In [ ]:
```
#Import required libraries
import pandas as pd
import os
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sqlalchemy import create_engine
engine = create_engine('sqlite:///:memory:')
```

In [ ]:
```
#Importing Data
db_dir = os.getcwd()
db_dir = db_dir + '/MMA860_Assignment1_Data_vf.xlsx'
dfWindfarmTurbineLocations = pd.read_excel(db_dir, sheet_name='Windfarm Turbine Locations')
dfTurbineSpecifications = pd.read_excel(db_dir, sheet_name='Turbine Specifications')
dfIslandAirportWeather = pd.read_excel(db_dir, sheet_name='Island Airport Weather',skiprows=16)
dfProductSales = pd.read_excel(db_dir, sheet_name='Product_Sales')
dfCollinearity = pd.read_excel(db_dir, sheet_name='Collinearity')
```

In [ ]:
```
#question 1a
# Since our weather data doesn’t include “air density”, you will need to calculate it
# for each hour of data using the following formula and save it to a new variable and print out the first 5 records
# Air density (kg/m^3)=([Stn Press (kPa)]  ×1000)/(287.05×([Temp (°C)]+273.15))

dfIslandAirportWeather['Air density'] = (dfIslandAirportWeather['Stn Press (kPa)']
                                          *1000)/(287.05*(dfIslandAirportWeather['Temp (°C)']+273.15))
dfIslandAirportWeather['Air density'].head(5)
```

Out[ ]:
```
0    1.236659
1    1.232853
2    1.230567
3    1.247307
4    1.253480
Name: Air density, dtype: float64
```

In [ ]:
```
# b.    Windspeed [Wind Spd (km/h)] will need to be converted from km/h to meters per second for
#       each hour of data (1 km/h ≈ 0.277778 m/s). Save this to a new variable and print out the
#       first 5 results. Include entries even if no power was generated. 2 marks

dfIslandAirportWeather['Wind Spd (m/s)'] = (dfIslandAirportWeather['Wind Spd (km/h)']*0.277778)
#Quality Check
selected_rows = dfIslandAirportWeather[['Wind Spd (km/h)','Wind Spd (m/s)']]
selected_rows.head(5)
```

Out[ ]:

	Wind Spd (km/h)	Wind Spd (m/s)
0	11	3.055558
1	4	1.111112
2	28	7.777784
3	34	9.444452
4	28	7.777784

In [ ]:
```
# c.    To calculate the power production at each turbine, use the following formula, keeping the following points 
#       and print out the first 5 records. 3 marks
#       power (watts)=[Air density]× [Turbine swept area]  ×0.5[Wind speed (m/s)]^3  ×[Maximum power coefficient]
            # Points:
            # i.        If the windspeed is above or below the minimum and maximum cutoff speeds, the wind turbines
            # ii.       Each turbine cannot produce more than [Turbine nominal power] at any given point in time. N
            #       given in Megawatts, where your power calculation is in watts (1 MW = 1,000,000 watts)

import numpy as np
TurbineSweptArea = dfTurbineSpecifications[dfTurbineSpecifications['Field'] == 'Turbine swept area'][['Values']]
MinCutoffSpeed = dfTurbineSpecifications[dfTurbineSpecifications['Field'] == 'Cut-in windspeed (m/s)'][['Values']]
MaxCutoffSpeed = dfTurbineSpecifications[dfTurbineSpecifications['Field'] == 'Cut-out windspeed (m/s)'][['Values']]
Maximumpowercoefficient = dfTurbineSpecifications[dfTurbineSpecifications['Field'] == 'Maximum power coefficient'][
#conversion Megawatts to Watts
Turbinenominalpower = dfTurbineSpecifications[dfTurbineSpecifications['Field'] == 'Turbine nominal power'][['Values

#covert for sql
dfIslandAirportWeather.to_sql('dfIslandAirportWeather', con=engine, if_exists='replace', index=False)
MinCutoffSpeed.to_sql('MinCutoffSpeed', con=engine, if_exists='replace', index=False)
MaxCutoffSpeed.to_sql('MaxCutoffSpeed', con=engine, if_exists='replace', index=False)
Maximumpowercoefficient.to_sql('Maximumpowercoefficient', con=engine, if_exists='replace', index=False)
Turbinenominalpower.to_sql('Turbinenominalpower', con=engine, if_exists='replace', index=False)
```

```
TurbineSweptArea.to_sql('TurbineSweptArea', con=engine, if_exists='replace', index=False)

newdfIslandAirportWeather = pd.read_sql_query(''' with df as(
                        SELECT *
                            ,case when [Wind Spd (m/s)] >= (select * from MinCutoffSpeed)
                                and [Wind Spd (m/s)] <= (select * from MaxCutoffSpeed)
                                then [Air density] * (select * from TurbineSweptArea) * power([Wind Spd (m/s)],3)
                                * (select * from Maximumpowercoefficient)
                            else 0
                                end as PowerProduction
                        FROM dfIslandAirportWeather
                        where [Wind Spd (m/s)] > (select * from MinCutoffSpeed) and [Wind Spd (m/s)] < (select * fr
                        )
                        SELECT *
                            ,case when PowerProduction > (select * from Turbinenominalpower) then (select * from Tu
                                else PowerProduction
                                end as [Power (Watts)]
                        from df
                  ''', con=engine)


newdfIslandAirportWeather = newdfIslandAirportWeather.drop(columns=['PowerProduction'])
newdfIslandAirportWeather['Power (Watts)'].head(5)
```

Out[ ]:

```
0    2.689784e+06
1    4.000000e+06
2    2.739869e+06
3    1.729322e+06
4    3.393210e+06
Name: Power (Watts), dtype: float64
```

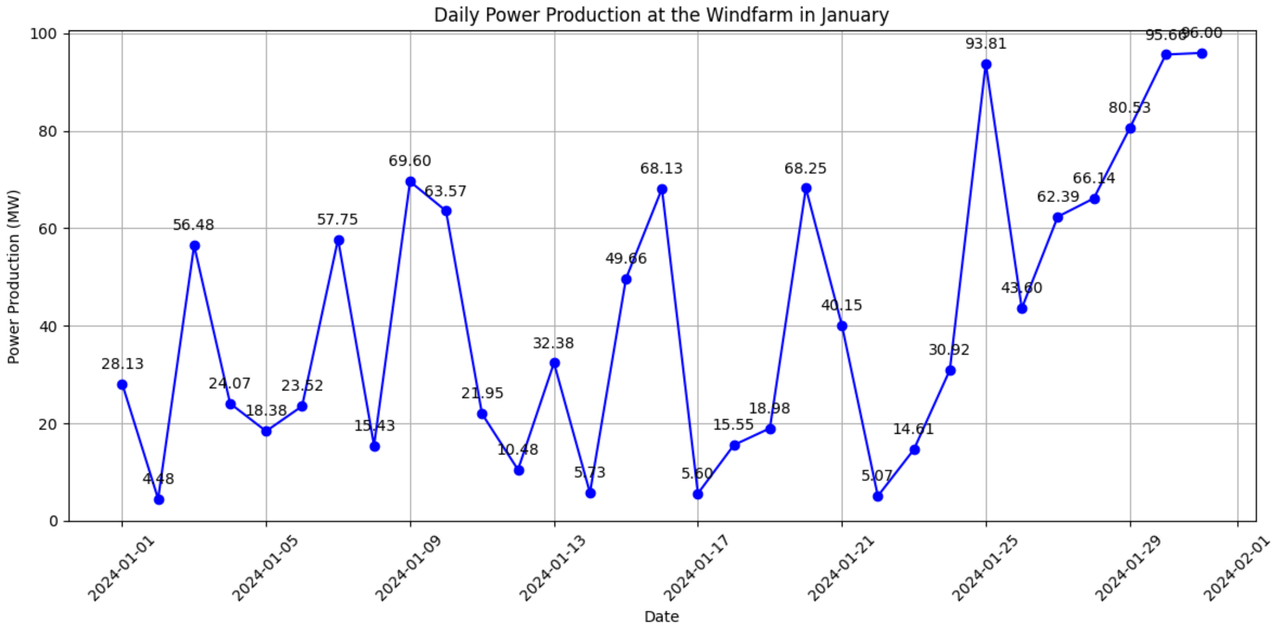
In [ ]:

```
# d. What is the total amount of electricity produced for the entire windfarm in January in Megawatts?
# Hint: it’s the sum of all power produced part c, above, converted to Megawatts. 1 mark

newdfIslandAirportWeather.to_sql('newdfIslandAirportWeather', con=engine, if_exists='replace', index=False)
df = pd.read_sql('''SELECT *, cast(sum([Power (Watts)]) as float)/1000000 as [Power (MW)]
                        FROM newdfIslandAirportWeather
                  ''', con=engine)
newdf = pd.read_sql('''SELECT *, cast([Power (Watts)] /1000000 as float) as [Power (MW)]
                        FROM newdfIslandAirportWeather
                  ''', con=engine)
print('Amount of electricity produced for the entire windfarm in January in Megawatts: '+ str(df['Power (MW)'].valu
```

```
Amount of electricity produced for the entire windfarm in January in Megawatts: 1287.0164146369561 Watts
```

In [ ]:

```
# e.    Create a visualization using the best practices we discussed showing power produced per day at the windfarm
newdf['Date'] = newdf.apply(lambda row: pd.to_datetime(f"2024-{row['Month']}-{row['Day']}-{row['Time']}"), axis=1)
newdf = newdf.resample('D', on='Date')['Power (MW)'].sum().reset_index()

plt.figure(figsize=(12, 6))
plt.plot(newdf['Date'], newdf['Power (MW)'], marker='o', linestyle='-',color='b')
for i, txt in enumerate(newdf['Power (MW)']):
    plt.annotate(f"{txt:.2f}", (newdf['Date'].iloc[i], newdf['Power (MW)'].iloc[i]),
                 textcoords="offset points", xytext=(0,10), ha='center')
plt.title('Daily Power Production at the Windfarm in January')
plt.xlabel('Date')
plt.ylabel('Power Production (MW)')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Question 3

In [ ]:
```
# a.    Make sure all variable formats are correct, and that Product_IDs are all of the same length, and a string. 
dfProductSales.dtypes
length = dfProductSales['Product_ID'].astype(str).str.len().max()
dfProductSales['Product_ID_New'] = dfProductSales['Product_ID'].astype(str).str.zfill(length)
dfProductSales.head(5)
```

Out[ ]:

	Obs	Product_ID	Sales_2016	Sales_2017	Import	Num_Retailers	Price	Product_ID_New
0	1	1	1162.91	235.19	1	5	$67.18	001
1	2	2	1191.11	944.87	1	3	$54.56	002
2	3	3	1214.96	737.06	0	5	$58.85	003
3	4	4	1336.07	986.15	0	7	$56.48	004
4	5	5	1343.29	871.33	1	7	$58.74	005

In [ ]:
```
dfProductSales.dtypes
```

Out[ ]:
```
Obs                 int64
Product_ID          int64
Sales_2016        float64
Sales_2017        float64
Import              int64
Num_Retailers       int64
Price              object
Product_ID_New     object
dtype: object
```

In [ ]:
```
# b.    Tidy the dataset (i.e., make sure all columns are unique variables, all rows are unique observations,
#       and there is a single data point in each cell). 2 marks
print()
dfProductSales = dfProductSales.drop_duplicates()
#from above dtypes price needs to be float
dfProductSales['Price'] = dfProductSales['Price'].replace(r'\$','',regex = True).astype(float)
dfProductSales
```

Out[ ]:

	Obs	Product_ID	Sales_2016	Sales_2017	Import	Num_Retailers	Price	Product_ID_New
0	1	1	1162.91	235.19	1	5	67.18	001
1	2	2	1191.11	944.87	1	3	54.56	002
2	3	3	1214.96	737.06	0	5	58.85	003
3	4	4	1336.07	986.15	0	7	56.48	004
4	5	5	1343.29	871.33	1	7	58.74	005
...	...	...	...	...	...	...	...	...
195	196	196	1334.69	879.47	0	7	58.43	196
196	197	197	2035.08	1251.54	0	21	64.51	197
197	198	198	1390.20	1327.56	1	6	51.16	198
198	199	199	1968.63	2656.05	0	15	37.27	199
199	200	200	1357.59	1206.39	1	6	52.80	200

200 rows × 8 columns

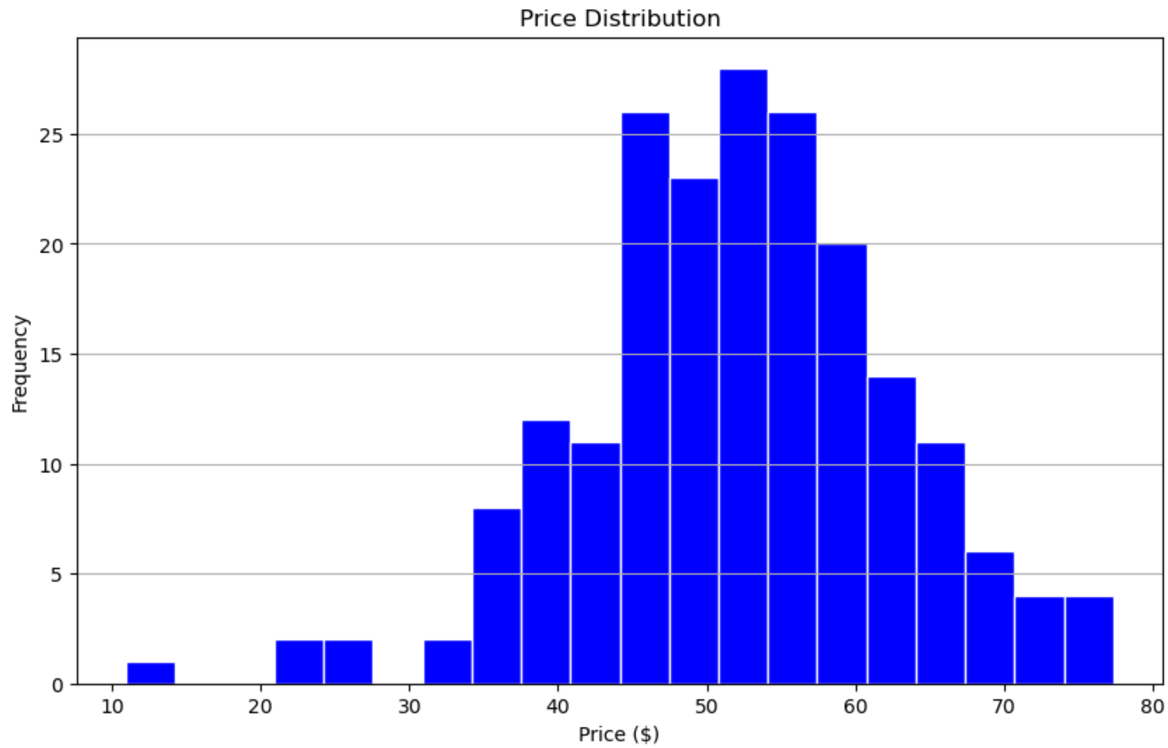
In [ ]:

```
print(dfProductSales.dtypes)
print(" ")
print("Count of Missing Values:  " +str(dfProductSales.isnull().sum()))
```

```
Obs                 int64
Product_ID          int64
Sales_2016        float64
Sales_2017        float64
Import              int64
Num_Retailers       int64
Price             float64
Product_ID_New     object
dtype: object

Count of Missing Values:  Obs               0
Product_ID        0
Sales_2016        0
Sales_2017        0
Import            0
Num_Retailers     0
Price             0
Product_ID_New    0
dtype: int64
```

In [ ]:

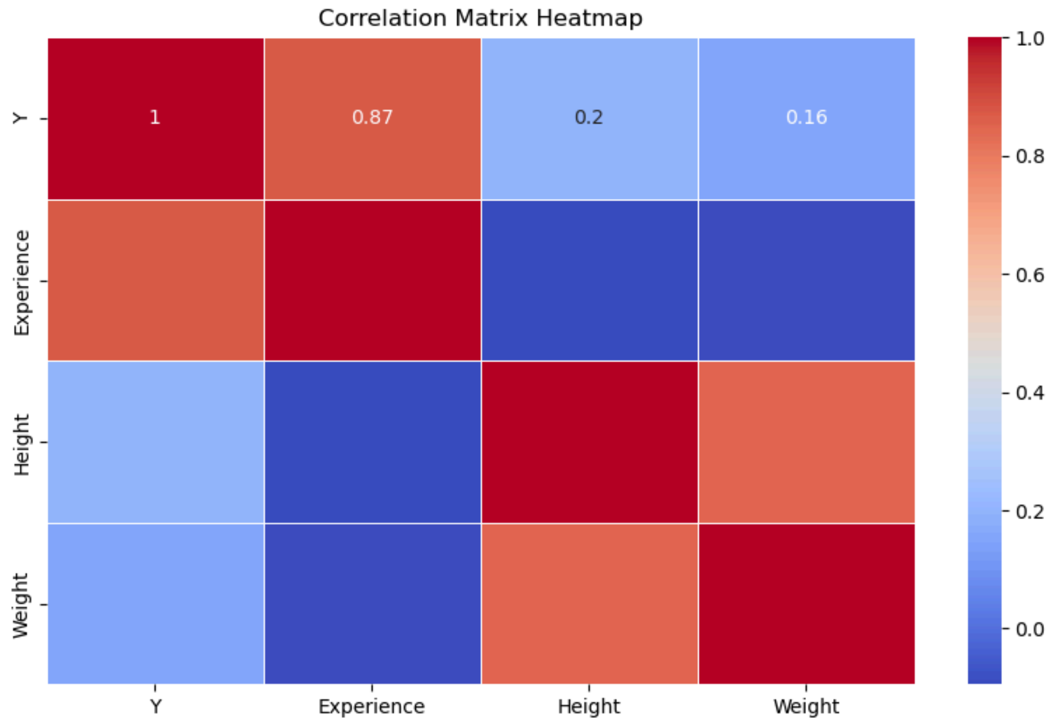
```
# c.    Create a visualization of price with appropriate labels. 2 marks

import matplotlib.pyplot as plt
# Histogram of the Price variable
plt.figure(figsize=(10, 6))
plt.hist(dfProductSales['Price'], bins=20, color='Blue', edgecolor='white')
plt.title('Price Distribution')
plt.xlabel('Price ($)')
plt.ylabel('Frequency')
plt.grid(axis='y')

plt.show()
```



In [ ]:
```
# d.    Calculate the correlation between price and number of retailers (In python: np.corrcoef()). Produce a scatt
#       and explain what the correlation means in practical terms (i.e., to a business owner)  3 marks

correlation = np.corrcoef(dfProductSales['Price'],dfProductSales['Num_Retailers'])
correlation = correlation[0, 1]
print(f"Correlation between  price and number of retailers: ", correlation)
```

```
Correlation between  price and number of retailers:  -0.050234651544592175
```

In [ ]:
```
# e.    Tell a story with the data - produce an easy to understand visualization and describe the insights you've i
#       Make sure the visualization can stand on its own without explanation, and that your story has a specific bu

import seaborn as sns
import statsmodels.api as sm

# Regression for all 100 observations
y_100 = dfCollinearity['Y']
X1_100 = dfCollinearity[['Experience', 'Height']]
X2_100 = dfCollinearity[['Experience', 'Weight']]
X3_100 = dfCollinearity[['Experience', 'Height', 'Weight']]

model1_100 = run_regression(y_100, X1_100)
model2_100 = run_regression(y_100, X2_100)
model3_100 = run_regression(y_100, X3_100)


plt.figure(figsize=(10, 6))
corr_matrix = dfCollinearity[['Y', 'Experience', 'Height', 'Weight']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



There is a high correlation between two things (height and weight) and (Experience and Y).

In [ ]:
```
# 4a.Collinearity: collections of variables that tend to move together, such as height and weight, are called colli
#       Using the data found on the tab ‘Collinear’:
    # a.        Filter the data to consider only the first 25 observations then run the following models; repeat th
    # heteroscedasticity, etc. You must run 6 regressions in total – i, ii, and iii with 25 observations + i, ii, i
        # i.    Run a linear regression to explain y in terms of experience and height.  Does height appear to expl
        # ii.   Run a linear regression to explain y in terms of experience and weight.  Does weight appear to expl
        # iii.  Run a linear regression to explain y in terms of experience and height and weight. Do height and we

import statsmodels.api as sm
from statsmodels.formula.api import ols

dfCollinearity25 = pd.read_excel(db_dir, sheet_name='Collinearity',nrows=25)

print('25 Observations: Y ~ Height + Experience')
model = ols('Y ~ Height + Experience',dfCollinearity25).fit()
print(model.summary())
```

```
Y ~ Height + Experience
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.878
Model:                            OLS   Adj. R-squared:                  0.867
Method:                 Least Squares   F-statistic:                     79.49
Date:                Sat, 29 Jun 2024   Prob (F-statistic):           8.56e-11
Time:                        17:19:34   Log-Likelihood:                -138.40
No. Observations:                  25   AIC:                             282.8
Df Residuals:                      22   BIC:                             286.5
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -39.5389     32.685     -1.210      0.239    -107.323      28.245
Height         1.6693      0.533      3.133      0.005       0.564       2.774
Experience   101.7732     11.530      8.827      0.000      77.861     125.685
==============================================================================
Omnibus:                        1.284   Durbin-Watson:                   2.380
Prob(Omnibus):                  0.526   Jarque-Bera (JB):                0.951
Skew:                           0.174   Prob(JB):                        0.621
Kurtosis:                       2.110   Cond. No.                         132.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Yes, height appears to explain Y since the P Value of height 0.005 is less than 0.05 making it significant. When height increases, Y is going to increase by 1.6693 units.

In [ ]:
```
print('25 Observations: Y ~ Weight + Experience')
model = ols('Y ~ Weight + Experience',dfCollinearity25).fit()
print(model.summary())
```

```
Y ~ Weight + Experience
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.855
Model:                            OLS   Adj. R-squared:                  0.842
Method:                 Least Squares   F-statistic:                     65.01
Date:                Sat, 29 Jun 2024   Prob (F-statistic):           5.83e-10
Time:                        17:32:44   Log-Likelihood:                -140.58
No. Observations:                  25   AIC:                             287.2
Df Residuals:                      22   BIC:                             290.8
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -69.9322     40.894     -1.710      0.101    -154.742      14.877
Weight         1.5973      0.735      2.174      0.041       0.073       3.121
Experience   110.3854     11.721      9.417      0.000      86.077     134.694
==============================================================================
Omnibus:                        0.590   Durbin-Watson:                   2.436
Prob(Omnibus):                  0.744   Jarque-Bera (JB):                0.678
Skew:                           0.283   Prob(JB):                        0.713
Kurtosis:                       2.426   Cond. No.                         150.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Yes, weight appears to explain Y since the P Value of height 0.04 is less than 0.05 making it significant. When weight increases, Y is going to increase by 1.5973 units.

In [ ]:
```
print('25 Observations: Y ~ Height + Weight + Experience')
model = ols('Y ~ Weight + Experience + Height',dfCollinearity25).fit()
print(model.summary())
```

```
Y ~ Weight + Experience
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.880
Model:                            OLS   Adj. R-squared:                  0.862
Method:                 Least Squares   F-statistic:                     51.17
Date:                Sat, 29 Jun 2024   Prob (F-statistic):           7.89e-10
Time:                        17:35:11   Log-Likelihood:                -138.27
No. Observations:                  25   AIC:                             284.5
Df Residuals:                      21   BIC:                             289.4
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -26.5362     43.584     -0.609      0.549    -117.173      64.101
Weight        -0.5822      1.260     -0.462      0.649      -3.202       2.038
Experience   101.1227     11.826      8.551      0.000      76.529     125.716
Height         2.0557      0.997      2.063      0.052      -0.017       4.128
==============================================================================
Omnibus:                        1.275   Durbin-Watson:                   2.351
Prob(Omnibus):                  0.529   Jarque-Bera (JB):                0.932
Skew:                           0.150   Prob(JB):                        0.628
Kurtosis:                       2.103   Cond. No.                         238.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Weight does not have a significant impact to Y since P Values 0.649 > 0.05 Height does not have a significant impact to Y since P Values 0.052 > 0.05.

In [ ]:
```
print('100 Observations: Y ~ Height+ Experience')
model = ols('Y ~ Height + Experience',dfCollinearity).fit()
print(model.summary())
```

```
100 Observations: Y ~ Height+ Experience
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.840
Model:                            OLS   Adj. R-squared:                  0.837
Method:                 Least Squares   F-statistic:                     255.0
Date:                Sat, 29 Jun 2024   Prob (F-statistic):           2.36e-39
Time:                        18:03:11   Log-Likelihood:                -540.07
No. Observations:                 100   AIC:                             1086.
Df Residuals:                      97   BIC:                             1094.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.9241     16.868      0.055      0.956     -32.555      34.403
Height         1.2762      0.184      6.935      0.000       0.911       1.641
Experience    96.2549      4.365     22.051      0.000      87.592     104.918
==============================================================================
Omnibus:                        0.048   Durbin-Watson:                   2.425
Prob(Omnibus):                  0.977   Jarque-Bera (JB):                0.080
Skew:                           0.046   Prob(JB):                        0.961
Kurtosis:                       2.897   Cond. No.                         170.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Height does have a significant impact to Y since P Values 0.000 < 0.05

In [ ]:
```
print('100 Observations: Y ~ Weight +Experience')
model = ols('Y ~ Weight + Experience',dfCollinearity).fit()
print(model.summary())
```

```
100 Observations: Y ~ Weight +Experience
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.817
Model:                            OLS   Adj. R-squared:                  0.814
Method:                 Least Squares   F-statistic:                     217.0
Date:                Sat, 29 Jun 2024   Prob (F-statistic):           1.55e-36
Time:                        18:01:14   Log-Likelihood:                -546.75
No. Observations:                 100   AIC:                             1100.
Df Residuals:                      97   BIC:                             1107.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -10.4614     20.496     -0.510      0.611     -51.141      30.218
Weight         1.5527      0.284      5.472      0.000       0.989       2.116
Experience    95.6817      4.665     20.512      0.000      86.424     104.940
==============================================================================
Omnibus:                        0.883   Durbin-Watson:                   2.311
Prob(Omnibus):                  0.643   Jarque-Bera (JB):                0.589
Skew:                           0.183   Prob(JB):                        0.745
Kurtosis:                       3.090   Cond. No.                         177.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Weight does have a significant impact to Y since P Values 0.000 < 0.05

Weight does not have a significant impact to Y since P Values 0.851 > 0.05 Height does have a significant impact to Y since P Values 0.000 < 0.05

b. Consider the results you have found from the work in a. Write a paragraph or two to explain to your manager the patterns you observed with respect to the significance of the t-statistics, why these results occurred, and the strategies for using explanatory variables that exhibit collinearity. 3 marks

When analyzing the regression analysis summary for the 3 items (25 Observations): Height, Weight, Height + Weight having on its significance on Y. The significance of Height and Weight independently have an influence Y. But, when looking at both Height and Weight together. Weight does not have significant impact to Y.

When comparing the models using the 25 and 100 observations, the results are very different. Looking at only 25 observations, each data point has more leverage to skew the results. By having a larger oberservations, it can reduce the influence of each data point and increase the reliability of the results.