

### Question 3 parts (b) through (h)

```
In [1]: import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import numpy as np
import re
```

```
In [2]: # Load the CSV file into a DataFrame
import pandas as pd

df = pd.read_csv('auto.txt', sep='\t')

pd.set_option('display.max_rows', None) # Display all rows
pd.set_option('display.max_columns', None) # Display all columns

df.head()
```

Out[2]:	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0 8 307.0 130.0 3504. 12...	chevrolet chevelle malibu	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	15.0 8 350.0 165.0 3693. 11...	buick skylark 320	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	18.0 8 318.0 150.0 3436. 11...	plymouth satellite	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	16.0 8 304.0 150.0 3433. 12...	amc rebel sst	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	17.0 8 302.0 140.0 3449. 10...	ford torino	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Reading the auto.txt file with relevant data cleaning steps, realised after hit-and-trial.

```
In [3]: file_path = 'auto.txt'

with open(file_path, 'r') as file:
    lines = file.readlines()

list_of_row_data = list()
columns_comma_separated = re.sub(r'\s+', ', ', lines[0])
columns_list = columns_comma_separated.split(",")[:-1]

for line in lines[1:]:

    metrics = line.split("\t")[0]
    metrics_comma_separated_string = re.sub(r'\s+', ', ', metrics)
    metrics_list = [float(value) if value != "NA" else np.nan for value in metrics_c
    last_col = line.split("\t")[1]
```

```

# print(metrics_list)
# print(metrics)
# print(last_col)

metrics_list.append(last_col)
list_of_row_data.append(metrics_list)

df = pd.DataFrame(list_of_row_data, columns=columns_list)

```

```

In [4]: pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
df.head()

```

```

Out[4]:
   mpg  cylinders  displacement  horsepower  weight  acceleration  year  origin  name
0   18.0         8.0         307.0         130.0   3504.0           12.0   70.0   1.0  "chevrolet chevelle malibu"
1   15.0         8.0         350.0         165.0   3693.0           11.5   70.0   1.0  "buick skylark 320"
2   18.0         8.0         318.0         150.0   3436.0           11.0   70.0   1.0  "plymouth satellite"
3   16.0         8.0         304.0         150.0   3433.0           12.0   70.0   1.0  "amc rebel sst"
4   17.0         8.0         302.0         140.0   3449.0           10.5   70.0   1.0  "ford torino"

```



```

In [ ]:

```

```

In [ ]:

```

```

In [5]: df.isnull().sum()

```

```

Out[5]: mpg          0
cylinders          0
displacement       0
horsepower         5
weight             0
acceleration       0
year              0
origin             0
name              0
dtype: int64

```

```

In [ ]:

```

```

In [6]: # Create the independent/derived variable year-squared

```

```
df['year_sq'] = df['year']**2
```

```
In [7]: df[ [ "year", "year_sq" ] ].head()
```

```
Out[7]:
```

	year	year_sq
0	70.0	4900.0
1	70.0	4900.0
2	70.0	4900.0
3	70.0	4900.0
4	70.0	4900.0

## Creating the uncentered regression model

```
In [8]: X_0 = df[['year' , 'year_sq']]
        y_0 = df['mpg']

# Add a constant, initialized to 1, to the independent variable. This will be the i
X_0 = sm.add_constant(X_0)
```

```
In [9]: # Fit the linear regression model
        model = sm.OLS(y_0, X_0).fit()

# Print the summary table of the regression model
        print(model.summary())
```

## OLS Regression Results

```

=====
Dep. Variable:          mpg      R-squared:            0.369
Model:                  OLS      Adj. R-squared:        0.366
Method:                 Least Squares      F-statistic:         115.4
Date:                   Tue, 08 Oct 2024    Prob (F-statistic):    3.61e-40
Time:                   02:29:25           Log-Likelihood:       -1288.1
No. Observations:       397             AIC:                 2582.
Df Residuals:           394             BIC:                 2594.
Df Model:                2
Covariance Type:        nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
const	577.2523	146.671	3.936	0.000	288.896	865.609
year	-15.8409	3.865	-4.098	0.000	-23.440	-8.242
year_sq	0.1123	0.025	4.419	0.000	0.062	0.162

```

=====
Omnibus:                21.346      Durbin-Watson:         0.809
Prob(Omnibus):           0.000      Jarque-Bera (JB):       18.130
Skew:                    0.446      Prob(JB):               0.000116
Kurtosis:                2.450      Cond. No.               2.73e+06
=====

```

### Notes:

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

[2] The condition number is large, 2.73e+06. This might indicate that there are strong multicollinearity or other numerical problems.

In [ ]:

```

In [10]: correlation_coefficient = df['year'].corr(df['year_sq'])
         correlation_coefficient

```

Out[10]: 0.999759011614808

```

In [11]: year_mean = np.mean(df['year'])
         year_mean

```

Out[11]: 75.99496221662469

## Creating centered year

```

In [12]: df['year_centered'] = df['year'] - year_mean
         df['year_centered'].head()

```

```

Out[12]: 0    -5.994962
         1    -5.994962
         2    -5.994962
         3    -5.994962
         4    -5.994962
         Name: year_centered, dtype: float64

```

In [ ]:

```
In [13]: # Creating derived variable for centered year squared
df['year_centered_sq'] = df['year_centered']**2
df[['year_centered_sq', 'year_centered']].head()
```

Out[13]:

	year_centered_sq	year_centered
0	35.939572	-5.994962
1	35.939572	-5.994962
2	35.939572	-5.994962
3	35.939572	-5.994962
4	35.939572	-5.994962

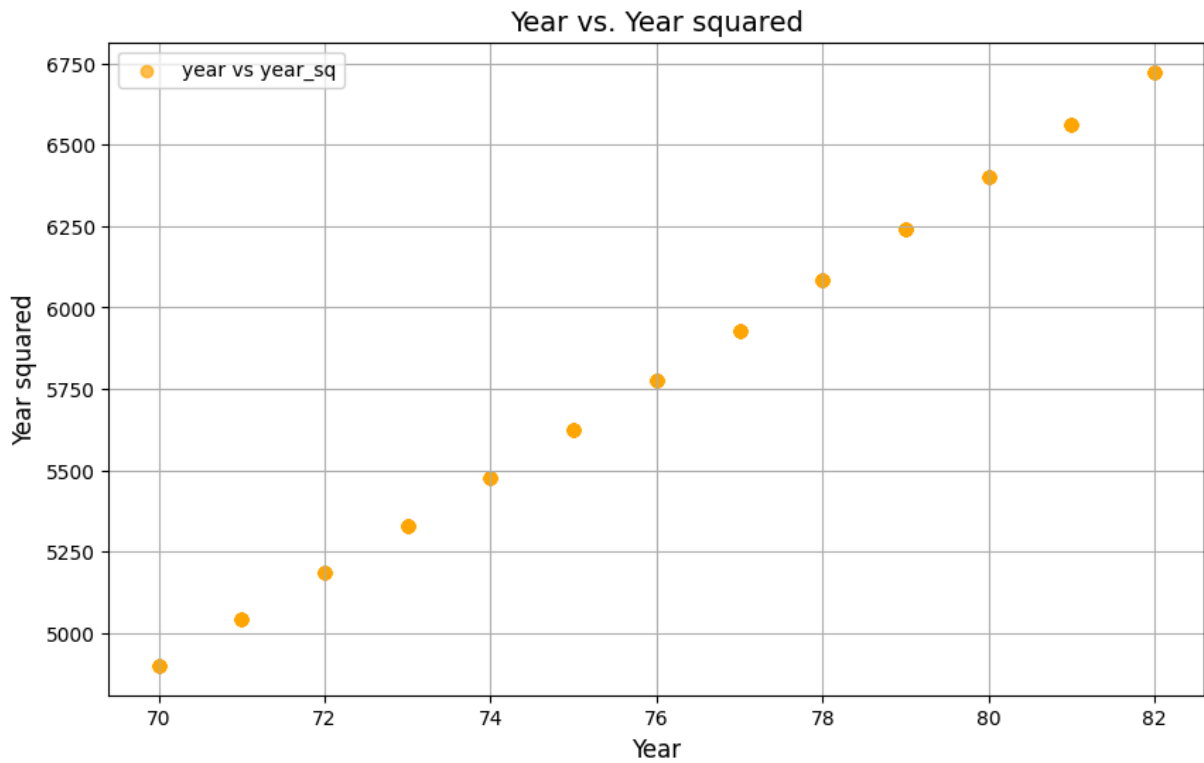
### Calculating correlation

```
In [14]: correlation_coefficient = df['year_centered'].corr(df['year_centered_sq'])
correlation_coefficient
```

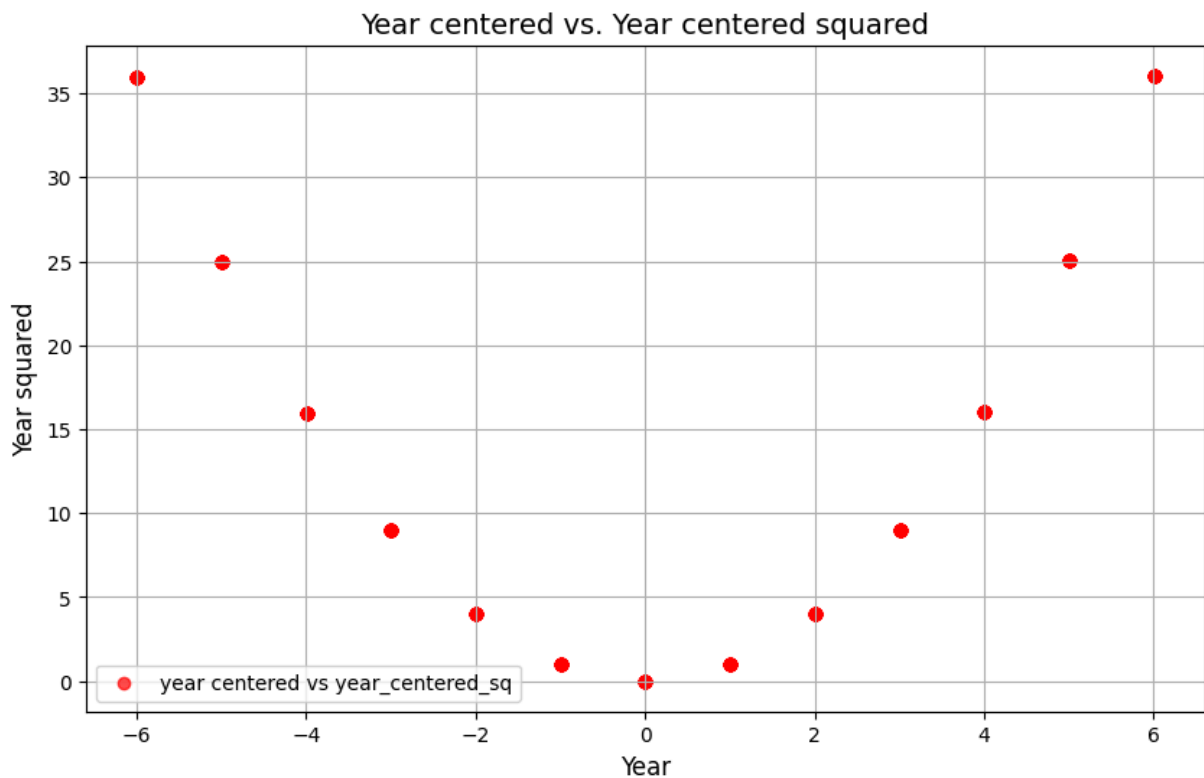
Out[14]: 0.014413995959565213

## Creating required scatter plots

```
In [15]: plt.figure(figsize=(10, 6))
plt.scatter(df['year'], df['year_sq'], color='orange', label='year vs year_sq', alp
# plt.scatter(df['year_centered'], df['year_centered_sq'], color='orange', label='Ap
plt.title('Year vs. Year squared', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Year squared', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```



```
In [16]: plt.figure(figsize=(10, 6))
plt.scatter(df['year_centered'], df['year_centered_sq'], color='red', label='year c
# plt.scatter(df['year_centered'], df['year_centered_sq'], color='orange', label='Ap
plt.title('Year centered vs. Year centered squared', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Year squared', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```



## Creating the centered model

```
In [17]: X_1 = df[['year_centered' , 'year_centered_sq']]
y_1 = df['mpg']

# Add a constant, initialized to 1, to the independent variable. This will be the i
X_1 = sm.add_constant(X_1)
```

In [ ]:

```
In [18]: # Fit the linear regression model
model = sm.OLS(y_1, X_1).fit()

# Print the summary table of the regression model
print(model.summary())
```



# OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.369			
Model:	OLS	Adj. R-squared:	0.366			
Method:	Least Squares	F-statistic:	115.4			
Date:	Tue, 08 Oct 2024	Prob (F-statistic):	3.61e-40			
Time:	02:29:25	Log-Likelihood:	-1288.1			
No. Observations:	397	AIC:	2582.			
Df Residuals:	394	BIC:	2594.			
Df Model:	2					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
-----						
const	21.9906	0.466	47.214	0.000	21.075	22.906
year_centered	1.2278	0.085	14.469	0.000	1.061	1.395
year_centered_sq	0.1123	0.025	4.419	0.000	0.062	0.162
=====						
Omnibus:	21.346	Durbin-Watson:	0.809			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18.130			
Skew:	0.446	Prob(JB):	0.000116			
Kurtosis:	2.450	Cond. No.	27.3			
=====						

## Notes:

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

In [ ]:

As can be observed ->

beta2 = gamma2 i.e. beta2 = 0.1123

evaluating ->

- beta1 = gamma1 - 2 \* gamma2 \* year\_mean

-----

gamma2 = 0.1123

gamma1 = 1.2278

year\_mean = 75.99

In [19]:

```
gamma1 = 1.2278
gamma2 = 0.1123
year_mean = 75.99

# b2= gamma2
# b1 = gamma1 - 2*gamma2

beta2 = gamma1 - 2*gamma2*year_mean
print(beta2, round(beta2,2))
```

-15.839553999999998 -15.84

As can be seen, the values derived for beta1 and beta2 in terms of gamma1 and gamma2 in the first part of the problem have been verified by the Betas produced by the OLS models for un-centered and centered model.

In [ ]:

In [ ]:

In [ ]:

In [ ]: