I have chosen data regarding the crime rates in Portland, Oregon. The data provided is reliable as data has been gathered from portlandoregon.gov and civicapps.org.

The data shows crime rate stats from the year 2015-2017. Each individual crime reported is lists the location, time and date of the incident as well as a the neighbourhood in which the event occurred.

This dataset can be useful to find out the top hotspots for crime, and what offences has been committed the most. These information can be vital to a lot of parties and organisations.

Firstly, the police department. Knowing which neighbourhood in Portland itself has the most crimes, they can deploy more patrols and police enforcement in that area itself. This is to deter crimes from happening in the area. The police could also look into the most committed crimes which had been done over the past years and look into the matter, maybe pass a new law or a stricter fine to reduce these number of cases. Police can also work hand in hand with the government, to also maybe create public education and raise awareness.

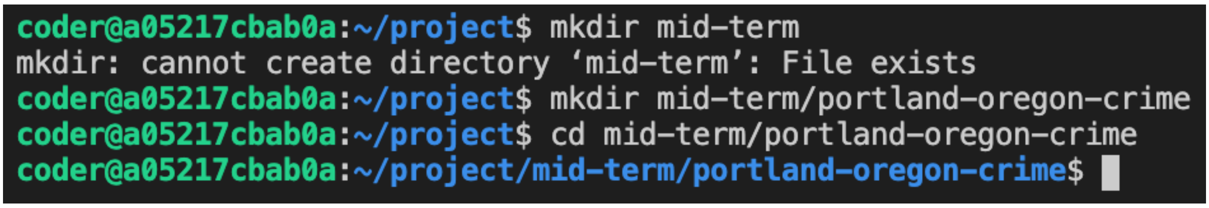
If the number of crimes keep dropping in the coming years, this may lead to an economic growth to Portland as well. Tourists would feel safe coming over to the city to enjoy what Portland has to offer, boosting the economy.

Coming from an Asian country, where owning guns are prohibited by law, I would like to know if there is a connection between the US law of being able to own a gun and the number of crime rates.

This is why it is an interesting dataset to work with as we can answer questions such as:

- How has crime changed over the years?
- Is it possible to predict where or when a crime will be committed?
- Which areas of the city have evolved over this time span?
- In which area most crimes are committed?
- What types of crimes are most common?
- Where are different types of crimes most likely to occur?
- Does the frequency of crimes change over the day? Week? Year?

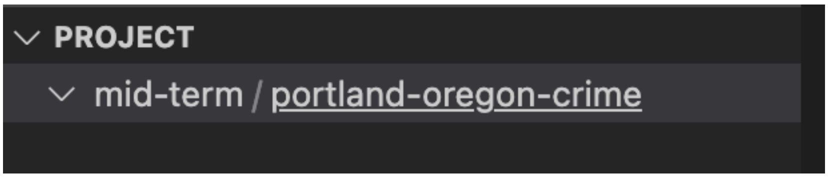
I created a new directory named ‘mid-term’ and ‘portland-oregon-crime’. ‘portland-oregon-crime’ will be located inside the ‘mid-term’ (subdirectory)



```
coder@a05217cbab0a:~/project$ mkdir mid-term
mkdir: cannot create directory ‘mid-term’: File exists
coder@a05217cbab0a:~/project$ mkdir mid-term/portland-oregon-crime
coder@a05217cbab0a:~/project$ cd mid-term/portland-oregon-crime
coder@a05217cbab0a:~/project/mid-term/portland-oregon-crime$ 
```

$mk dir mid-term
$ mk dir mid-term/portland-oregon-crime

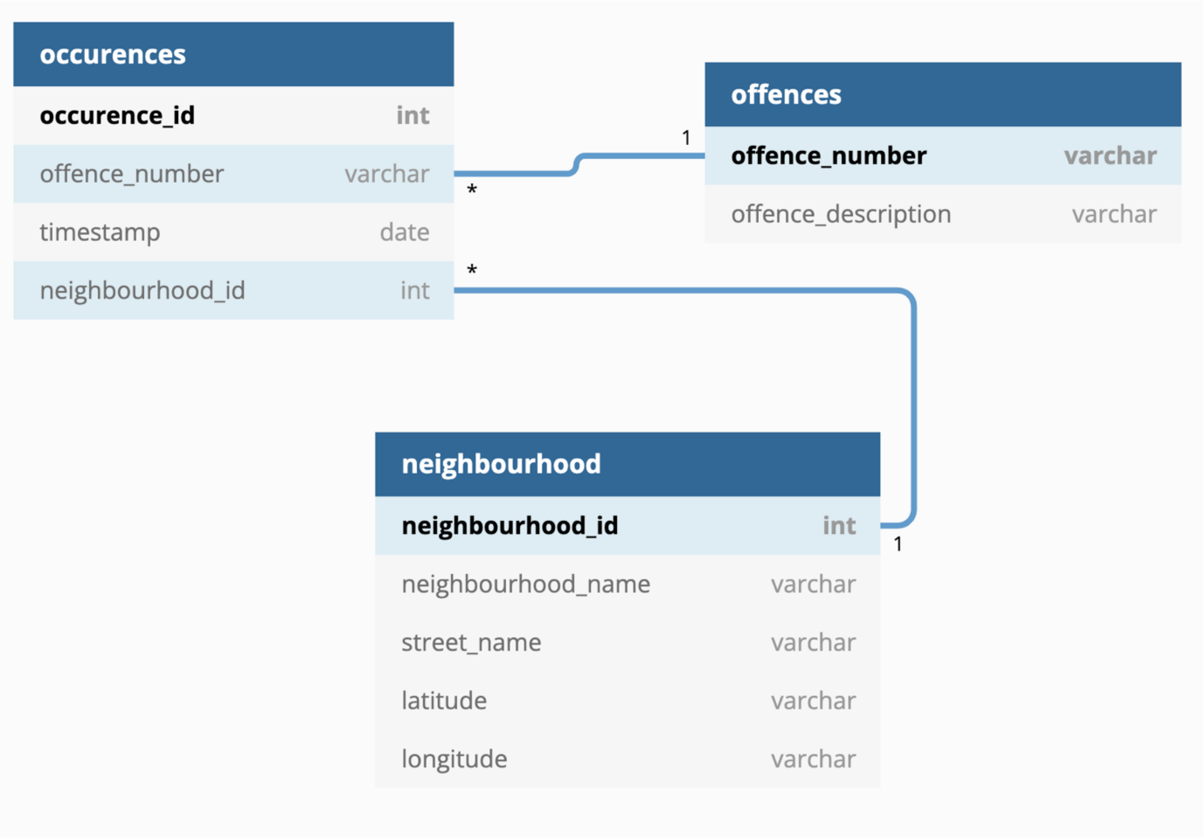
This is the result:



To go into the new directory, we can type
$ cd mid-term/portland-oregon-crime

type
$pwd
to check our current directory

Relational Schemas



occurences	
occurence_id	int
offence_number	varchar
timestamp	date
neighbourhood_id	int

offences	
offence_number	varchar
offence_description	varchar

neighbourhood	
neighbourhood_id	int
neighbourhood_name	varchar
street_name	varchar
latitude	varchar
longitude	varchar

created in: https://dbdiagram.io

neighbourhood.neighbourhood_id to occurences.neighbourhood_id is a one to many relationship as each neighbourhood will most likely will have one or more occurrences of crime.

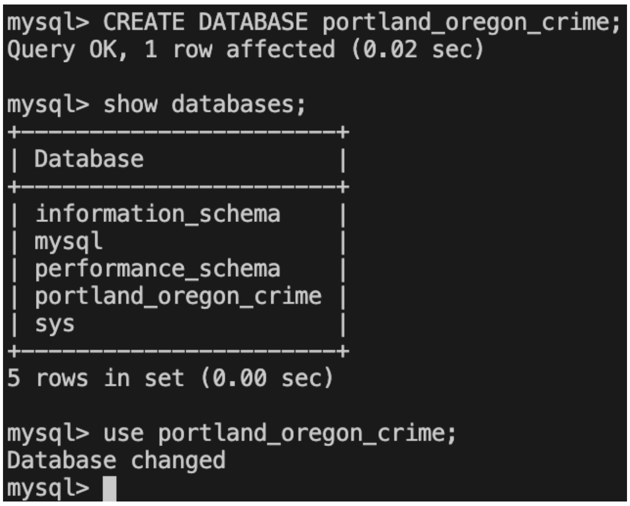
offences.offence_number to occurences.offence_number is also a one to many relationship as every occurrence has a unique offence_number.

Afterwards, it is time to run SQL console to create database. Open the terminal and type the command below:
$ mysql

Let us create a database called ‘portland_oregon_crime’ and use it

> CREATE DATABASE portland_oregon_crime;
> USE portland_oregon_crime;

The steps above are shown below. > SHOW databases; command is to check all databases which are available

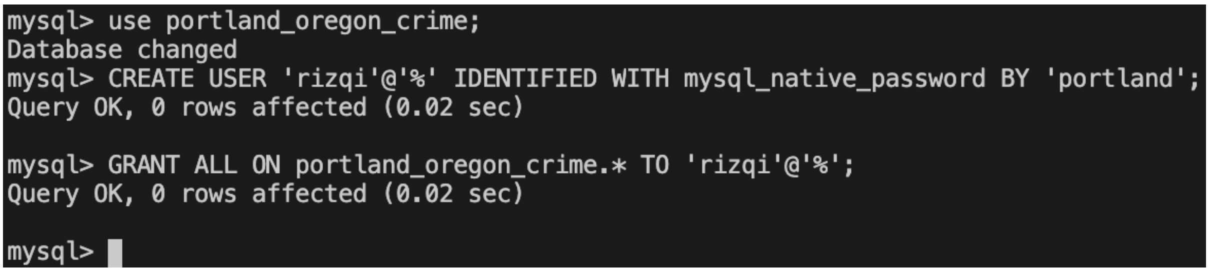


```
mysql> CREATE DATABASE portland_oregon_crime;
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
+-----------------------+
| Database              |
+-----------------------+
| information_schema    |
| mysql                 |
| performance_schema    |
| portland_oregon_crime |
| sys                   |
+-----------------------+
5 rows in set (0.00 sec)

mysql> use portland_oregon_crime;
Database changed
mysql> 
```

Let us now create a database user named ‘rizqi’. Grant access for this user to the database created above.

> CREATE USER ‘rizqi’@’%’ IDENTIFIED WITH mysql_native_password BY ‘california’;
> GRANT ALL ON portland_oregon_crime.* TO ‘rizqi’@’%’;

This is what is observed in the terminal:

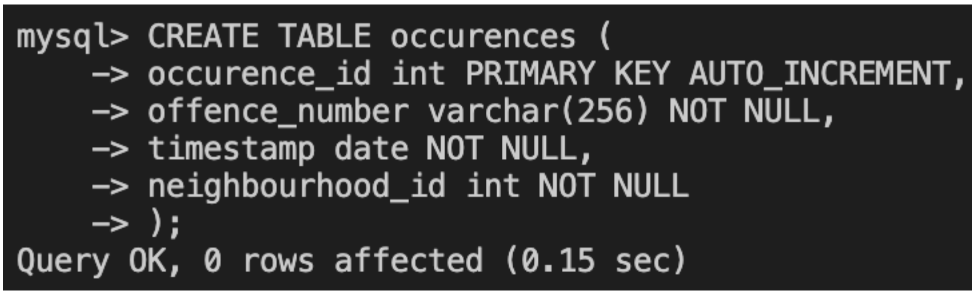


```
mysql> use portland_oregon_crime;
Database changed
mysql> CREATE USER 'rizqi'@'%' IDENTIFIED WITH mysql_native_password BY 'portland';
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT ALL ON portland_oregon_crime.* TO 'rizqi'@'%';
Query OK, 0 rows affected (0.02 sec)

mysql> 
```

Now it is time for **table creation**

To create **occurences** table:

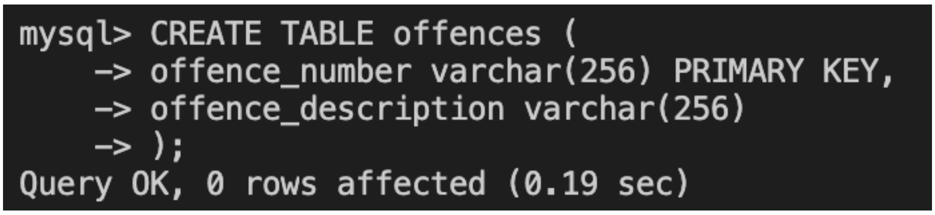
CREATE TABLE occurences (
occurence_id int PRIMARY KEY AUTO_INCREMENT,
offence_number varchar(256) NOT NULL,
timestamp date NOT NULL,
neighbourhood_id int NOT NULL
);



```
mysql> CREATE TABLE occurences (
    -> occurence_id int PRIMARY KEY AUTO_INCREMENT,
    -> offence_number varchar(256) NOT NULL,
    -> timestamp date NOT NULL,
    -> neighbourhood_id int NOT NULL
    -> );
Query OK, 0 rows affected (0.15 sec)
```

To create **offences** table:

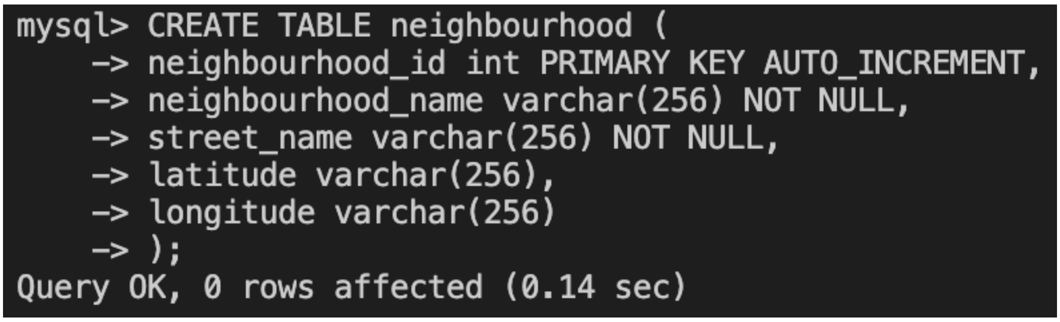
CREATE TABLE offences (
offence_number varchar(256) PRIMARY KEY,
offence_description varchar(256)
);



```
mysql> CREATE TABLE offences (
    -> offence_number varchar(256) PRIMARY KEY,
    -> offence_description varchar(256)
    -> );
Query OK, 0 rows affected (0.19 sec)
```

To create **neighbourhood** table:

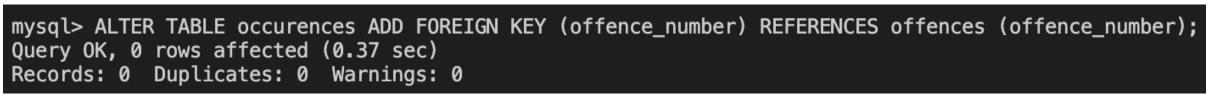
CREATE TABLE neighbourhood (
neighbourhood_id int PRIMARY KEY AUTO_INCREMENT,
neighbourhood_name varchar(256) NOT NULL,
street_name varchar(256) NOT NULL,
latitude varchar(256),
longitude varchar(256)
);



```
mysql> CREATE TABLE neighbourhood (
    -> neighbourhood_id int PRIMARY KEY AUTO_INCREMENT,
    -> neighbourhood_name varchar(256) NOT NULL,
    -> street_name varchar(256) NOT NULL,
    -> latitude varchar(256),
    -> longitude varchar(256)
    -> );
Query OK, 0 rows affected (0.14 sec)
```

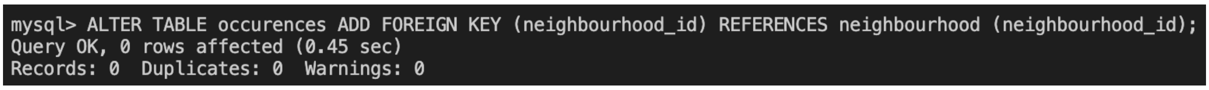
Adding **foreign key** to occurences table from offences and neighbourhood tables.

ALTER TABLE occurences ADD FOREIGN KEY (offence_number) REFERENCES offences (offence_number);



```
mysql> ALTER TABLE occurences ADD FOREIGN KEY (offence_number) REFERENCES offences (offence_number);
Query OK, 0 rows affected (0.37 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

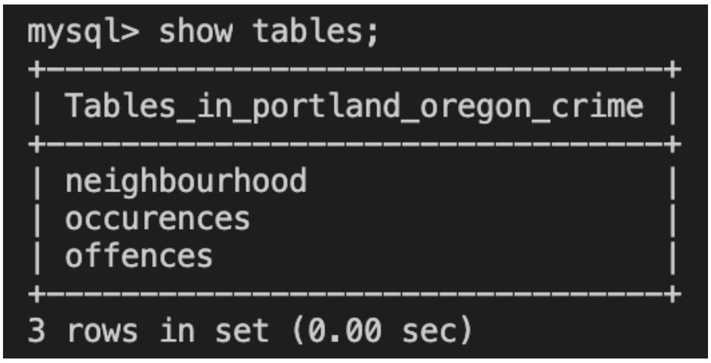
ALTER TABLE occurences ADD FOREIGN KEY (neighbourhood_id) REFERENCES neighbourhood (neighbourhood_id);



```
mysql> ALTER TABLE occurences ADD FOREIGN KEY (neighbourhood_id) REFERENCES neighbourhood (neighbourhood_id);
Query OK, 0 rows affected (0.45 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

We can check the tables which have been created under our database portland_oregon_crime

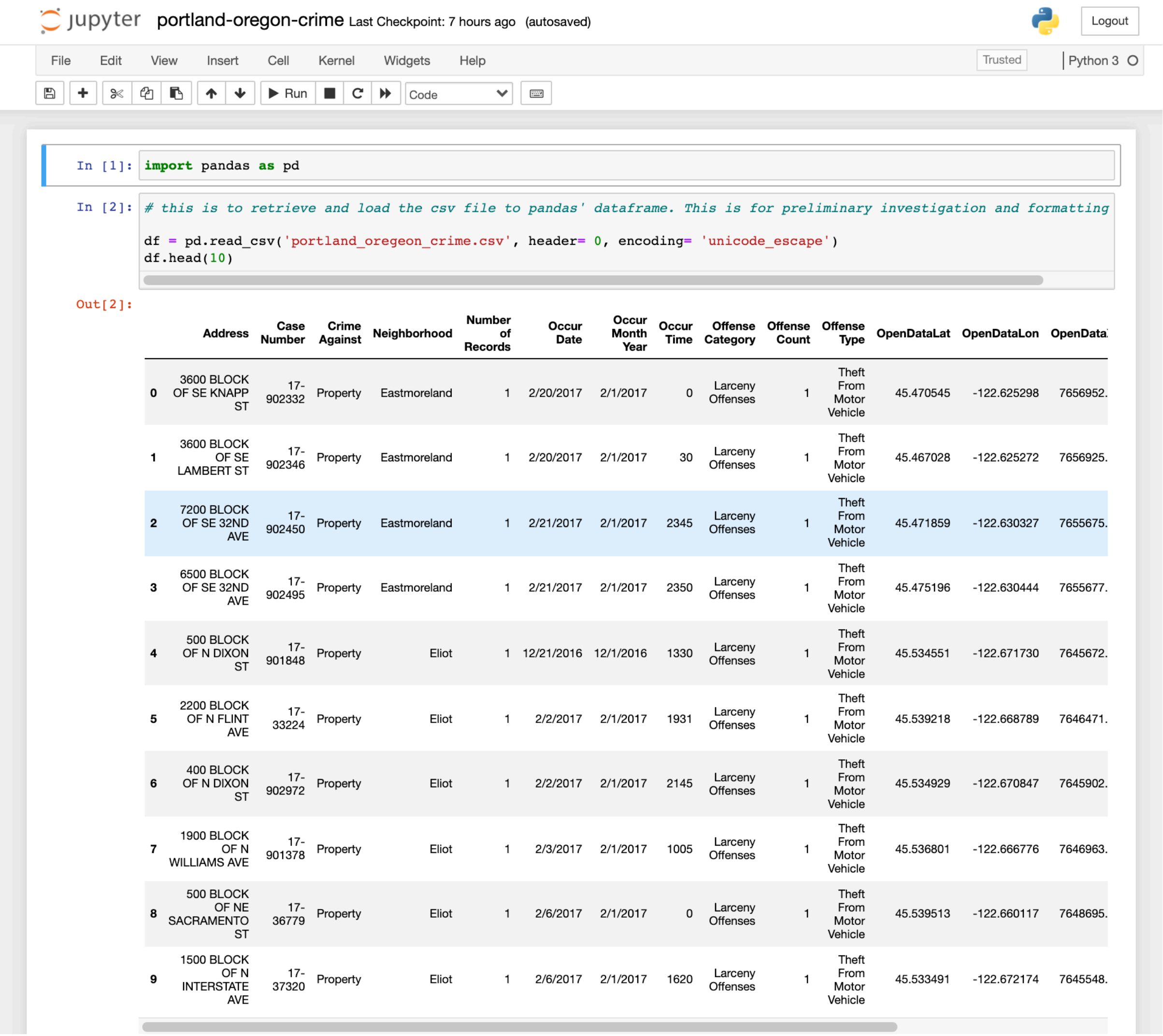
SHOW tables;



```
mysql> show tables;
+----------------------------------+
| Tables_in_portland_oregon_crime |
+----------------------------------+
| neighbourhood                    |
| occurences                       |
| offences                         |
+----------------------------------+
3 rows in set (0.00 sec)
```

Now it is time to **clean** our data

We import and load he csv file to pandas' dataframe first.

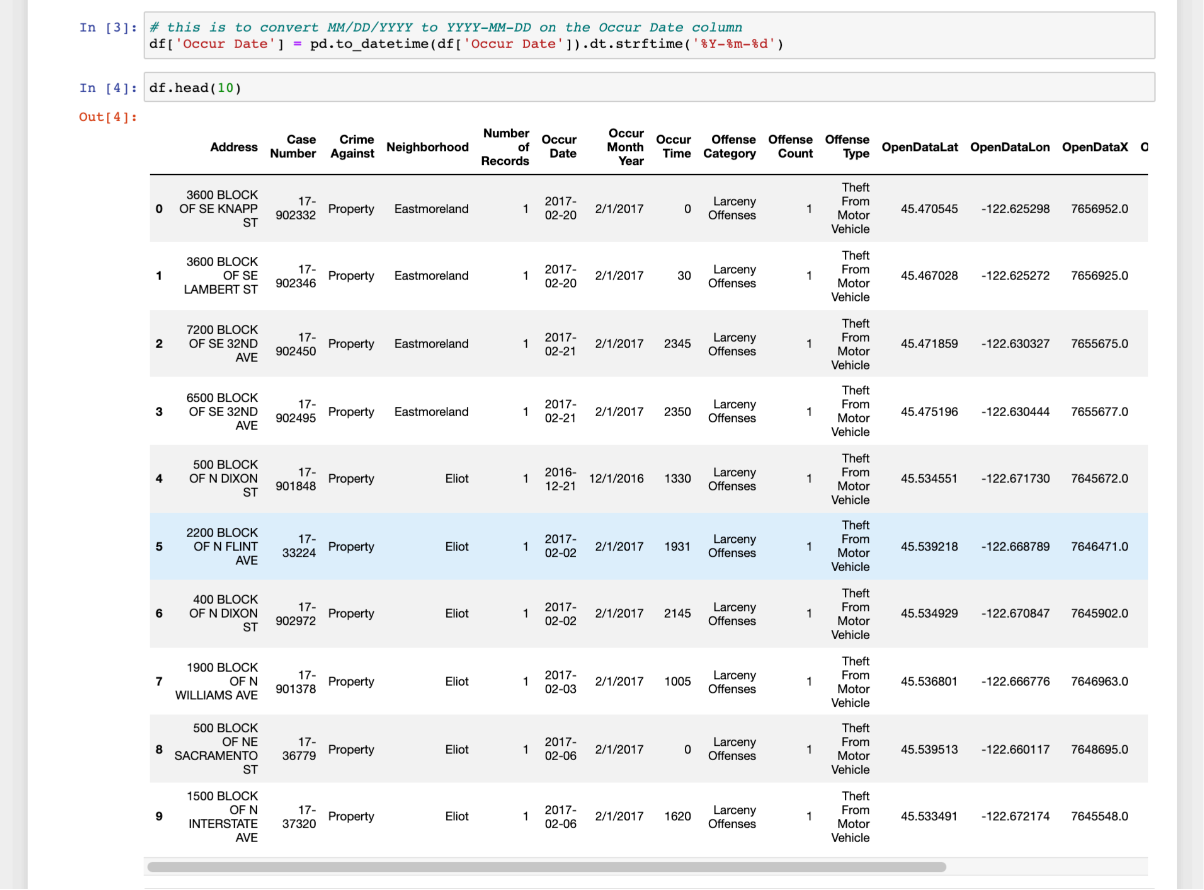


```
In [1]: import pandas as pd

In [2]: # this is to retrieve and load the csv file to pandas' dataframe. This is for preliminary investigation and formatting

        df = pd.read_csv('portland_oregeon_crime.csv', header= 0, encoding= 'unicode_escape')
        df.head(10)
```

Out[2]:

	Address	Case Number	Crime Against	Neighborhood	Number of Records	Occur Date	Occur Month Year	Occur Time	Offense Category	Offense Count	Offense Type	OpenDataLat	OpenDataLon	OpenData
0	3600 BLOCK OF SE KNAPP ST	17-902332	Property	Eastmoreland	1	2/20/2017	2/1/2017	0	Larceny Offenses	1	Theft From Motor Vehicle	45.470545	-122.625298	7656952.
1	3600 BLOCK OF SE LAMBERT ST	17-902346	Property	Eastmoreland	1	2/20/2017	2/1/2017	30	Larceny Offenses	1	Theft From Motor Vehicle	45.467028	-122.625272	7656925.
2	7200 BLOCK OF SE 32ND AVE	17-902450	Property	Eastmoreland	1	2/21/2017	2/1/2017	2345	Larceny Offenses	1	Theft From Motor Vehicle	45.471859	-122.630327	7655675.
3	6500 BLOCK OF SE 32ND AVE	17-902495	Property	Eastmoreland	1	2/21/2017	2/1/2017	2350	Larceny Offenses	1	Theft From Motor Vehicle	45.475196	-122.630444	7655677.
4	500 BLOCK OF N DIXON ST	17-901848	Property	Eliot	1	12/21/2016	12/1/2016	1330	Larceny Offenses	1	Theft From Motor Vehicle	45.534551	-122.671730	7645672.
5	2200 BLOCK OF N FLINT AVE	17-33224	Property	Eliot	1	2/2/2017	2/1/2017	1931	Larceny Offenses	1	Theft From Motor Vehicle	45.539218	-122.668789	7646471.
6	400 BLOCK OF N DIXON ST	17-902972	Property	Eliot	1	2/2/2017	2/1/2017	2145	Larceny Offenses	1	Theft From Motor Vehicle	45.534929	-122.670847	7645902.
7	1900 BLOCK OF N WILLIAMS AVE	17-901378	Property	Eliot	1	2/3/2017	2/1/2017	1005	Larceny Offenses	1	Theft From Motor Vehicle	45.536801	-122.666776	7646963.
8	500 BLOCK OF NE SACRAMENTO ST	17-36779	Property	Eliot	1	2/6/2017	2/1/2017	0	Larceny Offenses	1	Theft From Motor Vehicle	45.539513	-122.660117	7648695.
9	1500 BLOCK OF N INTERSTATE AVE	17-37320	Property	Eliot	1	2/6/2017	2/1/2017	1620	Larceny Offenses	1	Theft From Motor Vehicle	45.533491	-122.672174	7645548.

Afterwards, under the Occur Date column, we change the date format from MM/DD/YYYY to YYYY-MM-DD



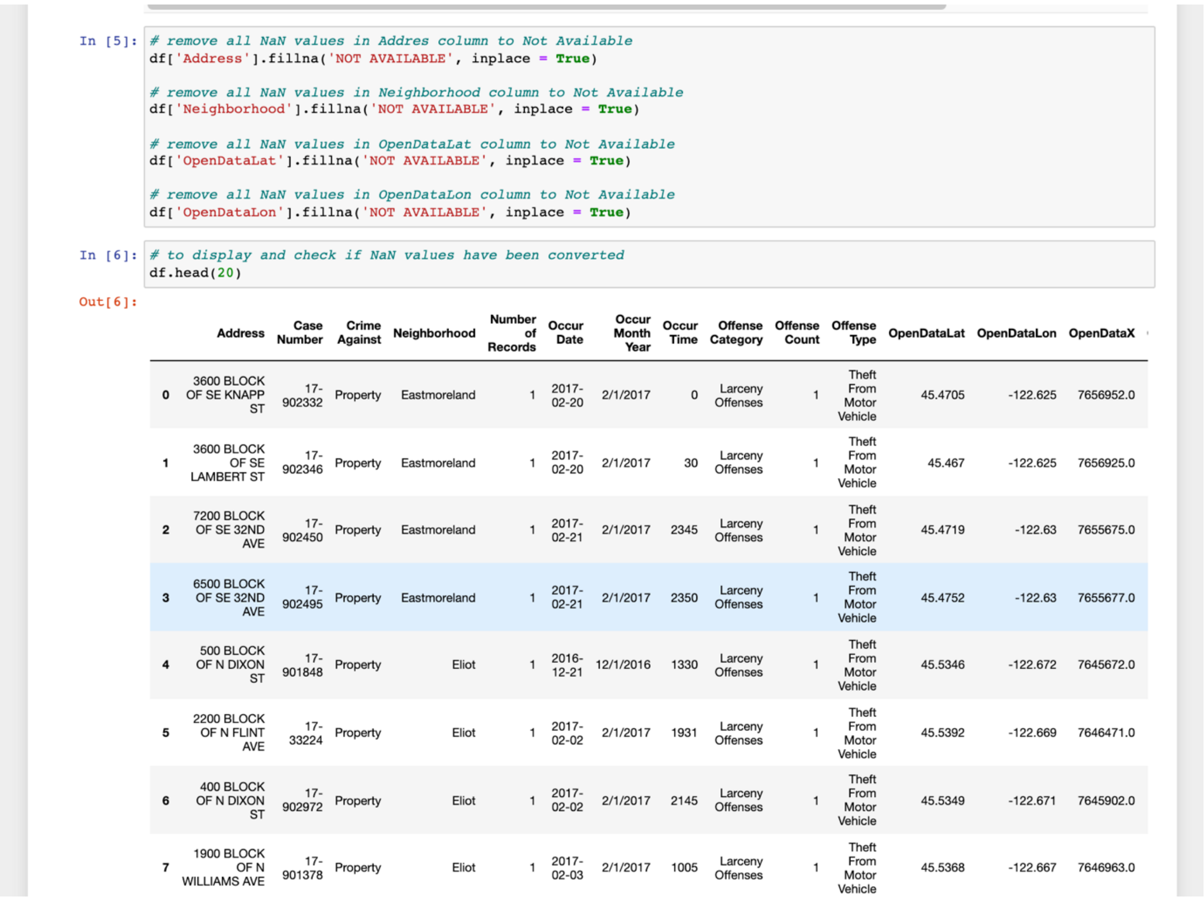
```
In [3]: # this is to convert MM/DD/YYYY to YYYY-MM-DD on the Occur Date column
        df['Occur Date'] = pd.to_datetime(df['Occur Date']).dt.strftime('%Y-%m-%d')
```

```
In [4]: df.head(10)
```

Out[4]:

	Address	Case Number	Crime Against	Neighborhood	Number of Records	Occur Date	Occur Month Year	Occur Time	Offense Category	Offense Count	Offense Type	OpenDataLat	OpenDataLon	OpenDataX	O
0	3600 BLOCK OF SE KNAPP ST	17-902332	Property	Eastmoreland	1	2017-02-20	2/1/2017	0	Larceny Offenses	1	Theft From Motor Vehicle	45.470545	-122.625298	7656952.0	
1	3600 BLOCK OF SE LAMBERT ST	17-902346	Property	Eastmoreland	1	2017-02-20	2/1/2017	30	Larceny Offenses	1	Theft From Motor Vehicle	45.467028	-122.625272	7656925.0	
2	7200 BLOCK OF SE 32ND AVE	17-902450	Property	Eastmoreland	1	2017-02-21	2/1/2017	2345	Larceny Offenses	1	Theft From Motor Vehicle	45.471859	-122.630327	7655675.0	
3	6500 BLOCK OF SE 32ND AVE	17-902495	Property	Eastmoreland	1	2017-02-21	2/1/2017	2350	Larceny Offenses	1	Theft From Motor Vehicle	45.475196	-122.630444	7655677.0	
4	500 BLOCK OF N DIXON ST	17-901848	Property	Eliot	1	2016-12-21	12/1/2016	1330	Larceny Offenses	1	Theft From Motor Vehicle	45.534551	-122.671730	7645672.0	
5	2200 BLOCK OF N FLINT AVE	17-33224	Property	Eliot	1	2017-02-02	2/1/2017	1931	Larceny Offenses	1	Theft From Motor Vehicle	45.539218	-122.668789	7646471.0	
6	400 BLOCK OF N DIXON ST	17-902972	Property	Eliot	1	2017-02-02	2/1/2017	2145	Larceny Offenses	1	Theft From Motor Vehicle	45.534929	-122.670847	7645902.0	
7	1900 BLOCK OF N WILLIAMS AVE	17-901378	Property	Eliot	1	2017-02-03	2/1/2017	1005	Larceny Offenses	1	Theft From Motor Vehicle	45.536801	-122.666776	7646963.0	
8	500 BLOCK OF NE SACRAMENTO ST	17-36779	Property	Eliot	1	2017-02-06	2/1/2017	0	Larceny Offenses	1	Theft From Motor Vehicle	45.539513	-122.660117	7648695.0	
9	1500 BLOCK OF N INTERSTATE AVE	17-37320	Property	Eliot	1	2017-02-06	2/1/2017	1620	Larceny Offenses	1	Theft From Motor Vehicle	45.533491	-122.672174	7645548.0	

We converted all the blank values under Address, Neighborhood, OpenDatLat and OpenDatLon to Not Available



In [5]:

```
# remove all NaN values in Addres column to Not Available
df['Address'].fillna('NOT AVAILABLE', inplace = True)

# remove all NaN values in Neighborhood column to Not Available
df['Neighborhood'].fillna('NOT AVAILABLE', inplace = True)

# remove all NaN values in OpenDataLat column to Not Available
df['OpenDataLat'].fillna('NOT AVAILABLE', inplace = True)

# remove all NaN values in OpenDataLon column to Not Available
df['OpenDataLon'].fillna('NOT AVAILABLE', inplace = True)
```

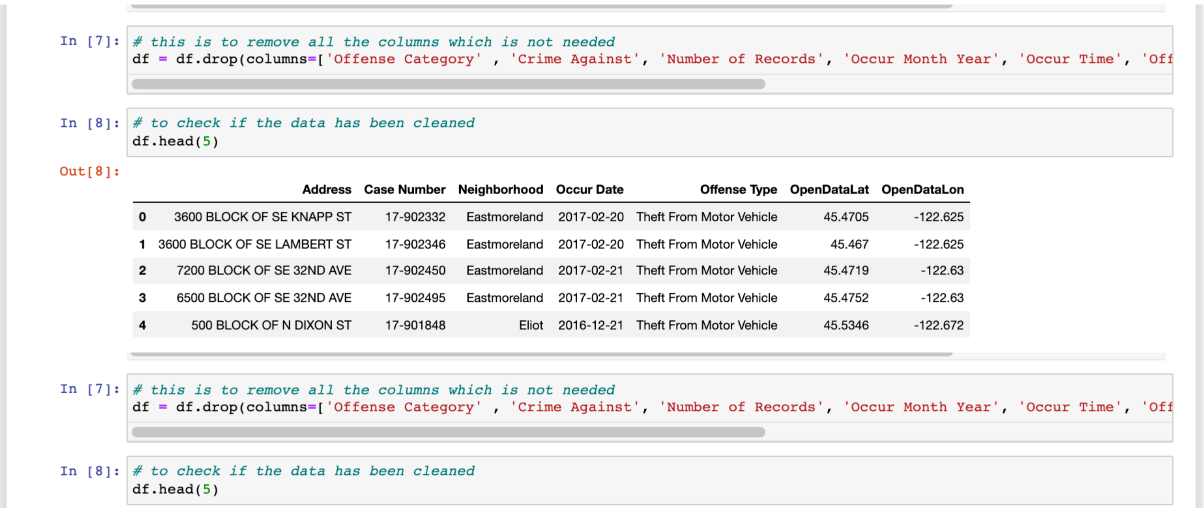
In [6]:

```
# to display and check if NaN values have been converted
df.head(20)
```

Out[6]:

	Address	Case Number	Crime Against	Neighborhood	Number of Records	Occur Date	Occur Month Year	Occur Time	Offense Category	Offense Count	Offense Type	OpenDataLat	OpenDataLon	OpenDataX
0	3600 BLOCK OF SE KNAPP ST	17-902332	Property	Eastmoreland	1	2017-02-20	2/1/2017	0	Larceny Offenses	1	Theft From Motor Vehicle	45.4705	-122.625	7656952.0
1	3600 BLOCK OF SE LAMBERT ST	17-902346	Property	Eastmoreland	1	2017-02-20	2/1/2017	30	Larceny Offenses	1	Theft From Motor Vehicle	45.467	-122.625	7656925.0
2	7200 BLOCK OF SE 32ND AVE	17-902450	Property	Eastmoreland	1	2017-02-21	2/1/2017	2345	Larceny Offenses	1	Theft From Motor Vehicle	45.4719	-122.63	7655675.0
3	6500 BLOCK OF SE 32ND AVE	17-902495	Property	Eastmoreland	1	2017-02-21	2/1/2017	2350	Larceny Offenses	1	Theft From Motor Vehicle	45.4752	-122.63	7655677.0
4	500 BLOCK OF N DIXON ST	17-901848	Property	Eliot	1	2016-12-21	12/1/2016	1330	Larceny Offenses	1	Theft From Motor Vehicle	45.5346	-122.672	7645672.0
5	2200 BLOCK OF N FLINT AVE	17-33224	Property	Eliot	1	2017-02-02	2/1/2017	1931	Larceny Offenses	1	Theft From Motor Vehicle	45.5392	-122.669	7646471.0
6	400 BLOCK OF N DIXON ST	17-902972	Property	Eliot	1	2017-02-02	2/1/2017	2145	Larceny Offenses	1	Theft From Motor Vehicle	45.5349	-122.671	7645902.0
7	1900 BLOCK OF N WILLIAMS AVE	17-901378	Property	Eliot	1	2017-02-03	2/1/2017	1005	Larceny Offenses	1	Theft From Motor Vehicle	45.5368	-122.667	7646963.0

Next, we remove all the unnecessary columns which are not need for this assignment.



In [7]:

```
# this is to remove all the columns which is not needed
df = df.drop(columns=['Offense Category' , 'Crime Against', 'Number of Records', 'Occur Month Year', 'Occur Time', 'Off
```

In [8]:

```
# to check if the data has been cleaned
df.head(5)
```

Out[8]:

	Address	Case Number	Neighborhood	Occur Date	Offense Type	OpenDataLat	OpenDataLon
0	3600 BLOCK OF SE KNAPP ST	17-902332	Eastmoreland	2017-02-20	Theft From Motor Vehicle	45.4705	-122.625
1	3600 BLOCK OF SE LAMBERT ST	17-902346	Eastmoreland	2017-02-20	Theft From Motor Vehicle	45.467	-122.625
2	7200 BLOCK OF SE 32ND AVE	17-902450	Eastmoreland	2017-02-21	Theft From Motor Vehicle	45.4719	-122.63
3	6500 BLOCK OF SE 32ND AVE	17-902495	Eastmoreland	2017-02-21	Theft From Motor Vehicle	45.4752	-122.63
4	500 BLOCK OF N DIXON ST	17-901848	Eliot	2016-12-21	Theft From Motor Vehicle	45.5346	-122.672

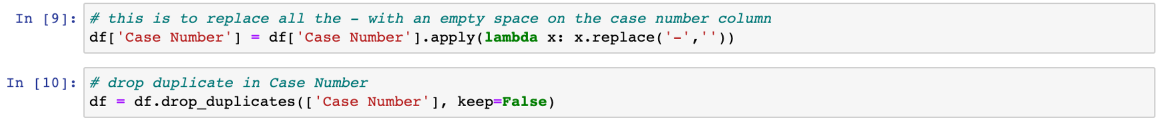
In [7]:

```
# this is to remove all the columns which is not needed
df = df.drop(columns=['Offense Category' , 'Crime Against', 'Number of Records', 'Occur Month Year', 'Occur Time', 'Off
```

In [8]:

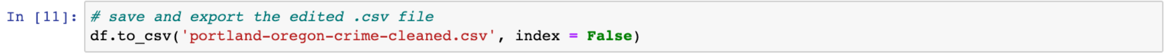
```
# to check if the data has been cleaned
df.head(5)
```

Next, we replace – with an empty space on the Case Number column, followed by removing all the duplicate values in that column as well. This is because I plan to use the Case Number column as a Primary Key (offence_number), and therefore needs to be unique.



```
In [9]:  # this is to replace all the - with an empty space on the case number column
         df['Case Number'] = df['Case Number'].apply(lambda x: x.replace('-',''))

In [10]: # drop duplicate in Case Number
         df = df.drop_duplicates(['Case Number'], keep=False)
```

And once we are done with cleaning, we export the file as 'portland-oregon-crime-cleaned.csv'



```
In [11]: # save and export the edited .csv file
         df.to_csv('portland-oregon-crime-cleaned.csv', index = False)
```

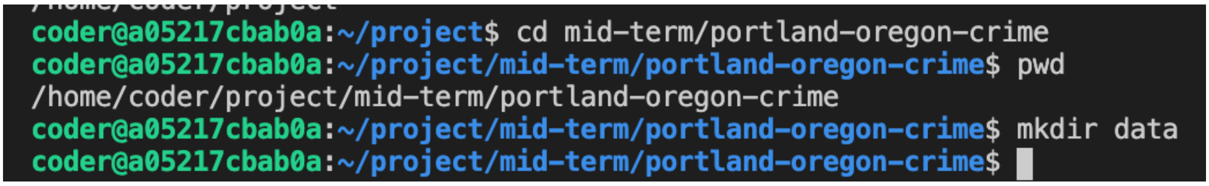
Now it is time to **load** our data

We need to ensure we are the correct directory. In this case, inside the 'mid-term/portland-oregon-crime/' folder. The next step is to make a new directory called 'data'. Place this directory under 'mid-term/portland-oregon-crime/'

$ pwd

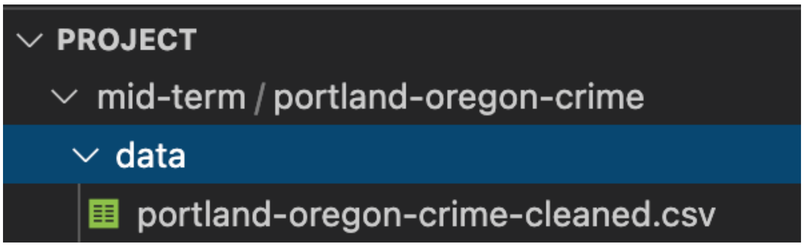
$ mkdir data

Steps in console are shown below:



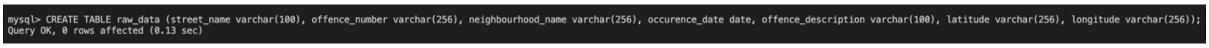
```
/home/coder/project
coder@a05217cbab0a:~/project$ cd mid-term/portland-oregon-crime
coder@a05217cbab0a:~/project/mid-term/portland-oregon-crime$ pwd
/home/coder/project/mid-term/portland-oregon-crime
coder@a05217cbab0a:~/project/mid-term/portland-oregon-crime$ mkdir data
coder@a05217cbab0a:~/project/mid-term/portland-oregon-crime$ 
```

The next step is to drag the 'portland-oregon-crime-cleaned.csv' file to the lab


PROJECT
mid-term / portland-oregon-crime
data
portland-oregon-crime-cleaned.csv

It is time to create a ‘raw_data’ table. This is to load the cleaned data which we just did.

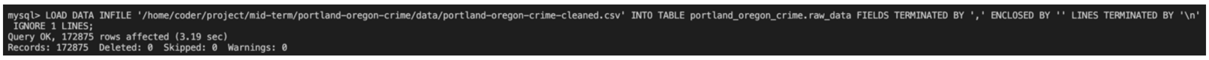
CREATE TABLE raw_data (street_name varchar(100), offence_number varchar(256), neighbourhood_name varchar(256), occurence_date date, offence_description varchar(256), latitude varchar(256), longitude varchar(256));



```
mysql> CREATE TABLE raw_data (street_name varchar(100), offence_number varchar(256), neighbourhood_name varchar(256), occurence_date date, offence_description varchar(100), latitude varchar(256), longitude varchar(256));
Query OK, 0 rows affected (0.13 sec)
```

After creating the table, we need to load our data in.

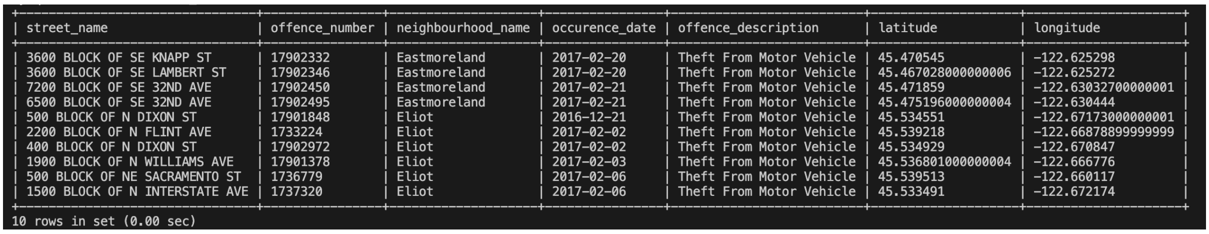
LOAD DATA INFILE '/home/coder/project/mid-term/portland-oregon-crime/data/portland-oregon-crime-cleaned.csv' INTO TABLE portland_oregon_crime.raw_data FIELDS TERMINATED BY ',' ENCLOSED BY '' LINES TERMINATED BY '\n' IGNORE 1 LINES;



```
mysql> LOAD DATA INFILE '/home/coder/project/mid-term/portland-oregon-crime/data/portland-oregon-crime-cleaned.csv' INTO TABLE portland_oregon_crime.raw_data FIELDS TERMINATED BY ',' ENCLOSED BY '' LINES TERMINATED BY '\n'
 IGNORE 1 LINES;
Query OK, 172875 rows affected (3.19 sec)
Records: 172875  Deleted: 0  Skipped: 0  Warnings: 0
```

Data has been successfully loaded. To view the data, we need to type

mysql > SELECT * FROM raw_data LIMIT 10;

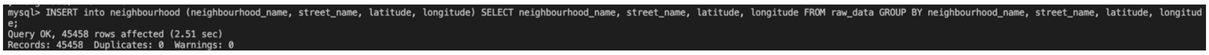


```
+--------------------------------+----------------+--------------------+----------------+--------------------------+--------------------+---------------------+
| street_name                    | offence_number | neighbourhood_name | occurence_date | offence_description      | latitude           | longitude           |
+--------------------------------+----------------+--------------------+----------------+--------------------------+--------------------+---------------------+
| 3600 BLOCK OF SE KNAPP ST      | 17902332       | Eastmoreland       | 2017-02-20     | Theft From Motor Vehicle | 45.470545          | -122.625298         |
| 3600 BLOCK OF SE LAMBERT ST    | 17902346       | Eastmoreland       | 2017-02-20     | Theft From Motor Vehicle | 45.467028000000006 | -122.625272         |
| 7200 BLOCK OF SE 32ND AVE      | 17902450       | Eastmoreland       | 2017-02-21     | Theft From Motor Vehicle | 45.471859          | -122.63032700000001 |
| 6500 BLOCK OF SE 32ND AVE      | 17902495       | Eastmoreland       | 2017-02-21     | Theft From Motor Vehicle | 45.475196000000004 | -122.630444         |
| 500 BLOCK OF N DIXON ST        | 17901848       | Eliot              | 2016-12-21     | Theft From Motor Vehicle | 45.534551          | -122.67173000000001 |
| 2200 BLOCK OF N FLINT AVE      | 1733224        | Eliot              | 2017-02-02     | Theft From Motor Vehicle | 45.539218          | -122.66878899999999 |
| 400 BLOCK OF N DIXON ST        | 17902972       | Eliot              | 2017-02-02     | Theft From Motor Vehicle | 45.534929          | -122.670847         |
| 1900 BLOCK OF N WILLIAMS AVE   | 17901378       | Eliot              | 2017-02-03     | Theft From Motor Vehicle | 45.536801000000004 | -122.666776         |
| 500 BLOCK OF NE SACRAMENTO ST  | 1736779        | Eliot              | 2017-02-06     | Theft From Motor Vehicle | 45.539513          | -122.660117         |
| 1500 BLOCK OF N INTERSTATE AVE | 1737320        | Eliot              | 2017-02-06     | Theft From Motor Vehicle | 45.533491          | -122.672174         |
+--------------------------------+----------------+--------------------+----------------+--------------------------+--------------------+---------------------+
10 rows in set (0.00 sec)
```

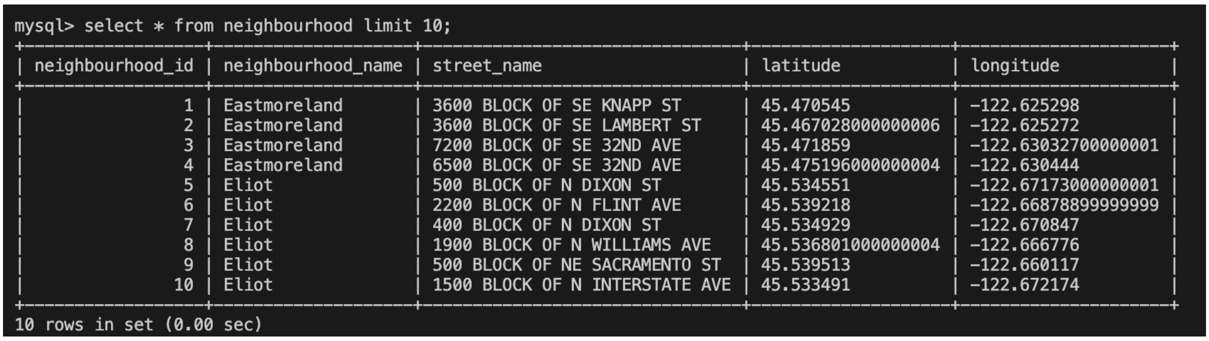
Now it is time to **insert** our raw_data into tables

neighbourhood table:

INSERT into neighbourhood (neighbourhood_name, street_name, latitude, longitude) SELECT neighbourhood_name, street_name, latitude, longitude FROM raw_data GROUP BY neighbourhood_name, street_name, latitude, longitude;



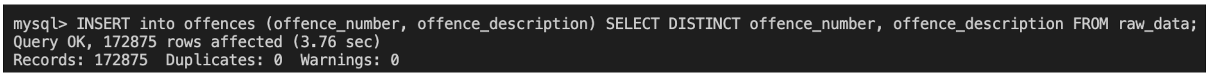
```
mysql> INSERT into neighbourhood (neighbourhood_name, street_name, latitude, longitude) SELECT neighbourhood_name, street_name, latitude, longitude FROM raw_data GROUP BY neighbourhood_name, street_name, latitude, longitud
e;
Query OK, 45458 rows affected (2.51 sec)
Records: 45458  Duplicates: 0  Warnings: 0
```



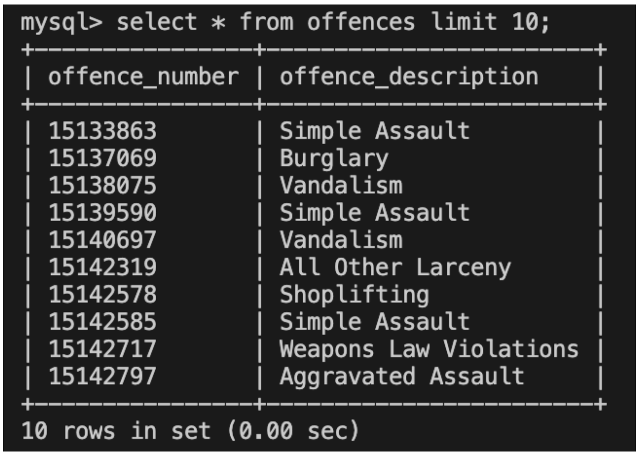
```
mysql> select * from neighbourhood limit 10;
+------------------+--------------------+--------------------------------+--------------------+---------------------+
| neighbourhood_id | neighbourhood_name | street_name                    | latitude           | longitude           |
+------------------+--------------------+--------------------------------+--------------------+---------------------+
|                1 | Eastmoreland       | 3600 BLOCK OF SE KNAPP ST      | 45.470545          | -122.625298         |
|                2 | Eastmoreland       | 3600 BLOCK OF SE LAMBERT ST    | 45.467028000000006 | -122.625272         |
|                3 | Eastmoreland       | 7200 BLOCK OF SE 32ND AVE      | 45.471859          | -122.63032700000001 |
|                4 | Eastmoreland       | 6500 BLOCK OF SE 32ND AVE      | 45.475196000000004 | -122.630444         |
|                5 | Eliot              | 500 BLOCK OF N DIXON ST        | 45.534551          | -122.67173000000001 |
|                6 | Eliot              | 2200 BLOCK OF N FLINT AVE      | 45.539218          | -122.66878899999999 |
|                7 | Eliot              | 400 BLOCK OF N DIXON ST        | 45.534929          | -122.670847         |
|                8 | Eliot              | 1900 BLOCK OF N WILLIAMS AVE   | 45.536801000000004 | -122.666776         |
|                9 | Eliot              | 500 BLOCK OF NE SACRAMENTO ST  | 45.539513          | -122.660117         |
|               10 | Eliot              | 1500 BLOCK OF N INTERSTATE AVE | 45.533491          | -122.672174         |
+------------------+--------------------+--------------------------------+--------------------+---------------------+
10 rows in set (0.00 sec)
```

offences table:

INSERT into offences (offence_number, offence_description) SELECT DISTINCT offence_number, offence_description FROM raw_data;



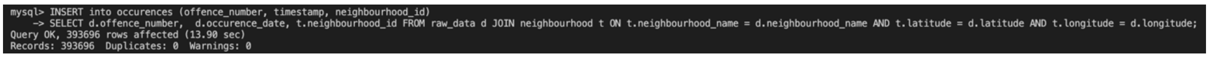
```
mysql> INSERT into offences (offence_number, offence_description) SELECT DISTINCT offence_number, offence_description FROM raw_data;
Query OK, 172875 rows affected (3.76 sec)
Records: 172875  Duplicates: 0  Warnings: 0
```



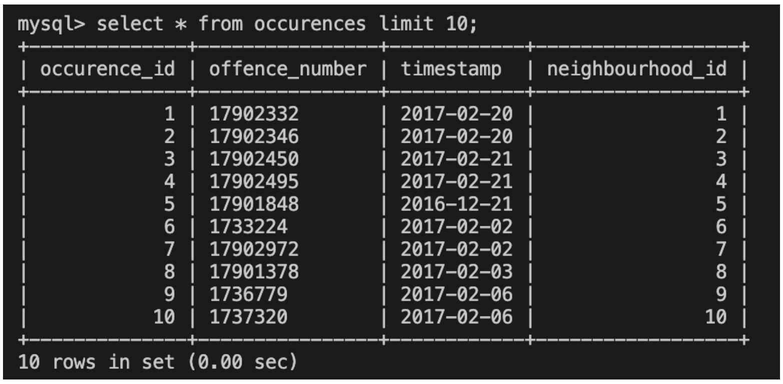
```
mysql> select * from offences limit 10;
+----------------+-------------------------+
| offence_number | offence_description     |
+----------------+-------------------------+
| 15133863       | Simple Assault          |
| 15137069       | Burglary                |
| 15138075       | Vandalism               |
| 15139590       | Simple Assault          |
| 15140697       | Vandalism               |
| 15142319       | All Other Larceny       |
| 15142578       | Shoplifting             |
| 15142585       | Simple Assault          |
| 15142717       | Weapons Law Violations  |
| 15142797       | Aggravated Assault      |
+----------------+-------------------------+
10 rows in set (0.00 sec)
```

occurences table:

INSERT into occurences (offence_number, timestamp, neighbourhood_id)
SELECT d.offence_number, d.occurence_date, t.neighbourhood_id FROM raw_data d JOIN neighbourhood t ON t.neighbourhood_name = d.neighbourhood_name AND t.latitude = d.latitude AND t.longitude = d.longitude;



```
mysql> INSERT into occurences (offence_number, timestamp, neighbourhood_id)
    -> SELECT d.offence_number,  d.occurence_date, t.neighbourhood_id FROM raw_data d JOIN neighbourhood t ON t.neighbourhood_name = d.neighbourhood_name AND t.latitude = d.latitude AND t.longitude = d.longitude;
Query OK, 393696 rows affected (13.90 sec)
Records: 393696  Duplicates: 0  Warnings: 0
```



```
mysql> select * from occurences limit 10;
+-------------+----------------+------------+------------------+
| occurence_id | offence_number | timestamp  | neighbourhood_id |
+-------------+----------------+------------+------------------+
|           1 | 17902332       | 2017-02-20 |                1 |
|           2 | 17902346       | 2017-02-20 |                2 |
|           3 | 17902450       | 2017-02-21 |                3 |
|           4 | 17902495       | 2017-02-21 |                4 |
|           5 | 17901848       | 2016-12-21 |                5 |
|           6 | 1733224        | 2017-02-02 |                6 |
|           7 | 17902972       | 2017-02-02 |                7 |
|           8 | 17901378       | 2017-02-03 |                8 |
|           9 | 1736779        | 2017-02-06 |                9 |
|          10 | 1737320        | 2017-02-06 |               10 |
+-------------+----------------+------------+------------------+
10 rows in set (0.00 sec)
```

Database Normalisation

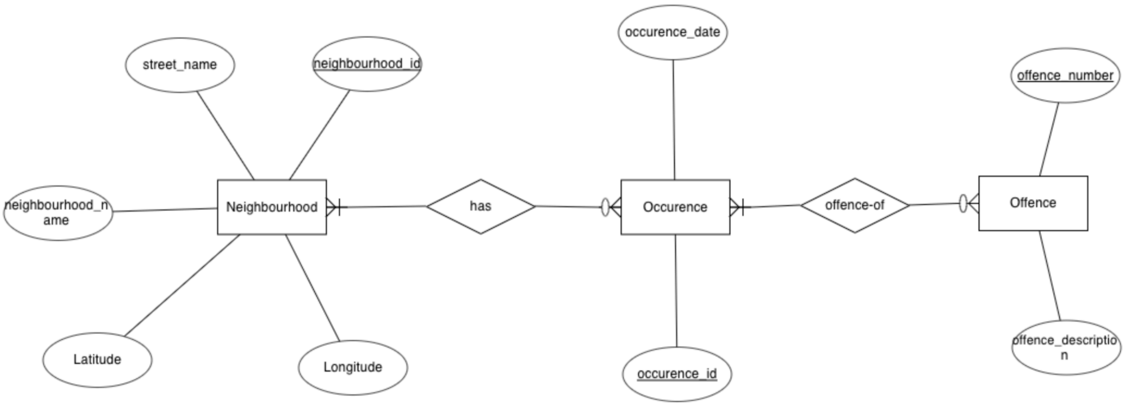
First Normal Form (1NF): Each cell keeps just one piece of information, and each column header only represents one piece of information. Column headers are not duplicated. All the data in a column header refers to the same attribute. The order in which the data is entered into the table is not vital for the table's proper operation.

Second Normal Form (2NF): Tables are already in 1NF and does not have any partial dependency.

Third Normal Form (3NF): Tables are already in 2NF and does not have any transitive dependency.

My table has currently achieved 3NF.

ERD Diagram



created in: https://erdplus.com

A neighbourhood may have zero or many occurrences of crimes, whereas occurrences of crime has happened in one or many neighbourhood.

An occurrence of crime may have one or many of offences which was done by the person who committed the crime. For example, a person can commit two offences at one go. A man assaulted the shop keeper (ASSAULT) and stole some of the items in the shop (SHOPLIFTING).

To build a web application, we need to initialise npm with ‘app.js’
$ npm init

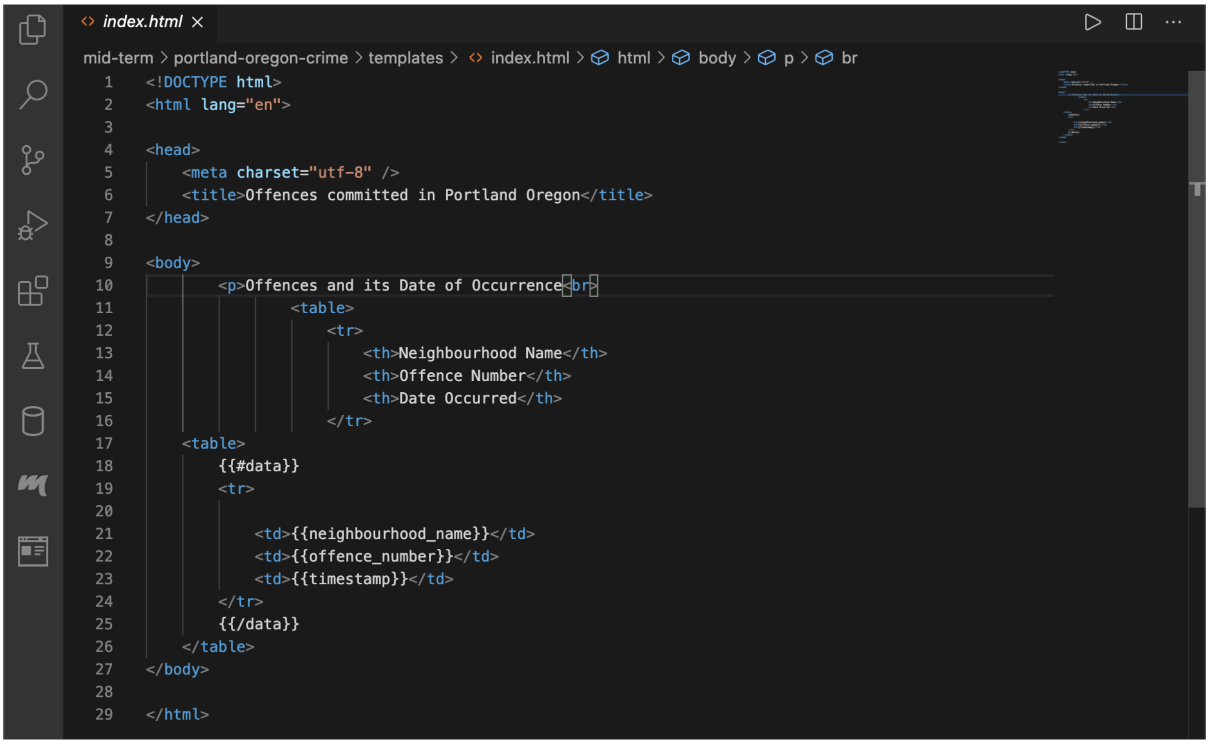
Second step is to install express
$ npm install express

To run our web app, we need to type the following web command
$ node app.js

Not to forget we need to install ‘mysql’
$ npm install nysql
$ npm install body-parser
$ npm install mustache-express
$ npm install yallist

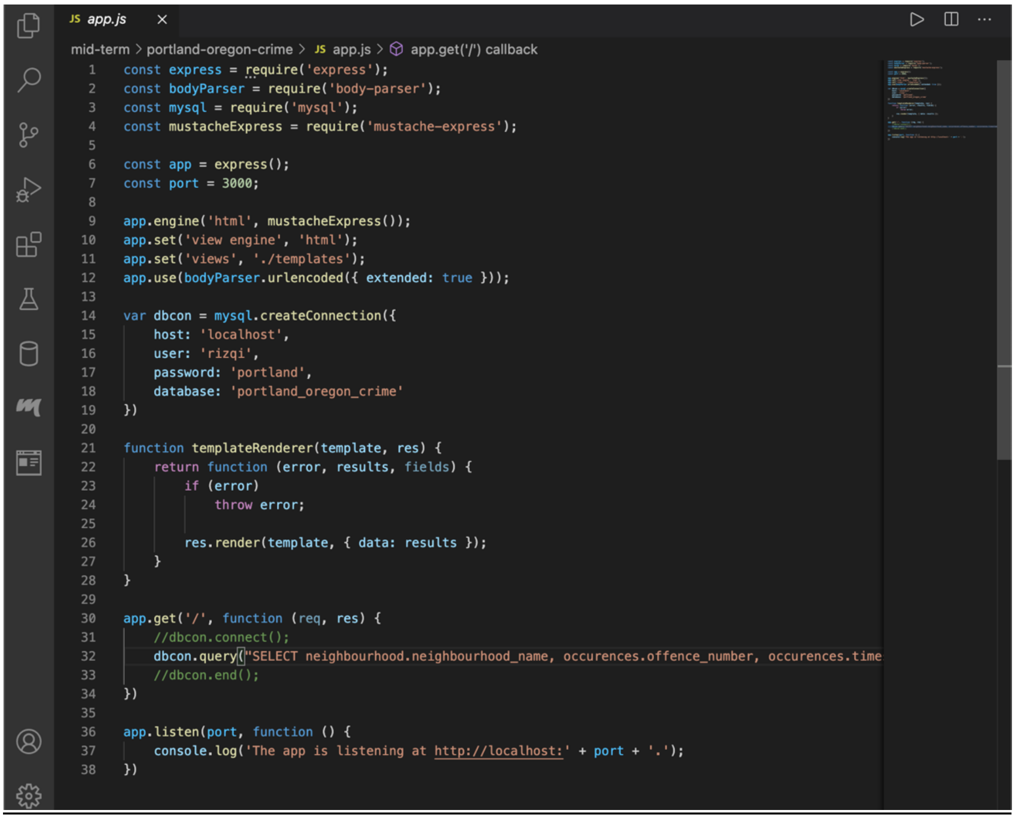
Create a new directory templates inside portland-oregon-crime, and then create index.html
$ touch index.html

my index.html looks like this:


index.html
mid-term > portland-oregon-crime > templates > index.html > html > body > p > br

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <title>Offences committed in Portland Oregon</title>
</head>

<body>
        <p>Offences and its Date of Occurrence<br>
                <table>
                    <tr>
                        <th>Neighbourhood Name</th>
                        <th>Offence Number</th>
                        <th>Date Occurred</th>
                    </tr>
    <table>
        {{#data}}
        <tr>

            <td>{{neighbourhood_name}}</td>
            <td>{{offence_number}}</td>
            <td>{{timestamp}}</td>
        </tr>
        {{/data}}
    </table>
</body>

</html>
```

my app.js file looks like this:

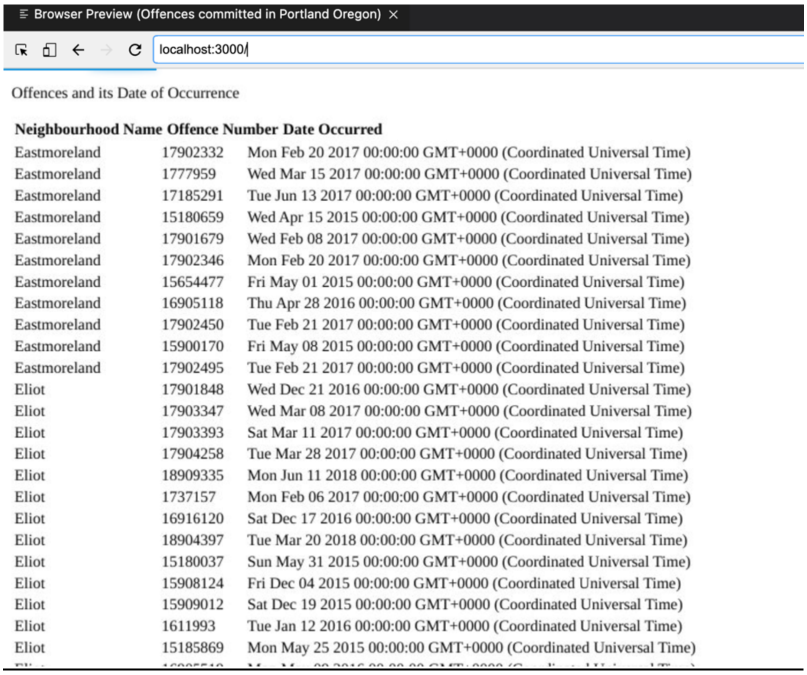

app.js
mid-term > portland-oregon-crime > app.js > app.get('/') callback

```
const express = require('express');
const bodyParser = require('body-parser');
const mysql = require('mysql');
const mustacheExpress = require('mustache-express');

const app = express();
const port = 3000;

app.engine('html', mustacheExpress());
app.set('view engine', 'html');
app.set('views', './templates');
app.use(bodyParser.urlencoded({ extended: true }));

var dbcon = mysql.createConnection({
    host: 'localhost',
    user: 'rizqi',
    password: 'portland',
    database: 'portland_oregon_crime'
})

function templateRenderer(template, res) {
    return function (error, results, fields) {
        if (error)
            throw error;

        res.render(template, { data: results });
    }
}

app.get('/', function (req, res) {
    //dbcon.connect();
    dbcon.query("SELECT neighbourhood.neighbourhood_name, occurences.offence_number, occurences.time
    //dbcon.end();
})

app.listen(port, function () {
    console.log('The app is listening at http://localhost:' + port + '.');
})
```

We can run our web app by typing the command:
$ node app.js

open the web browser on coursera lab and enter following URL:
localhost:3000

This is the result:



Offences and its Date of Occurrence

Neighbourhood Name	Offence Number	Date Occurred
Eastmoreland	17902332	Mon Feb 20 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	1777959	Wed Mar 15 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	17185291	Tue Jun 13 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	15180659	Wed Apr 15 2015 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	17901679	Wed Feb 08 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	17902346	Mon Feb 20 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	15654477	Fri May 01 2015 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	16905118	Thu Apr 28 2016 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	17902450	Tue Feb 21 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	15900170	Fri May 08 2015 00:00:00 GMT+0000 (Coordinated Universal Time)
Eastmoreland	17902495	Tue Feb 21 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	17901848	Wed Dec 21 2016 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	17903347	Wed Mar 08 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	17903393	Sat Mar 11 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	17904258	Tue Mar 28 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	18909335	Mon Jun 11 2018 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	1737157	Mon Feb 06 2017 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	16916120	Sat Dec 17 2016 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	18904397	Tue Mar 20 2018 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	15180037	Sun May 31 2015 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	15908124	Fri Dec 04 2015 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	15909012	Sat Dec 19 2015 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	1611993	Tue Jan 12 2016 00:00:00 GMT+0000 (Coordinated Universal Time)
Eliot	15185869	Mon May 25 2015 00:00:00 GMT+0000 (Coordinated Universal Time)

From the results above, this will provide me with all the information regarding the Neighbourhood name, Offence number and Date occurred.

```
SELECT ‘neighbourhood_name’,
COUNT(‘neighbourhood_name’) AS ‘value_occurrence’
FROM ‘neighbourhood’
GROUP BY ‘neighbourhood_name’
ORDER BY ‘value_occurrence’ DESC
LIMIT 1;
```

Using this syntax above, we can find the top neighbourhood which has the highest number of crime rates. This information will be vital so the police department can deploy more officers around the area in hopes to reduce the number of crimes there.

Shareable lab link:
https://hub.labs.coursera.org:443/connect/sharedlmcdlyfx?forceRefresh=false&path=%2F%3Ffolder%3D%2Fhome%2Fcoder%2Fproject