```
You will use ‘Admission_Predict.csv’ for Midterm. This dataset includes the
data of the applicants of an academic program. Each application has a unique
serial number, which represents a particular student. The dataset contains
several parameters which are considered important during the application for
Masters Programs. The parameters included are :
1)  GRE Scores (out of 340)
2) TOEFL Scores (out of 120)
3) University Rating (out of 5)
4) Statement of Purpose (SOP) (out of 5)
5) Letter of Recommendation (LOR) Strength (out of 5)
6) Undergraduate GPA (out of 10)
7) Research Experience (either 0 or 1)
8) Chance of Admit (ranging from 0 to 1)
```

Q1: Download "Simple linear regression.csv” dataset and load it as ‘data’.

In [1]:
```
import pandas as pd
```

In [2]:
```
data = pd.read_csv("/content/Simple linear regression.csv")
```

Q2: Display the first three rows in this dataset.

In [3]:
```
data.head(3)
```

Out[3]:

	SAT	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54

In [4]:
```
df = data
df.head(3)
```

Out[4]:

	SAT	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54

Q3: check the duplicate records. if yes then remove the duplicate records.

In [5]:
```
df.duplicated()
```

Out[5]:
```
0     False
1     False
2     False
3     False
4     False
      ...
79    False
80    False
81    False
82    False
83    False
Length: 84, dtype: bool
```

Q4: Are there any missing values in the dataset?

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
SAT    0
GPA    0
dtype: int64
```

Q5. Remove the missing value.

In [6]:
```

```

Q6: - Display the structure of all variables

In [7]:
```
df.shape
```

Out[7]:
```
(84, 2)
```

Q7: Check the datatypes of the attributes.

In [8]:
```
df.dtypes
```

Out[8]:
```
SAT      int64
GPA    float64
dtype: object
```

Q8: change the data type according to the discribution of the data set

In [9]:
```
#The data type is according to the distribution of the data set
```

Q9: Print the descriptive statistics of the Simple linear regression data to understand the data a little better (min, max, mean, median, 1st and 3rd quartiles).

In [10]:
```
df.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
SAT	84.0	1845.273810	104.530661	1634.0	1772.00	1846.00	1934.0000	2050.00
GPA	84.0	3.330238	0.271617	2.4	3.19	3.38	3.5025	3.81

Q10: Divide the dataset to training and test sets.

In [11]:
```
df = pd.get_dummies(df)
df.head(2)
```

Out[11]:

	SAT	GPA
0	1714	2.40
1	1664	2.52

In [12]:
```
dfx = df.drop('GPA', axis = 1)
```

In [13]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [14]:
```
scaler = MinMaxScaler()
```

In [15]:
```
scaler.fit_transform(dfx)
```

Out[15]:
```
array([[0.19230769],
       [0.07211538],
       [0.30288462],
       [0.12259615],
       [0.14182692],
       [0.08653846],
       [0.3125    ],
       [0.3125    ],
       [0.37980769],
       [0.51923077],
       [0.24278846],
       [0.33894231],
       [0.24278846],
       [0.1875    ],
       [0.33413462],
       [0.57211538],
       [0.29086538],
       [0.09615385],
       [0.5       ],
       [0.36538462],
       [0.30528846],
       [0.21153846],
       [0.06971154],
       [0.12740385],
       [0.81730769],
       [0.46153846],
       [0.36778846],
       [0.44951923],
       [0.92788462],
       [0.38461538],
       [0.32451923],
       [0.72115385],
       [0.33894231],
       [0.53125   ],
       [0.59134615],
       [0.51682692],
       [0.41826923],
       [0.76923077],
       [0.34375   ],
       [0.47355769],
       [0.55528846],
       [0.51923077],
       [0.79807692],
       [0.16346154],
       [0.85576923],
       [0.69951923],
       [0.45673077],
       [0.77403846],
       [0.53605769],
       [0.82932692],
       [0.40384615],
       [0.53125   ],
       [0.65625   ],
       [0.        ],
       [0.58894231],
       [0.60817308],
       [0.23076923],
```

```
       [0.76682692],
       [0.35336538],
       [0.61778846],
       [0.79326923],
       [0.41826923],
       [0.62259615],
       [0.97836538],
       [0.62259615],
       [0.47596154],
       [0.51923077],
       [0.72115385],
       [0.54567308],
       [0.71394231],
       [0.71875   ],
       [0.34615385],
       [0.81971154],
       [0.72115385],
       [0.93028846],
       [0.91586538],
       [0.87259615],
       [0.92788462],
       [0.50240385],
       [0.72596154],
       [0.42307692],
       [0.84855769],
       [0.78846154],
       [1.        ]])
```

In [16]:
```
dfx.columns
```

Out[16]:
```
Index(['SAT'], dtype='object')
```

In [17]:
```
pd.DataFrame (scaler.fit_transform(dfx))
```

Out[17]:

	0
0	0.192308
1	0.072115
2	0.302885
3	0.122596
4	0.141827
...	...
79	0.725962
80	0.423077
81	0.848558
82	0.788462
83	1.000000

84 rows × 1 columns

In [18]:
```
pd.DataFrame(scaler.fit_transform(dfx),columns = dfx.columns)
```

Out[18]:

	SAT
0	0.192308
1	0.072115
2	0.302885
3	0.122596
4	0.141827
...	...
79	0.725962
80	0.423077
81	0.848558
82	0.788462
83	1.000000

84 rows × 1 columns

In [19]:
```
scaler_dfx = pd.DataFrame(scaler.fit_transform(dfx),columns = dfx.columns)
scaler_dfx.head(10)
```

Out[19]:

	SAT
0	0.192308
1	0.072115
2	0.302885
3	0.122596
4	0.141827
5	0.086538
6	0.312500
7	0.312500
8	0.379808
9	0.519231

In [20]:
```
from sklearn.model_selection import train_test_split
```

In [21]:
```
input_train, input_test, output_train, output_test = train_test_split(scaler_d
```

In [22]:
```
input_train
```

Out[22]:

	SAT
36	0.418269
27	0.449519
43	0.163462
40	0.555288
2	0.302885
...	...
75	0.915865
9	0.519231
72	0.819712
12	0.242788
37	0.769231

67 rows × 1 columns

In [23]:
```
input_train,input_test,output_train,output_test
```

Out[23]:
```
(         SAT
36  0.418269
27  0.449519
43  0.163462
40  0.555288
2   0.302885
..       ...
75  0.915865
9   0.519231
72  0.819712
12  0.242788
37  0.769231

[67 rows x 1 columns],
         SAT
10  0.242788
67  0.721154
59  0.617788
33  0.531250
83  1.000000
77  0.927885
34  0.591346
38  0.343750
47  0.774038
56  0.230769
78  0.502404
68  0.545673
61  0.418269
66  0.519231
53  0.000000
65  0.475962
31  0.721154,
     SAT
36  1808
27  1821
43  1702
40  1865
2   1760
..   ...
75  2015
9   1850
72  1975
12  1735
37  1954

[67 rows x 1 columns],
     SAT
10  1735
67  1934
59  1891
33  1855
83  2050
77  2020
34  1880
38  1777
47  1956
56  1730
```

```
 78  1843
 68  1861
 61  1808
 66  1850
 53  1634
 65  1832
 31  1934)
```

Q11 Create the linear regression model.

In [24]:
```
import pandas as pd
import numpy as np
```

In [25]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()

SAT = df[["SAT"]]
GPA = df["GPA"]
```

Q12: Fit the linear regression model on training data set.

In [26]:
```
model.fit(SAT, GPA)
```

Out[26]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Q13 : Predicting the test set results

In [27]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()

SAT = df[["SAT"]]
GPA = df["GPA"]

model.fit(SAT, GPA)

X_test = SAT

y_pred = model.predict(X_test)
```

In [28]:
```
data_value = 85
```

In [29]:
```
predicted = 0.2750402996602799 + 0.00165569 * data_value
predicted
```

Out[29]:
```
0.4157739496602799
```

Q14: display regression coefficients

In [30]:
```
print(model.intercept_, model.coef_)
```

```
0.2750402996602799 [0.00165569]
```