Directions

1. This basically is a partially populated ipynb with blanks/empty code cells left deliberately to be filled out by you. Not just filling some stuff out is enough. Make sure it produces intended output.
2. You will come across code cells with a certain number of such blanks or commented cells without any code and each of which will carry two marks. Thus total marks for this tutorial is 10.
3. Avoid searching for the answers in the Internet or copying from your neighbor.
4. Upon the completion, the pdf exported ipynb can be downloaded and should be given a name that complies with the format "ROLLNO_NAME". The named file should be uploaded to the outlook form for which the link is shared in the very last text cell.

```
## Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
## Reading the dataset
sales = pd.read_csv("/content/company_sales_data.csv")
```

```
## Describing the dataset
sales.describe().T
```

	count	mean	std	min	25%	50%	75%	max
month_number	12.0	6.500000	3.605551	1.0	3.75	6.5	9.25	12.0
facecream	12.0	2873.333333	584.595172	1990.0	2460.00	2830.0	3435.00	3700.0
facewash	12.0	1542.916667	316.733745	1120.0	1305.00	1527.5	1765.00	2100.0
toothpaste	12.0	5825.833333	1242.032486	4550.0	4862.50	5530.0	6400.00	8300.0
bathingsoap	12.0	9500.833333	2348.095779	6100.0	8015.00	9090.0	10045.00	14400.0
shampoo	12.0	2117.500000	617.724931	1200.0	1795.00	1995.0	2325.00	3550.0
moisturizer	12.0	1542.916667	316.733745	1120.0	1305.00	1527.5	1765.00	2100.0
total_units	12.0	26027.500000	7014.365940	18330.0	21065.00	22935.0	29667.50	41280.0
total_profit	12.0	260275.000000	70143.659404	183300.0	210650.00	229350.0	296675.00	412800.0

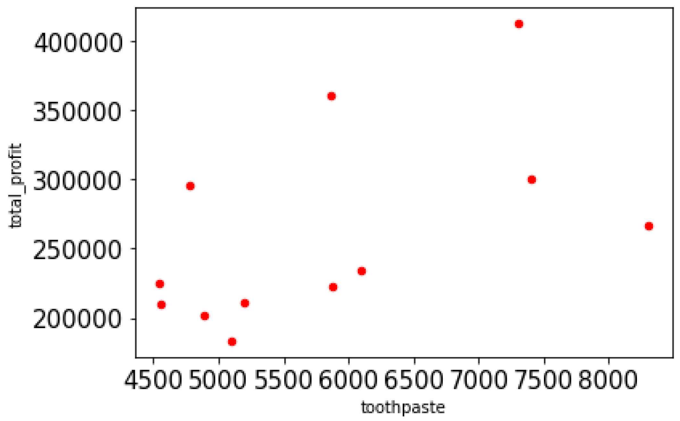
```
## Viewing 5 sample instances from the dataset
sales.sample(5)
```

	month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	total_units	tota
7	8	3700	1400	5860	9960	2860	1400	36140	
6	7	2980	1120	4780	8980	1780	1120	29550	
3	4	3400	1130	5870	8870	1870	1130	22270	
9	10	1990	1890	8300	10300	2300	1890	26670	
11	12	2900	1760	7400	14400	1800	1760	30020	

Question 1: Create a new datset from the sales dataset containing only the columns (toothpaste and total_profit). Draw scatter plots (using all three Pandas/Matplotlib/Seaborn) that show the relationship between the selected data. Make sure to include axis labels wherever necessary with font size as 15 and the font color as Red.

```
## The code should come here
df=sales.iloc[:, [3,8]]
print(df.head())
df.plot(kind = 'scatter', x = 'toothpaste', y = 'total_profit', color = 'red',fontsize=15)
```

```
   toothpaste  total_profit
0        5200        211000
1        5100        183300
2        4550        224700
3        5870        222700
4        4560        209600
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8a610dc10>
```



Question 2: Create training and test sets from the dataset with train-test split ratio 80:20 and random_state's value be 42.

```
## The code should come here
from sklearn.model_selection import train_test_split
x=df.iloc[:, [0]]
y=df.iloc[:, [1]]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(9, 1) (3, 1) (9, 1) (3, 1)
```

```
x_train = np.array(x_train)
x_test = np.array(x_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

Question 3: Fit a Linear Regression model (using Sklearn package) to the training data, test the fitted model, and report the model performance via MSE, MAE, and RMSE.

```
from sklearn.linear_model import LinearRegression

# Model creation and fitting
regressor = LinearRegression(fit_intercept =True)
```

```
regressor.fit(x_train.reshape(-1, 1),y_train.reshape(-1, 1))

# Printing the slope and intercept
print('Linear Model Coefficient (m): ', regressor.coef_)
print('Linear Model Coefficient (b): ', regressor.intercept_)

# Making predictions
y_predict = regressor.predict(x_test.reshape(-1, 1))
print(y_predict)

# Reporting the model performance
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, y_predict))
print('MSE:', metrics.mean_squared_error(y_test, y_predict))
print('R-squared:', metrics.r2_score(y_test, y_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_predict)))
```

```
Linear Model Coefficient (m):  [[27.33606292]]
Linear Model Coefficient (b):  [97160.67165092]
[[239308.19883285]
 [257350.00035979]
 [230834.01932777]]
MAE: 53930.73926694538
MSE: 4164706013.3587246
R-squared: 0.22416571755965098
RMSE: 64534.53349454635
```

Question 4: Approximate the slope and intercept using Ordinary Least Square method.

```
## Fill in the blanks
X=df.iloc[0]
Y=df.iloc[1]

X_mean = np.mean(X)
Y_mean = np.mean(Y)

num = 0
den = 0
for i in range(len(X)):
    num += (X[i] - X_mean)*(Y[i] - Y_mean)
    den += (X[i]-X_mean)**2
m = num / den
c = Y_mean - (m*X_mean)

print (m, c)
```

```
0.8658892128279884 597.376093294457
```

Question 5: Approximate the slope and intercept using Gradient Descent Optimization method with the learning rate 0.01 and 10 as number of iterations.

```
m = 0
c = 0

L = 0.01
epochs = 10

n = float(len(X)) # Number of elements in X

# Performing Gradient Descent
for i in range(epochs):
    Y_pred = m*X + c  # The current predicted value of Y
    D_m = (-2/n) * sum(X * (Y - Y_pred))  # Derivative wrt m
    D_c = (-2/n) * sum(Y - Y_pred)  # Derivative wrt c
    m = m - L * D_m  # Update m
    c = c - L * D_c  # Update c
print (m, c)
```

```
-2.674237712380318e+86 -1.2978577585657283e+81
```