

## ##### Problem 1 #####

The dataset USArrests.csv contains statistics in arrests per 100,000 residents for assault and murder, in each of the 50 US states, in 1973. Also given is the percentage of the population living in urban areas:<https://github.com/bforoura/IDS/tree/main/HW2>

Use the pre-processing techniques at your disposal to prepare the dataset for analysis.

Excel

1. Address all the missing values.
2. Look for outliers and smooth noisy data, if any.
3. Plot Murder rates for all 50 states
4. Plot histogram of assaults
5. Plot Murder rate vs. Assault rate.
6. Prepare the dataset to establish a relation between an urban population category and a crime type.  
Hint: Convert the urban population percentage into categories, for example, small (<50%), medium (<60%), large (<70%), and extra-large (70% and above) urban population.

MySQL

1. Import the original CSV file into MySQL and create the table USArrests.
2. Use SQL to replace all missing values in a column by the average.  
Hint: Use the Update command.
3. Find min, max, mean, and variance of all numeric attributes in SQL.
4. Use SQL to answer the following questions:
  - Which state has the maximum murder rate?
  - List of states in ascending order of urban population percentages.
  - How many states have higher murder rates than Arizona? List those states.

Ans:

	A	B	C	D	E	F	G	H	I
1	State	Murder	Assault	UrbanPop					
2	Alabama	13.2	236	58			169.939		
3	Alaska	10	263	48					
4	Arizona	8.1	294	80					
5	Arkansas	8.8	190	50					
6	California	9	276	91					
7	Colorado	7.9	204	78					
8	Connecticut	3.3	110	77					
9	Delaware	5.9	238	72					
10	Florida	15.4	335	80					
11	Georgia	17.4	169.94	60					
12	Hawaii	5.3	46	83					
13	Idaho	2.6	120	54					
14	Illinois	10.4	249	83					
15	Indiana	7.2	113	65					
16	Iowa	2.2	56	57					
17	Kansas	6	115	66					
18	Kentucky	9.7	109	52					
19	Louisiana	15.4	249	66					
20	Maine	2.1	83	51					
21	Maryland	11.3	300	67					
22	Massachusetts	4.4	149	85					
23	Michigan	12.1	255	74					
24	Minnesota	2.7	72	66					
25	Mississippi	16.1	259	44					
26	Missouri	9	178	70					
27	Montana	6	109	53					
28	Nebraska	4.3	102	62					
29	Nevada	12.2	252	81					
30	New Hampshire	2.1	57	56					
31	New Jersey	7.4	159	89					
32	New Mexico	11.4	285	70					
33	New York	11.1	254	86					
34	North Carolina	13	337	45					
35	North Dakota	0.8	45	44					
36	Ohio	7.3	120	75					
37	Oklahoma	6.6	151	68					
38	Oregon	4.9	159	67					
39	Pennsylvania	6.3	106	72					
40	Rhode Island	3.4	174	87					
41	South Carolina	14.4	279	48					
42	South Dakota	3.8	86	45					

categorical data      MurderAndAssault      state\_wise murder rate      assault hist  
Sheet 1 of 5      PageStyle\_in

Fig 1.1.1: Address missing value using =AVERAGE(C2:C10,C12:C51)

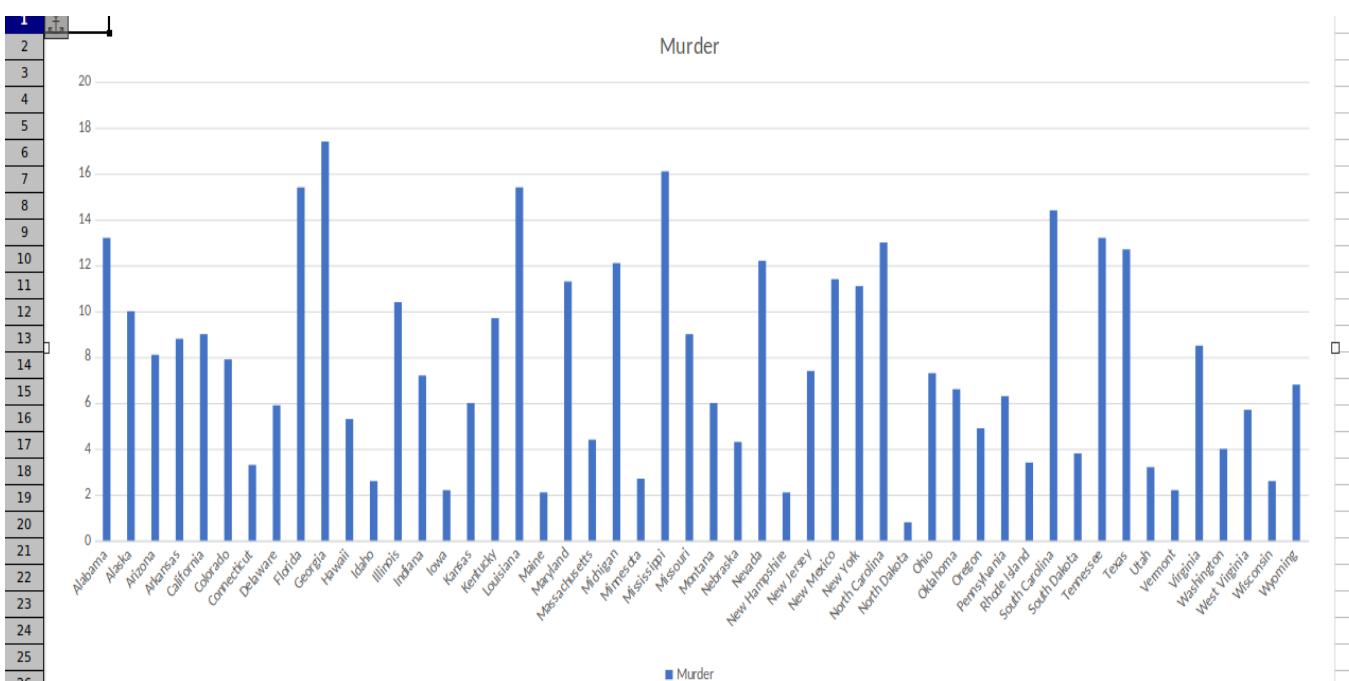


Fig 1.1.2: Plot Murder rates for all 50 states

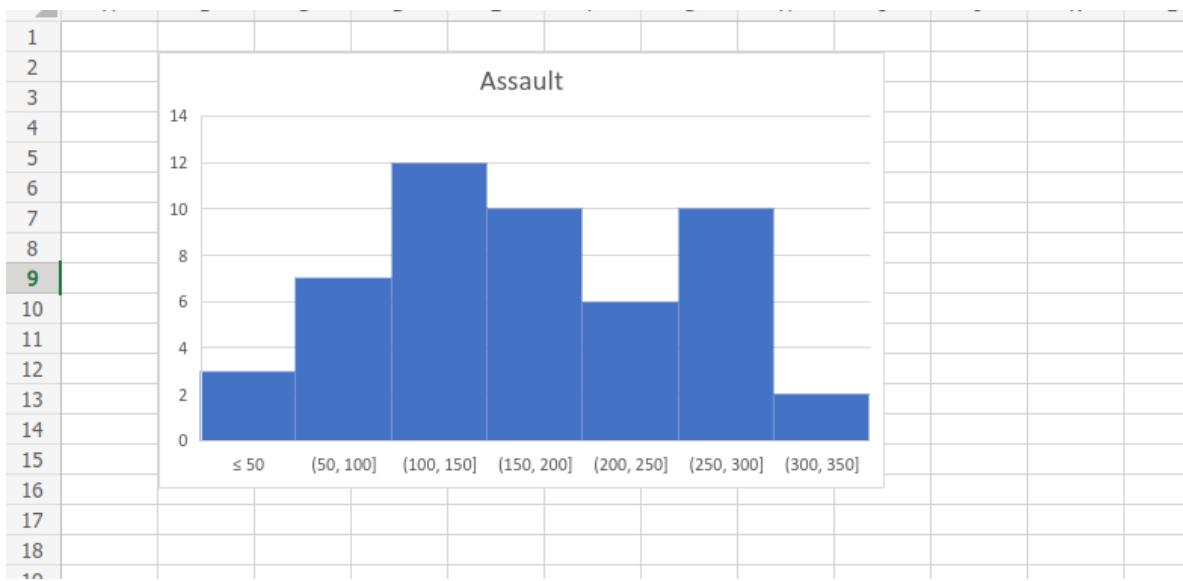


Fig 1.1.3: Plot histogram of assaults

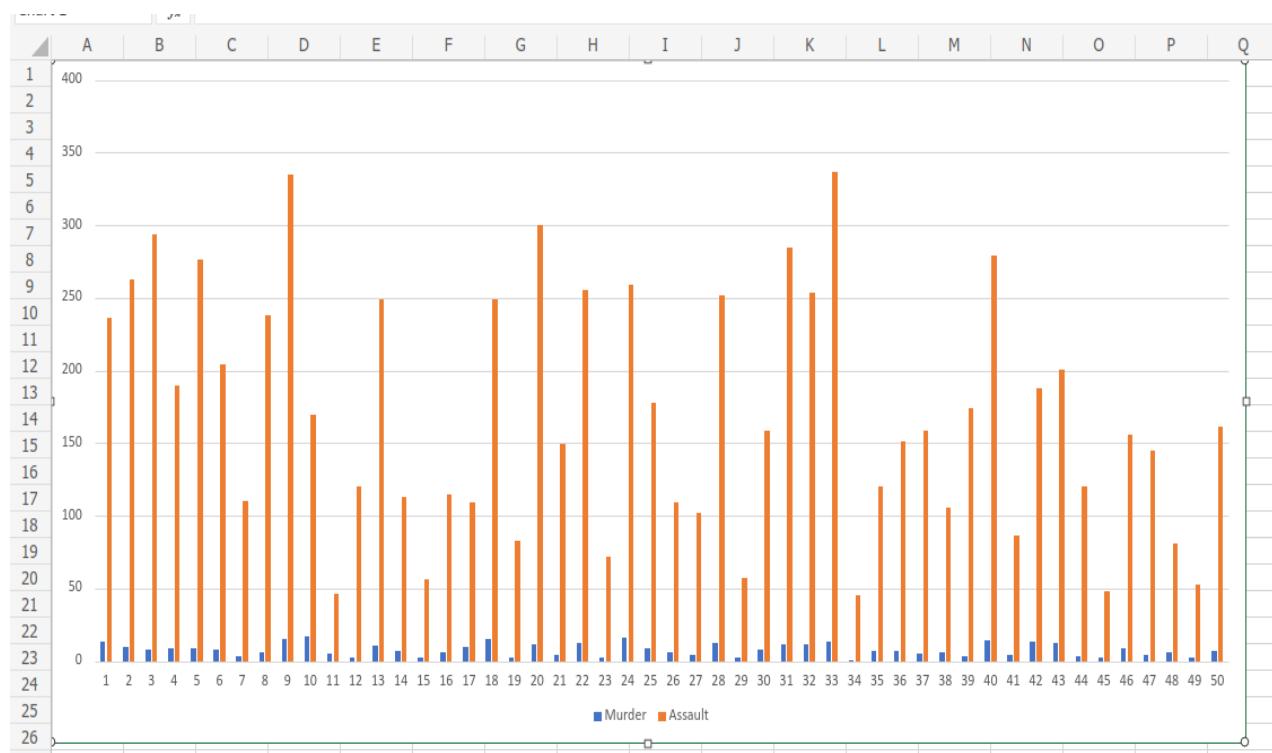


Fig 1.1.4: Plot Murder rate vs. Assault rate.

	A	B	C	D	E	F	G
1	State	Murder	Assault	UrbanPop	Urban_cat	crime_type	
2	Alabama	13.2	236	58	medium	most	
3	Alaska	10	263	48	small	most	
4	Arizona	8.1	294	80	Extra large	most	
5	Arkansas	8.8	190	50	medium	more	
6	California	9	276	91	Extra large	most	
7	Colorado	7.9	204	78	Extra large	most	
8	Connecticut	3.3	110	77	Extra large	less	
9	Delaware	5.9	238	72	Extra large	most	
10	Florida	15.4	335	80	Extra large	most	
11	Georgia	17.4	169.9388	60	large	most	
12	Hawaii	5.3	46	83	Extra large	less	
13	Idaho	2.6	120	54	medium	less	
14	Illinois	10.4	249	83	Extra large	most	
15	Indiana	7.2	113	65	large	less	
16	Iowa	2.2	56	57	medium	least	
17	Kansas	6	115	66	large	less	
18	Kentucky	9.7	109	52	medium	less	
19	Louisiana	15.4	249	66	large	most	
20	Maine	2.1	83	51	medium	least	
21	Maryland	11.3	300	67	large	most	
22	Massachusetts	4.4	149	85	Extra large	less	
23	Michigan	12.1	255	74	Extra large	most	
24	Minnesota	2.7	72	66	large	least	
25	Mississippi	16.1	259	44	small	most	
26	Missouri	9	178	70	Extra large	more	
27	Montana	6	109	53	medium	less	
28	Nebraska	4.3	102	62	large	less	
29	Nevada	12.2	252	81	Extra large	most	
30	New Hampshire	2.1	57	56	medium	least	
31	New Jersey	7.4	159	89	Extra large	more	
32	New Mexico	11.4	285	70	Extra large	most	
33	New York	11.1	254	86	Extra large	most	
34	North Carolina	13	337	45	small	most	
35	North Dakota	0.8	45	44	small	least	
36	Ohio	7.3	120	75	Extra large	less	

Fig 1.1.5: urban population category and a crime type  
#Formula:

1. =IF(D2<50, "small", IF(D2<60, "medium", IF(D2<70, "large", "Extra large")))
2. =IF(B2<5 AND C2<100, "least", IF(B2<10 AND C2<150, "less", IF(B2<15 AND C2<200, "more", "most")))

The screenshot shows the MySQL Workbench interface with two tabs open: 'Query 1' and 'USAreests'. The 'USAreests' tab is active, displaying the results of the 'desc USAreests;' command. The results are presented in a table with the following columns: #, Field, Type, Null, Key, Default, and Extra. The data shows four columns: State, Murder, Assault, and UrbanPop.

#	Field	Type	Null	Key	Default	Extra
1	State	text	YES		NULL	
2	Murder	double	YES		NULL	
3	Assault	int(11)	YES		NULL	
4	UrbanPop	int(11)	YES		NULL	

Fig 1.2.2: mysql describe table USAreests.

Command: desc USAreests;

The screenshot shows the MySQL Workbench interface with three tabs open: 'Query 1', 'USAreests', and 'USAreests'. The 'USAreests' tab is active, displaying the results of the 'SELECT \* FROM USAreests;' command. The results are presented in a table with columns: #, State, Murder, Assault, and UrbanPop. The data lists 50 US states with their corresponding values for Murder, Assault, and UrbanPop.

#	State	Murder	Assault	UrbanPop
1	Alabama	13.2	236	58
2	Alaska	10	263	48
3	Arizona	8.1	294	80
4	Arkansas	8.8	190	50
5	California	9	276	91
6	Colorado	7.9	204	78
7	Connecticut	3.3	110	77
8	Delaware	5.9	238	72
9	Florida	15.4	335	80
10	Georgia	17.4	0	60
11	Hawaii	5.3	46	83
12	Idaho	2.6	120	54
13	Illinois	10.4	249	83
14	Indiana	7.2	113	65
15	Iowa	2.2	56	57
16	Kansas	6	115	66
17	Kentucky	9.7	109	52
18	Louisiana	15.4	249	66
19	Maine	2.1	83	51
20	Maryland	11.3	300	67
21	Massach...	4.4	149	85
22	Michigan	12.1	255	74
23	Minnesota	2.7	72	66

Fig 1.2.2: mysql import data and select projection.

Sql: select \* from USAreests;

The screenshot shows the MySQL Workbench interface. At the top, there are three tabs: 'Query1', 'USArrests', and 'USArrests'. Below the tabs is a toolbar with various icons. A dropdown menu says 'Limit to 1000 rows'. The main area contains a code editor with the following SQL script:

```

1 • use mytestdb;
2 • SELECT * FROM mytestdb.USArrests;
3 • SET SQL_SAFE_UPDATES = 0;
4 • update mytestdb.USArrests set Assault=(select avg(Assault) from USArrests) where State='Georgia';
5

```

Below the code editor is an 'Action Output' table:

Action	Message	Duration / Fetch
16 05:12:42 SELECT * FROM mytestdb.USArrests LIMIT 0, 1000	50 row(s) returned	0.0012 sec / 0.0000...
17 05:12:45 update USArrests set Assault=(select avg(Assault) from U...	Error Code: 1175. You are using safe update mode ... To disable safe mode, toggle the option in Preferen...	0.00034 sec
18 05:13:06 update mytestdb.USArrests set Assault=(select avg(Assau...	Error Code: 1175. You are using safe update mode ... To disable safe mode, toggle the option in Preferen...	0.0011 sec
19 05:13:32 SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.00045 sec
20 05:13:35 update mytestdb.USArrests set Assault=(select avg(Assau...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.0023 sec

Fig 1.2.3 : address missing command.

Sql: update USArrests set assault=(select avg(assault) from USArrests) where state='Georgia';

The screenshot shows the MySQL Workbench interface. At the top, there are three tabs: 'Query1', 'USArrests', and 'USArrests'. Below the tabs is a toolbar with various icons. A dropdown menu says 'Limit to 1000 rows'. The main area contains a code editor with the same SQL script as Fig 1.2.3:

```

1 • use mytestdb;
2 • SELECT * FROM mytestdb.USArrests;
3 • SET SQL_SAFE_UPDATES = 0;
4 • update mytestdb.USArrests set Assault=(select avg(Assault) from USArrests) where State='Georgia';
5 • SELECT * FROM mytestdb.USArrests;

```

Below the code editor is a 'Result Grid' table:

#	State	Murder	Assault	UrbanPop
1	Alabama	13.2	236	58
2	Alaska	10	263	48
3	Arizona	8.1	294	80
4	Arkansas	8.8	190	50
5	California	9	276	91
6	Colorado	7.9	204	78
7	Connecticut	3.3	110	77
8	Delaware	5.9	238	72
9	Florida	15.4	335	80
10	Georgia	17.4	170	60
11	Hawaii	5.3	46	83
12	Idaho	2.6	120	54

Below the result grid is an 'Action Output' table:

Action	Message	Duration / Fetch
17 05:12:45 update USArrests set Assault=(select avg(Assault) from U...	Error Code: 1175. You are using safe update mode ... To disable safe mode, toggle the option in Preferen...	0.00034 sec
18 05:13:06 update mytestdb.USArrests set Assault=(select avg(Assau...	Error Code: 1175. You are using safe update mode ... To disable safe mode, toggle the option in Preferen...	0.0011 sec
19 05:13:32 SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.00045 sec
20 05:13:35 update mytestdb.USArrests set Assault=(select avg(Assau...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.0023 sec
21 05:15:12 SELECT * FROM mytestdb.USArrests LIMIT 0, 1000	50 row(s) returned	0.0013 sec / 0.0001...

Sql:Fig 1.2.4: Again selection.

Sql: select \* from USArrests;

```

1 • use mytestdb;
2 • SELECT * FROM mytestdb.USArrests;
3 • SET SQL_SAFE_UPDATES = 0;
4 • update mytestdb.USArrests set Assault=(select avg(Assault) from USArrests) where State='Georgia';
5 • SELECT * FROM mytestdb.USArrests;
6 • select min(Murder) as murder_min, max(Murder) as murder_max, avg(Murder) as murder_avg, variance(Murder) as murder_var,
7 min(Assault) as Assault_min, max(Assault) as Assault_max, avg(Assault) as Assault_avg, variance(Assault) as Assault_var,
8 min(UrbanPop) as UrbanPop_min, max(UrbanPop) as UrbanPop_max, avg(UrbanPop) as UrbanPop_avg, variance(UrbanPop) as UrbanPop_var from USArrests;

```

**Result Grid**

#	murder_min	murder_max	murder_avg	murder_var	Assault_min	Assault_max	Assault_avg	Assault_var	UrbanPop_min	UrbanPop_max	UrbanPop_avg	UrbanPop_var
1	0.8	17.4	7.787999999999999	18.591056000000005	45	337	169.9400	6773.2164	32	91	65.5400	205.3284

**Action Output**

#	Time	Action	Message	Duration / Fetch
30	05:20:40	SELECT min(murder), max(murder), avg(murder), variance(...)	1 row(s) returned	0.00033 sec / 0.000...
31	05:20:46	select min(Assault), avg(Assault),variance(A...)	1 row(s) returned	0.00045 sec / 0.00...
32	05:20:46	select min(UrbanPop), max(UrbanPop),avg(UrbanPop),vari...	1 row(s) returned	0.00039 sec / 0.00...
33	05:21:03	select min(UrbanPop), max(UrbanPop),avg(UrbanPop),vari...	1 row(s) returned	0.00031 sec / 0.00...
34	05:21:08	select min(UrbanPop), max(UrbanPop),avg(UrbanPop),vari...	1 row(s) returned	0.00054 sec / 0.00...
35	05:21:34	select min(Murder), max(Murder), avg(Murder), variance(...)	1 row(s) returned	0.00079 sec / 0.00...
36	05:21:50	select min(Murder) as murder_min, max(Murder), avg(Mur...)	1 row(s) returned	0.00058 sec / 0.00...
37	05:23:48	select min(Murder) as murder_min, max(Murder) as murd...	1 row(s) returned	0.00071 sec / 0.00...

Fig: 1.2.5: min max, avg

sql: select min(Murder) as murder\_min, max(Murder) as murder\_max, avg(Murder) as murder\_avg, variance(Murder) as murder\_var, min(Assault) as Assault\_min, max(Assault) as Assault\_max, avg(Assault) as Assault\_avg, variance(Assault) as Assault\_var, min(UrbanPop) as UrbanPop\_min, max(UrbanPop) as UrbanPop\_max, avg(urbanPop) as UrbanPop\_avg, variance(UrbanPop) as UrbanPop\_var from USArrests;

```

1 • SELECT * FROM USArrests where murder=(select max(murder) from USArrests);

```

**Result Grid**

#	State	Murder	Assault	UrbanPop
1	Georgia	17.4	170	60

Fig 1.2.6 : Which state has the maximum murder rate?

sql: select \* from USArrests where murder=(select max(murder) from USArrests);

```

Query1 × USArrests × USArrests ×
use mytestdb;
SELECT * FROM mytestdb.USArrests;
SET SQL_SAFE_UPDATES = 0;
update mytestdb.USArrests set Assault=(select avg(Assault) from USArrests) where State='Georgia';
SELECT * FROM mytestdb.USArrests;
select min(Murder) as murder_min, max(Murder) as murder_max, avg(Murder) as murder_avg, variance(Murder) as murder_var,
min(Assault) as Assault_min, max(Assault) as Assault_max, avg(Assault) as Assault_avg, variance(Assault) as Assault_var,
min(UrbanPop) as UrbanPop_min, max(UrbanPop) as UrbanPop_max, avg(UrbanPop) as UrbanPop_avg, variance(UrbanPop) as UrbanPop_var from USArrests;
select * from USArrests where murder=(select max(murder) from USArrests);
select state,UrbanPop from USArrests order by UrbanPop asc;

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

#	state	UrbanPop
1	Vermont	32
2	West Virginia	39
3	Mississippi	44
4	North Dakota	44
5	North Carolina	45
6	South Dakota	45
7	Alaska	48
8	South Carolina	48
9	Arkansas	50

Action Output ▾

#	Time	Action	Message	Duration / Fetch
1	05:28:11	select * from USArrests where murder=(select max(murder) from USArrests);	1 row(s) returned	0.00019 sec / 0.000...
2	05:29:17	select state from USArrests order by UrbanPop asc LIMIT ...	50 row(s) returned	0.00036 sec / 0.00...
3	05:29:26	select state,UrbanPop from USArrests order by UrbanPop ...	50 row(s) returned	0.00095 sec / 0.00...

Fig 1.2.7 List of states in ascending order of urban population percentages.

Sql: select state,urbanpop from USArrests order by urbanpop asc;

```

Query1 × USArrests × USArrests ×
select * from USArrests where murder=(select max(murder) from USArrests);
select state,UrbanPop from USArrests order by UrbanPop asc;
select * from USArrests where murder>(select murder from USArrests where State='Arizona');

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

#	State	Murder	Assault	UrbanPop
1	Alabama	13.2	236	58
2	Alaska	10	263	48
3	Arkansas	8.8	190	50
4	California	9	276	91
5	Florida	15.4	335	80
6	Georgia	17.4	170	60
7	Illinois	10.4	249	83
8	Kentucky	9.7	109	52
9	Louisiana	15.4	249	66
10	Maryland	11.3	300	67
11	Michigan	12.1	255	74
12	Mississ...	16.1	259	44
13	Missouri	9	178	70
14	Nevada	12.2	252	81
15	New M...	11.4	285	70
16	New York	11.1	254	86
17	North ...	13	337	45
18	South ...	14.4	279	48
19	Tennes...	13.2	188	59
20	Texas	12.7	201	80
21	Virginia	8.5	156	63

Action Output ▾

#	Time	Action	Message
3	05:29:26	select state,UrbanPop from USArrests order by UrbanPop ...	50 row(s) returned
4	05:31:35	select * from USArrests where murder>(select murder fro...)	21 row(s) returned

Fig 1.2.8 How many states have higher murder rates than Arizona? List those states.

Sql: select state from USArrests where murder>(select murder from USArrests where state='Arizona');

## ##### Problem 2 #####

. This problem's dataset involves child mortality rates and is inspired by data collected from UNICEF. The original dataset is available here: <https://github.com/bforoura/IDS/tree/main/HW2>

According to the report, the world has achieved substantial success in reducing child mortality during the last few decades. According to the UNICEF report, globally the under-five age mortality rate has decreased from 93 deaths per 1,000 live births in 1990 to less than 50 in 2016.

However, the dataset has a number of missing instances, which need to be fixed before a clear progress on child mortality can be explained from the year of 1990 to 2016. Use this dataset to complete the following tasks:

### Excel

1. Address all the missing values using the techniques at your disposal.
2. Prepare the dataset to establish the following relations (graphs in Excel):

--Under-five mortality rate and neonatal mortality rate.

--Infant mortality rate and neonatal mortality rate.

--Year and infant mortality rate.

Hint: You may think of converting the mortality rates into five-point Likert scale values. You may count the year before this dataset (i.e., 1989) as the starting point of this program, to assess the progress we have made as the years have passed.

### MySQL

1. Import the original data set into MySQL
2. Use SQL to fill in the missing values in each column using the medians.
3. Answer the following queries in SQL:

-- Display the entire table.

-- Which years have the lowest and highest infant mortality years, respectively?

-- In what years the neonatal mortality rates were above average?

-- Display the sorted infant mortality rates in descending order.

-- Display min, max, mean, variance, and standard deviation for each mortality rate.

-- Add a new column called Above-Five Mortality Rate and populate it with appropriate values. Hint: Use Alter Table Add Column.

-- Display the entire table again.

Ans:

A	B	C	D	E
1	Year	Under-five mortality rate	Infant mortality rate	Neonatal mortality rate
2	1990	93.4	64.8	36.8
3	1991	92.1	63.9	36.3
4	1992	90.9	63.1	35.9
5	1993	89.7	62.3	35.4
6	1994	88.7	61.4	39.24
7	1995	87.3	60.5	34.4
8	1996	85.6	59.4	33.7
9	1997	100.54	58.2	33.1
10	1998	82.1	56.9	32.3
11	1999	79.9	55.4	31.5
12	2000	77.5	53.9	30.7
13	2001	74.8	52.1	29.8
14	2002	72	72.69	28.9
15	2003	69.2	48.6	28
16	2004	66.7	46.9	38.4
17	2005	103.09	45.1	26.1
18	2006	61.1	43.4	25.3
19	2007	58.5	71.06	24.4
20	2008	56.2	40.3	23.6
21	2009	53.7	38.8	22.9
22	2010	105.64	37.4	22.2
23	2011	49.3	36	21.5
24	2012	47.3	34.7	20.8
25	2013	45.5	33.6	20.2
26	2014	43.7	69.44	19.6
27	2015	42.2	31.4	19.1
28	2016	40.8	30.5	18.6
29				
30				

Fig: 2.1.1 Address all missing data

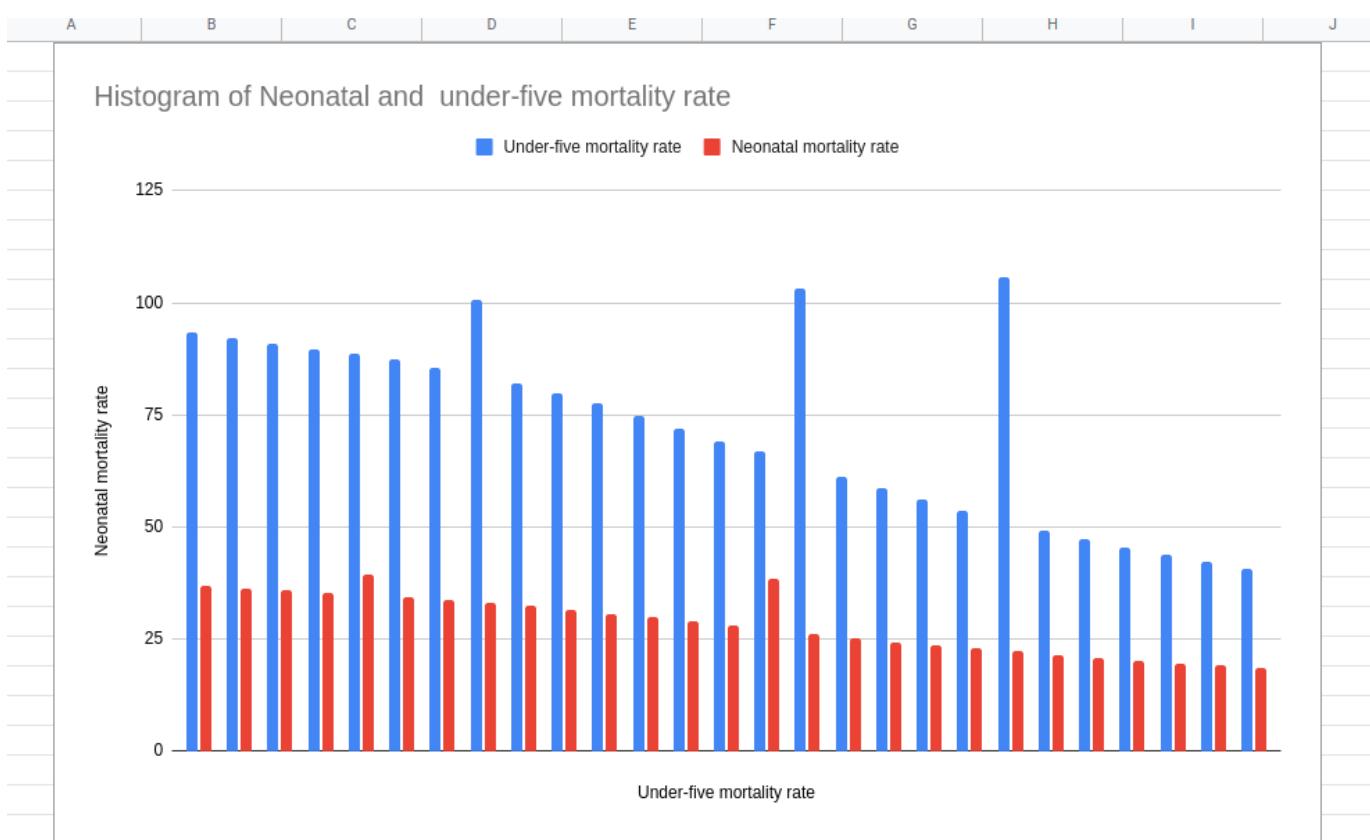


Fig 2.1.2 chart for Under-five mortality rate and neonatal mortality rate.

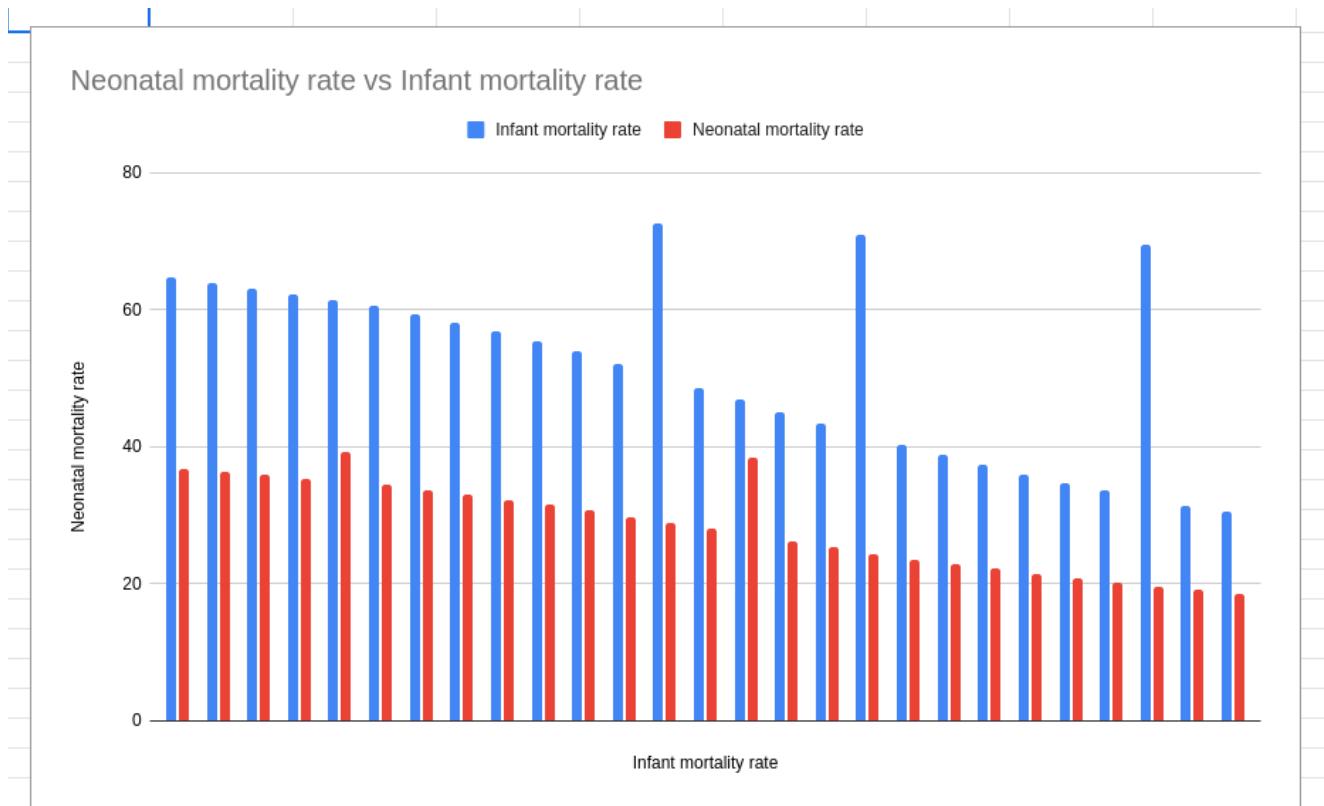


Fig 2.1.3: chart for Infant mortality rate and neonatal mortality rate.

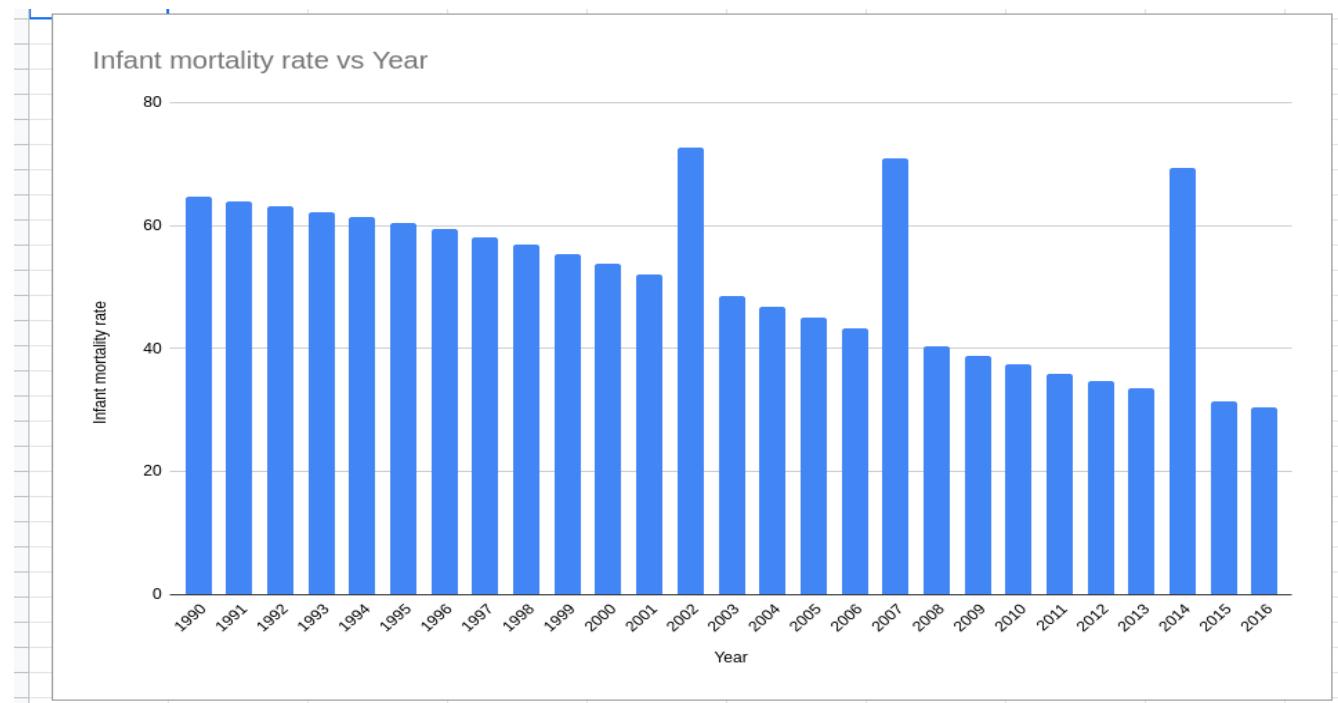


Fig 2.1.4 chart for --Year and infant mortality rate.

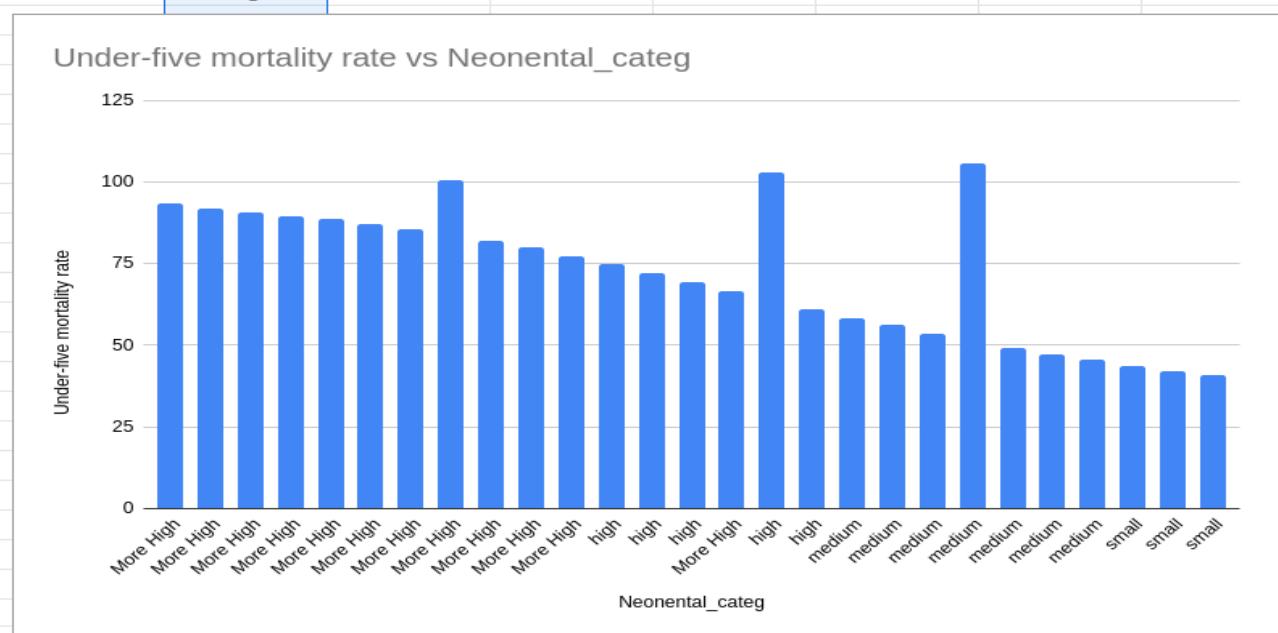


Fig 2.1.4 chart for Under-five mortality rate and neonatal mortality rate using categorical formula: =IF(D2<20, "small",IF(D2<25, "medium", IF(D2<30,"high","More High")))

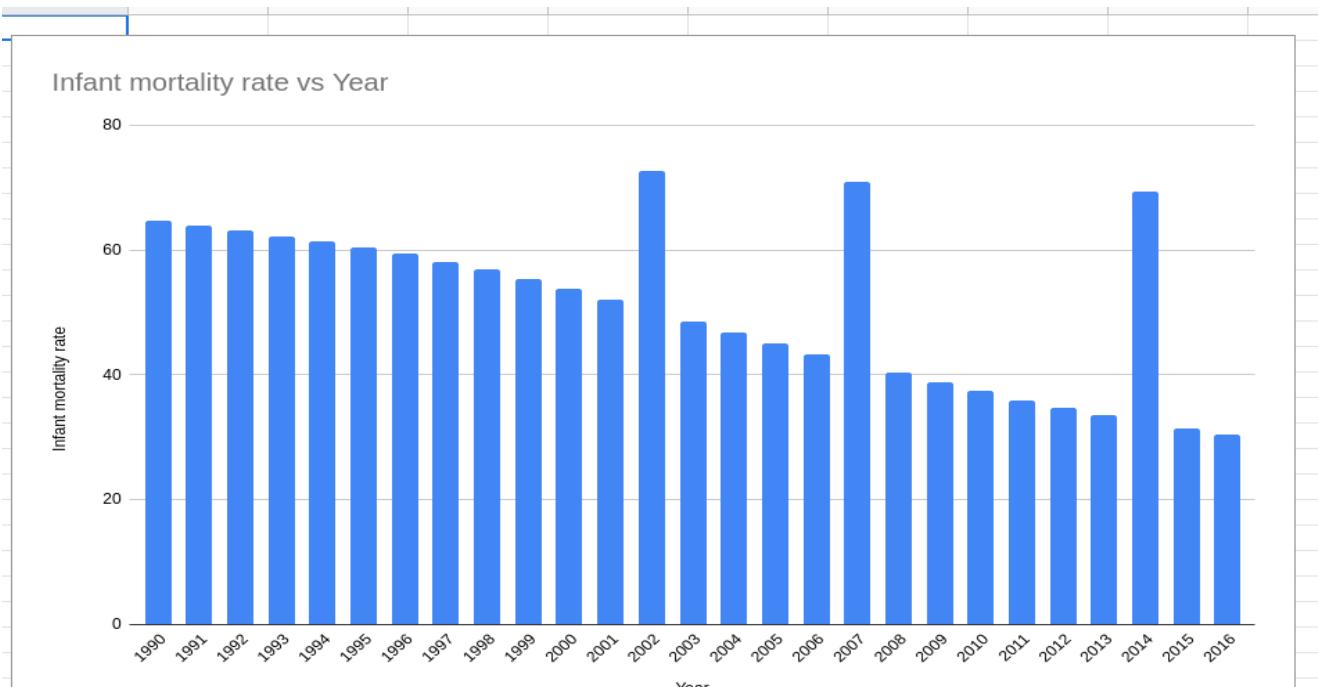


Fig 2.1.5 --Infant mortality rate and neonatal mortality rate.  
formula: =IF(D2<20, "small",IF(D2<25, "medium", IF(D2<30,"high","More High")))

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the title is "localhost - Warning - not supported". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, Help. The left sidebar shows the schema "mytestdb" with a table "USAArrests". The main area has a query editor titled "Query1" and a results grid titled "child\_mortality5".

```

1 create table child_mortality(
2     Years int(4),
3     Under_five_mortality_rate double,
4     Infant_mortality_rate double,
5     Neonatal_mortality_rate double
6 );
7 • desc USAArrests;
8 • desc child_mortality;
9 • select * from child_mortality; # 27 records
10
11

```

**Result Grid:**

#	Years	Under_five_mortality	Infant_mortality	Neonatal_mortality
1	1990	93.4	64.8	36.8
2	1991	92.1	63.9	36.3
3	1992	90.9	63.1	35.9
4	1993	89.7	62.3	35.4
5	1994	88.7	61.4	0
6	1995	87.3	60.5	34.4
7	1996	85.6	59.4	33.7
8	1997	0	33.1	0
9	1998	82.1	56.9	32.3
10	1999	79.9	55.4	31.5
11	2000	77.5	53.9	30.7
12	2001	74.8	52.1	29.8
13	2002	72	0	28.9

**Action Output:**

#	Time	Action	Message	Duration / Fetch
11	07:05:26	desc child_mortality	4 row(s) returned	0.0029 sec / 0.000...
12	07:05:32	select * from child_mortality LIMIT 0,1000	27 row(s) returned	0.00071 sec / 0.000...

Fig 2.2.1 : mysql import dataset and projection.

This screenshot is identical to Fig 2.2.1, showing the MySQL Workbench interface with the same schema, table, and data. The query editor contains the same SQL code, and the results grid shows the same data for the "child\_mortality" table. The action output table also shows the same log entries for the DESC and SELECT statements.

Fig 2.2.2 get median value

```

9
10 • SET @rowindex := -1;
11 • update child_mortality set Under_five_mortality_rate=(
12     SELECT
13         AVG(d.Under_five_mortality_rate) as Median
14     FROM
15         (SELECT @rowindex:="@rowindex + 1 AS rowindex,
16             child_mortality.Under_five_mortality_rate AS Under_five_mortality_rate
17         FROM child_mortality
18         ORDER BY child_mortality.Under_five_mortality_rate) AS d
19     WHERE
20         d.rowindex IN (FLOOR(@rowindex / 2), CEIL(@rowindex / 2))
21     ) where Under_five_mortality_rate=0;

```

Action Output ▾				
#	Time	Action	Message	Duration / Fetch
27	07:14:10	desc child_mortality	4 row(s) returned	0.0033 sec / 0.0000...
28	07:14:29	select * from child_mortality LIMIT 0,1000	27 row(s) returned	0.00051 sec / 0.000...
29	07:20:27	desc child_mortality	4 row(s) returned	0.0016 sec / 0.0000...
30	10:20:26	select * from child_mortality LIMIT 0,1000	27 row(s) returned	0.0015 sec / 0.0000...
31	10:22:15	SET @rowindex:=-1	0 row(s) affected	0.00013 sec
32	10:22:15	update child_mortality set Under_five_mortality_rate=( ...	3 row(s) affected Rows matched: 3 Changed: 3 Warnings: 0	0.0014 sec

Fig 2.2.3 update missing value by median

The screenshot shows the MySQL Workbench interface with a query editor titled "Query1" running against the "USArests" schema. The query is:

```
1 • select * from child_mortality;
```

The result grid displays 14 rows of data:

#	Years	Under_five_mortality_ra	Infant_mortality_ra	Neonatal_mortality_ra
1	1990	93.4	64.8	36.8
2	1991	92.1	63.9	36.8
3	1992	90.9	63.1	35.9
4	1993	89.7	62.3	35.4
5	1994	88.7	61.4	26.1
6	1995	87.3	60.5	34.4
7	1996	85.6	59.4	33.7
8	1997	66.7	58.2	33.1
9	1998	82.1	56.9	32.3
10	1999	79.9	55.4	31.5
11	2000	77.5	53.9	30.7
12	2001	74.8	52.1	29.8
13	2002	72	46.9	28.9
14	2003	69.2	48.6	28

The "Action Output" pane shows the execution log with entries 53 through 58, detailing the update and subsequent queries.

Fig: 2.2.4: projection after updating missing values

The screenshot shows the MySQL Workbench interface with a query editor running against the "USArests" schema. The query is:

```
1 select min(Under_five_mortality_rate) as UnderFive_min, max(Under_five_mortality_rate) as UnderFive_max, avg(Under_five_mortality_rate) as UnderFive_avg, variance(Under_five_mortality_rate) as UnderFive_var, 2 min(Infant_mortality_rate) as Infant_min, max(Infant_mortality_rate) as Infant_max, avg(Infant_mortality_rate) as Infant_avg, variance(Infant_mortality_rate) as Infant_var, 3 min(Neonatal_mortality_rate) as Neonatal_min, max(Neonatal_mortality_rate) as Neonatal_max, avg(Neonatal_mortality_rate) as Neonatal_avg, variance(Neonatal_mortality_rate) as Neonatal_var;
```

The result grid displays the calculated statistics:

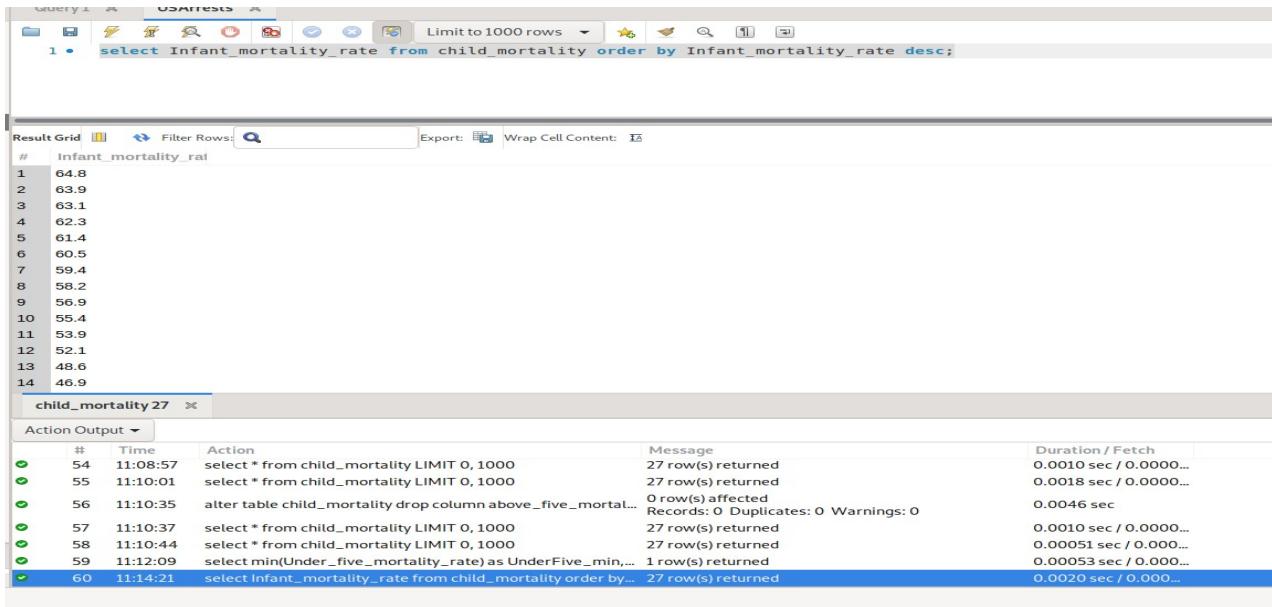
#	UnderFive_min	UnderFive_max	UnderFive_avg	UnderFive_var	Infant_min	Infant_max	Infant_avg	Infant_var	Neonatal_min	Neonatal_max	Neonatal_avg	Neonatal_var
1	40.8	93.4	68.45555555555555	281.0928395061728	30.5	64.8	48.86296296296296	114.96159122085056	18.6	36.8	27.529629629629635	33.41541838134432

The "Action Output" pane shows the execution log with entries 53 through 59, detailing the calculation process.

Fig: 2.2.5: Display min, max, mean, variance, and standard deviation for each mortality rate.

Sql: `select min(Under_five_mortality_rate) as UnderFive_min, max(Under_five_mortality_rate) as UnderFive_max, avg(Under_five_mortality_rate) as UnderFive_avg, variance(Under_five_mortality_rate) as UnderFive_var, min(Infant_mortality_rate) as Infant_min, max(Infant_mortality_rate) as Infant_max, avg(Infant_mortality_rate) as Infant_avg, variance(Infant_mortality_rate) as Infant_var,`

`min(Neonatal_mortality_rate) as Neonatal_min, max(Neonatal_mortality_rate) as Neonatal_max,  
 avg(Neonatal_mortality_rate) as Neonatal_avg, variance(Neonatal_mortality_rate) as Neonatal_var  
 from child_mortality;`



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 • select Infant_mortality_rate from child_mortality order by Infant_mortality_rate desc;
```

The results grid displays the following data:

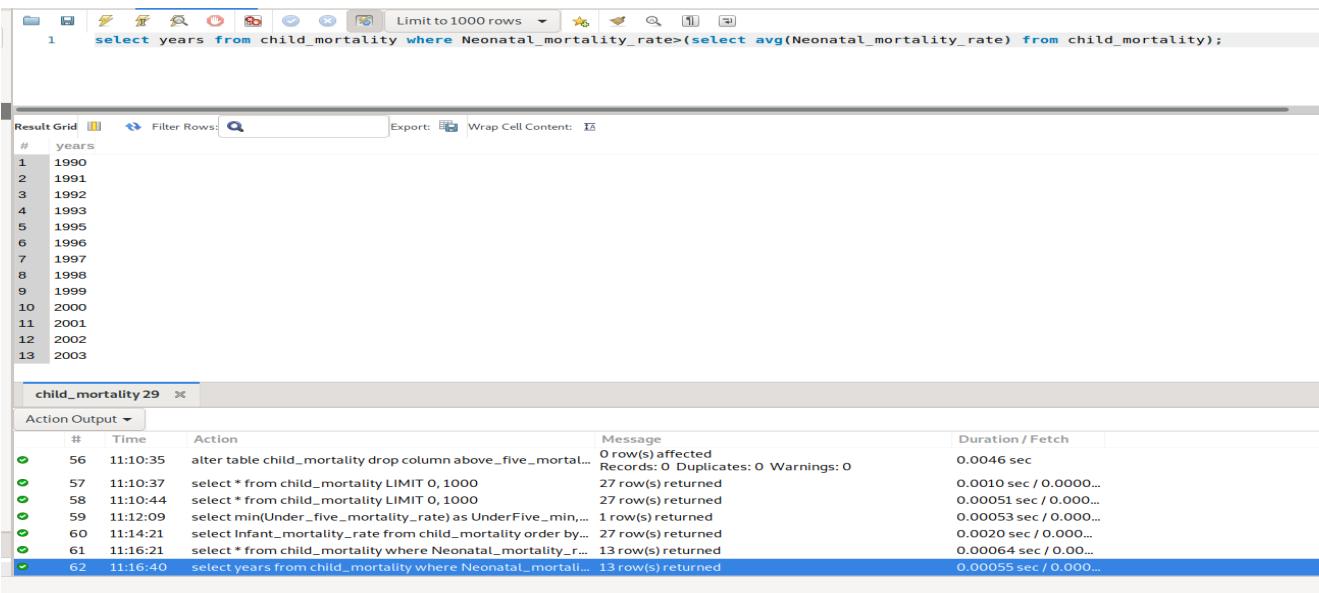
#	Infant_mortality_rate
1	64.8
2	63.9
3	63.1
4	62.3
5	61.4
6	60.5
7	59.4
8	58.2
9	56.9
10	55.4
11	53.9
12	52.1
13	48.6
14	46.9

The action output window shows the following log entries:

#	Time	Action	Message	Duration / Fetch
54	11:08:57	select * from child_mortality LIMIT 0, 1000	27 row(s) returned	0.0010 sec / 0.0000...
55	11:10:01	select * from child_mortality LIMIT 0, 1000	27 row(s) returned	0.0018 sec / 0.0000...
56	11:10:35	alter table child_mortality drop column above_five_mortal...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.0046 sec
57	11:10:37	select * from child_mortality LIMIT 0, 1000	27 row(s) returned	0.0010 sec / 0.0000...
58	11:10:44	select * from child_mortality LIMIT 0, 1000	27 row(s) returned	0.00051 sec / 0.000...
59	11:12:09	select min(UnderFive_mortality_rate) as UnderFive_min,...	1 row(s) returned	0.00053 sec / 0.000...
60	11:14:21	select Infant_mortality_rate from child_mortality order by...	27 row(s) returned	0.0020 sec / 0.000...

Fig 2.2.6: Display the sorted infant mortality rates in descending order.

Sql: `select Infant_mortality_rate from child_mortality order by Infant_mortality_rate desc;`



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 • select years from child_mortality where Neonatal_mortality_rate > (select avg(Neonatal_mortality_rate) from child_mortality);
```

The results grid displays the following data:

#	years
1	1990
2	1991
3	1992
4	1993
5	1995
6	1996
7	1997
8	1998
9	1999
10	2000
11	2001
12	2002
13	2003

The action output window shows the following log entries:

#	Time	Action	Message	Duration / Fetch
56	11:10:35	alter table child_mortality drop column above_five_mortal...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.0046 sec
57	11:10:37	select * from child_mortality LIMIT 0, 1000	27 row(s) returned	0.0010 sec / 0.0000...
58	11:10:44	select * from child_mortality LIMIT 0, 1000	27 row(s) returned	0.00051 sec / 0.000...
59	11:12:09	select min(UnderFive_mortality_rate) as UnderFive_min,...	1 row(s) returned	0.00053 sec / 0.000...
60	11:14:21	select Infant_mortality_rate from child_mortality order by...	27 row(s) returned	0.0020 sec / 0.000...
61	11:16:21	select * from child_mortality where Neonatal_mortality_r...	13 row(s) returned	0.00064 sec / 0.000...
62	11:16:40	select years from child_mortality where Neonatal_mortali...	13 row(s) returned	0.00055 sec / 0.000...

Fig: 2.2.7 - In what years the neonatal mortality rates were above average?

sql: `select years from child_mortality where Neonatal_mortality_rate > (select avg(Neonatal_mortality_rate) from child_mortality);`

The screenshot shows the MySQL Workbench interface with two tabs: "Query 1" and "child\_mortality 45".

**Query 1:**

```

1 • select years from child_mortality where Infant_mortality_rate=(select min(Infant_mortality_rate) from child_mortality);
2 • select years from child_mortality where Infant_mortality_rate=(select max(Infant_mortality_rate) from child_mortality);
3

```

**Result Grid:**

#	years
1	2016

**Action Output:**

#	Time	Action	Message	Duration / Fetch
76	11:25:11	select years from child_mortality where infant_mortality_rate=(select min(infant_mortality_rate) from child_mortality);	1 row(s) returned	0.00025 sec / 0.000...
77	11:25:45	select years from child_mortality where Infant_mortality_rate=(select max(Infant_mortality_rate) from child_mortality);	Error Code: 1241. Operand should contain 1 column...	0.00032 sec
78	11:26:01	select min(Infant_mortality_rate),max(Infant_mortality_r...	1 row(s) returned	0.00030 sec / 0.00...
79	11:26:08	select years from child_mortality where Infant_mortality...	Error Code: 1241. Operand should contain 1 column...	0.00035 sec
80	11:26:23	select min(Infant_mortality_rate),max(Infant_mortality_r...	1 row(s) returned	0.0025 sec / 0.000...
81	11:26:43	select years from child_mortality where Infant_mortality...	2 row(s) returned	0.00039 sec / 0.00...
82	11:26:55	select years from child_mortality where Infant_mortality...	Error Code: 1241. Operand should contain 1 column...	0.0011 sec
83	11:28:03	select years from child_mortality where Infant_mortality...	1 row(s) returned	0.0039 sec / 0.000...

Fig 2.28 -- Which years have the lowest and highest infant mortality years, respectively?

Sql:    select    years    from    child\_mortality    where    Infant\_mortality\_rate=(select min(Infant\_mortality\_rate) from child\_mortality);

select years from child\_mortality where Infant\_mortality\_rate=(select max(Infant\_mortality\_rate) from child\_mortality);

The screenshot shows the MySQL Workbench interface with two tabs: "Query 1" and "Result 47".

**Query 1:**

```

1 • alter table child_mortality add column Above_Five_Mortality_Rate double;
2 • desc child_mortality;

```

**Result Grid:**

#	Field	Type	Null	Key	Default	Extra
1	Years	int(4)	YES			
2	Under_five_mortality_rate	double	YES			
3	Infant_mortality_rate	double	YES			
4	Neonatal_mortality_rate	double	YES			
5	Above_Five_Mortality_Rate	double	YES			

**Action Output:**

#	Time	Action	Message	Duration / Fetch
1	11:30:52	alter table child_mortality add column Above_Five_Mortality_Rate double;	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.0039 sec
2	11:31:01	desc child_mortality	5 row(s) returned	0.0052 sec / 0.000...
3	11:31:13	desc child_mortality	5 row(s) returned	0.00069 sec / 0.00...

Fig 2.2.9: Add a new column called Above-Five Mortality Rate

sql: alter table child\_mortality add column Above\_Five\_Mortality\_Rate double;

The screenshot shows the MySQL Workbench interface. The top bar has tabs for 'Query 1' and 'USArrests'. Below the tabs are various icons and a dropdown menu for 'Limit to 1000 rows'. The main area contains two code snippets:

```

1 • select * from child_mortality;
2 • update child_mortality set above_five_mortality_rate=format((Under_five_mortality_rate+Infant_mortality_rate+Neonatal_mortality_rate)/3,2);

```

Below the code is a 'Result Grid' table with columns: #, Years, Under\_five\_mortality\_ra, Infant\_mortality\_ra, Neonatal\_mortality\_ra, and Above\_Five\_Mortality\_Rate. The data is as follows:

#	Years	Under_five_mortality_ra	Infant_mortality_ra	Neonatal_mortality_ra	Above_Five_Mortality_Rate
1	1990	93.4	64.8	36.8	65
2	1991	92.1	63.9	36.3	64.1
3	1992	90.9	63.1	35.9	63.3
4	1993	89.7	62.3	35.4	62.47
5	1994	88.7	61.4	26.1	58.73
6	1995	87.3	60.5	34.4	60.73
7	1996	85.6	59.4	33.7	59.57
8	1997	66.7	58.2	33.1	52.67
9	1998	82.1	56.9	32.3	57.1
10	1999	79.9	55.4	31.5	55.6

Below the result grid is a 'child\_mortality 51' tab showing 'Action Output' with 9 rows of log data.

#	Time	Action	Message	Duration / Fetch
3	11:31:13	desc child_mortality	5 row(s) returned	0.00069 sec / 0.00...
4	11:33:02	select * from child_mortality LIMIT 0, 1000	27 row(s) returned	0.00092 sec / 0.00...
5	11:33:52	desc child_mortality	5 row(s) returned	0.0012 sec / 0.0000...
6	11:34:35	update child_mortality set above_five_mortality_rate=(U...	27 row(s) affected	Rows matched: 27 Changed: 27 Warnings: 0 0.0030 sec
7	11:34:47	select * from child_mortality LIMIT 0, 1000	27 row(s) returned	0.0043 sec / 0.000...
8	11:35:05	update child_mortality set above_five_mortality_rate=for...	20 row(s) affected	Rows matched: 27 Changed: 20 Warnings: 0 0.0016 sec
9	11:35:07	select * from child_mortality LIMIT 0, 1000	27 row(s) returned	0.0034 sec / 0.000...

Fig: 2.3.0 Populate Above-Five Mortality Rate with appropriate values.

Sql:

```

update child_mortality
set above_five_mortality_rate=format((Under_five_mortality_rate+Infant_mortality_rate+Neonatal_mortality_rate)/3,2);

```

## XML/JSON

Convert the cleansed data set into XML and JSON formats.

### Submission Guidelines

- All the data files (CSV, XML, JSON), Excel analysis sheets, and SQL scripts must reside in IDS/HW2folder in your repository
- Summarize your findings for the problems in a PDF report (HW2.pdf) and upload it into Canvas. Thoroughly explain your data cleaning and analysis steps for each problem.
- Include the link to your public GitHub repository on top of your report.