Matt Caraher

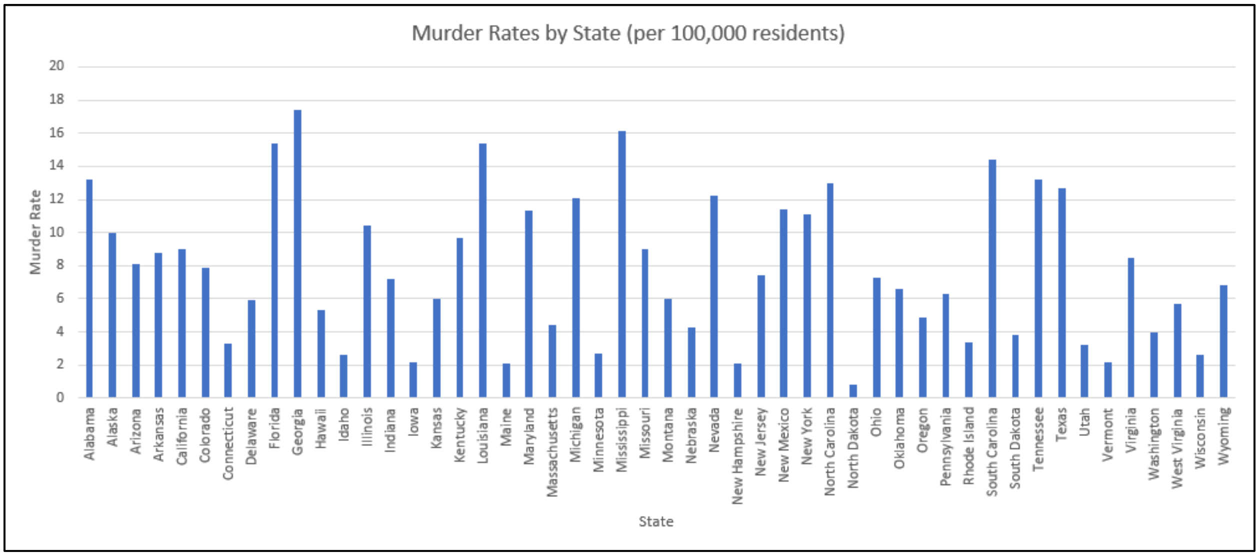
Introduction to Data Science

Project 2

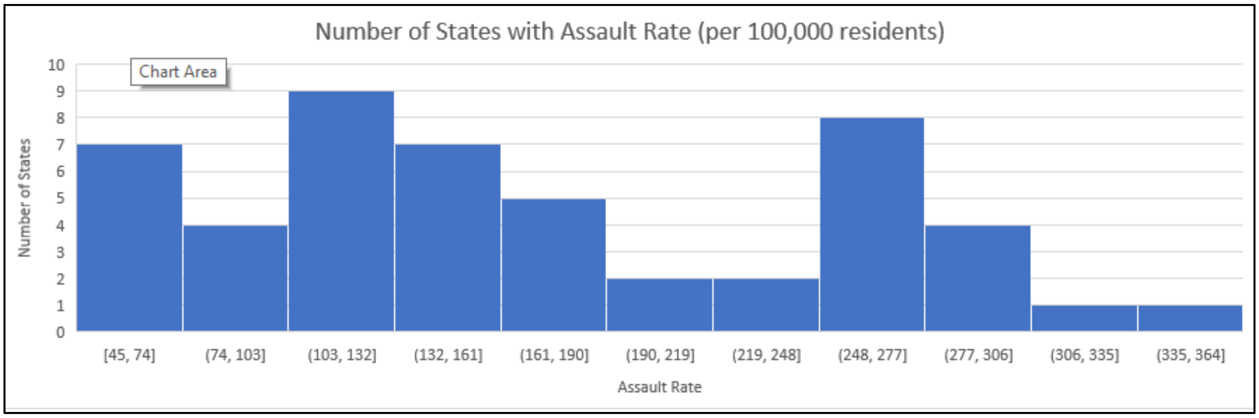
15 February 2022

Project 2

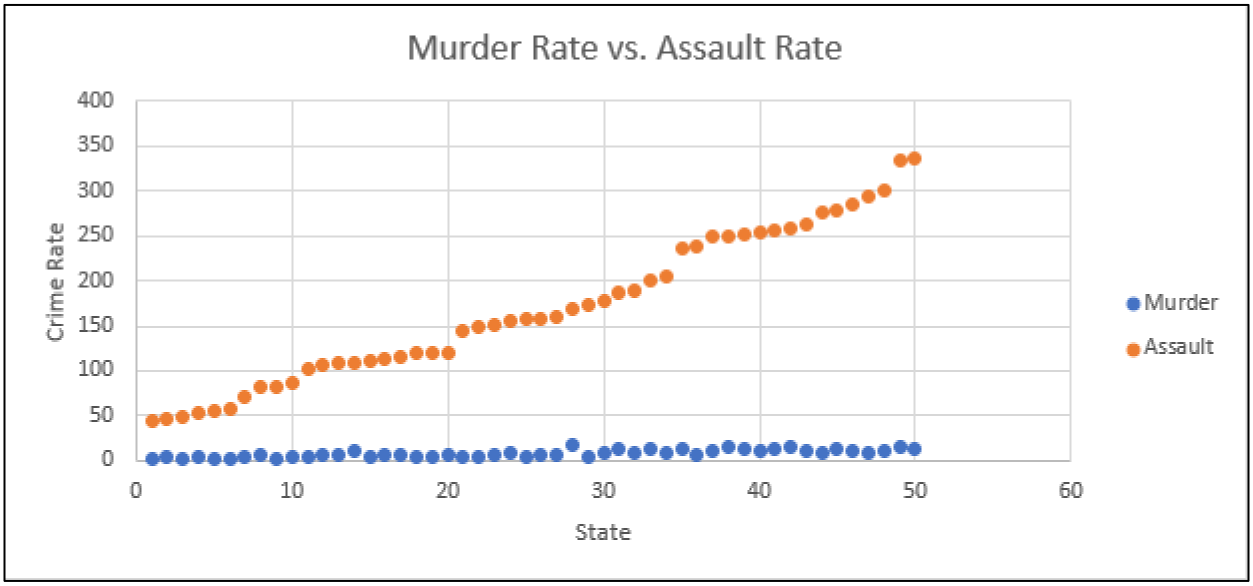
Problem 1 Excel: To address the missing value for Georgia in the assault column, I took the average for assaults in the other 49 states. After finding an average of 170, I used this number as Georgia's assault number. To minimize noisy data, I first put each row in descending order to check for outliers, which there were none of. I then made the murder column into integers because the decimal numbers seemed like noisy data.



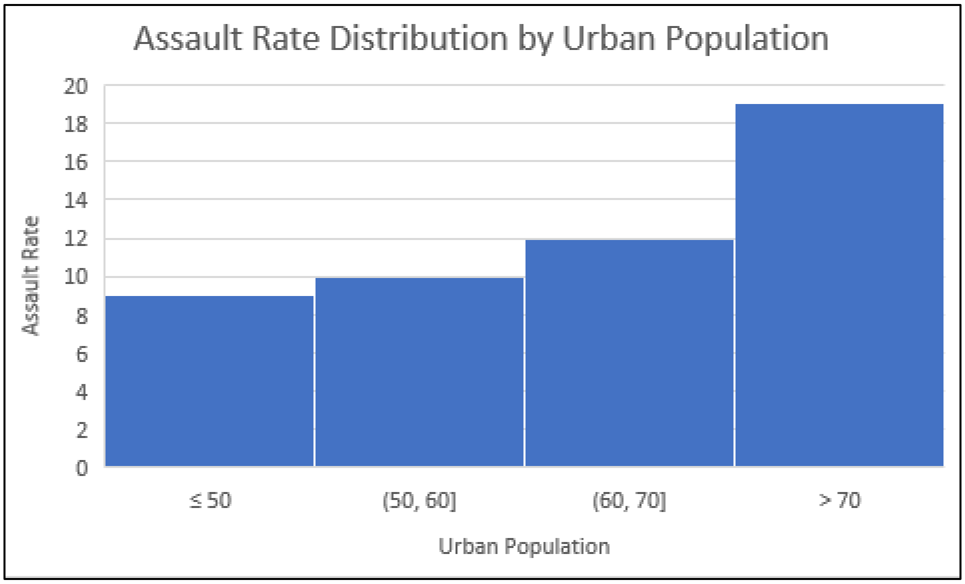
The next chart made is a histogram of assault rates by state. I divided the data into 5 bins to get an accurate representation of how many states fall into each bin. I concluded that murder rates were not commonly over 306, with only 2 states being above.



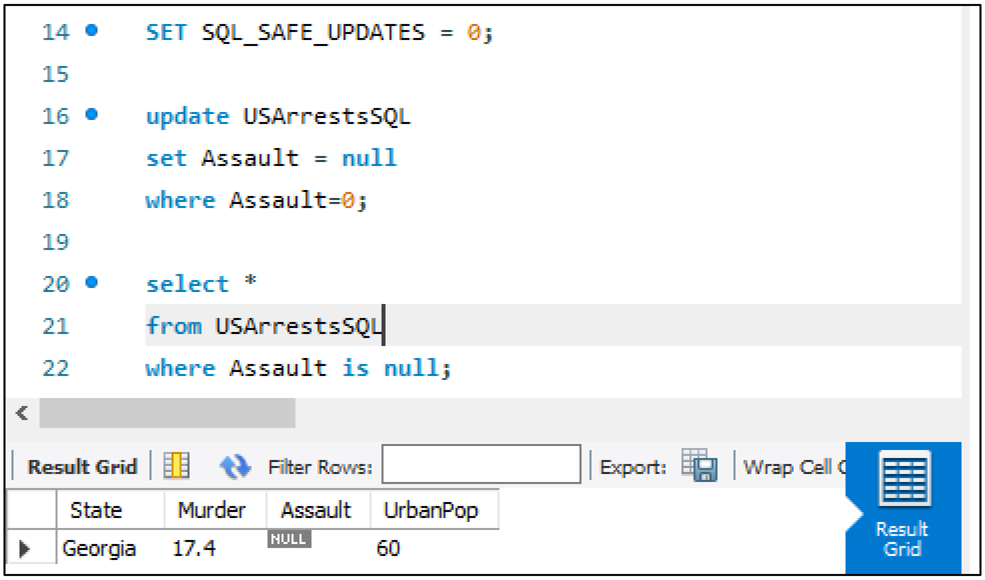
I then plotted the murder rate vs. assault rate for each state. The assault rate was significantly higher than the murder rate and I did not find a correlation between the two. The highest murder rate was nearly halfway through the data, which was sorted by assault rate. When I sorted by murder rate, the assault rate points became very sporadic, confirming that there was no clear trend.



Next, I created a chart to show the distribution of assault by urban population. I did this by using a histogram and adding custom bins to specify below 50 and above 70 as min and max overflow.



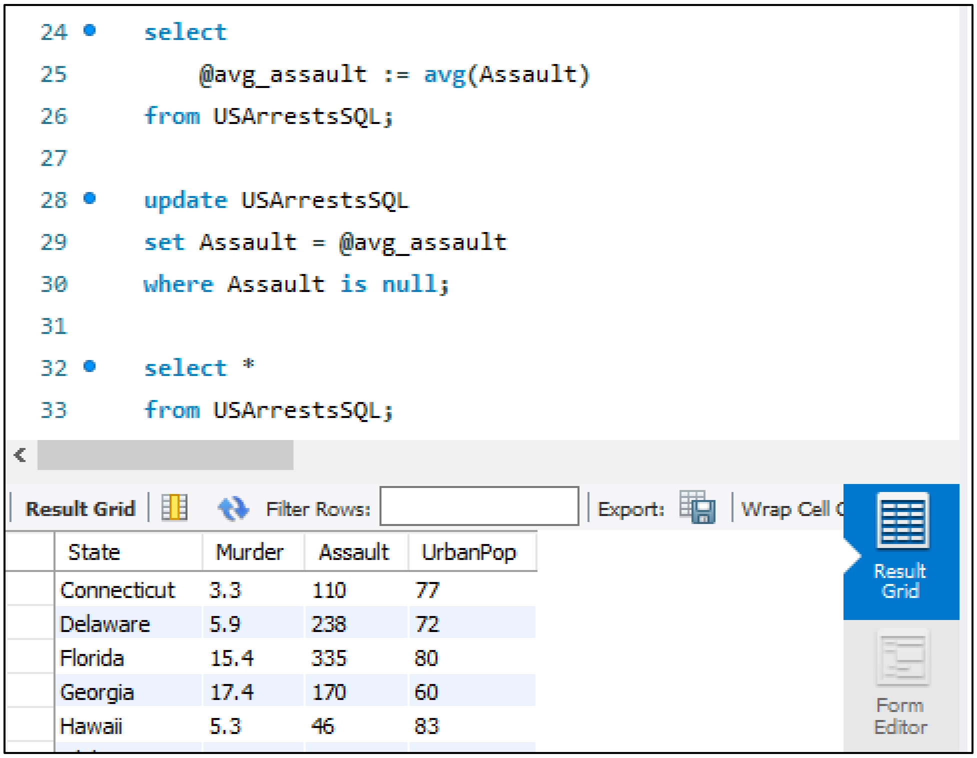
Problem 1 MySQL: I used the table data wizard to import as a CSV and noticed that when I took the average of my data, it was counting the 0 into it. So before updating the missing value, I ran a query to replace the 0 with null, so that it would not count towards the average.



```
SET SQL_SAFE_UPDATES = 0;

update USArrestsSQL
set Assault = null
where Assault=0;

select *
from USArrestsSQL
where Assault is null;
```

State	Murder	Assault	UrbanPop
Georgia	17.4	NULL	60

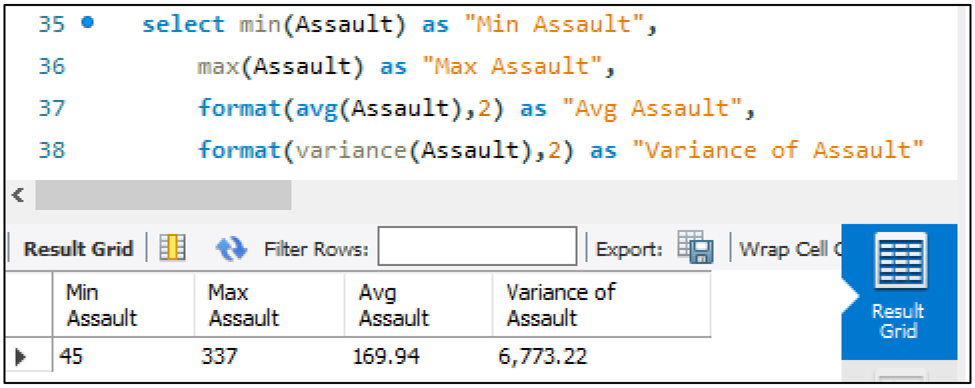
I then replaced null with average using these queries:



```sql
select
    @avg_assault := avg(Assault)
from USArrestsSQL;

update USArrestsSQL
set Assault = @avg_assault
where Assault is null;

select *
from USArrestsSQL;
```

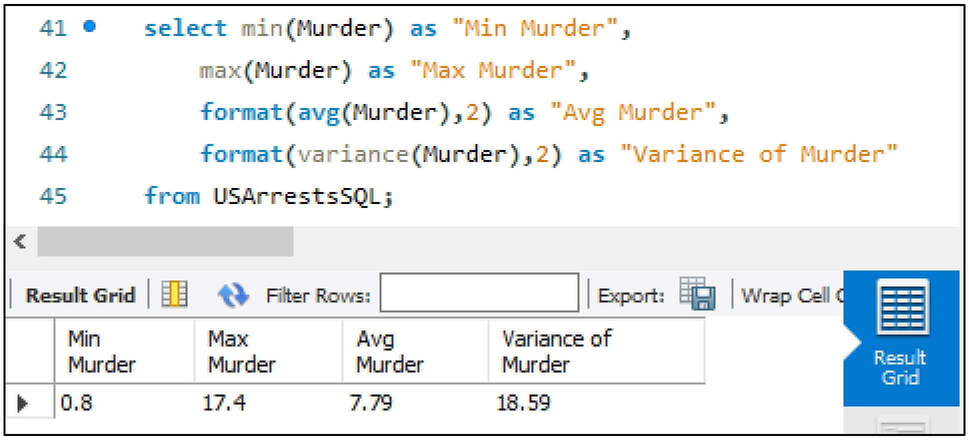
State	Murder	Assault	UrbanPop
Connecticut	3.3	110	77
Delaware	5.9	238	72
Florida	15.4	335	80
Georgia	17.4	170	60
Hawaii	5.3	46	83

The next images are the min, max, avg, and variance of the three numeric attributes in the table. I changed each column heading and edited the format of average and variance to make it clearer.



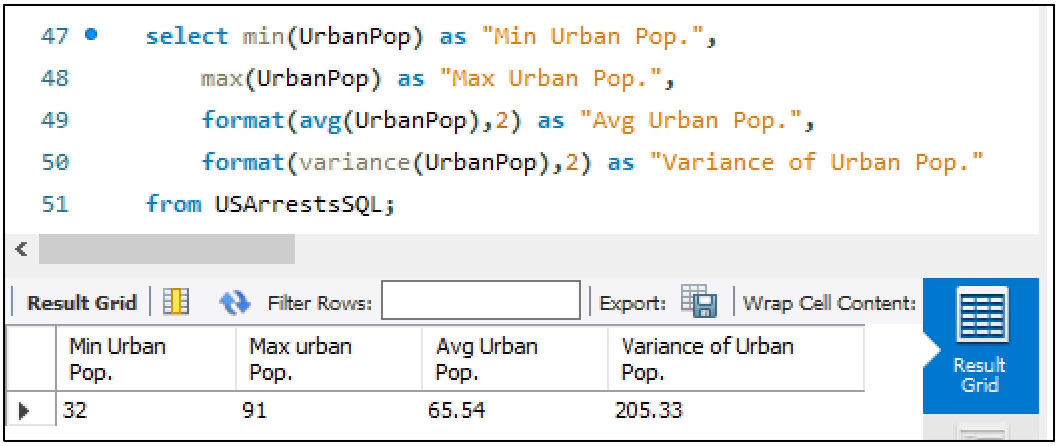
```sql
select min(Assault) as "Min Assault",
    max(Assault) as "Max Assault",
    format(avg(Assault),2) as "Avg Assault",
    format(variance(Assault),2) as "Variance of Assault"
```

Min Assault	Max Assault	Avg Assault	Variance of Assault
45	337	169.94	6,773.22



```sql
select min(Murder) as "Min Murder",
    max(Murder) as "Max Murder",
    format(avg(Murder),2) as "Avg Murder",
    format(variance(Murder),2) as "Variance of Murder"
from USArrestsSQL;
```

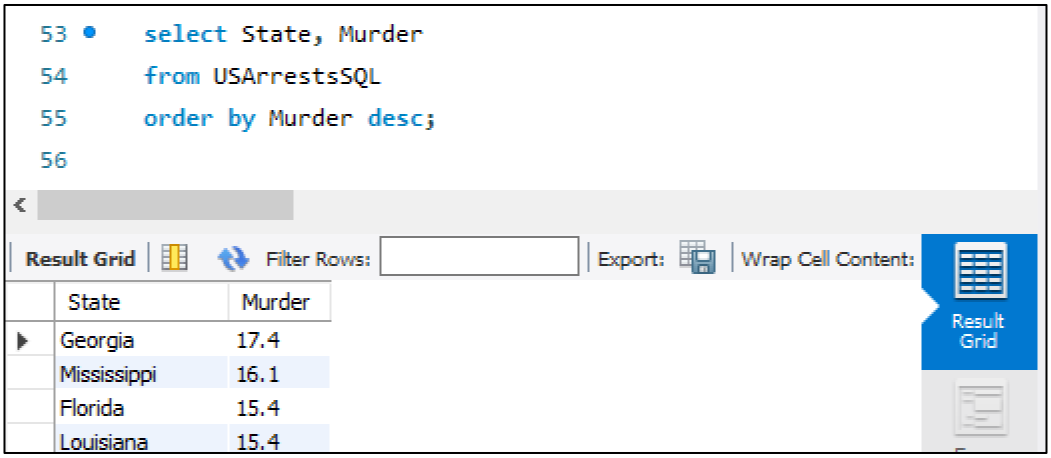
Min Murder	Max Murder	Avg Murder	Variance of Murder
0.8	17.4	7.79	18.59



```sql
select min(UrbanPop) as "Min Urban Pop.",
    max(UrbanPop) as "Max Urban Pop.",
    format(avg(UrbanPop),2) as "Avg Urban Pop.",
    format(variance(UrbanPop),2) as "Variance of Urban Pop."
from USArrestsSQL;
```

Min Urban Pop.	Max urban Pop.	Avg Urban Pop.	Variance of Urban Pop.
32	91	65.54	205.33

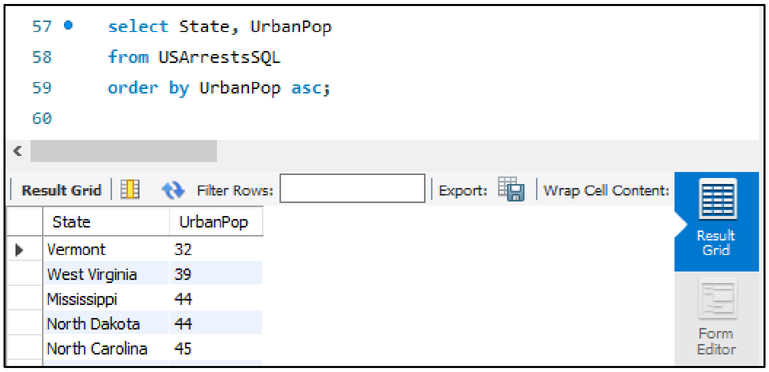
To find which state has the maximum murder rate, I used this query to find out that Georgia had the max murder rate of 17.4.



```sql
select State, Murder
from USArrestsSQL
order by Murder desc;
```

State	Murder
Georgia	17.4
Mississippi	16.1
Florida	15.4
Louisiana	15.4

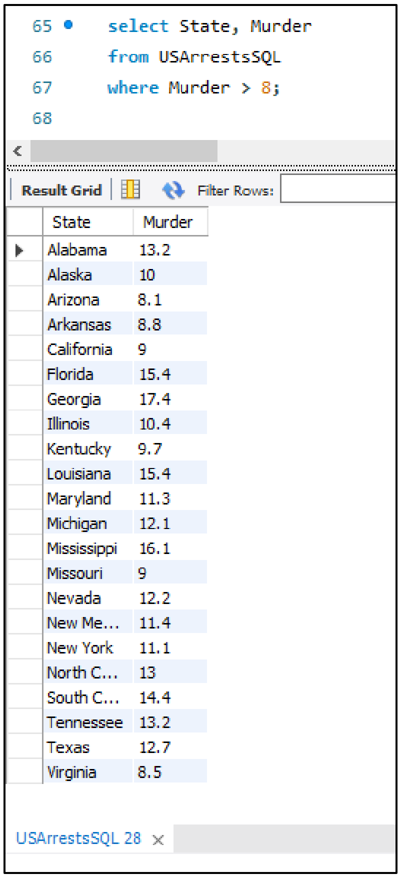
Here is the query I used to find the urban population percentages ascending and the resulting table. I was able to see the min and max using the data.



```
select State, UrbanPop
from USArrestsSQL
order by UrbanPop asc;
```

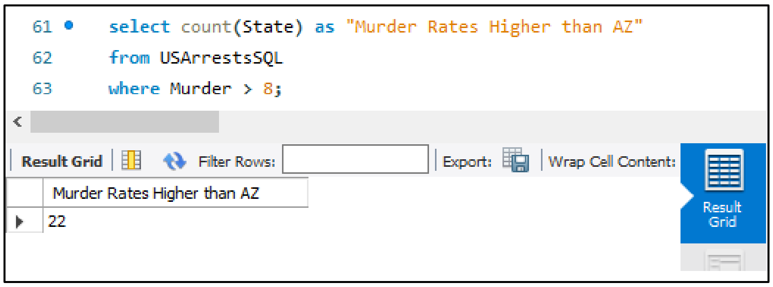
State	UrbanPop
Vermont	32
West Virginia	39
Mississippi	44
North Dakota	44
North Carolina	45

I found the number of states with a higher murder rate than Arizona by asking for a count of the number of states with murder rates higher than 8 (Arizona's). The result was 22:



```
select State, Murder
from USArrestsSQL
where Murder > 8;
```

State	Murder
Alabama	13.2
Alaska	10
Arizona	8.1
Arkansas	8.8
California	9
Florida	15.4
Georgia	17.4
Illinois	10.4
Kentucky	9.7
Louisiana	15.4
Maryland	11.3
Michigan	12.1
Mississippi	16.1
Missouri	9
Nevada	12.2
New Me...	11.4
New York	11.1
North C...	13
South C...	14.4
Tennessee	13.2
Texas	12.7
Virginia	8.5



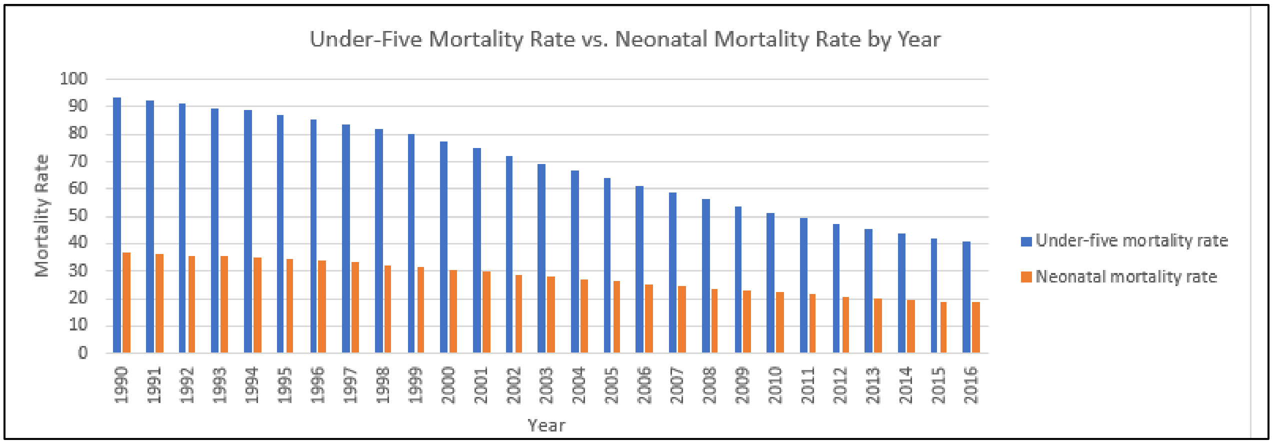
```
select count(State) as "Murder Rates Higher than AZ"
from USArrestsSQL
where Murder > 8;
```

Murder Rates Higher than AZ
22

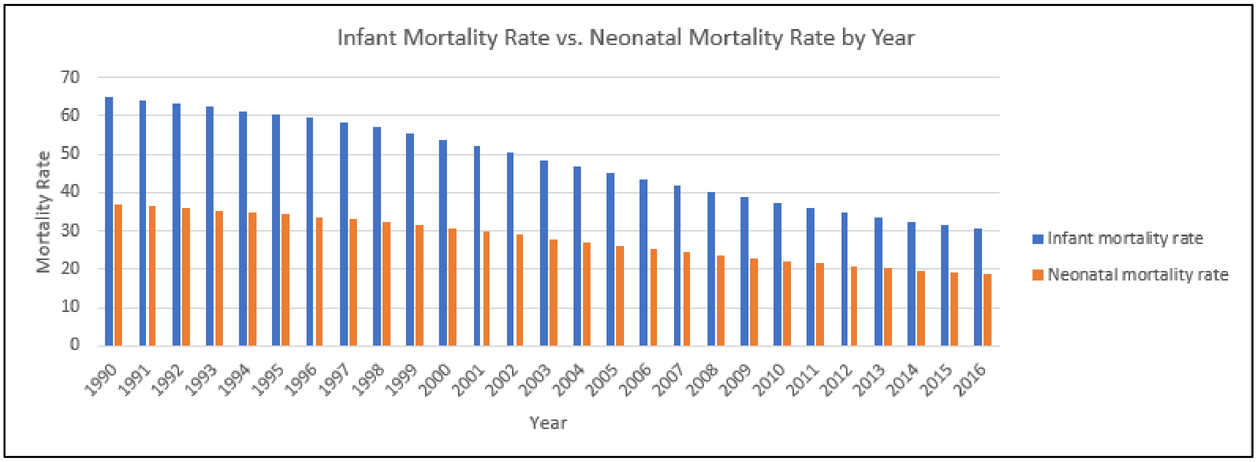
Problem 2 Excel:

When I opened the data, I noticed that the values were already in descending order for each numeric column. For that reason, it seemed best for me to take the average of the years adjacent to each missing value. For example, I filled in the missing under-five mortality rate in 2005 with the average of those values for 2004 and 2006. I did this process for all missing values in the table. I used this data to create the following relations between the data.

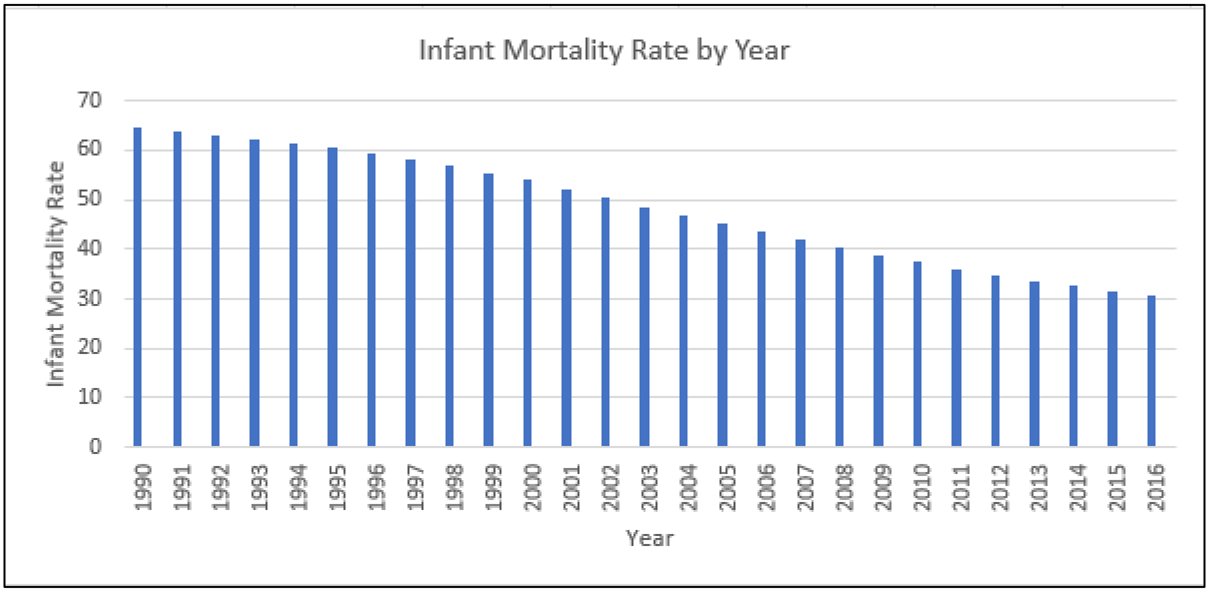
Under-five mortality rate and neonatal mortality rate:



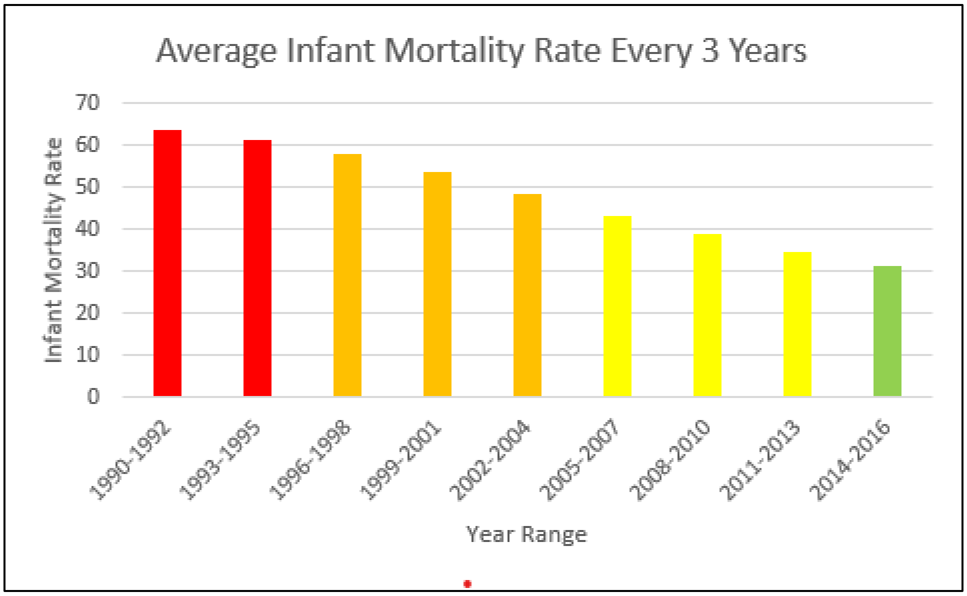
Infant mortality rate and neonatal mortality rate:



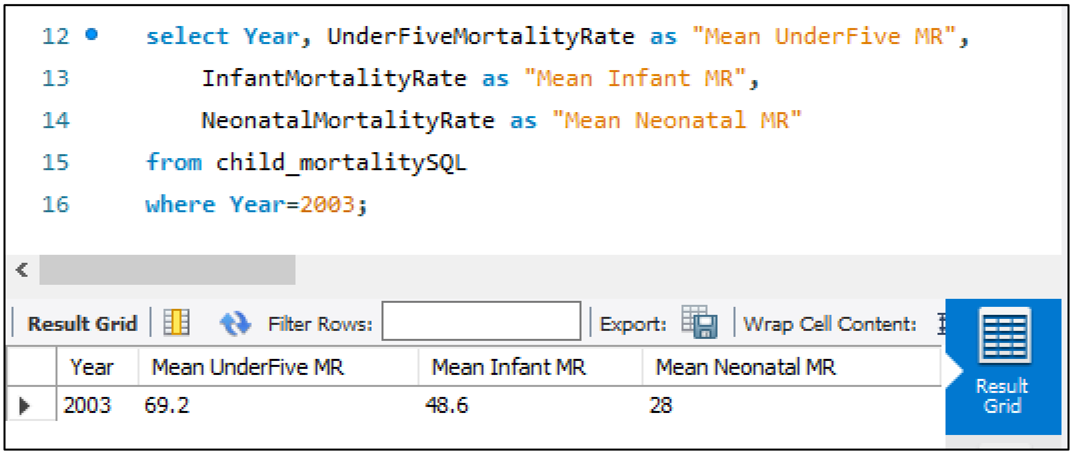
Year and infant mortality rate:



With these graphs, I was able to conclude that the mortality rate of under-five, infant, and neonatal has decreased steadily from 1990 to 2016. I also experimented with taking the average across spans of years to take a different look at the data, with colors to aid the graph.



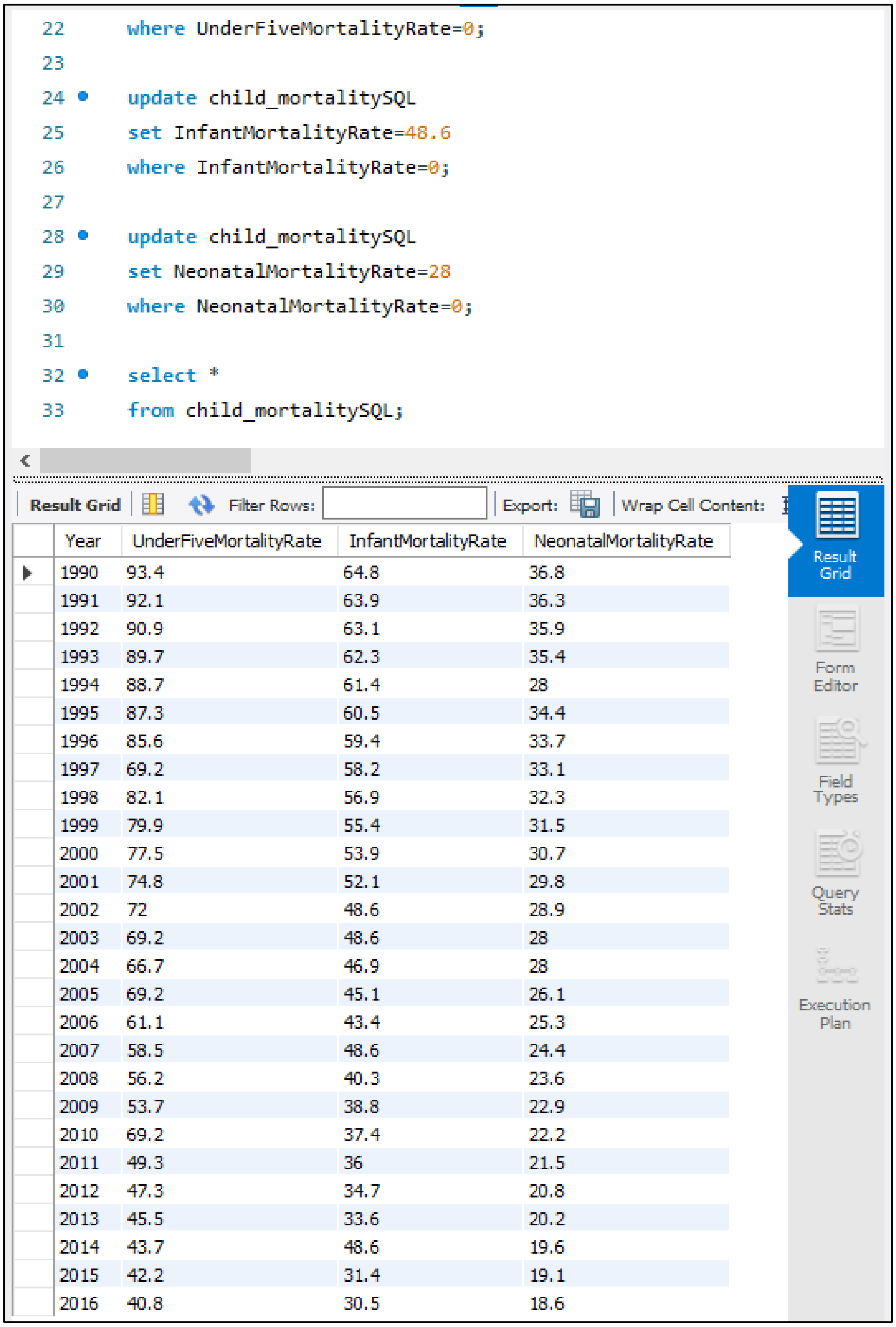
Problem 2 MySQL: To find the median of each column in the child_mortality.csv table, I first did a quick count query to confirm that there are 27 years in the table. This would mean the median is at the year 2003 in each column.



```
select Year, UnderFiveMortalityRate as "Mean UnderFive MR",
    InfantMortalityRate as "Mean Infant MR",
    NeonatalMortalityRate as "Mean Neonatal MR"
from child_mortalitySQL
where Year=2003;
```

Year	Mean UnderFive MR	Mean Infant MR	Mean Neonatal MR
2003	69.2	48.6	28

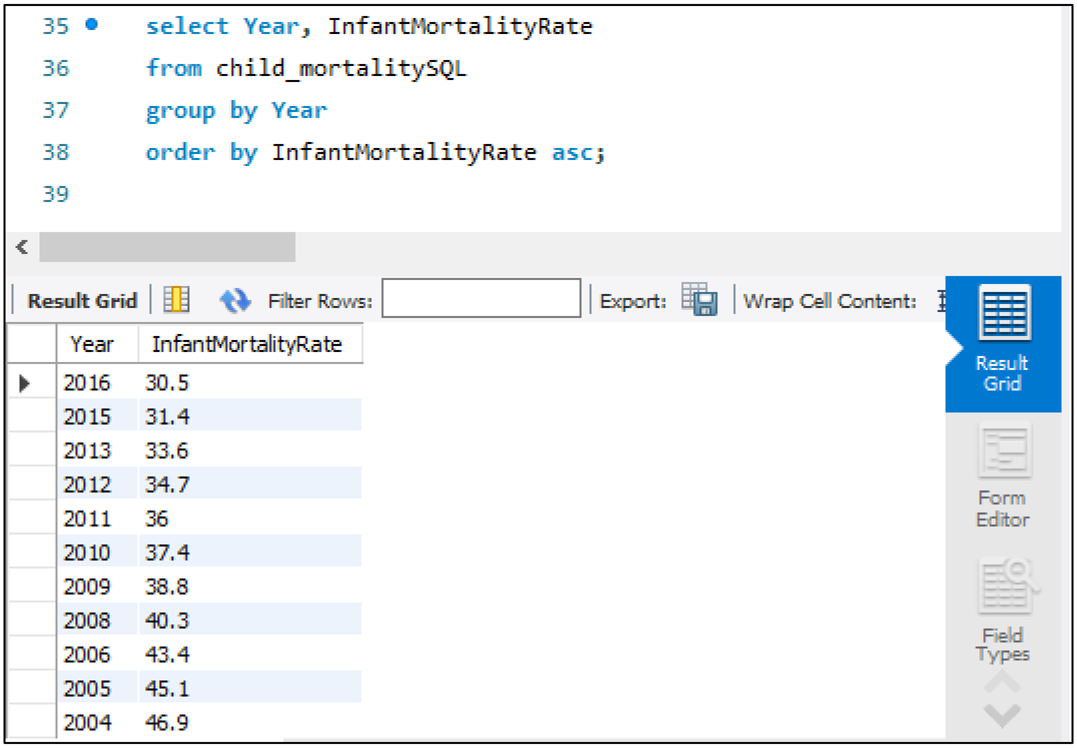
This image shows how I located the empty mortality rates from each row and updated them with their corresponding median value. This is also the entire table displayed (the first part of C).



```
where UnderFiveMortalityRate=0;

update child_mortalitySQL
set InfantMortalityRate=48.6
where InfantMortalityRate=0;

update child_mortalitySQL
set NeonatalMortalityRate=28
where NeonatalMortalityRate=0;

select *
from child_mortalitySQL;
```

Year	UnderFiveMortalityRate	InfantMortalityRate	NeonatalMortalityRate
1990	93.4	64.8	36.8
1991	92.1	63.9	36.3
1992	90.9	63.1	35.9
1993	89.7	62.3	35.4
1994	88.7	61.4	28
1995	87.3	60.5	34.4
1996	85.6	59.4	33.7
1997	69.2	58.2	33.1
1998	82.1	56.9	32.3
1999	79.9	55.4	31.5
2000	77.5	53.9	30.7
2001	74.8	52.1	29.8
2002	72	48.6	28.9
2003	69.2	48.6	28
2004	66.7	46.9	28
2005	69.2	45.1	26.1
2006	61.1	43.4	25.3
2007	58.5	48.6	24.4
2008	56.2	40.3	23.6
2009	53.7	38.8	22.9
2010	69.2	37.4	22.2
2011	49.3	36	21.5
2012	47.3	34.7	20.8
2013	45.5	33.6	20.2
2014	43.7	48.6	19.6
2015	42.2	31.4	19.1
2016	40.8	30.5	18.6

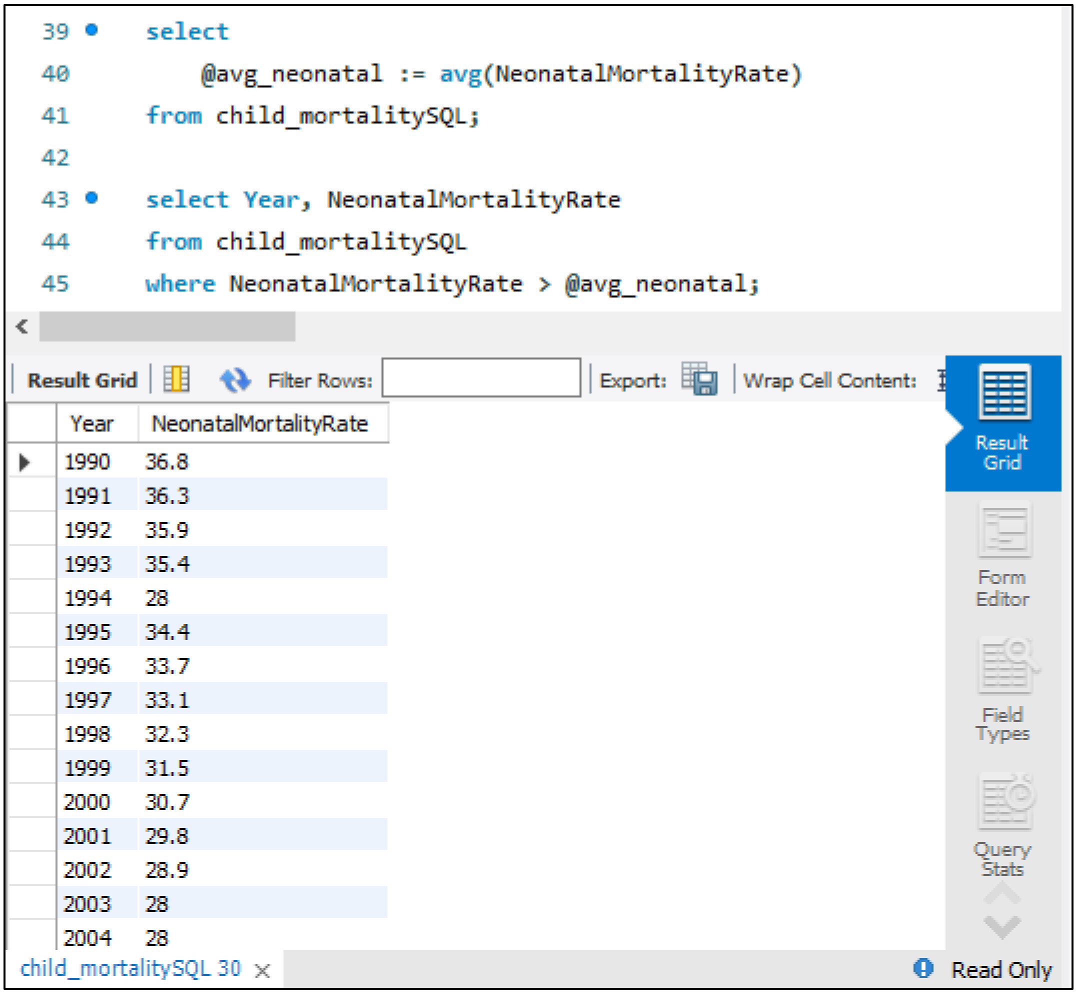
I found that the minimum infant mortality rate is 30.5 in 2016 and the maximum is 64.8 1990. I used the following query to find the minimum and replaced “asc” with “desc” to display the maximum.



```
select Year, InfantMortalityRate
from child_mortalitySQL
group by Year
order by InfantMortalityRate asc;
```

Year	InfantMortalityRate
2016	30.5
2015	31.4
2013	33.6
2012	34.7
2011	36
2010	37.4
2009	38.8
2008	40.3
2006	43.4
2005	45.1
2004	46.9

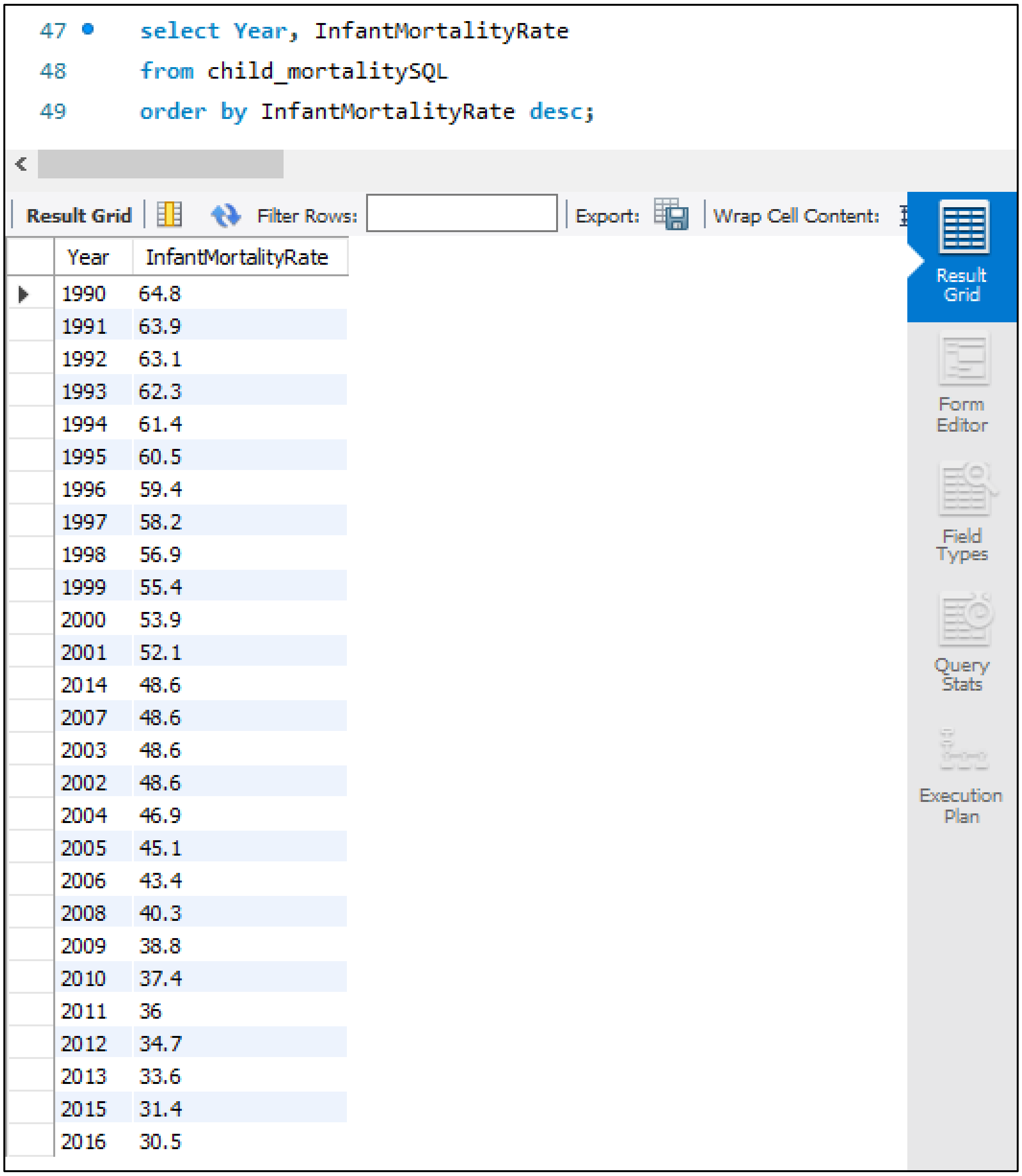
To find which years the neonatal mortality rates were above average, I first found the average of 27.7 and then used a query to display all years greater than it.



```
select
    @avg_neonatal := avg(NeonatalMortalityRate)
from child_mortalitySQL;

select Year, NeonatalMortalityRate
from child_mortalitySQL
where NeonatalMortalityRate > @avg_neonatal;
```

Year	NeonatalMortalityRate
1990	36.8
1991	36.3
1992	35.9
1993	35.4
1994	28
1995	34.4
1996	33.7
1997	33.1
1998	32.3
1999	31.5
2000	30.7
2001	29.8
2002	28.9
2003	28
2004	28

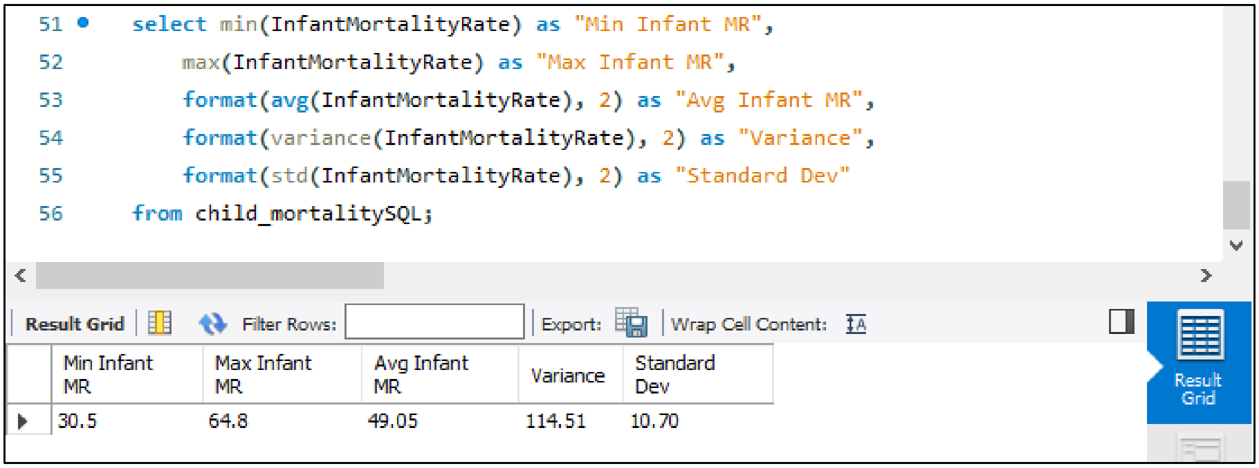
When displaying the sorted infant mortality rates, I noticed that some years would be out of chronological order because of the inputted median in missing values.



```
select Year, InfantMortalityRate
from child_mortalitySQL
order by InfantMortalityRate desc;
```

Year	InfantMortalityRate
1990	64.8
1991	63.9
1992	63.1
1993	62.3
1994	61.4
1995	60.5
1996	59.4
1997	58.2
1998	56.9
1999	55.4
2000	53.9
2001	52.1
2014	48.6
2007	48.6
2003	48.6
2002	48.6
2004	46.9
2005	45.1
2006	43.4
2008	40.3
2009	38.8
2010	37.4
2011	36
2012	34.7
2013	33.6
2015	31.4
2016	30.5

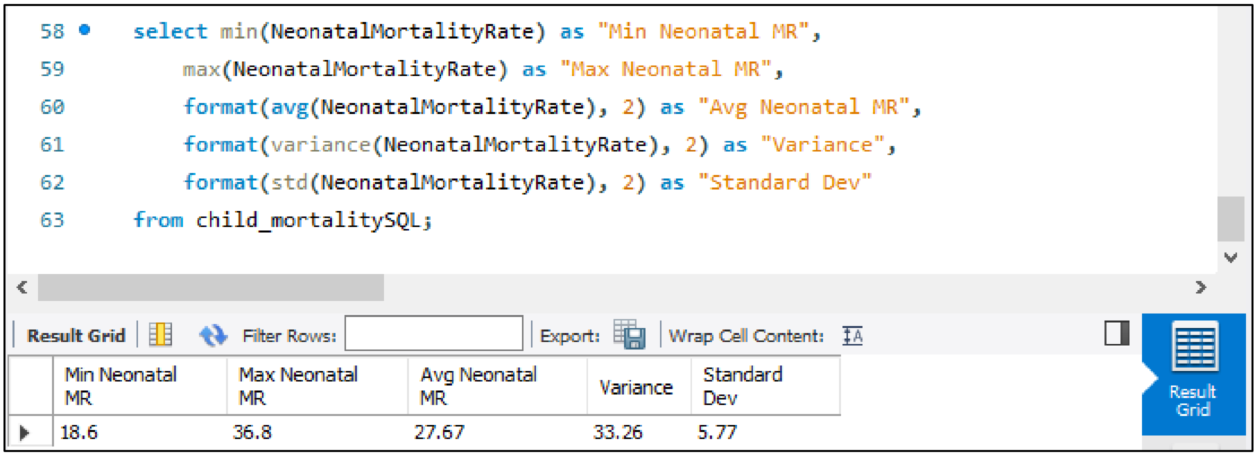
Infant Statistics:



```sql
select min(InfantMortalityRate) as "Min Infant MR",
    max(InfantMortalityRate) as "Max Infant MR",
    format(avg(InfantMortalityRate), 2) as "Avg Infant MR",
    format(variance(InfantMortalityRate), 2) as "Variance",
    format(std(InfantMortalityRate), 2) as "Standard Dev"
from child_mortalitySQL;
```

Min Infant MR	Max Infant MR	Avg Infant MR	Variance	Standard Dev
30.5	64.8	49.05	114.51	10.70

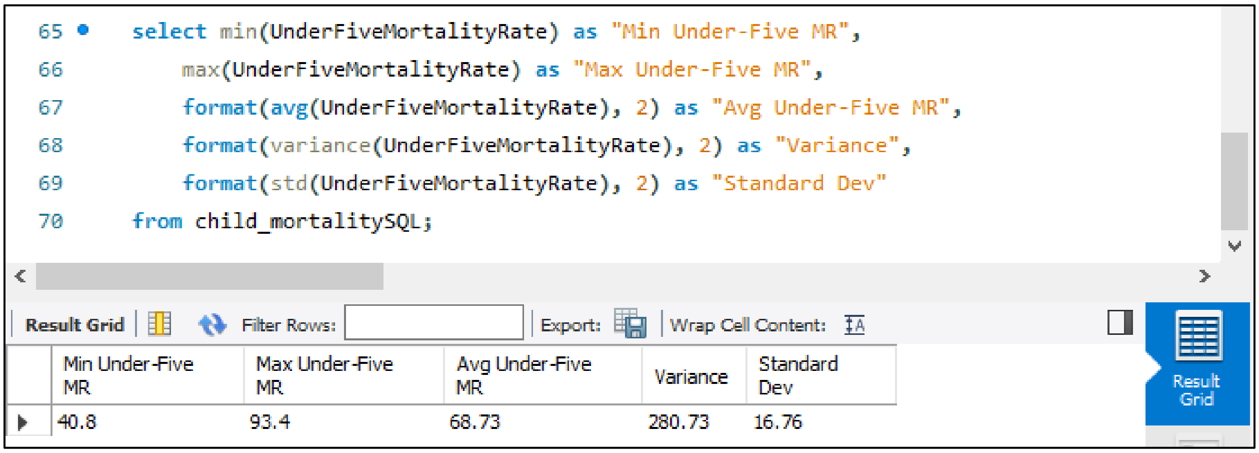
Neonatal Statistics:



```sql
select min(NeonatalMortalityRate) as "Min Neonatal MR",
    max(NeonatalMortalityRate) as "Max Neonatal MR",
    format(avg(NeonatalMortalityRate), 2) as "Avg Neonatal MR",
    format(variance(NeonatalMortalityRate), 2) as "Variance",
    format(std(NeonatalMortalityRate), 2) as "Standard Dev"
from child_mortalitySQL;
```

Min Neonatal MR	Max Neonatal MR	Avg Neonatal MR	Variance	Standard Dev
18.6	36.8	27.67	33.26	5.77

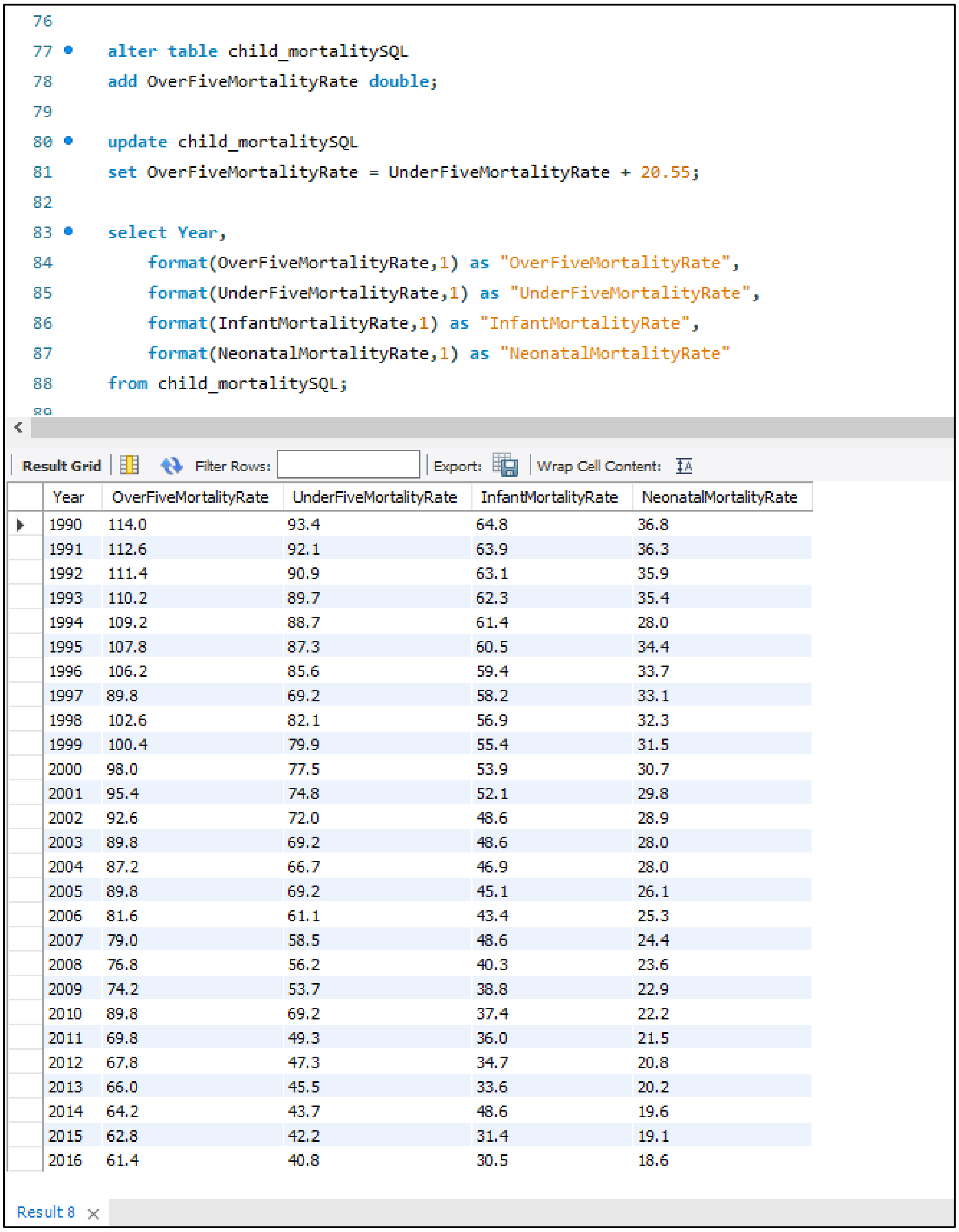
Under-Five Statistics:



```sql
select min(UnderFiveMortalityRate) as "Min Under-Five MR",
    max(UnderFiveMortalityRate) as "Max Under-Five MR",
    format(avg(UnderFiveMortalityRate), 2) as "Avg Under-Five MR",
    format(variance(UnderFiveMortalityRate), 2) as "Variance",
    format(std(UnderFiveMortalityRate), 2) as "Standard Dev"
from child_mortalitySQL;
```

Min Under-Five MR	Max Under-Five MR	Avg Under-Five MR	Variance	Standard Dev
40.8	93.4	68.73	280.73	16.76

To add the above-five mortality rate column, I first found the average value in each column. Then, I took the difference between under-five MR and infant MR (19.7), as well as the difference between infant MR and neonatal MR (21.4). The average of these differences was 20.6, and to find an appropriate over-five mortality rate I added this average to each under-five mortality rate, continuing the trend.



```
alter table child_mortalitySQL
add OverFiveMortalityRate double;

update child_mortalitySQL
set OverFiveMortalityRate = UnderFiveMortalityRate + 20.55;

select Year,
    format(OverFiveMortalityRate,1) as "OverFiveMortalityRate",
    format(UnderFiveMortalityRate,1) as "UnderFiveMortalityRate",
    format(InfantMortalityRate,1) as "InfantMortalityRate",
    format(NeonatalMortalityRate,1) as "NeonatalMortalityRate"
from child_mortalitySQL;
```

Year	OverFiveMortalityRate	UnderFiveMortalityRate	InfantMortalityRate	NeonatalMortalityRate
1990	114.0	93.4	64.8	36.8
1991	112.6	92.1	63.9	36.3
1992	111.4	90.9	63.1	35.9
1993	110.2	89.7	62.3	35.4
1994	109.2	88.7	61.4	28.0
1995	107.8	87.3	60.5	34.4
1996	106.2	85.6	59.4	33.7
1997	89.8	69.2	58.2	33.1
1998	102.6	82.1	56.9	32.3
1999	100.4	79.9	55.4	31.5
2000	98.0	77.5	53.9	30.7
2001	95.4	74.8	52.1	29.8
2002	92.6	72.0	48.6	28.9
2003	89.8	69.2	48.6	28.0
2004	87.2	66.7	46.9	28.0
2005	89.8	69.2	45.1	26.1
2006	81.6	61.1	43.4	25.3
2007	79.0	58.5	48.6	24.4
2008	76.8	56.2	40.3	23.6
2009	74.2	53.7	38.8	22.9
2010	89.8	69.2	37.4	22.2
2011	69.8	49.3	36.0	21.5
2012	67.8	47.3	34.7	20.8
2013	66.0	45.5	33.6	20.2
2014	64.2	43.7	48.6	19.6
2015	62.8	42.2	31.4	19.1
2016	61.4	40.8	30.5	18.6

Problem 2 XML/JSON: To change the file into JSON, I used the table data export wizard. For XML, I used an online converter to transfer the JSON file to an XML file.

JSON to XML converter: https://www.convertjson.com/json-to-xml.htm