orange
HUNTINGTON AVE	Forest Hills Station	orange
HUNTINGTON AVE	Stony Brook Station	orange
WASHINGTON ST	Hynes Convention Center	green
TREMONT ST	Back Bay Station	orange
TREMONT ST	Bowdoin Station	blue
BEACON ST	Green Street Station	orange
WASHINGTON ST	Kenmore Station	green
BLUE HILL AVE	Symphony Station	green
BOYLSTON ST	Bowdoin Station	blue
DORCHESTER AVE	Hynes Convention Center	green

16. To find the closest station, distance from a particular neighbourhood and street.

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the schema structure, including tables tab1 and tab3. The table tab3 has columns: STREET_ID, station, line_001, closest_station_distance, closest_station_1km, closest_station_2km, closest_station_3km, and ID. The query window contains the following T-SQL code:

```

1 •  select DISTINCT tab1.neighbourhood,tab1.STREET, tab3.closest_station_distance,tab3.closest_station_1km
2   from tab1 inner join tab3 on tab1.ID=tab3.ID

```

The results grid shows the output of the query:

neighbourhood	STREET	closest_station_distance	closest_station_1km
Back Bay	HUNTINGTON AVE	0.388674513	Copley Station
Jamaica Plain	MASSACHUSETTS AVE	0.756237262	Green Street Station
Dorchester	MASSACHUSETTS AVE	1.576817878	
Back Bay	WASHINGTON ST	0.091817796	Prudential Station
Jamaica Plain	HUNTINGTON AVE	0.151704646	Green Street Station
Jamaica Plain	HUNTINGTON AVE	0.378969426	Forest Hills Station
Jamaica Plain	HUNTINGTON AVE	0.471992524	Forest Hills Station
Jamaica Plain	HUNTINGTON AVE	0.680906111	Stony Brook Station
Back Bay	WASHINGTON ST	0.412716007	Hynes Convention Center
Jamaica Plain	HUNTINGTON AVE	1.065981674	
South End	TREMONT ST	0.14220921	Back Bay Station
Beacon Hill	TREMONT ST	0.171207855	Bowdoin Station
Jamaica Plain	BEACON ST	0.323239118	Green Street Station
Fenway	WASHINGTON ST	0.605215481	Kenmore Station
DORCHESTER AVE			

17.To create a stored procedure to show the street which are connected green line

The screenshot shows the SQL Server Management Studio interface. On the left, the Navigator pane is open with sections for MANAGEMENT, INSTANCE, and PERFORMANCE. The central area contains a query window titled 'Query 1' with the following T-SQL code:

```
1 delimiter &&
2 • create procedure top_STREETK()
3 begin
4 select STREET, Line_001
5 from neut.database where line_001='green';
6 End &&
7 delimiter ;
8 • call top_STREETK();
```

Below the code, the 'Result Grid' shows the output of the stored procedure:

STREET	Line_001
NORMANDY ST	green
LAWN ST	green
LESLIE ST	green
OCEAN VIEW DR	green
DALESSIO CT	green
MARLBOROUGH ST	green
WOODROW AVE	green
MULVEY ST	green
STONEHURST ST	green
HUNTINGTON AVE	green
SHIRLEY ST	green
WOLCOTT	green
WASHINGTON ST	green
ANNAPOLIS ST	green
BLUE HILL AVE	green
CLARKWOOD ST	-----

18.We have created a view and given the name (namegiven)

The screenshot shows the SQL Server Management Studio interface. On the left, the Navigator pane is open with sections for SCHEMAS, TABLES, and other database objects. The central area contains a query window titled 'SQL File 6*' with the following T-SQL code:

```
1 • create view namegiven as select neighbourhood, street , closest_station_distance,REPORTING_AREA,
2 OFFENSE_DESCRIPTION,station
3 from neut.maindatabase
4
5
```

At the bottom of the screen, the status bar displays the command history and execution details:

178 23:17:53 create view namegiven as select neighbourhood, street , closest_station_dist... 0 row(s) affected 0.016 sec

Now we have called the view which is to find the neighbourhood , street , closest station , reporting area offense description and station

neighbourhood	street	closest_station_distance	REPORTING_AREA	OFFENSE_DESCRIPTION	station
Back Bay	HUNTINGTON AVE	0.388674513	146	LARCENY SHORPLIFTING	Copley Station
Jamaica Plain	MASSACHUSETTS AVE	0.756237262	185	VERBAL DISPUTE	Green Street Station
Dorchester	MASSACHUSETTS AVE	1.576817878	185	LARCENY ALL OTHERS	Shawmut Station
Back Bay	WASHINGTON ST	0.09181796	450	HARASSMENT	Prudential Station
Jamaica Plain	HUNTINGTON AVE	0.151704646	600	ASSAULT - AGGRAVATED	Green Street Station
Jamaica Plain	HUNTINGTON AVE	0.378969426	600	ASSAULT SIMPLE - BATTERY	Forest Hills Station
Jamaica Plain	HUNTINGTON AVE	0.471992524	600	DISORDERLY CONDUCT	Forest Hills Station
Jamaica Plain	HUNTINGTON AVE	0.68090611	600	TRESPASSING	Stony Brook Station
Back Bay	WASHINGTON ST	0.412716007	562	LARCENY THEFT FROM MV - NON-ACCESSORY	Hynes Convention Center
Jamaica Plain	HUNTINGTON AVE	1.065981674	594	VERBAL DISPUTE	Forest Hills Station
South End	TREMONT ST	0.14220921	111	ASSAULT - AGGRAVATED - BATTERY	Back Bay Station
Beacon Hill	TREMONT ST	0.171207855	122	M/V - LEAVING SCENE - PROPERTY DAMAGE	Bowdoin Station

19. To find street name, neighbourhood and offense description on the basis on reporting crime

neighbourhood	STREET	REPORTING_AREA	OFFENSE_DESCRIPTION
Jamaica Plain	BEACON ST	749	INVESTIGATE PROPERTY
Allston	BRIGHTON AVE	796	LARCENY ALL OTHERS
Allston	BRIGHTON AVE	800	LARCENY THEFT FROM BUILDING
Allston	BRIGHTON AVE	802	VAL - VIOLATION OF AUTO LAW - OTHER
Dorchester	CAMBRIDGE ST	795	VAL - VIOLATION OF AUTO LAW - OTHER
Downtown	COMMONWEALTH AVE	786	INVESTIGATE PERSON
Downtown	COMMONWEALTH AVE	788	TOWED MOTOR VEHICLE
Downtown	COMMONWEALTH AVE	793	TOWED MOTOR VEHICLE
Downtown	COMMONWEALTH AVE	794	LANDLORD - TENANT SERVICE
Downtown	COMMONWEALTH AVE	796	LARCENY ALL OTHERS
Back Bay	DORCHESTER AVE	817	TOWED MOTOR VEHICLE
West End	FANEUIL ST	759	LARCENY THEFT FROM MV - NON-ACCES...
Back Bay	FIDELIS WAY	938	SICK/INJURED/MEDICAL - PERSON
Allston	HEATH ST	906	VAL - VIOLATION OF AUTO LAW - OTHER
Allston	OVER ST	951	SICK/INJURED/MEDICAL - PERSON

20.To find the station which is farest from the street along with the distance using “PARTITION BY” clause

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the schema structure, including tables like 'tab3' and columns such as 'STREET', 'station', and 'distance'. The central pane shows a T-SQL query:

```

1 •  SELECT STREET, station,
2      MAX(closest_station_distance) OVER (PARTITION BY ID) AS distance
3  FROM neut.maindatabase;
4

```

The results grid on the right displays the data:

STREET	station	distance
HUNTINGTON AVE	Copley Station	0.388674513
MASSACHUSETTS AVE	Green Street Station	0.756237262
MASSACHUSETTS AVE	Shawmut Station	1.576817878
WASHINGTON ST	Prudential Station	0.091817796
HUNTINGTON AVE	Green Street Station	0.151704646
HUNTINGTON AVE	Forest Hills Station	0.378969426
HUNTINGTON AVE	Forest Hills Station	0.471992524
HUNTINGTON AVE	Stony Brook Station	0.68090611
WASHINGTON ST	Hynes Convention Center	0.412716007
HUNTINGTON AVE	Forest Hills Station	1.065981674
TREMONT ST	Back Bay Station	0.14220921
TREMONT ST	Bowdoin Station	0.171207855
BEACON ST	Green Street Station	0.323239118
WASHINGTON ST	Kenmore Station	0.605215481

Steps from choosing dataset to getting the final DB:

Step 1: Searched for datasets related to the topic and finalized 2 datasets and web scraped using Browse AI.

Links for the Datasets are:

- https://github.com/kanguyn/Interactive-Data-Visualization-MBTA-airbnb/blob/master/data/mbta_metadata.csv
- <https://www.kaggle.com/datasets/AnalyzeBoston/crimes-in-boston>

Step 2: The datasets were cleaned as it includes null values and duplicate values and they needed to clean for further steps. These steps were done on Google collab.

Data Cleaning:

1. Importing the CSV file

The screenshot shows a Google Colab interface with a notebook titled "database.ipynb". The code cell contains the following Python code:

```
[ ] from google.colab import drive  
drive.mount('/content/drive')  
  
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).  
  
[ ] import pandas as pd  
import matplotlib.pyplot as plt  
  
[ ] path="/content/drive/MyDrive/database2.csv"  
df=pd.read_csv(path)  
  
[ ] df.describe
```

Below the code cell, the output shows the descriptive statistics for the DataFrame:

	<bound method NDFrame.describe of OFFENSE_DESCRIPTION
AFFRAY	1
ANIMAL CONTROL - DOG BITES - ETC.	4
ANIMAL INCIDENTS	1
ANNOYING AND ACCOSTING	1

File Edit View Insert Runtime Tools Help Last edited on December 9

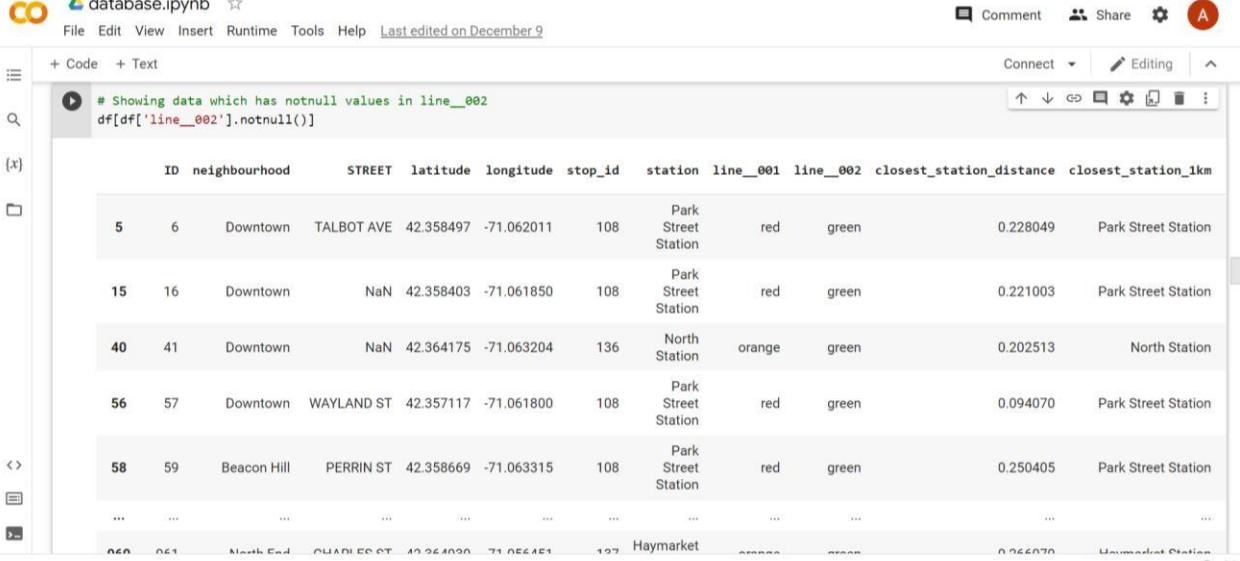
+ Code + Text

df.isna()

ID	neighbourhood	STREET	latitude	longitude	stop_id	station	line_001	line_002	closest_station_distance	closest_station_1km	closest_
0	False	False	False	False	False	False	False	True	False	False	False
1	False	False	False	False	False	False	False	True	False	False	False
2	False	False	False	False	False	False	False	True	False	False	False
3	False	False	False	False	False	False	False	True	False	True	True
4	False	False	False	False	False	False	False	True	False	False	False
...
1026	False	False	False	False	False	False	False	True	False	False	False
1027	False	False	False	False	False	False	False	True	False	False	False
1028	False	False	False	False	False	False	False	True	False	False	False
1029	False	False	False	False	False	False	False	True	False	False	False
1030	False	False	False	False	False	False	False	True	False	False	False

1031 rows x 16 columns

2. Finding the data which has no null value in the column line_002



```
# Showing data which has notnull values in line_002
df[df['line_002'].notnull()]
```

ID	neighbourhood	STREET	latitude	longitude	stop_id	station	line_001	line_002	closest_station_distance	closest_station_1km		
5	6	Downtown	TALBOT AVE	42.358497	-71.062011	108	Park Street Station	red	green	0.228049	Park Street Station	
15	16	Downtown		NaN	42.358403	-71.061850	108	Park Street Station	red	green	0.221003	Park Street Station
40	41	Downtown		NaN	42.364175	-71.063204	136	North Station	orange	green	0.202513	North Station
56	57	Downtown	WAYLAND ST	42.357117	-71.061800	108	Park Street Station	red	green	0.094070	Park Street Station	
58	59	Beacon Hill	PERRIN ST	42.358669	-71.063315	108	Park Street Station	red	green	0.250405	Park Street Station	
...	
1026	1027	North End	QUADIER ST	42.364020	-71.064151	107	Haymarket	orange	green	0.244070	Haymarket Station	

3. Finding the duplicated values in the dataset



```
#Finding duplicated values
df.duplicated()
```

```
0    False
1    False
2    False
3    False
4    False
...
1026  False
1027  False
1028  False
1029  False
1030  False
Length: 1031, dtype: bool
```

```
#Finding sum of duplicated values
df.duplicated().sum()
```

```
0
```

```
[ ] #Finding sum of duplicated values of STREET column
df['STREET'].duplicated().sum()
```

```
531
```

```

File Edit View Insert Runtime Tools Help Last edited on December 9
Comment Share A
+ Code + Text
[ ] #Finding duplicated values
df.duplicated()

[ ] #Finding sum of duplicated values
df.duplicated().sum()

[ ] #Finding sum of duplicated values of STREET column
df['STREET'].duplicated().sum()

```

4. Finding the duplicated values of column STREET and removing the duplicated value

```

File Edit View Insert Runtime Tools Help Last edited on December 9
Comment Share A
+ Code + Text
[ ] df.STREET.duplicated().sum()
531
[ ] df.loc[df.duplicated(keep='last'), :]

ID neighbourhood STREET latitude longitude stop_id station line_001 closest_station_distance closest_station_1km closest_station_2km close
[ ] df.duplicated(subset=['STREET']).head(1000)

0 False
1 False
2 False
3 False
4 False
...
995 True
996 True
997 True
998 True
999 True
Length: 1000, dtype: bool

```

```

File Edit View Insert Runtime Tools Help Last edited on December 9
Comment Share A
+ Code + Text
[ ] df.duplicated(subset=['STREET']).head(1000)

0 False
1 False
2 False
3 False
4 False
...
995 True
996 True
997 True
998 True
999 True
Length: 1000, dtype: bool
[ ] #Drop null values row for column Street
df=df.dropna(subset=["STREET"])

[ ] #Converting the cleaned data into CSV file
df.to_csv('/content/drive/MyDrive/final4database.csv', index = False, header=True)

```

Step 3: The cleaned dataset was then further converted into a CSV file named “maindatabase”.

Step 4: According to the dataset and creation of tables, ER diagram was prepared

Step 5: Further the cleaned data CSV file was imported into SQL Workbench

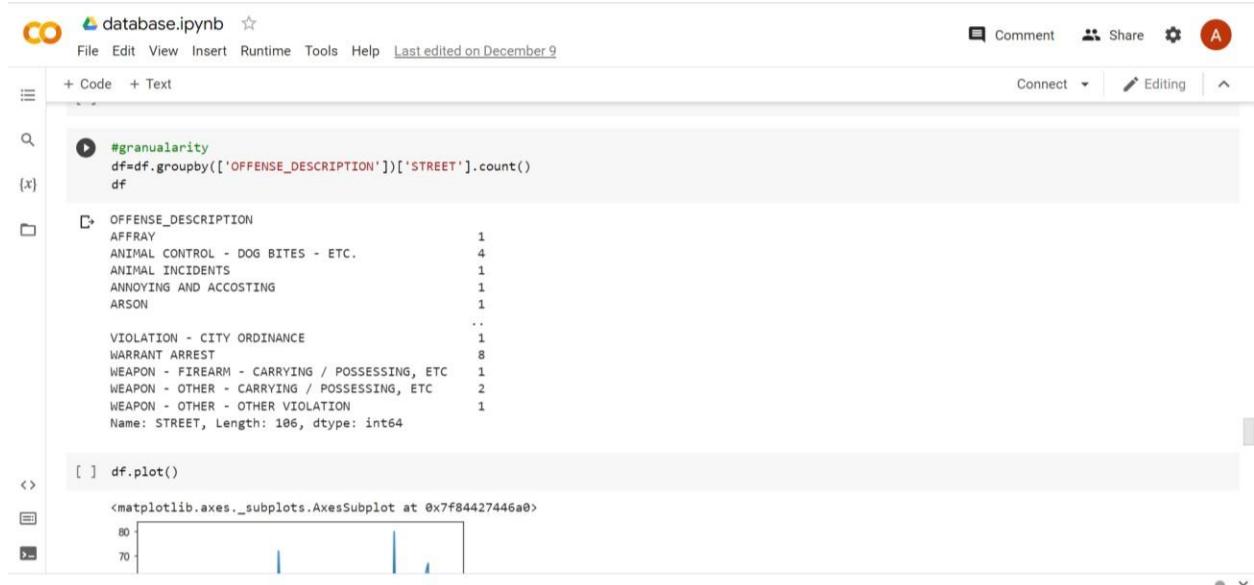
Step 6: The 20 sub-queries were implemented on SQL Workbench

Step 7: After completing the queries, data visualization was implemented on the final dataset on Google Collab

Step 8: Granularity, Bar graphs, and Histograms have been produced on the basis of the final dataset

Data Visualization:

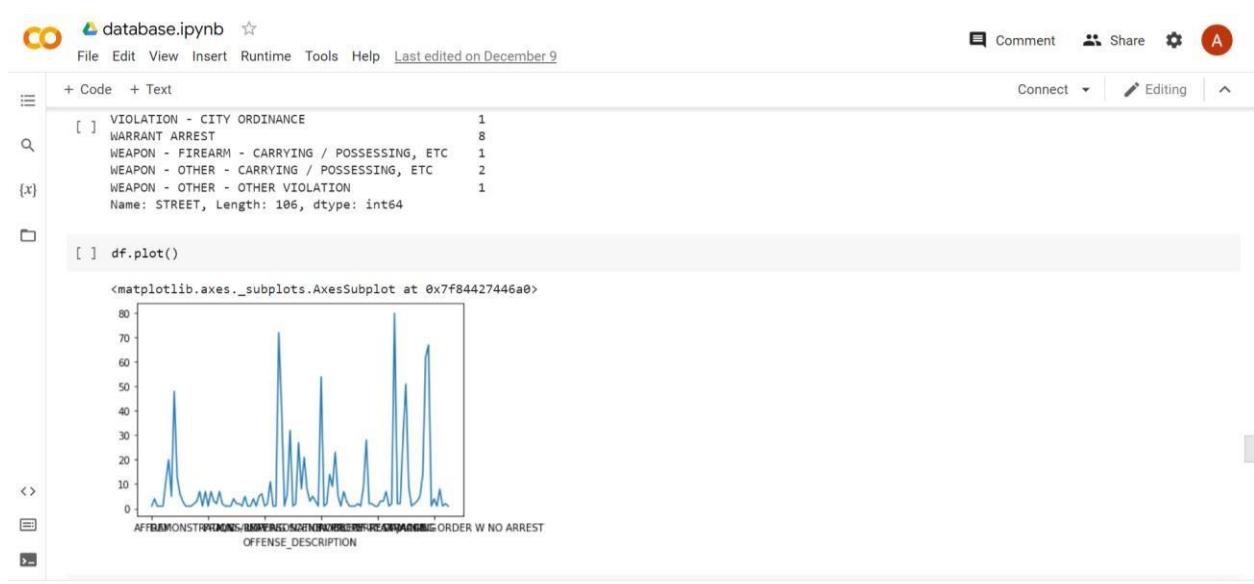
1. Granularity



```
#granularity
df=df.groupby(['OFFENSE_DESCRIPTION'])['STREET'].count()
df
```

OFFENSE_DESCRIPTION	Count
AFFRAY	1
ANIMAL CONTROL - DOG BITES - ETC.	4
ANIMAL INCIDENTS	1
ANNOYING AND ACCOSTING	1
ARSON	1
..	..
VIOLATION - CITY ORDINANCE	1
WARRANT ARREST	8
WEAPON - FIREARM - CARRYING / POSSESSING, ETC	1
WEAPON - OTHER - CARRYING / POSSESSING, ETC	2
WEAPON - OTHER - OTHER VIOLATION	1
Name: STREET, Length: 106, dtype: int64	

```
[ ] df.plot()
```



2. Histogram

database.ipynb

File Edit View Insert Runtime Tools Help Last edited on December 9

Comment Share A

+ Code + Text

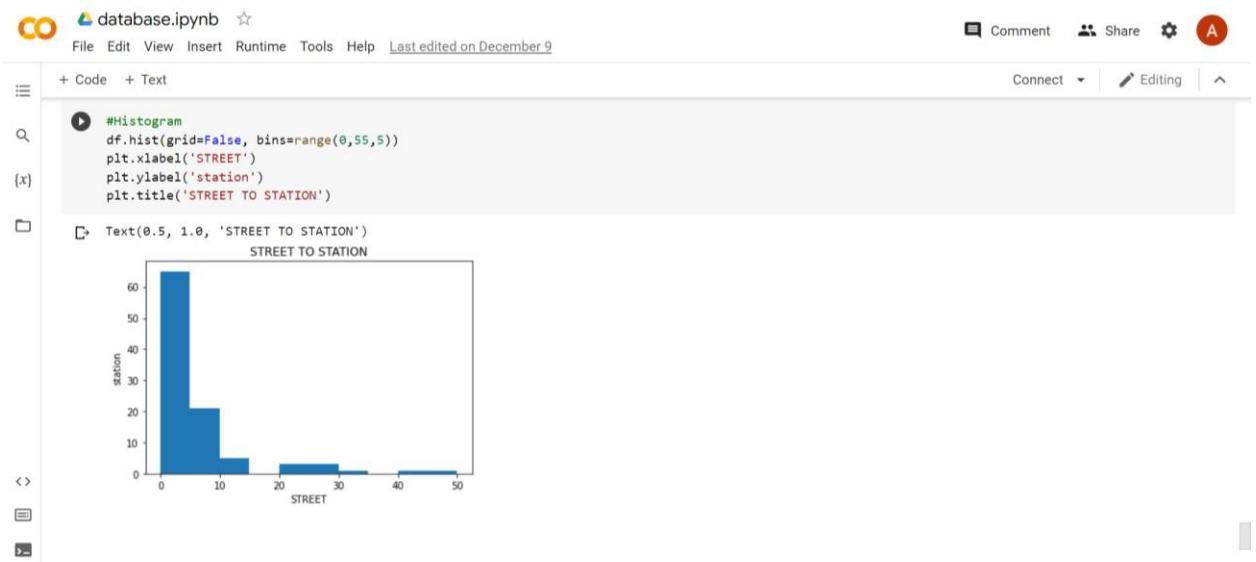
```
[ ] #Histogram  
df.hist()
```

{x}

<matplotlib.axes._subplots.AxesSubplot at 0x7f8441373670>

```
▶ #Histogram  
df.hist(grid=False, bins=range(0,55,5))  
plt.xlabel('STREET')  
plt.ylabel('station')  
plt.title('STREET TO STATION')  
Text(0.5, 1.0, 'STREET TO STATION')
```

3. Histogram



Step 9: After data visualization, we have performed Data Normalization.

Data Normalization:

In the below process, we have checked whether our tables are in 1NF, 2NF and 3NF.

1NF:

Here, in order to check whether our table is in 1NF or not we have performed the following steps –

Step 1 — We have created a stored procedure named “new

_procedure” in order to check whether the values in each column are atomic and there is no repeating groups.

The screenshot shows two instances of MySQL Workbench. The top instance displays the 'new_procedure - Routine' editor with the following DDL code:

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `new_procedure`(out var int)
2 • BEGIN
3 •     select count(*) into var from neutdatabase.maindatabase where
4 •     OFFENSE_DESCRIPTION like '%,%';
5 • END
```

The bottom instance shows the execution of the stored procedure in the 'SQL File 5*' tab:

```
1
2 • set @var = 0;
3 • call neutdatabase.new_procedure(@var);
4 • select @var;
```

The result grid shows the value of @var:

@var
0

Step 2- Then we have created a function named “new_function” to check and display the result .

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar displays the database structure under the 'Schemas' tab, specifically for the 'neutdatabase' schema. The 'Functions' section is selected. The main area shows the DDL code for creating a function:

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `new_function`(a int) RETURNS varchar(20) C
2     READS SQL DATA
3     DETERMINISTIC
4     BEGIN
5         declare result varchar(20);
6         SET global log_bin_trust_function_creators=1;
7         IF a>0 THEN
8             SET result = "table is not in 1NF";
9         ELSE
10            SET result = 'table in 1 NF';
11        END IF;
12        RETURN result;
13    END
```

Below the DDL, a query is run to execute the function:

```
5
6 • select neutdatabase.new_function(@var);
7
8
```

The results grid shows the output of the function:

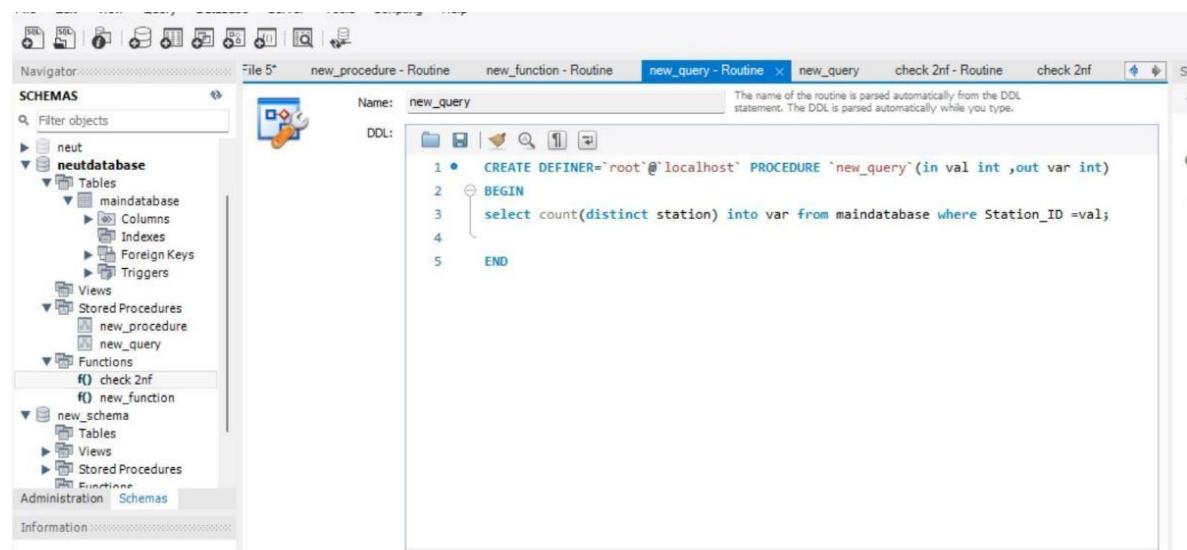
Result Grid
neutdatabase.new_function(@var)
table in 1 NF

At the bottom, the schema is identified as 'neutdatabase'.

2NF:

Here, in order to check whether our table is in 2NF or not we have performed the following steps –

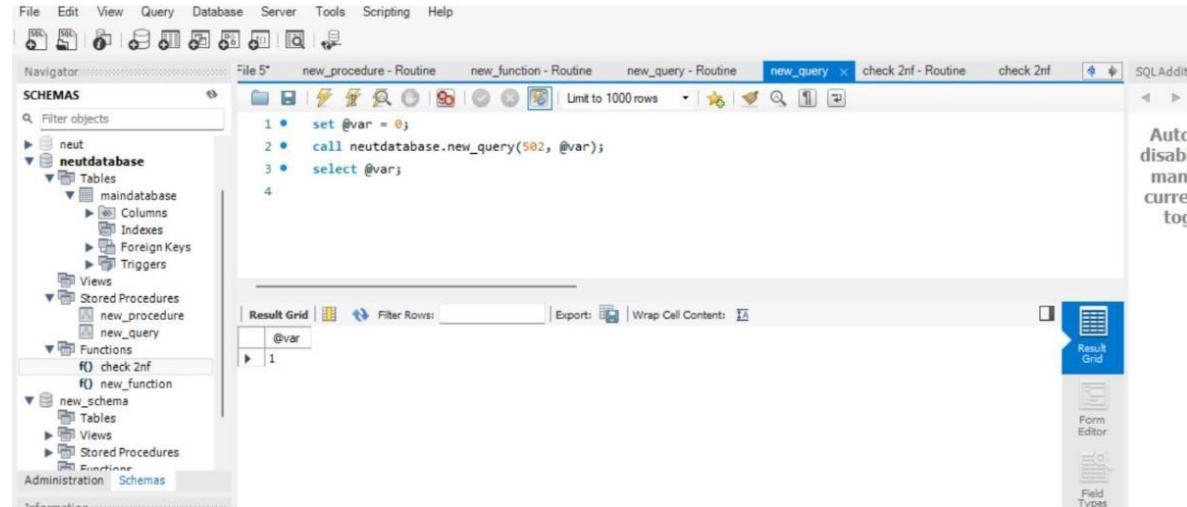
Step 1 – We have created a stored procedure named “new_query” in order to check that there is no partial dependencies.



The screenshot shows the MySQL Workbench interface. In the top navigation bar, the 'new_query - Routine' tab is selected. The 'Name:' field contains 'new_query'. The 'DDL:' pane displays the following SQL code:

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `new_query`(in val int ,out var int)
2 • BEGIN
3 •     select count(distinct station) into var from maindatabase where Station_ID =val;
4 •
5 • END
```

The left sidebar shows the database schema with the 'neutdatabase' schema expanded, displaying tables like 'maindatabase', stored procedures like 'new_procedure' and 'new_query', and functions like 'check 2nf' and 'new_function'.



The screenshot shows the MySQL Workbench interface with the 'new_query' tab selected. The 'File 5*' tab is also visible. The 'Result Grid' pane shows the output of the stored procedure:

```
1 • set @var = 0;
2 • call neutdatabase.new_query(502, @var);
3 • select @var;
```

The result grid displays a single row with the value '1' in the '@var' column. The right sidebar includes tabs for 'Result Grid', 'Form Editor', and 'Field Types'.

Step 2- Then we have created a function named “check 2nf” to check and display the result .

The screenshot shows two instances of MySQL Workbench. The top instance is in the 'Routines' tab, displaying the DDL for creating a function named 'check 2nf'. The bottom instance is in the 'SQL' tab, executing a query to run the function and displaying the results.

Detailed Description:

Top Window (Routines Tab):

- Schemas:** neut, neutdatabase
- Tables:** maindatabase
- Functions:** check 2nf (selected)
- DDL:**

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `check 2nf`(a int) RETURNS varchar(20) CHARACTER SET latin1
2 BEGIN
3     declare result varchar(20);
4
5     IF a>1 THEN
6         SET result = "IN NF 2";
7     ELSE
8         SET result = 'not IN NF 2';
9     END IF;
10    RETURN result;
11
12 END
```

Bottom Window (SQL Tab):

- Schemas:** neut, neutdatabase
- Query:**

```
1 • select neutdatabase.`check 2nf`(505);
```

Result Grid:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
neutdatabase.`check 2nf` 505			
IN NF 2			

3NF:

Here, in order to check whether our table is in 3NF or not we have performed the following steps –

Step 1 – We have created a stored procedure named “new_procedure2” in order to check that there is no transitive dependencies.

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree view shows a database named 'MAS' containing a table 'tab3'. The 'Stored Procedures' node under 'MAS' has a single entry: 'new_procedure2'. On the right, the main workspace displays the DDL code for 'new_procedure2':

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `new_procedure2`(in val int , out var INT)
2 • BEGIN
3     select count(distinct station) into var from tab3 where Station_ID =val;
4     END
```

Below the DDL, a result grid shows the output of the stored procedure:

@var
1

Step 2- Then we have created a function named “checknf3b3” to check and display the result .

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree view shows a database named 'MAS' containing a table 'tab3' and several other tables like 'top_STREETH', 'top_STREETP', etc. The 'Functions' node under 'MAS' has a single entry: 'checknf3b3'. On the right, the main workspace displays the DDL code for 'checknf3b3':

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `checknf3b3`(a int ) RETURNS varchar(20) CHARSET utf8mb4
2     READS SQL DATA
3     DETERMINISTIC
4     BEGIN
5         declare result varchar(20);
6         SET GLOBAL log_bin_trust_function_creators = 1;
7         IF a1 THEN
8             SET result = " NOT IN NF 3 ";
9         ELSE
10            SET result = " IN NF 3 ";
11        END IF;
```

```

MAS
  |__ Iter objects
    |__ Foreign Keys
    |__ Triggers
    |__ tab3
      |__ Columns
        |__ line_001
          |__ closest_station_distance
          |__ closest_station_1km
          |__ closest_station_2km
          |__ closest_station_3km
          |__ ID
        |__ Indexes
        |__ Triggers
      |__ Views
    |__ Stored Procedures
      |__ new procedure2

```

```

1 • call neut.new_procedure(510,@v1);
2 • select neut.checknf3b3(@v1);

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content: 
neut.checknf3b3(@v1)
IN NF 3

```

Team Number- 9

Team Members Description:

1. Name: Deepak Sawalka

NU-ID: 002910793

NU Email-ID: sawalka.d@northeastern.edu

GitHub-ID: DeepakSawalka

2. Name: Krishana Kapadia

NU-ID: 002762329

NU Email-ID: kapadia.kr@northeastern.edu

GitHub-ID: krishnakapadia29

3. Name: Janhvi Shah

NU-ID: 002783375

NU Email-ID: shah.janh@northeastern.edu

GitHub-ID: janhvishah

4. Name: Anjali Kabra

NU-ID: 002762324

NU Email-ID: kabra.an@northeastern.edu

GitHub-ID: anjalikabra18