**Sankeerth Sai Shabad**

**Introduction to Data Science Project-2**

**15/02/2022**

## PURPOSE OF THE PROBLEM-1:

- To perform data pre-processing techniques to analyze the dataset by using Excel and MySQL Workbench. And converting cleansed dataset into XML and JSON.

## Methodology:

- COLLECTION OF DATA: All the data is collected from the dataset with their values from https://github.com/SankeerthShabad/IDS/blob/main/USArrests.csv
- OPERATIONS: Addressing missing values by using the median, plotting graphs for required datasets by using Excel, performing required queries to find min, max, mean, and variance, And sorting, filling missing values by using SQL commands by using MySQL Workbench. And converting cleansed data into XML and JSON
- OBSERVATIONS: Insert plot graphs for required data and divide Urban populations into 4 sub-divisions to analyze the relationship between urban population and crimes and execute SQL queries and convert cleansed data into XML and JSON.

## RESULTS:

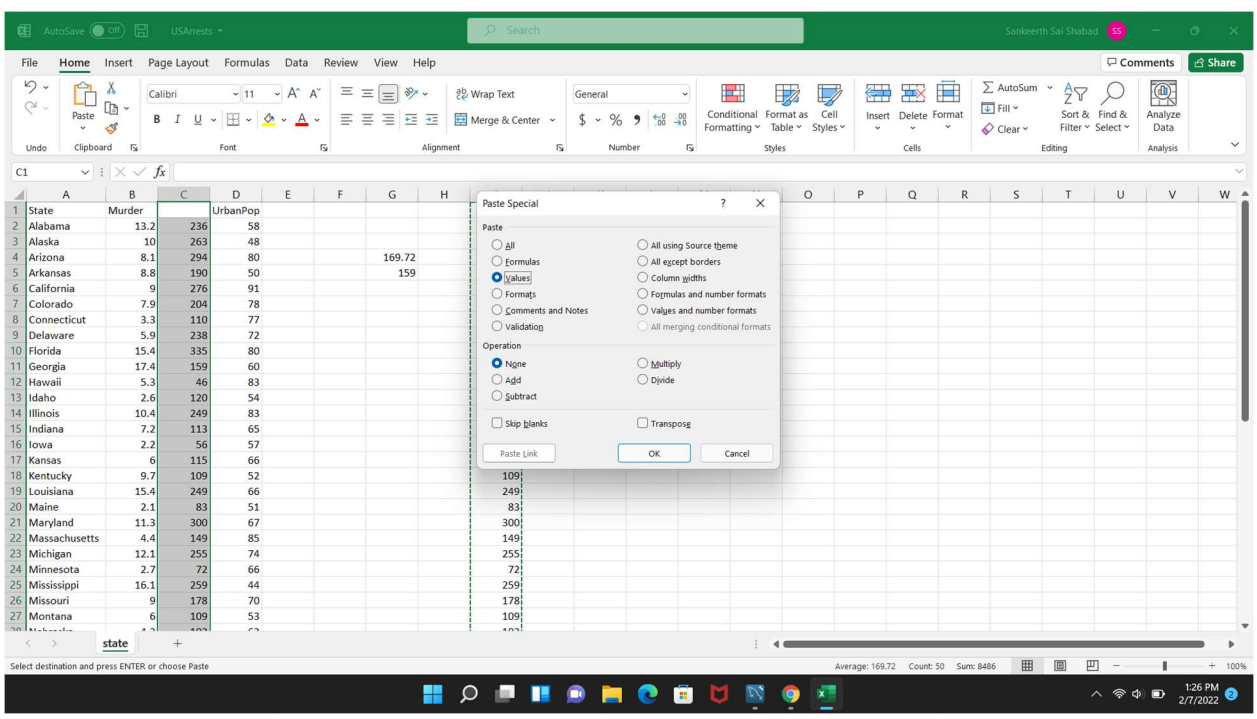



**Fig:** excel sheet showing inserting missing value

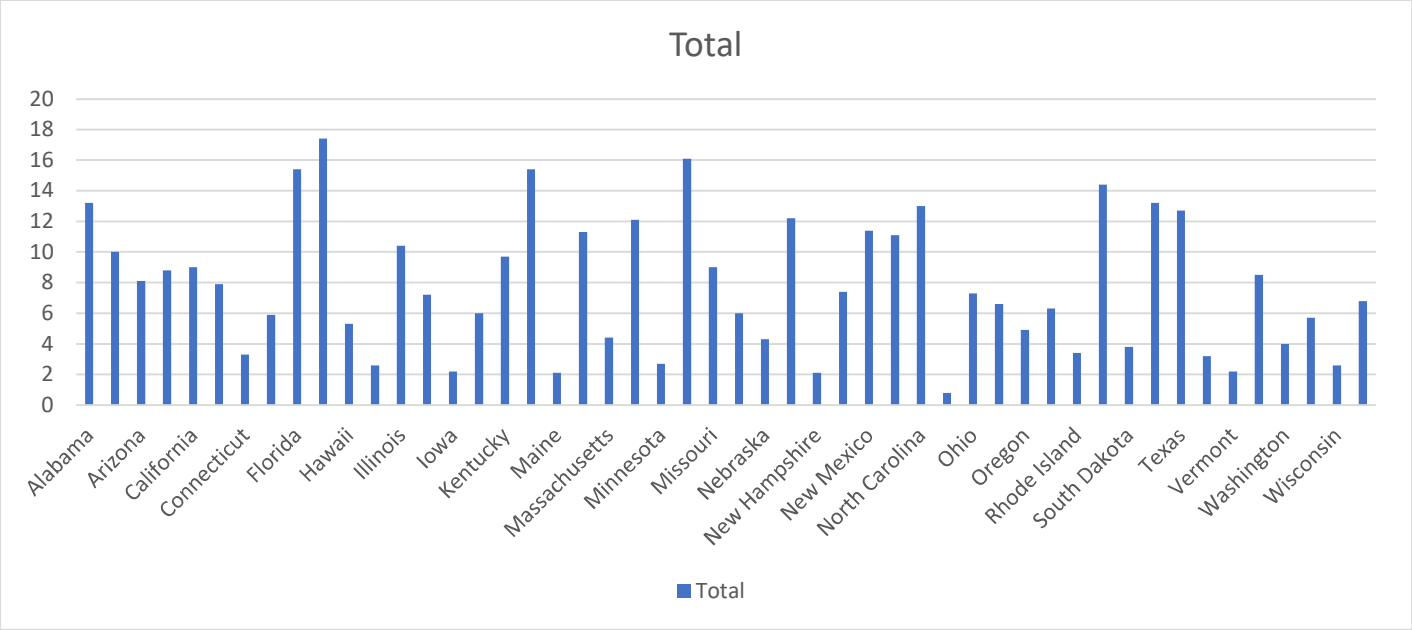



**Fig:** sum of murders by states graph

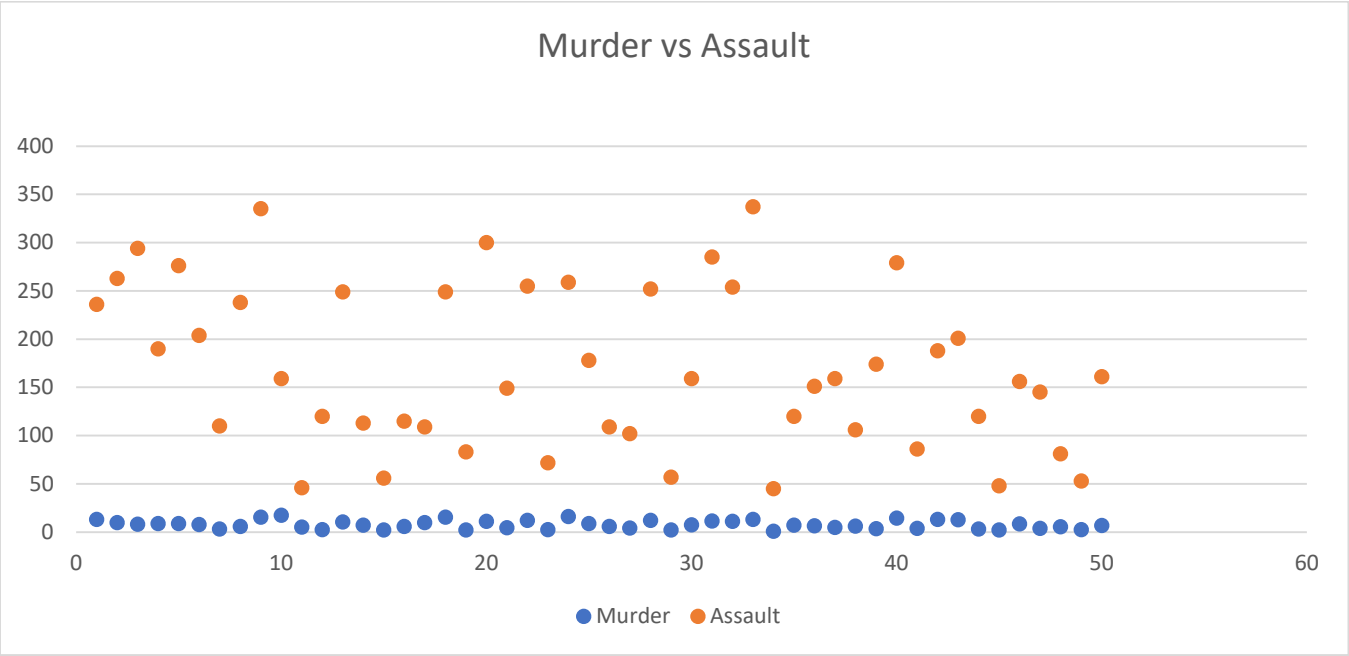



**Fig:** Murder vs Assault scatter graph

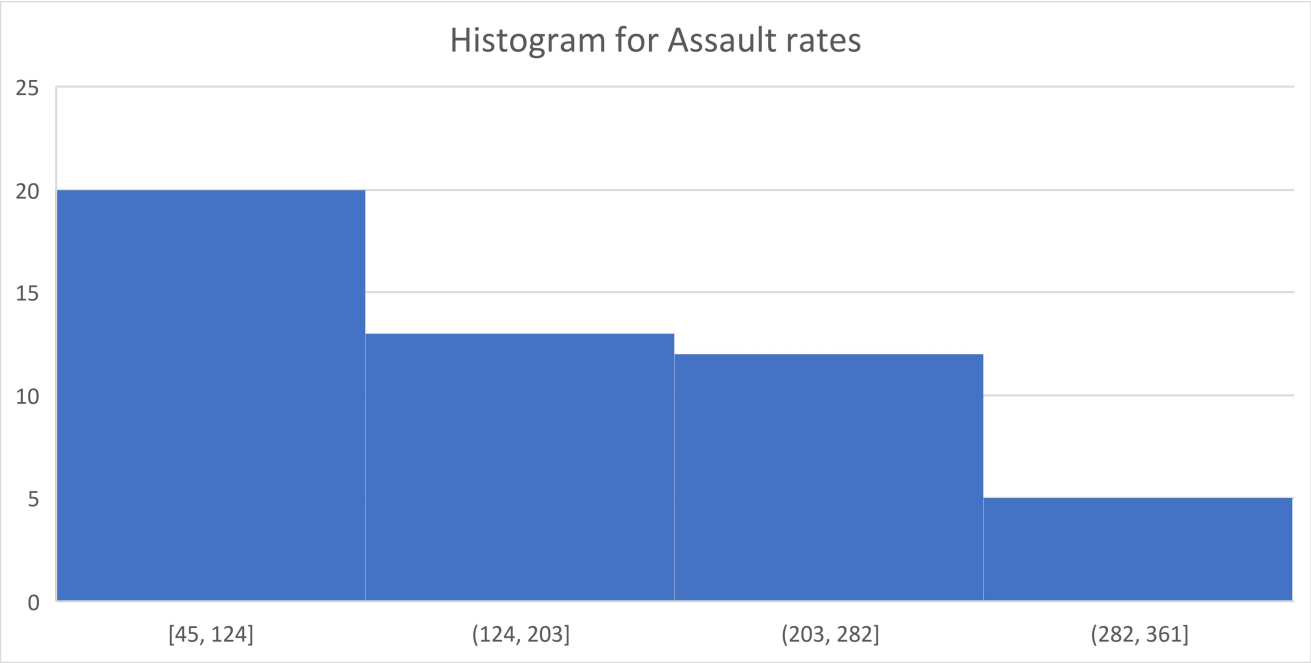



**Fig:** Assault rates Histogram graph

| Row Labels | Count of UrbanPop | Sum of Assault | Sum of Murder |
|---|---|---|---|
| Extra large | 21 | 4150 | 169.8 |
| Large | 12 | 1790 | 93.7 |
| Medium | 9 | 1148 | 59.9 |
| Small | 8 | 1398 | 66 |
| **Grand Total** | **50** | **8486** | **389.4** |

**Fig**: pivot table showing relationship b/w UrbanPop and crimes

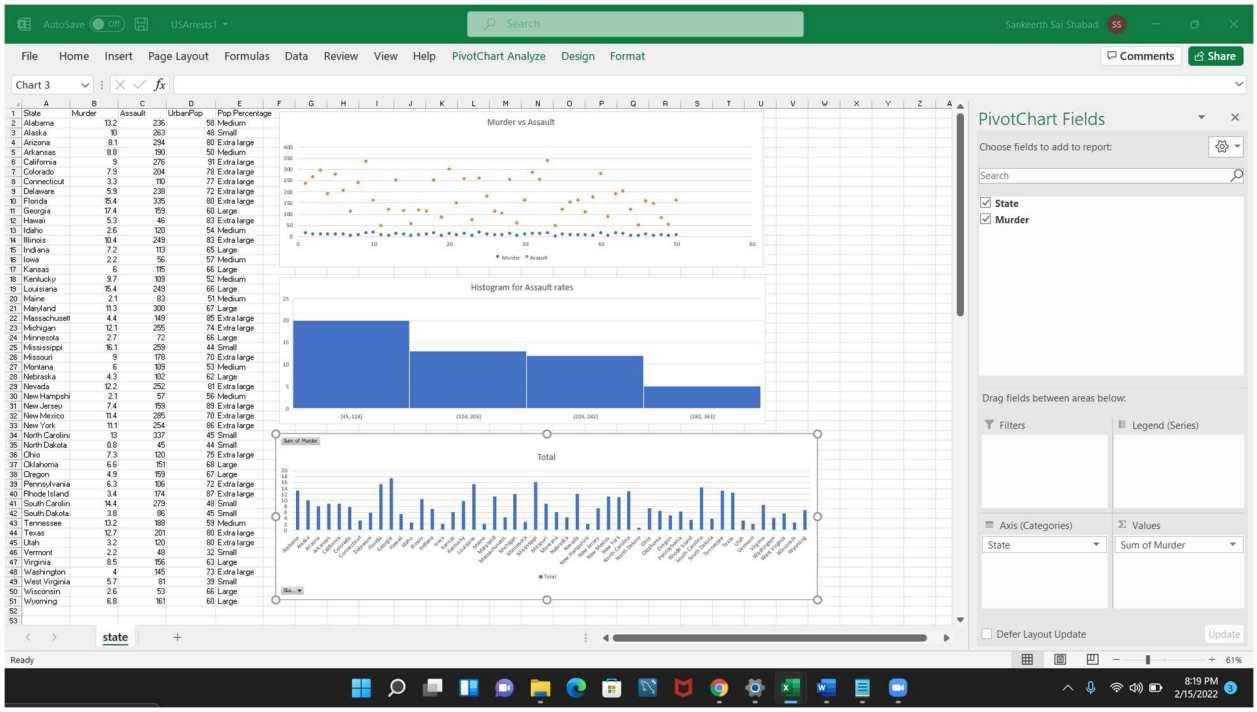



**Fig:** excel sheet showing dataset and plot graphs

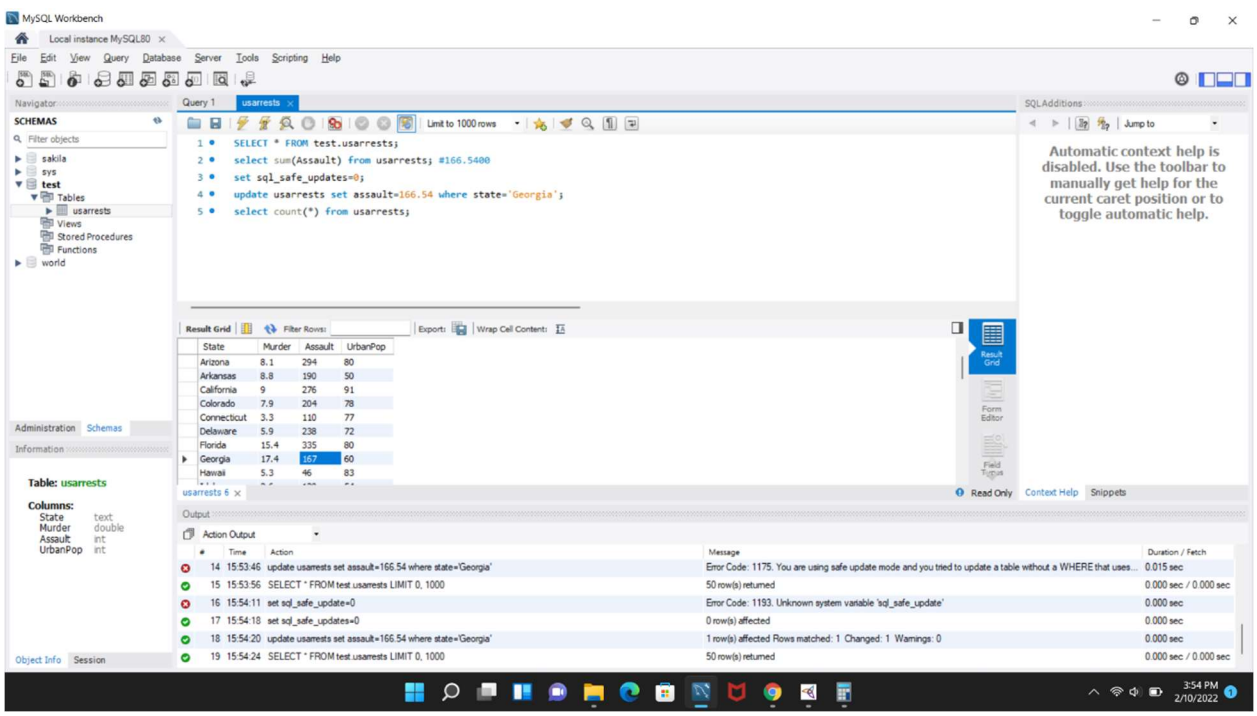



**Fig**: Syntax for updating missing values

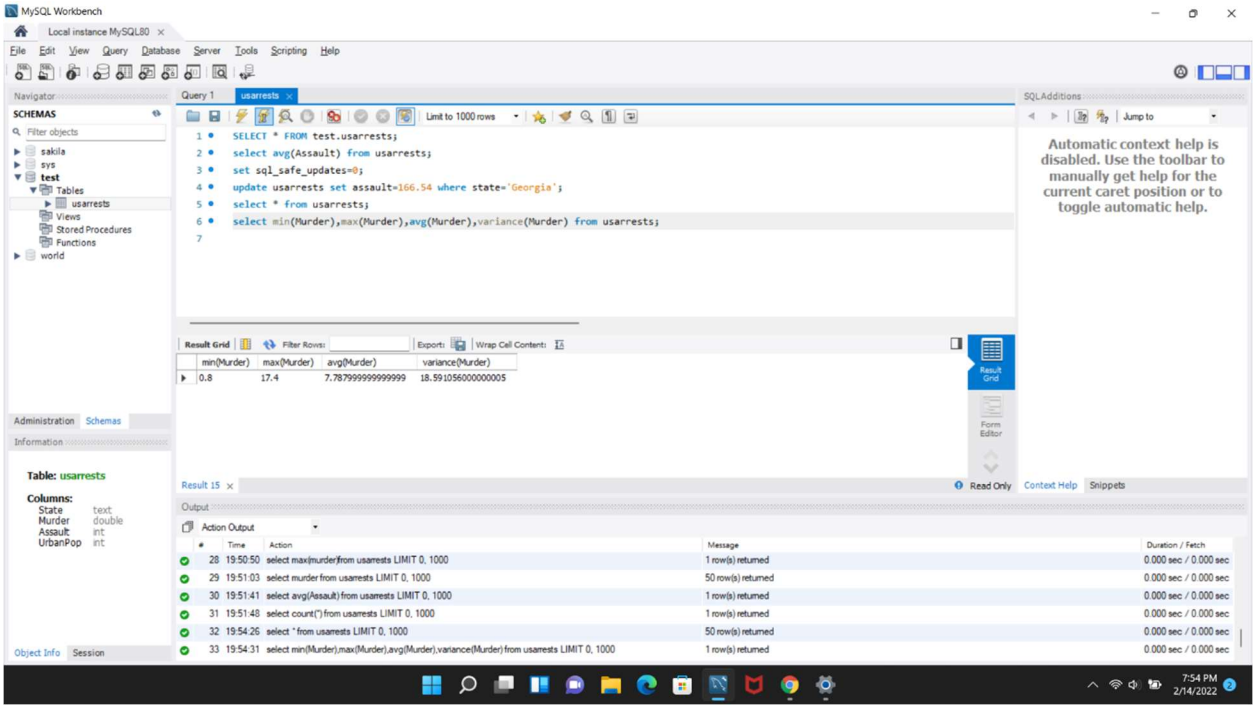



**Fig:** Syntax for a min, max, avg, and variance

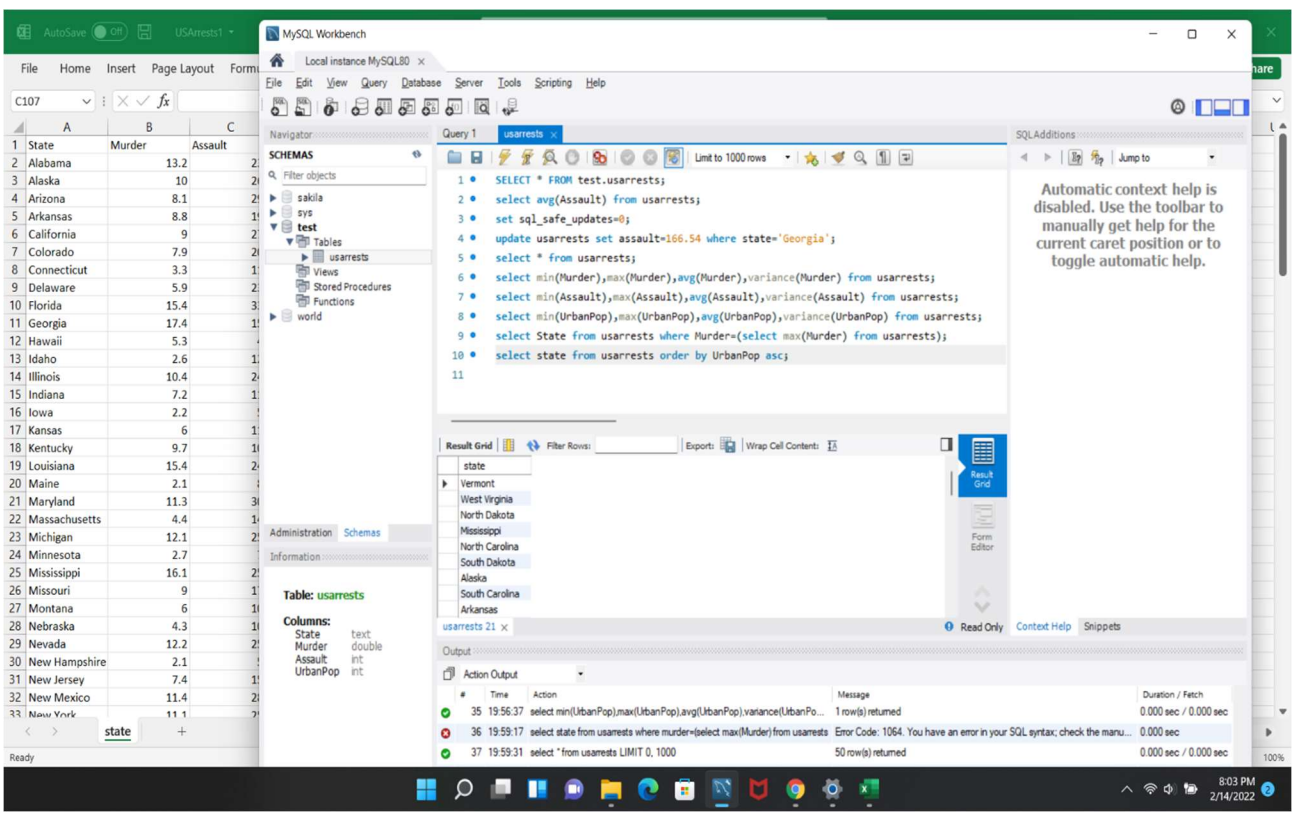
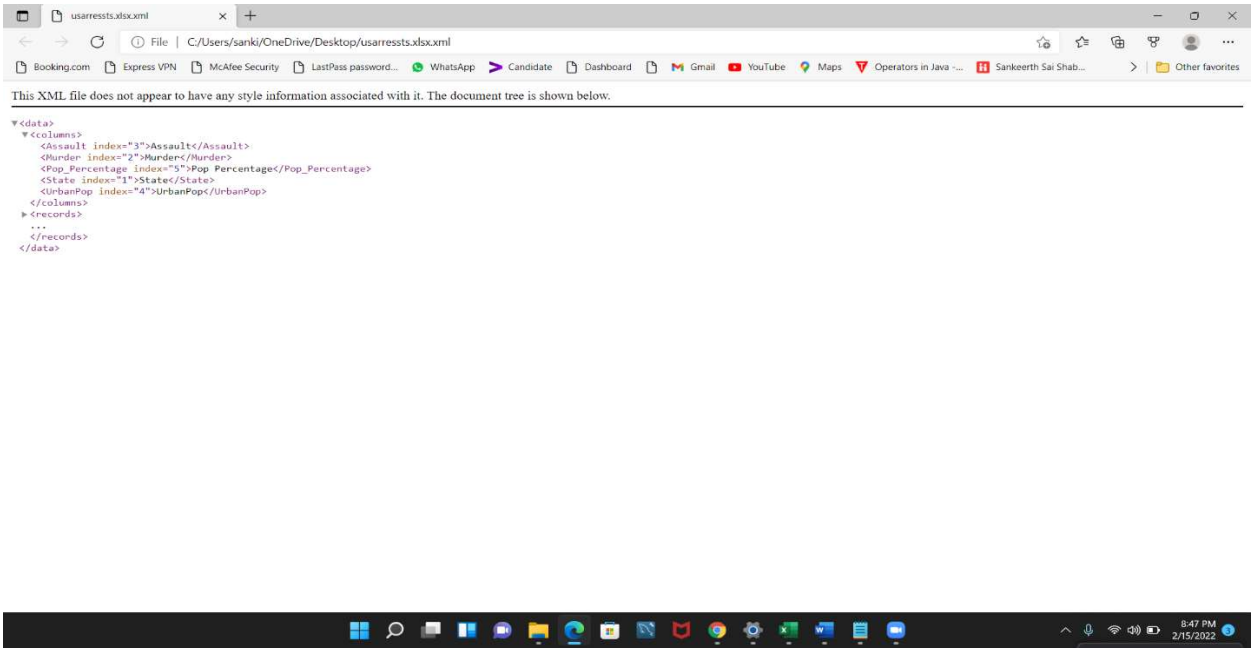



**Fig:** syntax for order of the state by UrbanPop values

**Fig:** Cleansed data after converting into XML



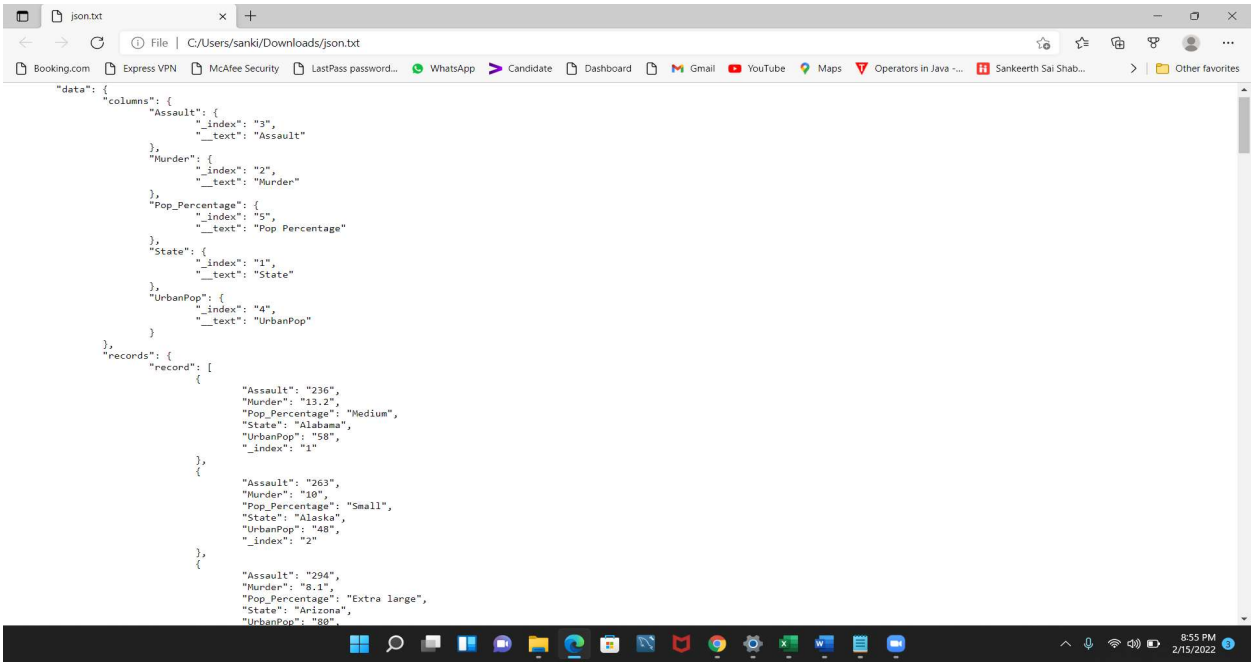




**Fig:** Cleansed data after converting into JSON

**CONCLUSION:** All the required operations like addressing the missing values smoothing noisy data, plotting graphs, and establishing a relation between urbanPop and crimes have been done in Excel and all the SQL commands have been performed in MYSQL Workbench. And cleansed data is converted to XML and JSON.

## PURPOSE OF THE PROBLEM-2:

- To perform data pre-processing techniques to analyze the dataset by using Excel and MySQL Workbench. And converting cleansed dataset into XML and JSON.

## Methodology:

- COLLECTION OF DATA: All the data is collected from the dataset with their values from https://github.com/SankeerthShabad/IDS/blob/main/child_mortality.csv
- OPERATIONS: Addressing missing values by using the median, plotting graphs for required datasets by using Excel, performing required queries to find min, max, mean, and variance, and sorting, filling missing values by using SQL commands by using MySQL Workbench and converting cleansed data into XML and JSON.
- OBSERVATIONS: Plotting required graphs to analyze the relationship between three different mortalities and years and perform SQL queries by using required commands and logics and converting cleansed data into XML and JSON.

## RESULTS:

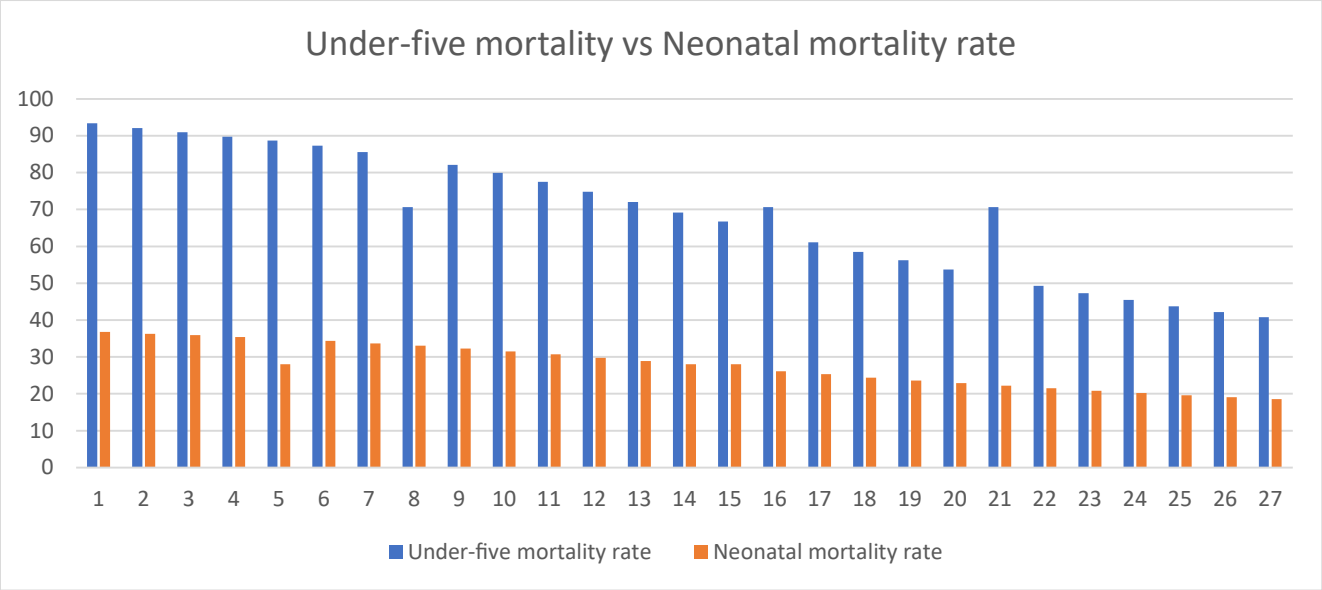



**Fig:** Under-five mortality vs Neonatal mortality rate plot graph

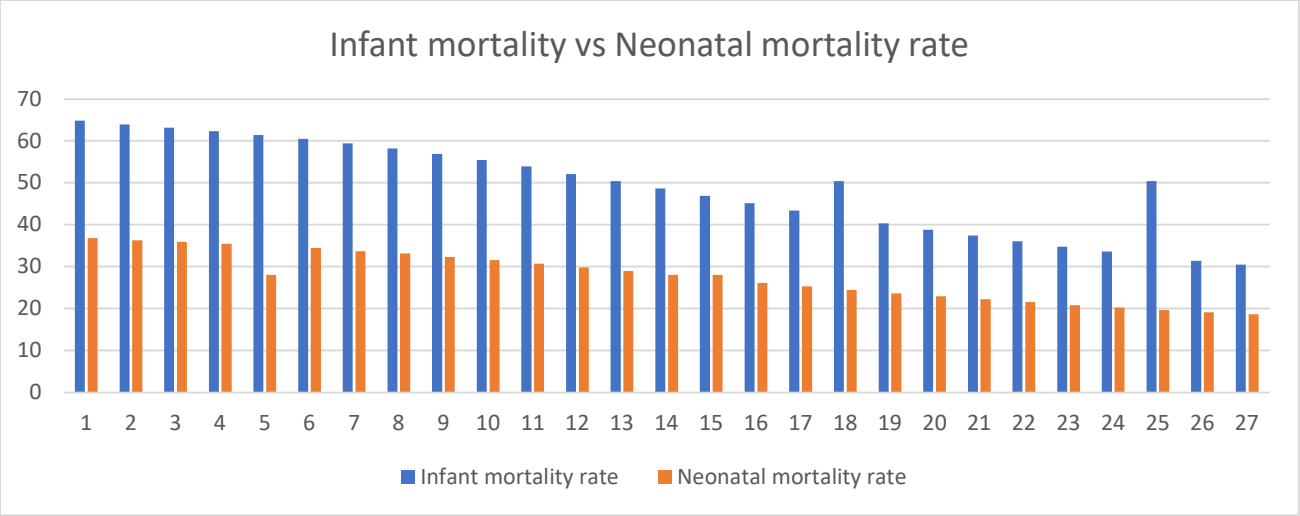



**Fig**: Infant mortality vs Neonatal mortality rate plot graph

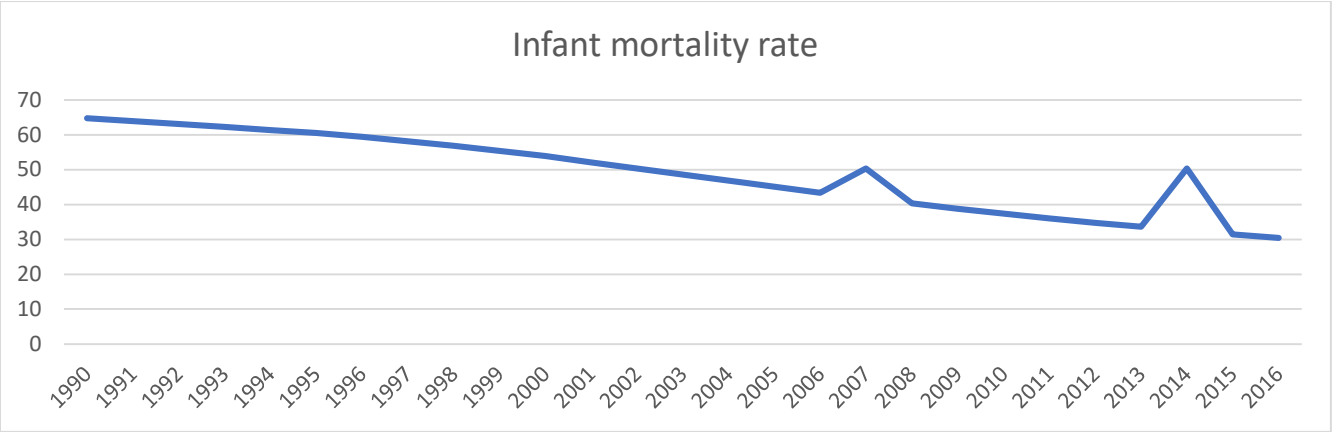



**Fig:** Infant mortality rate as per years plot graph

| Row Labels | Count of Under-five mortality rate |
|---|---|
| high | 10 |
| medium | 9 |
| very high | 8 |
| **Grand Total** | **27** |

| Row Labels | Count of Infant mortality rate |
|---|---|
| high | 6 |
| low | 7 |
| medium | 14 |
| **Grand Total** | **27** |

| Row Labels | Sum of Neonatal mortality rate |
|---|---|
| low | 689.8 |
| very low | 57.3 |
| **Grand Total** | **747.1** |

**Fig**: Converting mortality rates into five Likert scale value

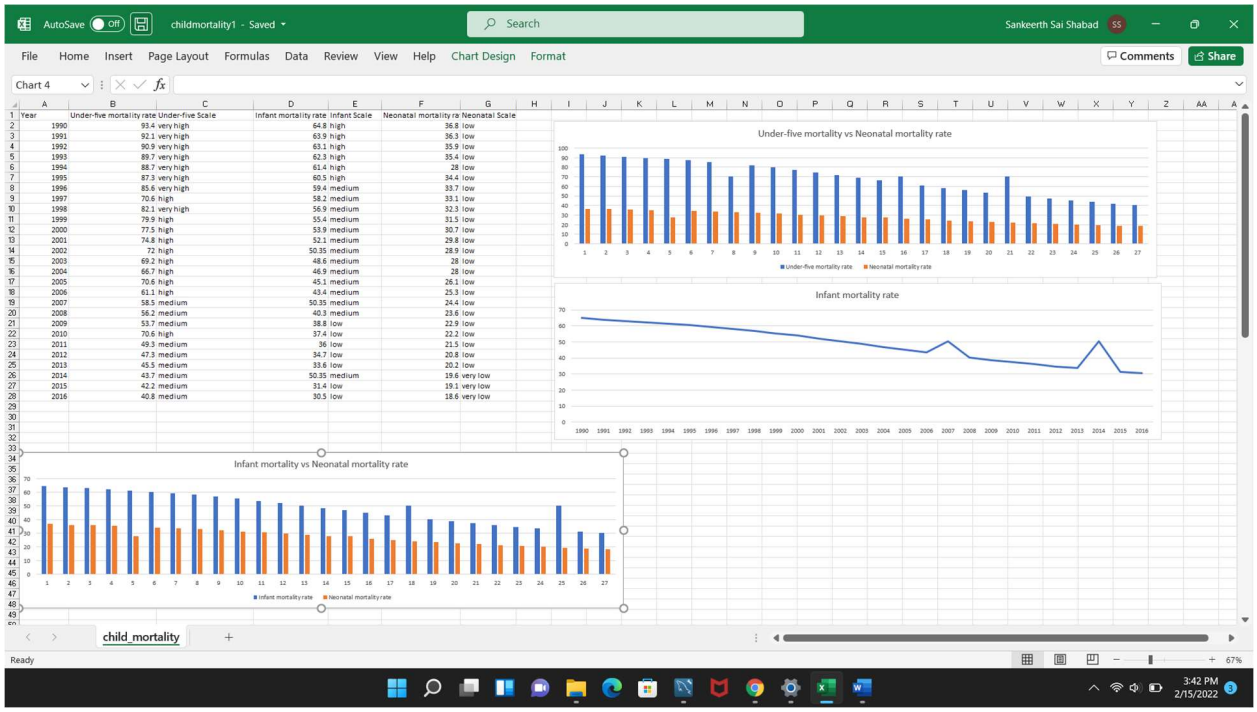



**Fig:** Excel sheet showing dataset and plot graphs

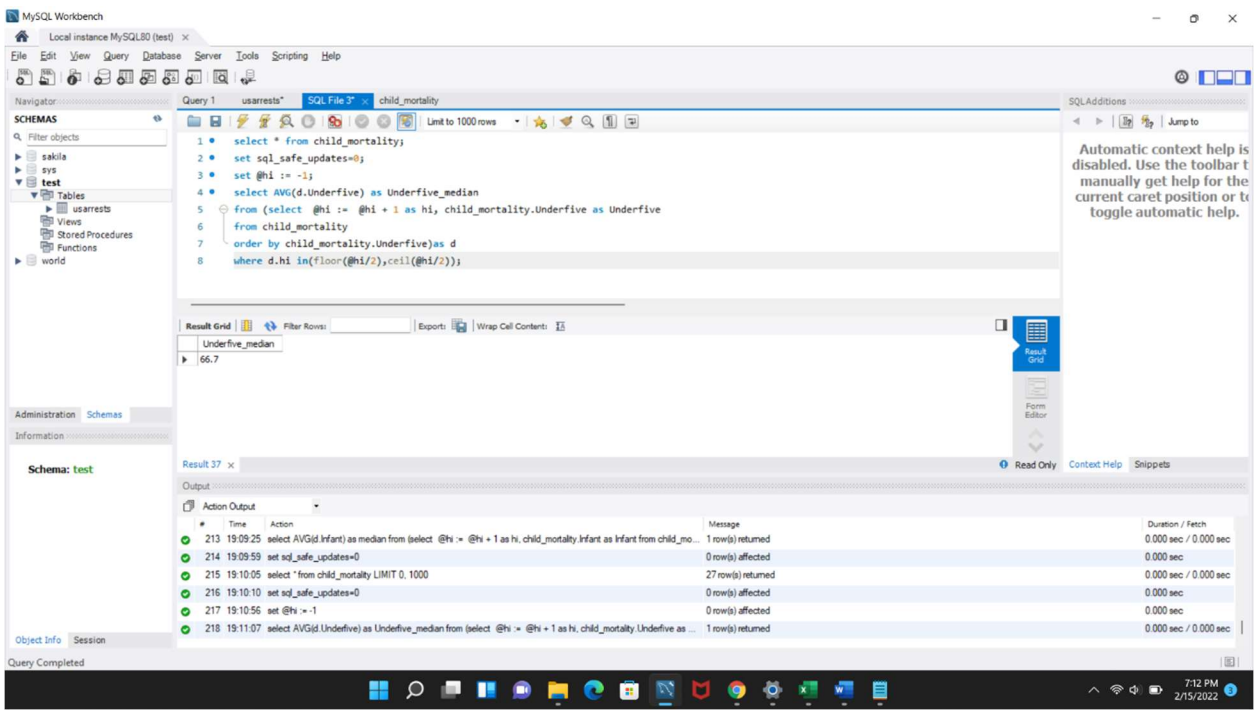



**Fig:** Median command using MySQL workbench

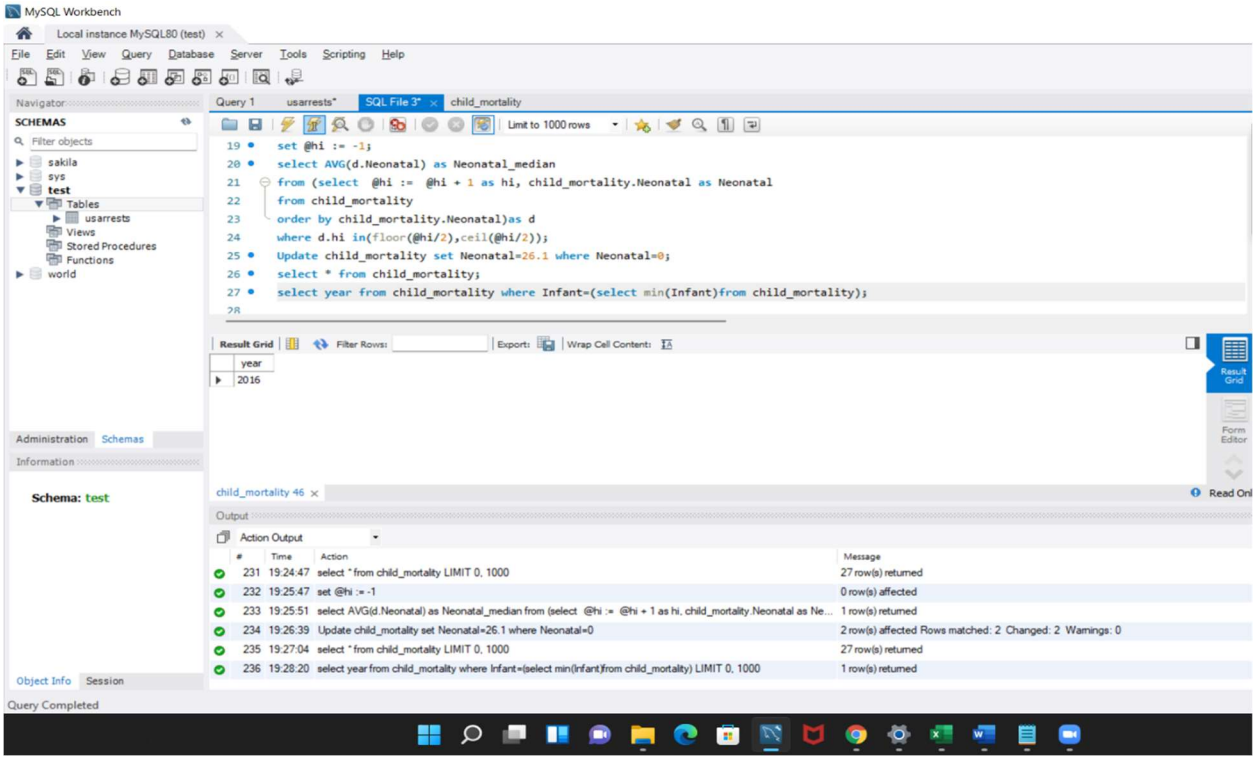



**Fig:** Table displaying the year of minimum mortality rate

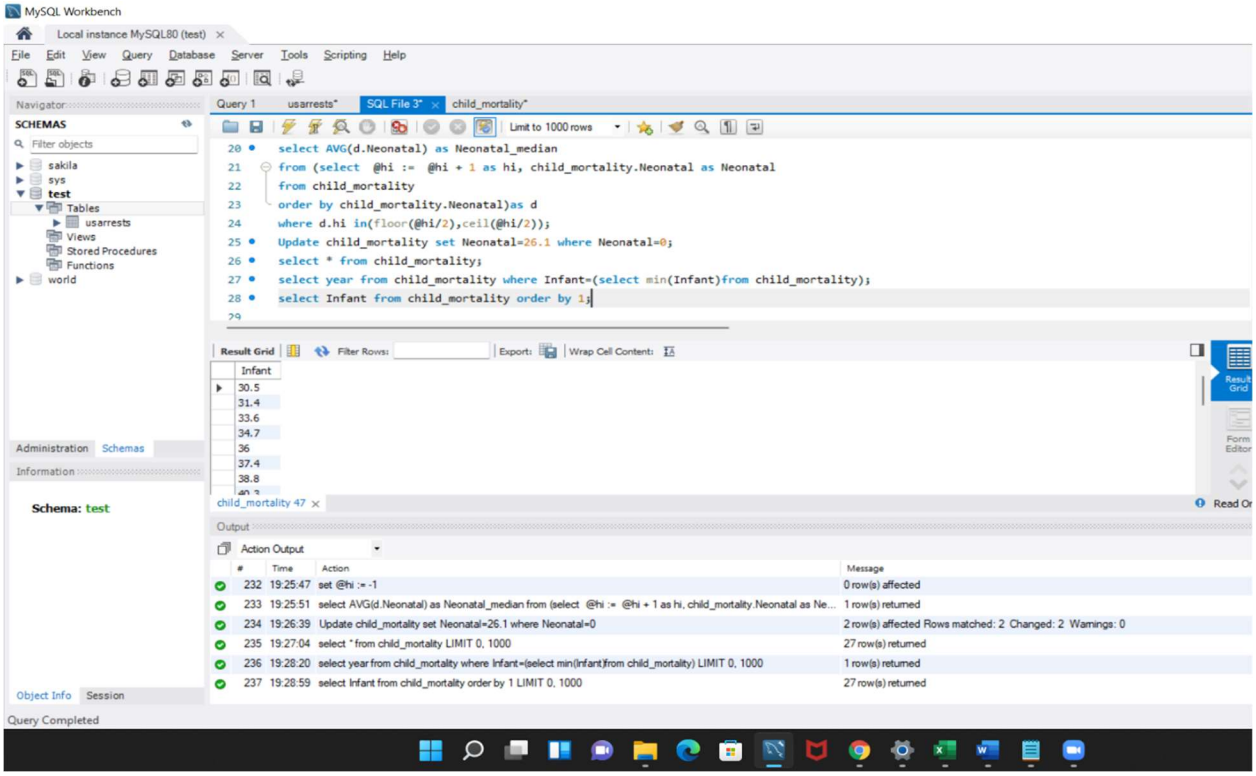



**Fig**: Table showing Infant values in sorting order

MySQL Workbench

Local instance MySQL80 (test)

File Edit View Query Database Server Tools Scripting Help

Query 1 usarrests* SQL File 3* child_mortality*

```
where d.hi in(floor(@hi/2),ceil(@hi/2));
Update child_mortality set Neonatal=26.1 where Neonatal=0;
select * from child_mortality;
select year from child_mortality where Infant=(select min(Infant)from child_mortality);
select Infant from child_mortality order by 1;
select avg(Infant),min(Infant),max(Infant),variance(Infant),stddev_pop(Infant) from child_mortality;
select avg(Neonatal),min(Neonatal),max(Neonatal),variance(Neonatal),stddev_pop(Neonatal) from child_mortality;
select avg(Underfive),min(Underfive),max(Underfive),variance(Underfive),stddev_pop(Underfive) from child_mortality;
```

Result Grid

| avg(Infant) | min(Infant) | max(Infant) | variance(Infant) | stddev_pop(Infant) |
|---|---|---|---|---|
| 48.86296296296296 | 30.5 | 64.8 | 114.96159122085056 | 10.722014326648262 |

Result 48

Output

| # | Time | Action | Message |
|---|---|---|---|
| 233 | 19:25:51 | select AVG(d.Neonatal) as Neonatal_median from (select @hi := @hi + 1 as hi, child_mortality.Neonatal as Ne... | 1 row(s) returned |
| 234 | 19:26:39 | Update child_mortality set Neonatal=26.1 where Neonatal=0 | 2 row(s) affected Rows matched: 2 Changed: 2 Warnings: 0 |
| 235 | 19:27:04 | select * from child_mortality LIMIT 0, 1000 | 27 row(s) returned |
| 236 | 19:28:20 | select year from child_mortality where Infant=(select min(Infant)from child_mortality) LIMIT 0, 1000 | 1 row(s) returned |
| 237 | 19:28:59 | select Infant from child_mortality order by 1 LIMIT 0, 1000 | 27 row(s) returned |
| 238 | 19:29:28 | select avg(Infant),min(Infant),max(Infant),variance(Infant),stddev_pop(Infant) from child_mortality LIMIT 0, 1000 | 1 row(s) returned |

Schema: test

Query Completed

MySQL Workbench

Local instance MySQL80 (test)

File Edit View Query Database Server Tools Scripting Help

Query 1 usarrests* SQL File 3* child_mortality*

```
order by child_mortality.Neonatal)as d
where d.hi in(floor(@hi/2),ceil(@hi/2));
Update child_mortality set Neonatal=26.1 where Neonatal=0;
select * from child_mortality;
select year from child_mortality where Infant=(select min(Infant)from child_mortality);
select Infant from child_mortality order by 1;
select avg(Infant),min(Infant),max(Infant),variance(Infant),stddev_pop(Infant) from child_mortality;
select avg(Neonatal),min(Neonatal),max(Neonatal),variance(Neonatal),stddev_pop(Neonatal) from child_mortality;
select avg(Underfive),min(Underfive),max(Underfive),variance(Underfive),stddev_pop(Underfive) from child_mortality;
```

Result Grid

| avg(Underfive) | min(Underfive) | max(Underfive) | variance(Underfive) | stddev_pop(Underfive) |
|---|---|---|---|---|
| 68.45555555555555 | 40.8 | 93.4 | 281.0928395061728 | 16.765823555858294 |

Result 49

Output

| # | Time | Action | Message |
|---|---|---|---|
| 234 | 19:26:39 | Update child_mortality set Neonatal=26.1 where Neonatal=0 | 2 row(s) affected Rows matched: 2 Changed: 2 Warnings: 0 |
| 235 | 19:27:04 | select * from child_mortality LIMIT 0, 1000 | 27 row(s) returned |
| 236 | 19:28:20 | select year from child_mortality where Infant=(select min(Infant)from child_mortality) LIMIT 0, 1000 | 1 row(s) returned |
| 237 | 19:28:59 | select Infant from child_mortality order by 1 LIMIT 0, 1000 | 27 row(s) returned |
| 238 | 19:29:28 | select avg(Infant),min(Infant),max(Infant),variance(Infant),stddev_pop(Infant) from child_mortality LIMIT 0, 1000 | 1 row(s) returned |
| 239 | 19:29:55 | select avg(Underfive),min(Underfive),max(Underfive),variance(Underfive),stddev_pop(Underfive) from child_mo... | 1 row(s) returned |

Schema: test

Query Completed

**Fig:** SQL displaying a table of avg, max, variance, standard deviation

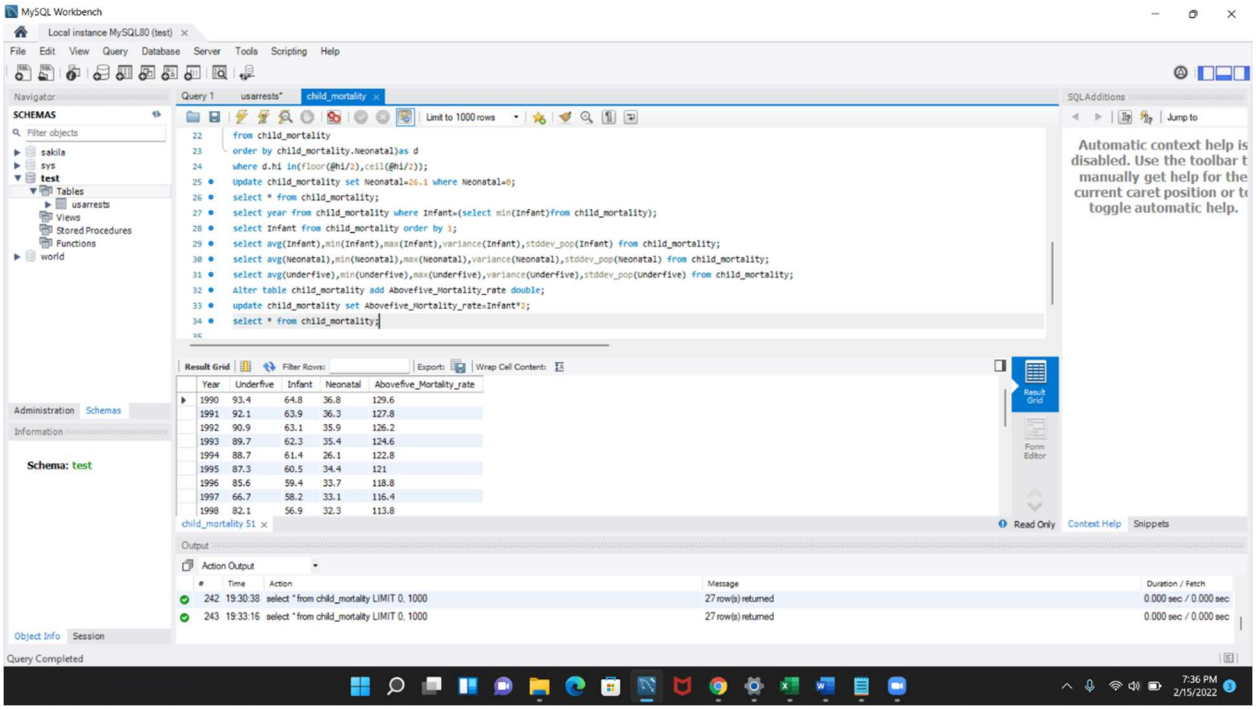



Fig: Child_mortality table after updating values

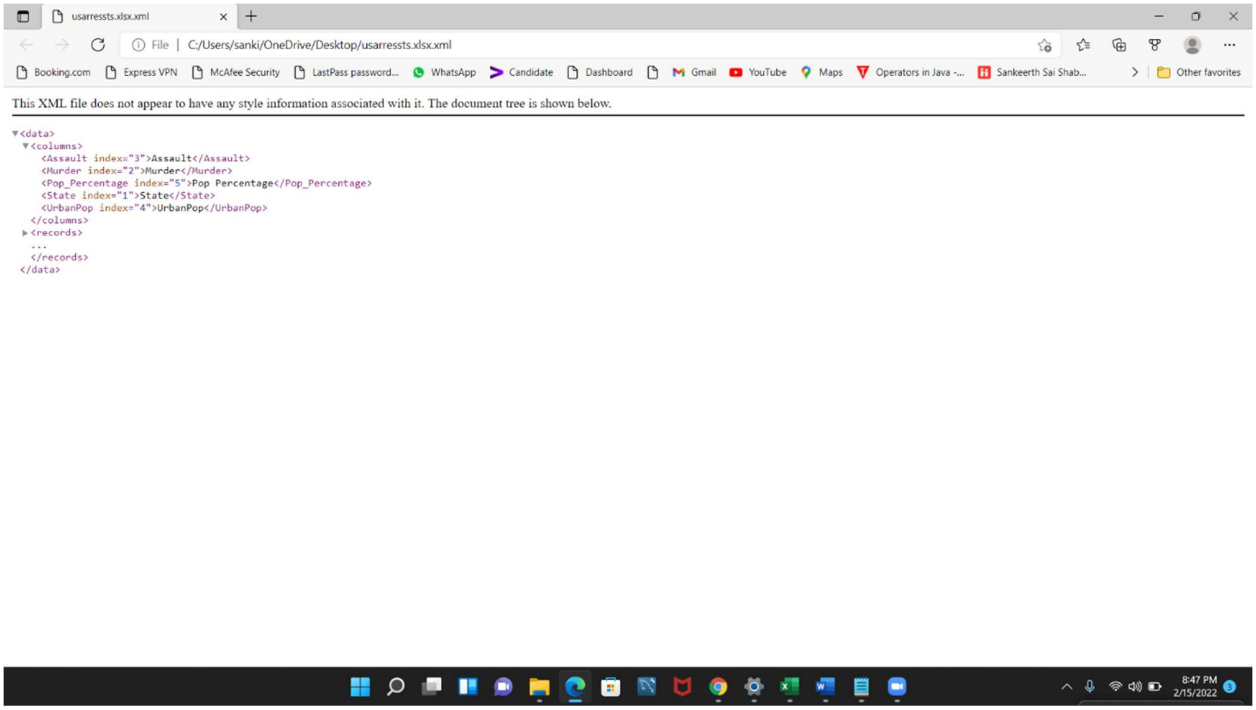
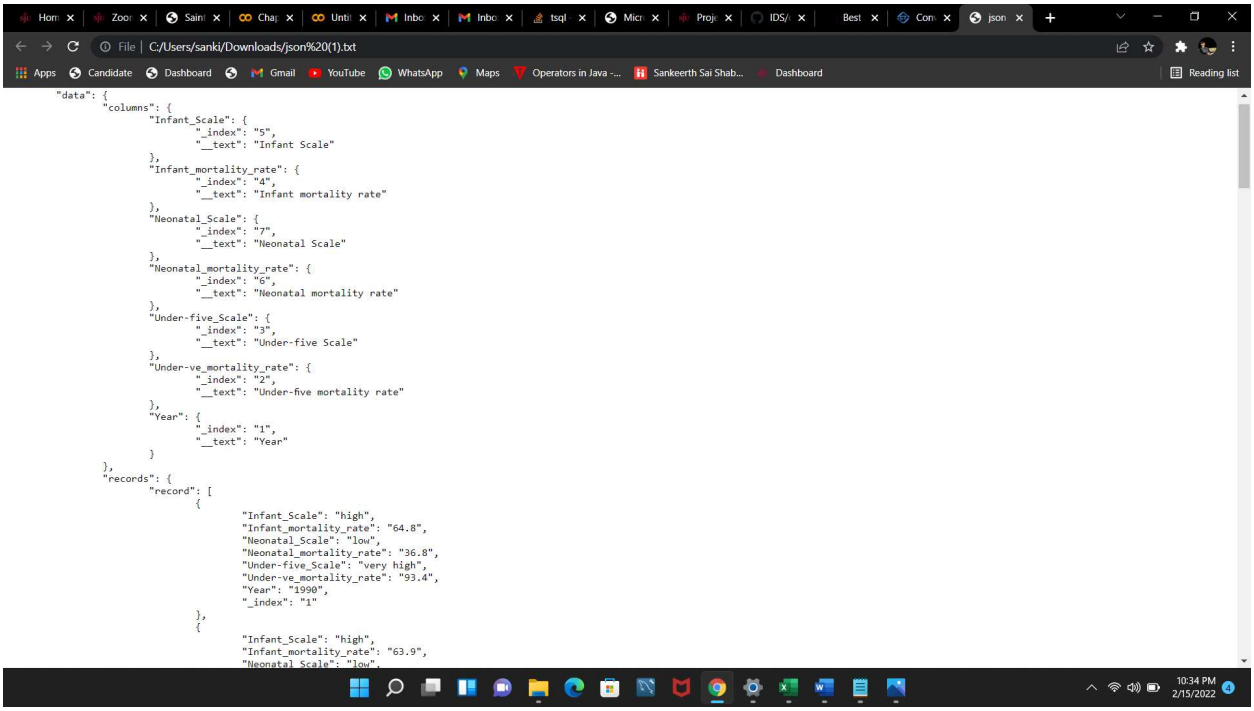



**Fig:** Cleansed data after converting into XML




```
"data": {
        "columns": {
                "Infant_Scale": {
                        "_index": "5",
                        "__text": "Infant Scale"
                },
                "Infant_mortality_rate": {
                        "_index": "4",
                        "__text": "Infant mortality rate"
                },
                "Neonatal_Scale": {
                        "_index": "7",
                        "__text": "Neonatal Scale"
                },
                "Neonatal_mortality_rate": {
                        "_index": "6",
                        "__text": "Neonatal mortality rate"
                },
                "Under-five_Scale": {
                        "_index": "3",
                        "__text": "Under-five Scale"
                },
                "Under-ve_mortality_rate": {
                        "_index": "2",
                        "__text": "Under-five mortality rate"
                },
                "Year": {
                        "_index": "1",
                        "__text": "Year"
                }
        },
        "records": {
                "record": [
                        {
                                "Infant_Scale": "high",
                                "Infant_mortality_rate": "64.8",
                                "Neonatal_Scale": "low",
                                "Neonatal_mortality_rate": "36.8",
                                "Under-five_Scale": "very high",
                                "Under-ve_mortality_rate": "93.4",
                                "Year": "1990",
                                "_index": "1"
                        },
                        {
                                "Infant_Scale": "high",
                                "Infant_mortality_rate": "63.9",
                                "Neonatal_Scale": "low",
```


**Fig:** Cleansed data after converting into JSON

**CONCLUSION:** All the required operations like addressing the missing values smoothing noisy data, plotting graphs in Excel, and all the SQL commands have been performed in MYSQL Workbench. And cleansed data is converted to XML and JSON.