Matt Caraher

Dr. Forouraghi

Project 3

1 March 2022

Project 3

Part 1 MySQL: Importing Life_Expectancy.csv took about 10-15 minutes, and I started the SQL file by selecting my database and setting safe updates to 0. Running the first command, I deleted the rows that had a population equal to 0. There were 252 rows affected.

```
use ids4db;
set SQL_SAFE_UPDATES = 0;

delete from Life_Expectancy
where population = 0;
```

To continue cleaning the data, I searched through each column and found that most of them had at least a few missing values and decided to replace them all with the average for each year. For example, I found the average life expectancy for the year 2013 and then replaced missing life expectancy values from the year 2013 with that average. I decided to find the averages dependent on year because I figured it would help to identify patterns in the data over the years. Here is the first column affected (life expectancy) and the year 2013, the same was done for each other column by each year respectively:

```
#Data cleaning
select Country, Year
from Life_Expectancy
where Life_Expectancy = 0 and Year = 2013;

select
    @LifeExpectancy := avg(Life_Expectancy)
from Life_Expectancy
where Year = 2013;

update Life_Expectancy
set Life_Expectancy = @Life_Expectancy
where Year = 2013 and Life_Expectancy = 0;
```

Output

Action Output

#	Time	Action	Message
38	19:08:29	select Country, Year from Life_Expectancy where Life_...	2 row(s) returned
39	19:14:50	select Country, Year from Life_Expectancy where Life_...	2 row(s) returned
40	19:15:09	select @LifeExpectancy := avg(Life_Expectancy) from ...	1 row(s) returned
41	19:15:49	update Life_Expectancy set Life_Expectancy = @Life_...	2 row(s) affected Rows matched: 2 C

For Alcohol, Percentage Expenditure, and Total Expenditure, in year 2015, there were over 100 values equal to 0 in these rows, and so I had to specify that the avg variable is only taken from the values over 0. With the other columns and years, this was irrelevant because they each had 2 rows missing at most, so the effect of one 0 missing value was not problematic when I tested. Here is an example:

```sql
#Alcohol 2015
select
    @Alcohol2015 := avg(Alcohol)
from Life_Expectancy
where Year = 2015 and Alcohol > 0;

update Life_Expectancy
set Alcohol = @Alcohol2015
where Year = 2015 and Alcohol = 0;
```

To display the total count of countries after the cleaning, I used the following command to get 145 countries:

```sql
select count(distinct Country) as "Total Countries"
from Life_Expectancy;
```

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

Total Countries
145

To find the min and max of each mortality rate and list them by year in a table, I did the following (with the other year's commands that were cut off in the picture):

```sql
select Year as "Year (first min, then max)", Country, Adult_Mortality
from Life_Expectancy
where Adult_Mortality = (select min(Adult_Mortality) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, Adult_Mortality
from Life_Expectancy
where Adult_Mortality = (select max(Adult_Mortality) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, Adult_Mortality
from Life_Expectancy
where Adult_Mortality = (select min(Adult_Mortality) from Life_Expectancy where Year = 2011) and Year = 2011
union all
select Year, Country, Adult_Mortality
from Life_Expectancy
```

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

Year (first min, then max)	Country	Adult_Mortality
2010	Fiji	2
2010	Haiti	682
2011	Italy	6
2011	Zimbabwe	464
2012	Senegal	2
2012	Lesotho	513
2013	Iceland	5
2013	Lesotho	518
2014	Kiribati	2
2014	Lesotho	522
2015	Tunisia	1
2015	Lesotho	484

I did the same commands for the next tasks but replacing mortality rates with population, GDP, or schooling:

```sql
select Year as "Year (first min, then max)", Country, Population
from Life_Expectancy
where Population = (select min(Population) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, Population
from Life_Expectancy
where Population = (select max(Population) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, Population
from Life_Expectancy
where Population = (select min(Population) from Life_Expectancy where Year = 2011) and Year = 2011
union all
select Year, Country, Population
from Life_Expectancy
where Population = (select max(Population) from Life_Expectancy where Year = 2011) and Year = 2011
union all
select Year, Country, Population
from Life_Expectancy
```

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

Year (first min, then max)	Country	Population
2010	Hungary	123
2010	Indonesia	242524123
2011	Maldives	377
2011	Brazil	198686688
2012	Maldives	385
2012	Indonesia	248883232
2013	Palau	292
2013	Pakistan	181712595
2014	Maldives	41
2014	India	1293859294
2015	Sri Lanka	2966
2015	Indonesia	258162113

```sql
select Year as "Year (first min, then max)", Country, GDP
from Life_Expectancy
where GDP = (select min(GDP) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, GDP
from Life_Expectancy
where GDP = (select max(GDP) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, GDP
from Life_Expectancy
where GDP = (select min(GDP) from Life_Expectancy where Year = 2011) and Year = 2011
union all
select Year, Country, GDP
from Life_Expectancy
where GDP = (select max(GDP) from Life_Expectancy where Year = 2011) and Year = 2011
union all
select Year, Country, GDP
from Life_Expectancy
```

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

Year (first min, then max)	Country	GDP
2010	Mauritius	8.376432
2010	Norway	87646.75346
2011	Senegal	18.25321
2011	Luxembourg	115761.577
2012	Guinea	52.3485646
2012	Switzerland	83164.38795
2013	Tajikistan	14.214412
2013	Luxembourg	113751.85
2014	Romania	12.27733
2014	Luxembourg	119172.7418
2015	Burundi	33.681223
2015	Australia	56554.3876

```sql
select Year as "Year (first min, then max)", Country, Schooling
from Life_Expectancy
where Schooling = (select min(Schooling) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, Schooling
from Life_Expectancy
where Schooling = (select max(Schooling) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, Schooling
from Life_Expectancy
where Schooling = (select min(Schooling) from Life_Expectancy where Year = 2011) and Year = 2011
union all
select Year, Country, Schooling
from Life_Expectancy
where Schooling = (select max(Schooling) from Life_Expectancy where Year = 2011) and Year = 2011
union all
select Year, Country, Schooling
from Life_Expectancy
```

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

Year (first min, then max)	Country	Schooling
2010	Niger	4.5
2010	Australia	19.5
2011	Niger	4.8
2011	Australia	19.8
2012	South Sudan	4.9
2012	Australia	20.1
2013	South Sudan	4.9
2013	Australia	20.3
2014	South Sudan	4.9
2014	Australia	20.4
2015	South Sudan	4.9
2015	Australia	20.4

For the Alcohol min-max table, there were a lot of repeated 0.01 values that tied for the min spot:

```sql
select Year as "Year (first min, then max)", Country, Alcohol
from Life_Expectancy
where Alcohol = (select min(Alcohol) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, Alcohol
from Life_Expectancy
where Alcohol = (select max(Alcohol) from Life_Expectancy where Year = 2010) and Year = 2010
union all
select Year, Country, Alcohol
from Life_Expectancy
where Alcohol = (select min(Alcohol) from Life_Expectancy where Year = 2011) and Year = 2011
union all
select Year, Country, Alcohol
from Life_Expectancy
where Alcohol = (select max(Alcohol) from Life_Expectancy where Year = 2011) and Year = 2011
union all
select Year, Country, Alcohol
from Life_Expectancy
where Alcohol = (select min(Alcohol) from Life_Expectancy where Year = 2012) and Year = 2012
```

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

Year (first min, then max)	Country	Alcohol
2010	Afghanistan	0.01
2010	Bangladesh	0.01
2010	Mauritania	0.01
2010	Estonia	14.97
2011	Afghanistan	0.01
2011	Bangladesh	0.01
2011	Estonia	0.01
2011	Fiji	0.01
2011	Mauritania	0.01
2011	Mongolia	0.01
2011	Belarus	17.31
2012	Afghanistan	0.01
2012	Azerbaijan	0.01
2012	Bangladesh	0.01

To decide if there is a correlation between population and life expectancy, I first tried sorting the countries by population and then by life expectancy. There was no clear trend, and so I edited the previous min-max chart for population and added the life expectancy column. With the following chart showing the min and max population from each year, the life expectancy was higher every year in the countries with smaller populations.

Year (first min, then max)	Country	Population	Life_Expectancy
2010	Hungary	123	74.5
2010	Indonesia	242524123	68.1
2011	Maldives	377	77.3
2011	Brazil	198686688	74.1
2012	Maldives	385	77.6
2012	Indonesia	248883232	68.5
2013	Palau	292	NULL
2013	Pakistan	181712595	66
2014	Maldives	41	78.2
2014	India	1293859294	68
2015	Sri Lanka	2966	74.9
2015	Indonesia	258162113	69.1

Part 2 Python: The following picture is of my setup commands in Jupyter Notebooks. I had to install tabulate and the SQL connector, and the bottom of the picture includes the inputted column names.

```
[6]: import pandas as pd

[9]: pip install tabulate

Collecting tabulate
  Downloading tabulate-0.8.9-py3-none-any.whl (25 kB)
Installing collected packages: tabulate
Successfully installed tabulate-0.8.9
Note: you may need to restart the kernel to use updated packages.

[10]: from tabulate import tabulate

[11]: pip install mysql-connector-python

Requirement already satisfied: mysql-connector-python in c:\users\matt\anaconda3\lib\site-packages (8.0.28)
Requirement already satisfied: protobuf>=3.0.0 in c:\users\matt\anaconda3\lib\site-packages (from mysql-connector-python) (3.19.4)
Note: you may need to restart the kernel to use updated packages.

[12]: import mysql.connector as sql

[15]: db_connection = sql.connect(host='208.109.18.154', database='ids4db', user='ids4', password='Wst2394')
db_cursor = db_connection.cursor()
db_cursor.execute('SELECT * FROM Life_Expectancy')
table_rows = db_cursor.fetchall()
LifeExpDF = pd.DataFrame(table_rows)
LifeExpDF.columns = ['Country','Year','Life_Expectancy' ,'Adult_Mortality','Alcohol','Percentage_Expenditure', 'BMI' ,'Total_Expenditure','GDP','Population','Schooling']
print(tabulate(LifeExpDF, headers='keys', tablefmt='fancy_grid'))
```

	Country	Year	Life_Expectancy	Adult_Mortality	Alcohol	Percentage_Expenditure	BMI	Total_Expenditure	GDP	Population	Schooling
0	Afghanistan	2010	58.8	279	0.01	79.6794	16.7	9.2	553.329	2883167	9.2
1	Afghanistan	2011	59.2	275	0.01	7.09711	17.2	7.87	63.5372	2978599	9.5
2	Afghanistan	2012	59.5	272	0.01	78.1842	17.6	8.52	669.959	3696958	9.8
3	Afghanistan	2013	59.9	268	0.01	73.2192	18.1	8.13	631.745	31731688	9.9
4	Afghanistan	2014	59.9	271	0.01	73.5236	18.6	8.18	612.697	327582	10
5	Afghanistan	2015	65	263	0.01	71.2796	19.1	8.16	584.259	33736494	10.1
6	Albania	2010	76.2	91	5.28	41.8228	54.3	5.34	494.359	291321	12.5

I then looked at how adult mortality rates affect life expectancy. The Pearson r value being -0.69788 indicates that there is a somewhat strong negative correlation between adult mortality and life expectancy. As adult mortality goes up, life expectancy goes down.

```
# Calculate Pearson's correlation between adult mortality and life expectancy

from scipy.stats import pearsonr

data1 = LifeExpDF['Life_Expectancy']
data2 = LifeExpDF['Adult_Mortality']
```

```
corr, _ = pearsonr(data1, data2)
print('Pearsons correlation: %.5f' % corr)

Pearsons correlation: -0.69788
```

To test the correlation between life expectancy and eating habits, I compared life expectancy to BMI. With a Pearson's correlation of 0.4528, I know that these two have a weak but positive correlation.

```
In [21]: # Correlation between life expectancy and eating habits (BMI)

         lifeExpectancyCol = LifeExpDF['Life_Expectancy']
         bmiCol = LifeExpDF['BMI']

         lifeExpectancyCol = np.nan_to_num(lifeExpectancyCol)

         corr1, _ = pearsonr(lifeExpectancyCol, bmiCol)
         print('Pearsons correlation : %.5f' % corr1)

         Pearsons correlation : 0.45280
```

Life expectancy and alcohol have a weak positive correlation of 0.41422, like BMI. This number may be less or even negative if the data set included a greater number of extremes, in this case alcoholics. This is because alcoholics likely do not live as long as non-alcoholics.

```
In [24]: # Correlation between life expectancy and eating habits (BMI)

         data1 = LifeExpDF['Life_Expectancy']
         data2 = LifeExpDF['Alcohol']

         data1 = np.nan_to_num(data1)

         corr1, _ = pearsonr(data1, data2)
         print('Pearsons correlation : %.5f' % corr1)

         Pearsons correlation : 0.41422
```

To test the relation of a social factor and life expectancy, I chose schooling. This gave a strong positive correlation of 0.74597. This suggests that as the amount of schooling a sample has goes up, so does their life expectancy. Population had a correlation of -0.02695.

```
In [27]: # Correlation between life expectancy and schooling

         data1 = LifeExpDF['Life_Expectancy']
         data2 = LifeExpDF['Schooling']

         data1 = np.nan_to_num(data1)

         corr1, _ = pearsonr(data1, data2)
         print('Pearsons correlation : %.5f' % corr1)

         Pearsons correlation : 0.74597
```

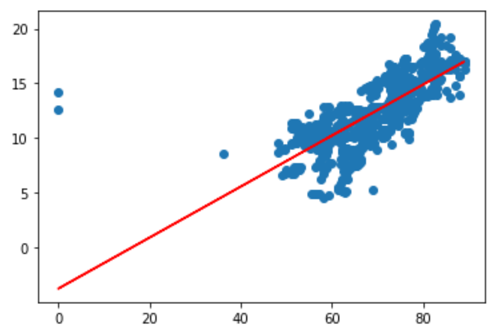
The last relation I tested was economic factors, by comparing GDP to life expectancy. This gave a positive correlation of 0.44083. This is a very weak correlation much like alcohol an BMI to life expectancy.

```
In [29]: # Correlation between life expectancy and GDP

         data1 = LifeExpDF['Life_Expectancy']
         data2 = LifeExpDF['GDP']

         data1 = np.nan_to_num(data1)

         corr1, _ = pearsonr(data1, data2)
         print('Pearsons correlation : %.5f' % corr1)

         Pearsons correlation : 0.44083
```

I then compared total expenditure to life expectancy and got 0.17442, meaning that it has a very weak positive correlation. The strongest correlation seen is schooling to life expectancy, which previously showed that more schooling means a higher life expectancy.

The next step I took was to plot a linear regression model for life expectancy and schooling, to show the positive correlation. Below is also the line equation along with the $R2$ and MSE values:

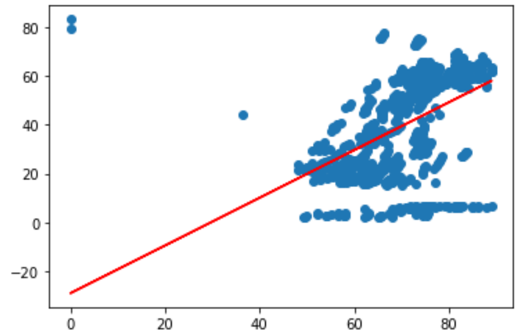
```
In [77]: # Linear Regression between life expectancy and schooling
         from sklearn.linear_model import LinearRegression

         # Save life expectancy in X
         X = LifeExpDF.iloc[:, 2].values.reshape(-1,1)
         X = np.nan_to_num(X)

         # Save schooling in Y
         Y = LifeExpDF.iloc[:, 10].values.reshape(-1,1)

         # Create model
         LRmodel = LinearRegression()

         # Perform linear regression
         LRmodel.fit(X, Y)

         # Make predictions
         Y_pred = LRmodel.predict(X)

         # Visualize the dataset and the regression line:
         plt.scatter(X, Y)
         plt.plot(X, Y_pred, color='red')

         plt.show()
```




```
In [90]: # Equation of the line
         print("The slope: ", LRmodel.coef_)
         print("The intercept: ", LRmodel.intercept_)
         print()

         # Error analysis: MSE and R2
         from sklearn.metrics import mean_squared_error, r2_score

         print("MSE: ", mean_squared_error(Y, Y_pred))
         print("R2: ", r2_score(Y, Y_pred))

         The slope:  [[0.23308557]]
         The intercept:  [-3.75920734]

         MSE:  3.91800952000835
         R2:  0.5564667277917688
```

Although BMI has a weak correlation, it is the next highest in the relations I tested. Here is a linear regression model along with R2 and MSE values:

In [78]:
```
# Linear Regression between life expectancy and schooling
from sklearn.linear_model import LinearRegression

# Save life expectancy in X
X = LifeExpDF.iloc[:, 2].values.reshape(-1,1)
X = np.nan_to_num(X)

# Save schooling in Y
Y = LifeExpDF.iloc[:, 6].values.reshape(-1,1)

# Create model
LRmodel = LinearRegression()

# Perform linear regression
LRmodel.fit(X, Y)

# Make predictions
Y_pred = LRmodel.predict(X)

# Visualize the dataset and the regression line:
plt.scatter(X, Y)
plt.plot(X, Y_pred, color='red')

plt.show()
```



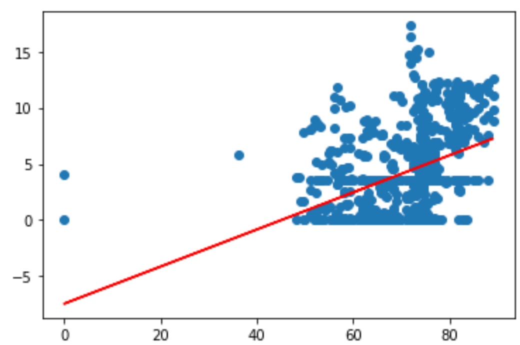
In [86]:
```
# Equation of the line
print("The slope: ", LRmodel.coef_)
print("The intercept: ", LRmodel.intercept_)
print()

# Error analysis: MSE and R2
from sklearn.metrics import mean_squared_error, r2_score

print("MSE: ", mean_squared_error(Y, Y_pred))
print("R2: ", r2_score(Y, Y_pred))
```

```
The slope:  [[0.97937808]]
The intercept:  [-29.05118008]

MSE:  336.4977643451747
R2:  0.20502985069323476
```

The last linear regression model I made was of life expectancy and alcohol, which has slightly less of a correlation than BMI. This chart, along with BMI, helped to target some of the outliers that still existed even after cleansing the data of 0's and null values. Below is the line equation along with the R2 and MSE values:

In [79]:

```
# Linear Regression between life expectancy and schooling
from sklearn.linear_model import LinearRegression

# Save life expectancy in X
X = LifeExpDF.iloc[:, 2].values.reshape(-1,1)
X = np.nan_to_num(X)

# Save schooling in Y
Y = LifeExpDF.iloc[:, 4].values.reshape(-1,1)

# Create model
LRmodel = LinearRegression()

# Perform linear regression
LRmodel.fit(X, Y)

# Make predictions
Y_pred = LRmodel.predict(X)

# Visualize the dataset and the regression line:
plt.scatter(X, Y)
plt.plot(X, Y_pred, color='red')

plt.show()
```



In [84]:

```
# Equation of the line
print("The slope: ", LRmodel.coef_)
print("The intercept: ", LRmodel.intercept_)
print()

# Error analysis: MSE and R2
from sklearn.metrics import mean_squared_error, r2_score

print("MSE: ", mean_squared_error(Y, Y_pred))
print("R2: ", r2_score(Y, Y_pred))
```

```
The slope:  [[0.16523573]]
The intercept:  [-7.45210736]

MSE:  11.927643414389264
R2:  0.1715750457910561
```

Out of the three models I created, I found that the first one, life expectancy and schooling, performs the best. This model had the highest R2 value meaning it was the most accurate line relating to the data, and the MSE was the lowest meaning that the forecast line is the most accurate of the models.