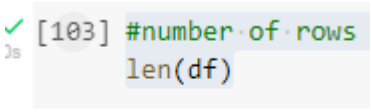
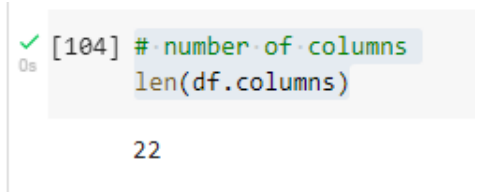
FIT1043 Assignment 2: Solution

A1. Dataset size
How many rows and columns exist in this dataset?
Solution:
There are 9140 rows and 22 columns in the dataset.
Code:

```
#number of rows
len(df)

# number of columns
len(df.columns)
```

Output:





A2. The number of unique values in some columns
Count the number of unique values for National Remoteness Areas, SA4 Name 2016, National LGA Name 2017, and National Road Type in this dataset.
Solution (Code):

```
# Counting the number of unique value in National Remoteness Areas
len(pd.unique(df['National Remoteness Areas']))

# Counting the number of unique value in SA4 Name 2016
len(pd.unique(df['SA4 Name 2016']))

# Counting the number of unique value in National LGA Name 2017
len(pd.unique(df['National LGA Name 2017']))

# Counting the number of unique value in National Road Type
len(pd.unique(df['National Road Type']))
```

Output:

```
# Counting the number of unique value in National Remoteness Areas
len(pd.unique(df['National Remoteness Areas']))
```

```
6
```

```
[107] # Counting the number of unique value in SA4 Name 2016
len(pd.unique(df['SA4 Name 2016']))
```

```
89
```

```
[108] # Counting the number of unique value in National LGA Name 2017
len(pd.unique(df['National LGA Name 2017']))
```

```
501
```

```
# Counting the number of unique value in National Road Type
len(pd.unique(df['National Road Type']))
```

```
10
```

A3. Missing values and duplicates

A3.1 How many rows contain missing values (Unspecified or Undetermined or blank) in this dataset?

Solution:

There are 2287 rows contain missing values (Unspecified or Undetermined or blank) in this dataset.

Code:

```
# rows containing missing values (Unspecified or Undetermined or
sum([True for idx,row in df.iterrows() if any(row.isnull())])
```

Output:

```
# rows containing missing values (Unspecified or Undetermined or
sum([True for idx,row in df.iterrows() if any(row.isnull())])
```

```
2287
```

A3.2 List the months with no missing values in them.

Solution:

There are no months with missing values.

Code:

```
# Listing the months with no missing values in them (A3.2)
```

```
bool_series = pd.notnull(df["YYYYMM"])
df[bool_series]
```

Output:

```
9140 rows × 22 columns
```

A3.3 Remove the records with missing values.
Code:

```
#Remove the records with missing values.
data=df.dropna()
```

A3.4 Remove duplicates as well after removing the missing values

Code:

```
#A3.4 Remove duplicates as well after removing the missing values
data1=data.drop_duplicates()
```

A4. Number of crashes in each month
List the number of crashes in each month. In which two months are the number of crashes at their largest?
Solution: In January and March 2019, the number of crashes are at their largest.
Code:

```
#List the number of crashes in each month.
data1.groupby('YYYYMM')['Crash ID'].count().sort_values(ascending=False).reset_index(name
='count')
```

Output:

```
#List the number of crashes in each month.
data1.groupby('YYYYMM')['Crash ID'].count().sort_values(ascending=False).reset_index(name='count')
```

	YYYYMM	count
0	201901	115
1	201903	112
2	201803	112
3	201912	107
4	201508	106
...	...	...
88	201408	21
89	201404	21
90	201405	19
91	201402	18
92	201401	17

93 rows × 2 columns

A5. Investigating crashes over different months for specific road user
A5.1 Compute the average number of crashes against Month for car drivers.

a. Extract Year and Month as separate columns
b. Compute the number of crashes by both Year and Month for car drivers
c. Based on task A5-1-b result, compute again the average number of crashes against Month. For each month, the average number of crashes is calculated over different years for which we have collected data for.

Solution (Code):

```
#changing the format of date
data1['YYYYMM']=pd.to_datetime(data1['YYYYMM'],format='%Y%m', errors='coerce')

#A5.1.a
#creating year column
data1['YYYY'] = data1['YYYYMM'].dt.year
data1['MM'] = data1['YYYYMM'].dt.month

#A.5.1.b
#no. of crashes by year for car drivers
data2=data1.groupby('YYYY')['Road User'].apply(lambda x: (x=='Car driver').sum()).reset_inde
x(name='No. of Crashes yearly by Car drivers')

#A.5.1.b
```

```
#no. of crashes monthly for car drivers
data3=data1.groupby('MM')['Road User'].apply(lambda x: (x=='Car driver').sum()).reset_index(name='Average crashes monthly by car drivers')
data3['Average crashes monthly by car drivers'] = (data3['Average crashes monthly by car drivers'] / 8).round(2)
display(data3)
```

Output:

	MM	Average crashes monthly by car drivers
0	1	34.38
1	2	28.88
2	3	39.38
3	4	33.38
4	5	32.88
5	6	33.62
6	7	34.12
7	8	38.88
8	9	34.25
9	10	30.75
10	11	30.62
11	12	33.88

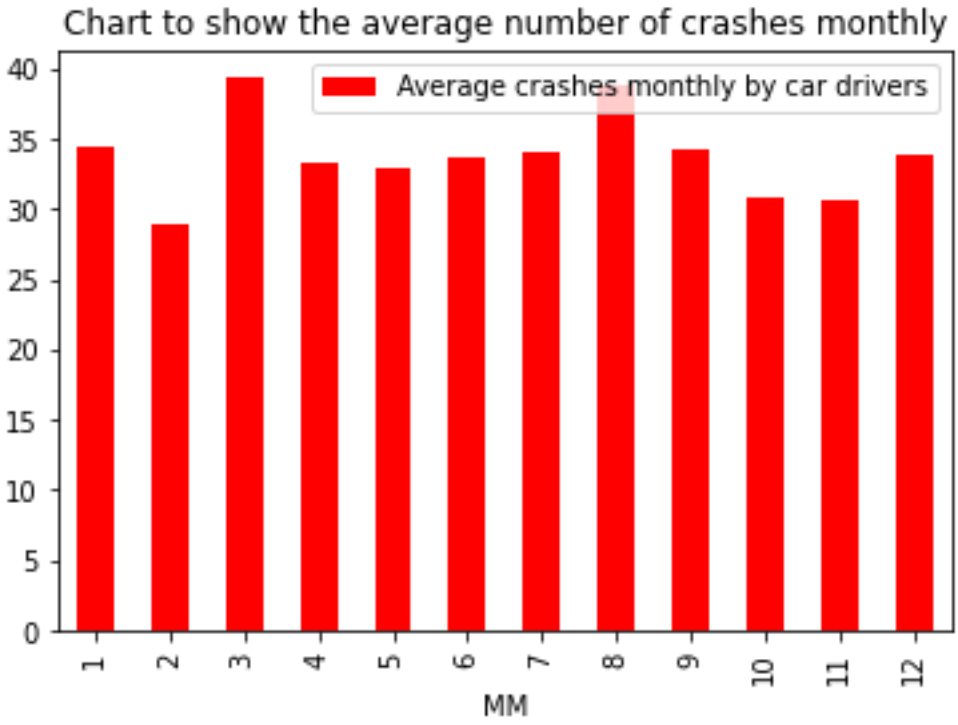
A5.2 Draw a chart showing the average number of crashes over different months computed in task A5-1.

Code:

```
#A5.2
#Draw a chart showing the average number of crashes over different months computed in task A 5-1
import matplotlib.pyplot as plt
data3.plot(kind = 'bar',
    x = 'MM',
    y = 'Average crashes monthly by car drivers',
    color = 'red')

# set the title
plt.title('Chart to show the average number of crashes monthly')
```

```
# show the plot
plt.show()
```

Output:



A5.3 Discuss any interesting point in the chart.
Solution:
The average of crashes is highest in the month of March of every year followed by the month of August as it is second highest.
The average of crashes is lowest in the month of February of every year.

A6. Exploring Speed, National Road Type, and Age
A6.1 Draw a chart showing the average speed against National Road Type for car drivers.

Code:

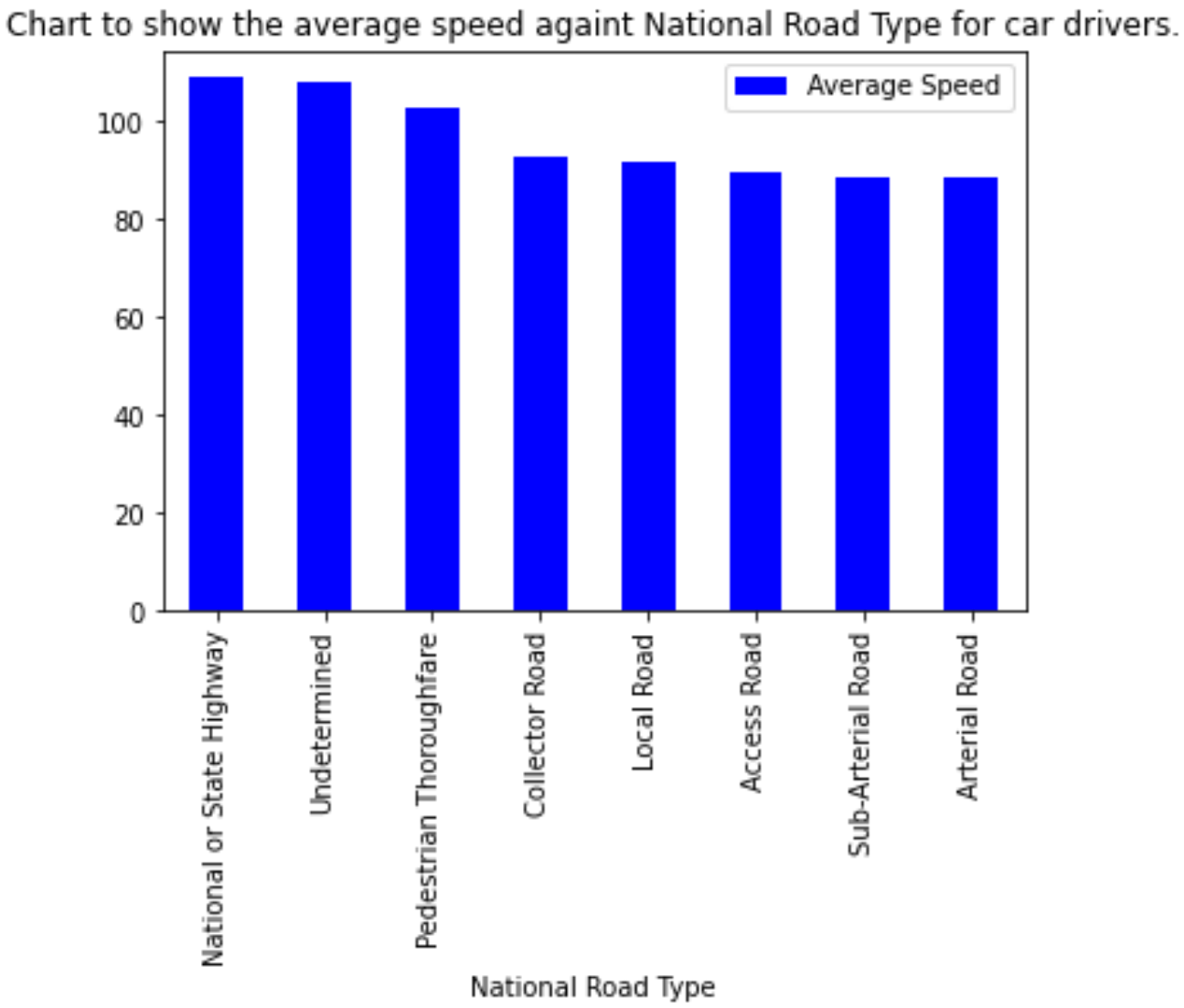
```
a6 = pd.DataFrame(data1,columns=['Road User','National Road Type','Speed'])

x = a6[(a6 ['Road User'] == 'Car driver')]

#A6
data4=x.groupby('National Road Type')['Speed'].mean().sort_values(ascending=False).reset_index(name='Average Speed')
print(data4)

#A6.1
```

#Draw a chart showing the average speed against National Road Type for car drivers.

```
data4.plot(kind = 'bar',
    x = 'National Road Type',
    y = 'Average Speed',
    color = 'blue')

# set the title
plt.title('Chart to show the average speed againt National Road Type for car drivers.')

# show the plot
plt.show()
```

Output:

	National Road Type	Average Speed
0	National or State Highway	109.177102
1	Undetermined	108.166667
2	Pedestrian Thoroughfare	103.000000
3	Collector Road	92.652439
4	Local Road	91.803957
5	Access Road	89.785714
6	Sub-Arterial Road	88.595573
7	Arterial Road	88.571262



A6.2 Due to measurement error, there are some counter-intuitive values in Age column. Identify those values and replace them with zero.

Code:

```
#A6.2
#Due to measurement error, there are some counter
intuitive values in Age column. Identify those values and replace them with zero.

data1['Age'].mask(data1['Age'] <0 ,0, inplace=True)
```

A7. Relationship between Age, Speed, and Driving Experiences

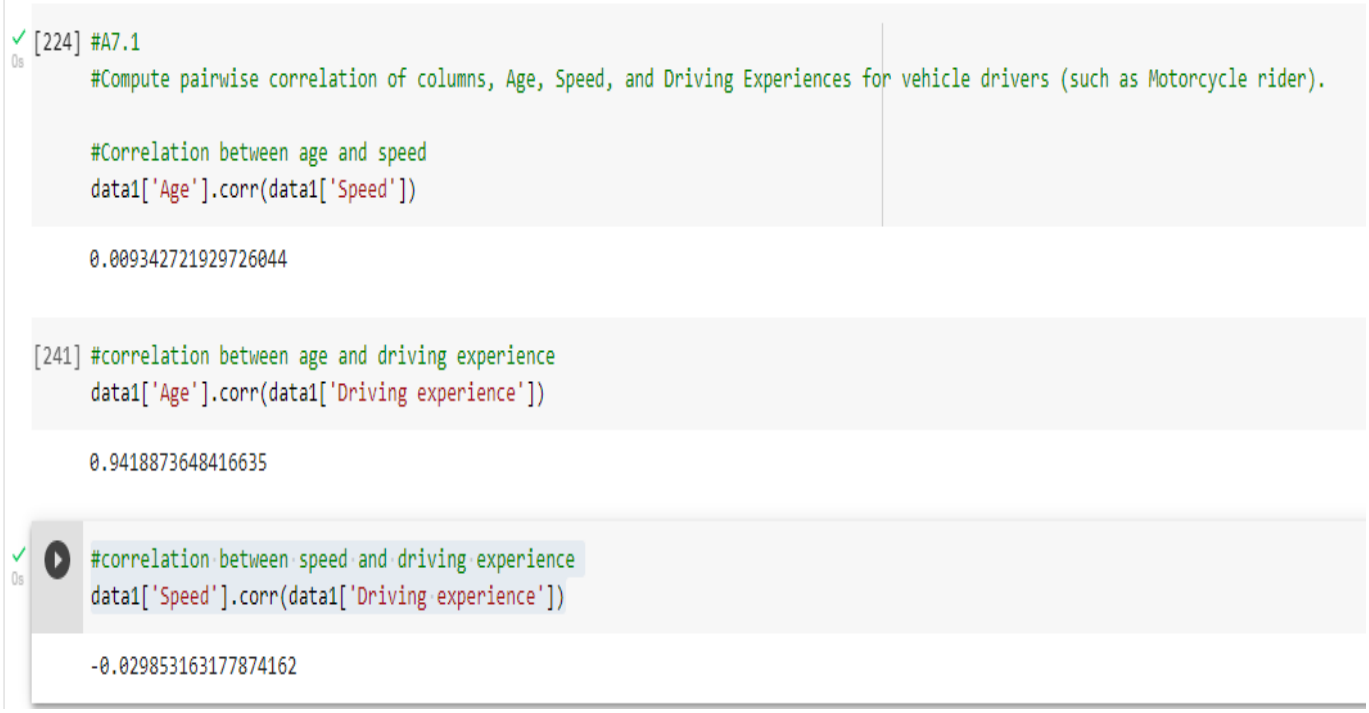
A7.1 Compute pairwise correlation of columns, Age, Speed, and Driving Experiences for vehicle drivers (such as Motorcycle rider). Which two features have the highest linear association?

Solution: The linear association is highest between age and driving experience.

Code:

```
#A7.1
#Compute pairwise correlation of columns, Age, Speed, and Driving Experiences for vehicle dri
vers (such as Motorcycle rider).

#Correlation between age and speed
data1['Age'].corr(data1['Speed'])
#correlation between age and driving experience
data1['Age'].corr(data1['Driving experience'])
#correlation between speed and driving experience
data1['Speed'].corr(data1['Driving experience'])
```

Output:



A7.2 Now let's look at the relationship between the number of crashes and Driving Experiences. To do this, first compute the number of crashes against Driving Experiences for vehicle drivers and plot the values of these two features against each other. Is there any relationship between these two features? Describe it.

Solution:

The correlation between the number of crashes and Driving Experiences is -0.6768982048231171 which means they are inversely correlated. It means higher the driving experience lower the crashes.

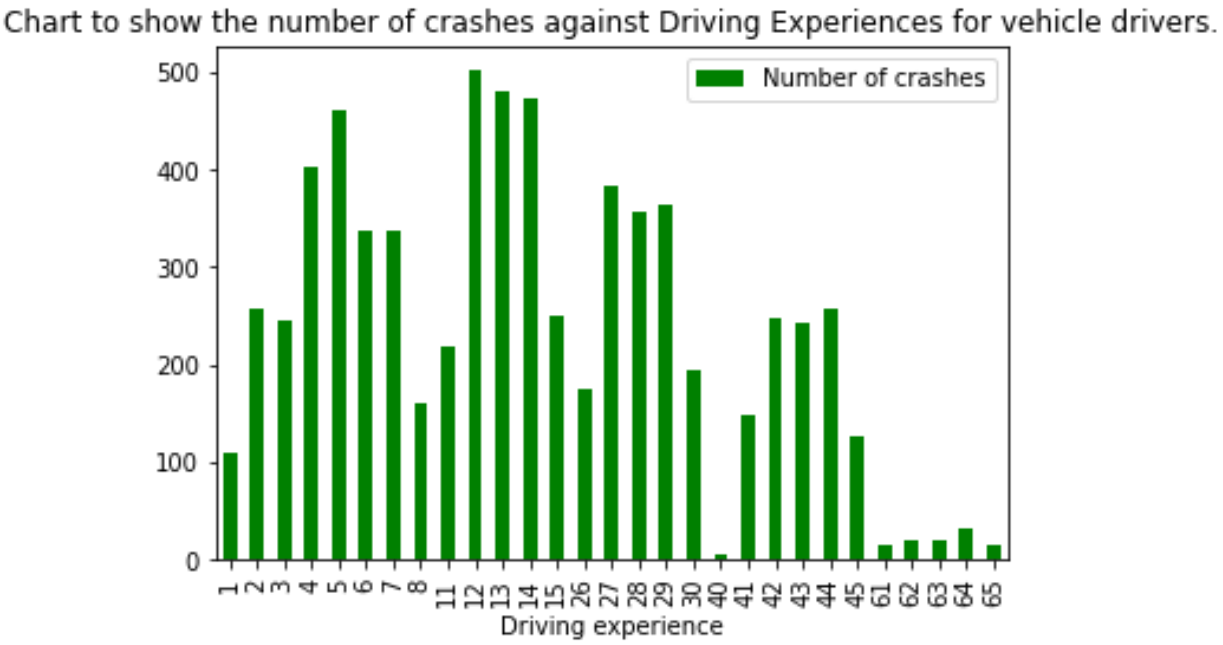
Code:

```
#A7.2
```

```
data6=data1.groupby('Driving experience')['Crash ID'].count().reset_index(name='Number of cra
shes')
print(data6)

#A7.2
data6.plot(kind = 'bar',
    x = 'Driving experience',
    y = 'Number of crashes',
    color = 'green')

# set the title
plt.title('Chart to show the number of crashes against Driving Experiences for vehicle drivers.')

# show the plot
plt.show()
#A7.2
#relationship between the number of crashes and Driving Experiences
data6['Driving experience'].corr(data6['Number of crashes'])
```

Output:

	Driving experience	Number of crashes
0	1	110
1	2	257
2	3	244
3	4	402
4	5	461
5	6	338
6	7	337
7	8	161
8	11	218
9	12	502
10	13	481
11	14	474
12	15	251
13	26	175
14	27	384
15	28	356
16	29	364
17	30	193
18	40	4
19	41	148
20	42	248
21	43	243
22	44	257
23	45	127
24	61	15




```
#A7.2
#relationship between the number of crashes and Driving Experiences
data6['Driving experience'].corr(data6['Number of crashes'])
```

```
-0.6768982048231171
```

A8. Investigating yearly trend of crash

A8. 1 Fit a linear regression using Python to this data (The number of crashes over different years) and plot the linear fit.

Code:

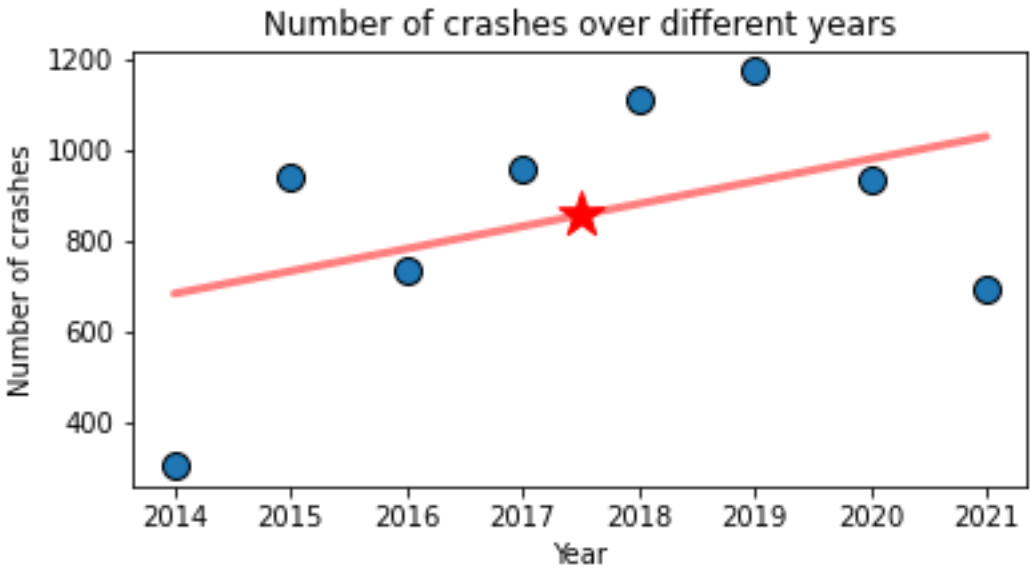
```
#A8.1
#investigate the trend in the crash over years
data7=data1.groupby('YYYY')['Crash ID'].count().reset_index(name='No. of Crashes yearly')
display(data7)
#A8.1
#Fit a linear regression using Python to this data (The number of crashes over different years) an
d plot the linear fit.
import numpy as np
import matplotlib.pyplot as plt
x = data7['YYYY']
y = data7['No. of Crashes yearly']
def linear_regression(x, y):
    N = len(x)
    x_mean = x.mean()
    y_mean = y.mean()

    B1_num = ((x - x_mean) * (y - y_mean)).sum()
```

```
    B1_den = ((x - x_mean)**2).sum()
    B1 = B1_num / B1_den

    B0 = y_mean - (B1*x_mean)

    reg_line = 'y = {} + {}β'.format(B0, round(B1, 3))

    return (B0, B1, reg_line)
N = len(x)
x_mean = x.mean()
y_mean = y.mean()

B1_num = ((x - x_mean) * (y - y_mean)).sum()
B1_den = ((x - x_mean)**2).sum()
B1 = B1_num / B1_den
B0 = y_mean - (B1 * x_mean)
def corr_coef(x, y):
    N = len(x)

    num = (N * (x*y).sum()) - (x.sum() * y.sum())
    den = np.sqrt((N * (x**2).sum() - x.sum()**2) * (N * (y**2).sum() - y.sum()**2))
    R = num / den
    return R
B0, B1, reg_line = linear_regression(x, y)
print('Regression Line: ', reg_line)
R = corr_coef(x, y)
print('Correlation Coef.: ', R)
print('"Goodness of Fit": ', R**2)
plt.figure(figsize=(6,3))
plt.scatter(x, y,s=100, linewidths=1,edgecolor='black')
plt.title('Number of crashes over different years')
plt.xlabel('Year', fontsize=10)
plt.ylabel('Number of crashes', fontsize=10)
plt.plot(x, B0 + B1*x, c = 'r', linewidth=3, alpha=.5, solid_capstyle='round')
plt.scatter(x=x.mean(), y=y.mean(), marker='*', s=10**2.5, c='r')
```

Output:

	YYYY	No. of Crashes yearly
0	2014	302
1	2015	939
2	2016	733
3	2017	958
4	2018	1111
5	2019	1173
6	2020	931
7	2021	690

```
[236]
0s
Regression Line:  y = -98795.46428571429 + 49.393β
Correlation Coef.:  0.43640572730969585
"Goodness of Fit":  0.1904499588287046
```



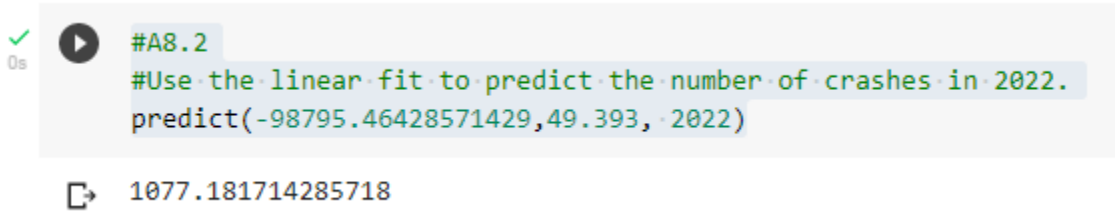
A8.2 Use the linear fit to predict the number of crashes in 2022.

Code:

```
def predict(B0, B1, new_x):
    y = B0 + B1 * new_x
    return y
#A8.2
```

```
#Use the linear fit to predict the number of crashes in 2022.
predict(-98795.46428571429,49.393, 2022)
```

Output:



A8.3 Can you think of a better model that well captures the trend of yearly crash? Develop a new model and explain why it is better suited for this task.

Solution:

Yes, I can implement ridge regression model that well captures the trend of yearly crash.
It is better suited because:
It is an extension of linear regression.
Here, the loss function is modified to minimize the complexity of model.
It shrinks the coefficients which do not contribute much in prediction.

Code:

```
#Building new model
# define model
from sklearn.linear_model import Ridge
x = data7[['YYYY']]
y = data7['No. of Crashes yearly']
# define model
model = Ridge(alpha=1.0)
# fit model
model.fit(x, y)
```

A8.4 Use your new model to predict the number of crashes in 2022.

Code:

```
#A8.4
#Use your new model to predict the number of crashes in 2022.
r=[[2022]]
# make a prediction
a = model.predict(r)
# summarize prediction
print('Predicted: %.3f' % a)
```

Output: