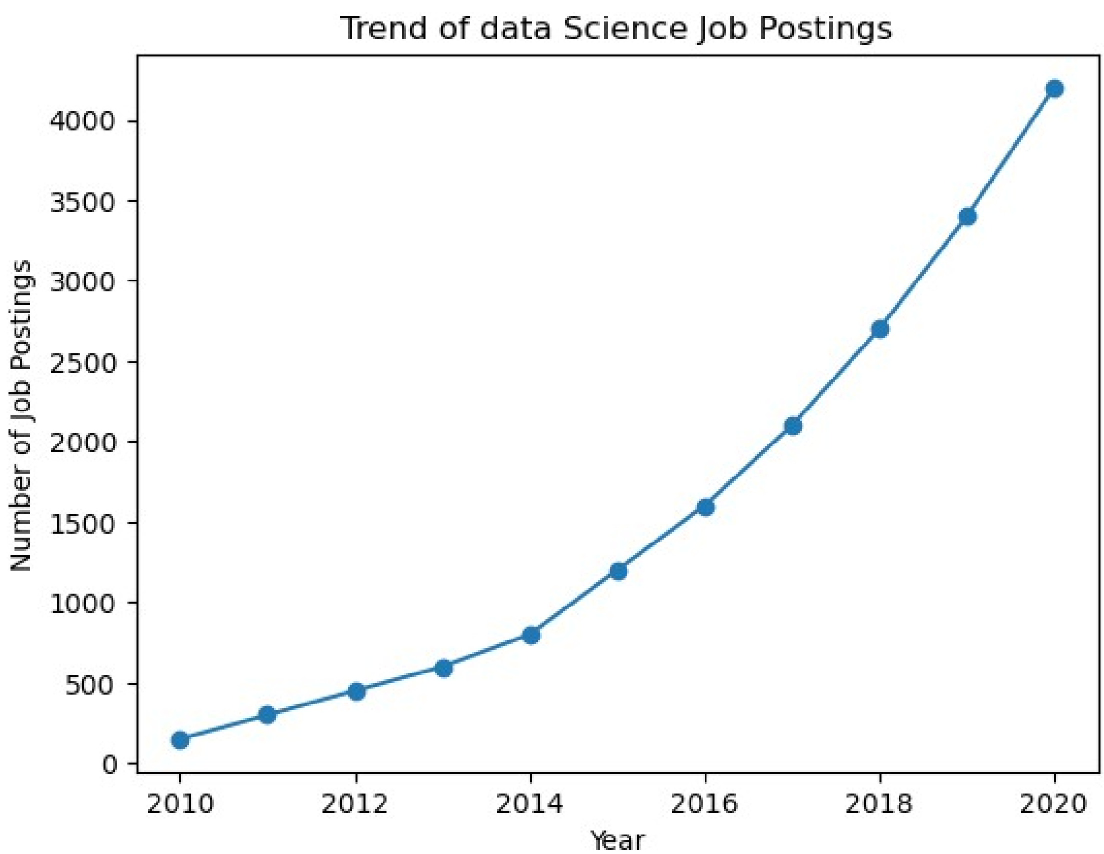
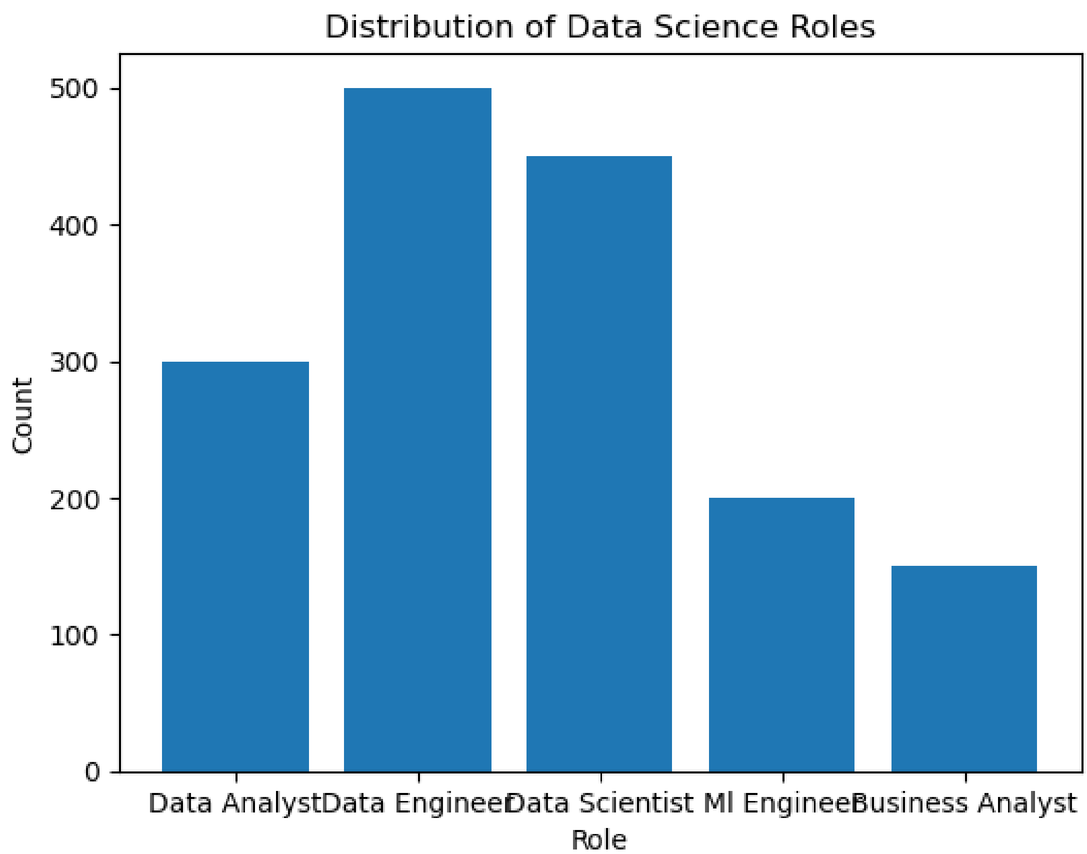
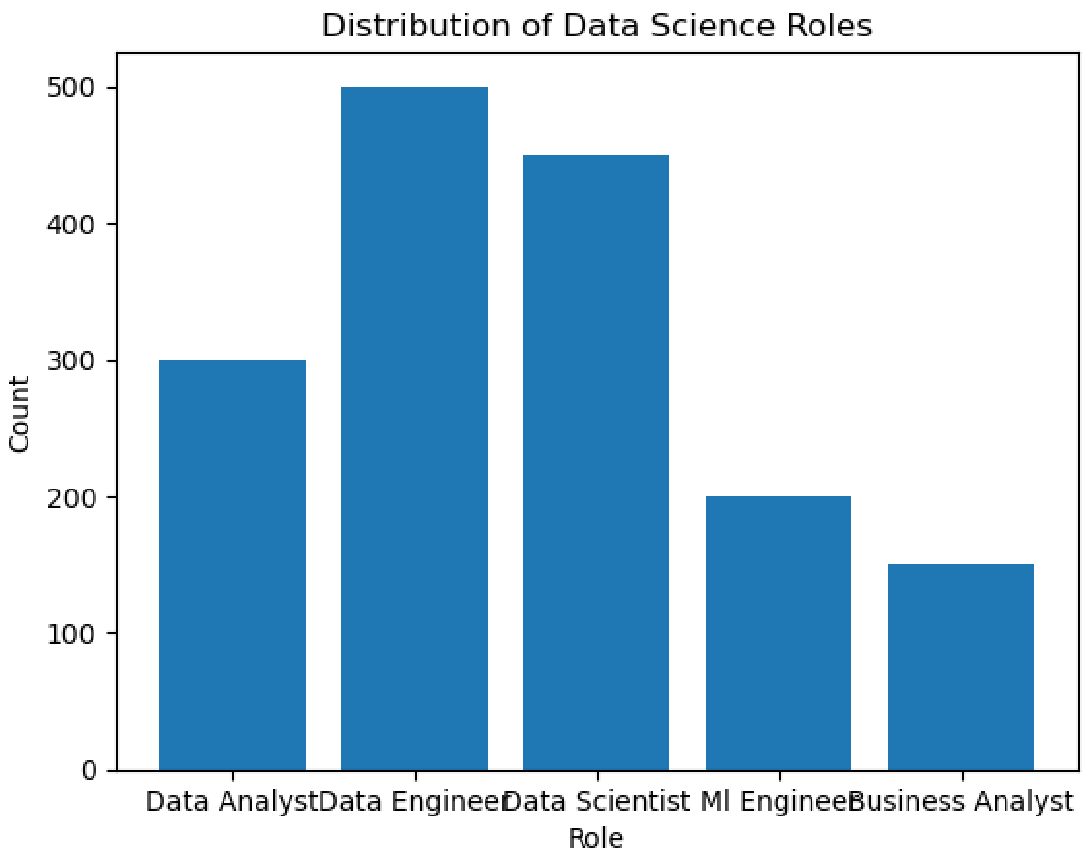
```
import pandas as pd
import matplotlib.pyplot as plt


data={'Year':(range(2010,2021)),
              'Job Postings':
[150,300,450,600,800,1200,1600,2100,2700,3400,4200]}
df=pd.DataFrame(data)
plt.plot(df['Year'],df['Job Postings'],marker='o')
plt.title('Trend of data Science Job Postings')
plt.xlabel('Year')
plt.ylabel('Number of Job Postings')
plt.show()
```





Distribution of Data Science Roles
Count
500
400
300
200
100
0
Data Analyst
Data Engineer
Data Scientist
Ml Engineer
Business Analyst
Role

```
import pandas as pd
import matplotlib.pyplot as plt
roles=['Data Analyst','Data Engineer','Data Scientist','Ml
Engineer','Business Analyst']
counts=[300,500,450,200,150]
plt.bar(roles,counts)
plt.title('Distribution of Data Science Roles')
plt.xlabel('Role')
plt.ylabel('Count')
plt.show()
```



```python
import pandas as pd
structured_data=pd.DataFrame({'ID':[1,2,3],'name':
['Alice','Bob','Charlie'],'age':[25,30,23]})
print("structured data:\n",structured_data)
```

```
structured data:
    ID     name  age
0   1    Alice   25
1   2      Bob   30
2   3  Charlie   23
```

```python
unstructured_data="This is an example of unstructured data.It can be a
piece of text"
print("Unstructured data:\n",unstructured_data)
```

```
Unstructured data:
 This is an example of unstructured data.It can be a piece of text
```

```python
semistructured_data={'ID':[1,2,3],'name':
['Alice','Bob','Charlie'],'age':[25,30,23]}
print("semistructured_data:\n",semistructured_data)
```

```
semistructured_data:
 {'ID': [1, 2, 3], 'name': ['Alice', 'Bob', 'Charlie'], 'age': [25,
30, 23]}
```

```python
file=open('data.txt','w')
file.writelines(["hello\n","this is file handling using python\n"])
file.close()
```

```python
#unstructured data
f=open("data.txt",'r')
unstruct_data=f.readline()
print("unstructured data:\n",unstruct_data)
```

```
unstructured data:
 he
```

```
from cryptography.fernet import Fernet
key= Fernet.generate_key()
f=Fernet(key)
token=f.encrypt(b"This is plain text.")
")
f.decrypt(token)
key=Fernet.generate_key()
cipher_suite=Fernet(key)
plain_text=b"This is plain text."
cipher_text=cipher_suite.encrypt(plain_text)

decrypted_text=cipher_suite.decrypt(cipher_text)
print("original data: ",plain_text)
print("encrypted data: ",cipher_text)
print("decrypted data: ", decrypted_text)
```

```
original data:  b'This is plain text.'
encrypted data:
b'gAAAAABmtEPcNs087rNAdIGFtE78RR2iTR1t9NDsbKZ2iP_p4uXZDtE3bVcl0MzV-
Cqu2you0EY4-jSd7zQLp0sYj7W1uaVGl2SZt_bkVWE5c6uLk4WLqTQ='
decrypted data:  b'This is plain text.'
```

```
print("hello world")
```

```
hello world
```

```
import pandas as pd
db=pd.read_csv("C:\Users\DELL\Desktop\diabetes.csv
```

```
  Cell In[4], line 2
    db=pd.read_csv("C:\Users\DELL\Desktop\diabetes.csv")
                                                       ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes
in position 2-3: truncated \UXXXXXXXX escape
```

```
import pandas as pd
db=pd.read_csv("diabetes.csv")
print(db.head())
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin
BMI  \
0            6      148             72             35        0  33.6

1            1       85             66             29        0  26.6

2            8      183             64              0        0  23.3

3            1       89             66             23       94  28.1

4            0      137             40             35      168  43.1


   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
print(db.info())
```

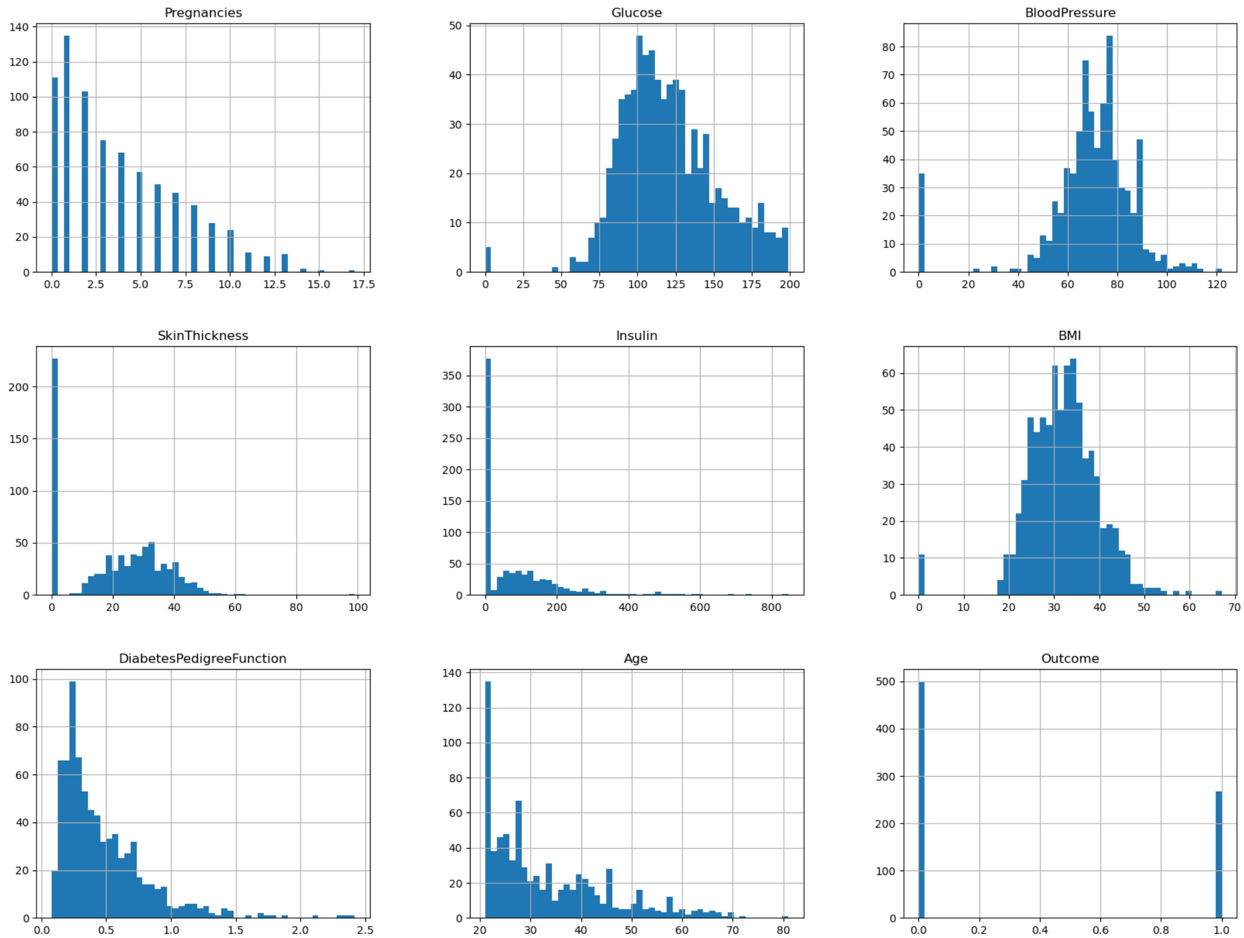
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
```

```
memory usage: 54.1 KB
None
```

```python
print(db.describe())
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness
Insulin  \
count   768.000000  768.000000     768.000000     768.000000
768.000000
mean      3.845052  120.894531      69.105469      20.536458
79.799479
std       3.369578   31.972618      19.355807      15.952218
115.244002
min       0.000000    0.000000       0.000000       0.000000
0.000000
25%       1.000000   99.000000      62.000000       0.000000
0.000000
50%       3.000000  117.000000      72.000000      23.000000
30.500000
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

```python
import matplotlib.pyplot as plt
import seaborn as sns
db.hist(bins=50,figsize=(20,15))
plt.show()
```




```
db.duplicated().any()
```

```
False
```

```
db.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
import pandas as pd
sales_df=pd.read_csv("Downloads/Sales_Transactions_Dataset_Weekly.csv"
)
sales_df.head()
```

```
  Product_Code  W0  W1  W2  W3  W4  W5  W6  W7  W8  ...  Normalized 42
\
0           P1  11  12  10   8  13  12  14  21   6  ...           0.06

1           P2   7   6   3   2   7   1   6   3   3  ...           0.20

2           P3   7  11   8   9  10   8   7  13  12  ...           0.27

3           P4  12   8  13   5   9   6   9  13  13  ...           0.41

4           P5   8   5  13  11   6   7   9  14   9  ...           0.27


   Normalized 43  Normalized 44  Normalized 45  Normalized 46
Normalized 47  \
0           0.22           0.28           0.39           0.50
0.00
1           0.40           0.50           0.10           0.10
0.40
2           1.00           0.18           0.18           0.36
0.45
3           0.47           0.06           0.12           0.24
0.35
4           0.53           0.27           0.60           0.20
0.20

   Normalized 48  Normalized 49  Normalized 50  Normalized 51
0           0.22           0.17           0.11           0.39
1           0.50           0.10           0.60           0.00
2           1.00           0.45           0.45           0.36
3           0.71           0.35           0.29           0.35
4           0.13           0.53           0.33           0.40

[5 rows x 107 columns]
```

```
import numpy as np
import matplotlib.pyplot as plt
file_path="Downloads/Sales_Transactions_Dataset_Weekly.csv"
sales_df.isnull().sum()
```

```
Product_Code     0
W0               0
W1               0
W2               0
W3               0
                ..
```

```
Normalized 47    0
Normalized 48    0
Normalized 49    0
Normalized 50    0
Normalized 51    0
Length: 107, dtype: int64
```

```
sales_df.describe()
```

```
                W0          W1          W2          W3          W4
W5  \
count  811.000000  811.000000  811.000000  811.000000  811.000000
811.000000
mean     8.902589    9.129470    9.389642    9.717633    9.574599
9.466091
std     12.067163   12.564766   13.045073   13.553294   13.095765
12.823195
min      0.000000    0.000000    0.000000    0.000000    0.000000
0.000000
25%      0.000000    0.000000    0.000000    0.000000    0.000000
0.000000
50%      3.000000    3.000000    3.000000    4.000000    4.000000
3.000000
75%     12.000000   12.000000   12.000000   13.000000   13.000000
12.500000
max     54.000000   53.000000   56.000000   59.000000   61.000000
52.000000

                W6          W7          W8          W9  ...  Normalized
42  \
count  811.000000  811.000000  811.000000  811.000000  ...
811.000000
mean     9.720099    9.585697    9.784217    9.681874  ...
0.299149
std     13.347375   13.049138   13.550237   13.137916  ...
0.266993
min      0.000000    0.000000    0.000000    0.000000  ...
0.000000
25%      0.000000    0.000000    0.000000    0.000000  ...
0.000000
50%      4.000000    4.000000    4.000000    4.000000  ...
0.280000
75%     13.000000   12.500000   13.000000   13.000000  ...
0.490000
max     56.000000   62.000000   63.000000   52.000000  ...
1.000000

       Normalized 43  Normalized 44  Normalized 45  Normalized 46  \
count     811.000000     811.000000     811.000000     811.000000
mean        0.287571       0.304846       0.316017       0.334760
```

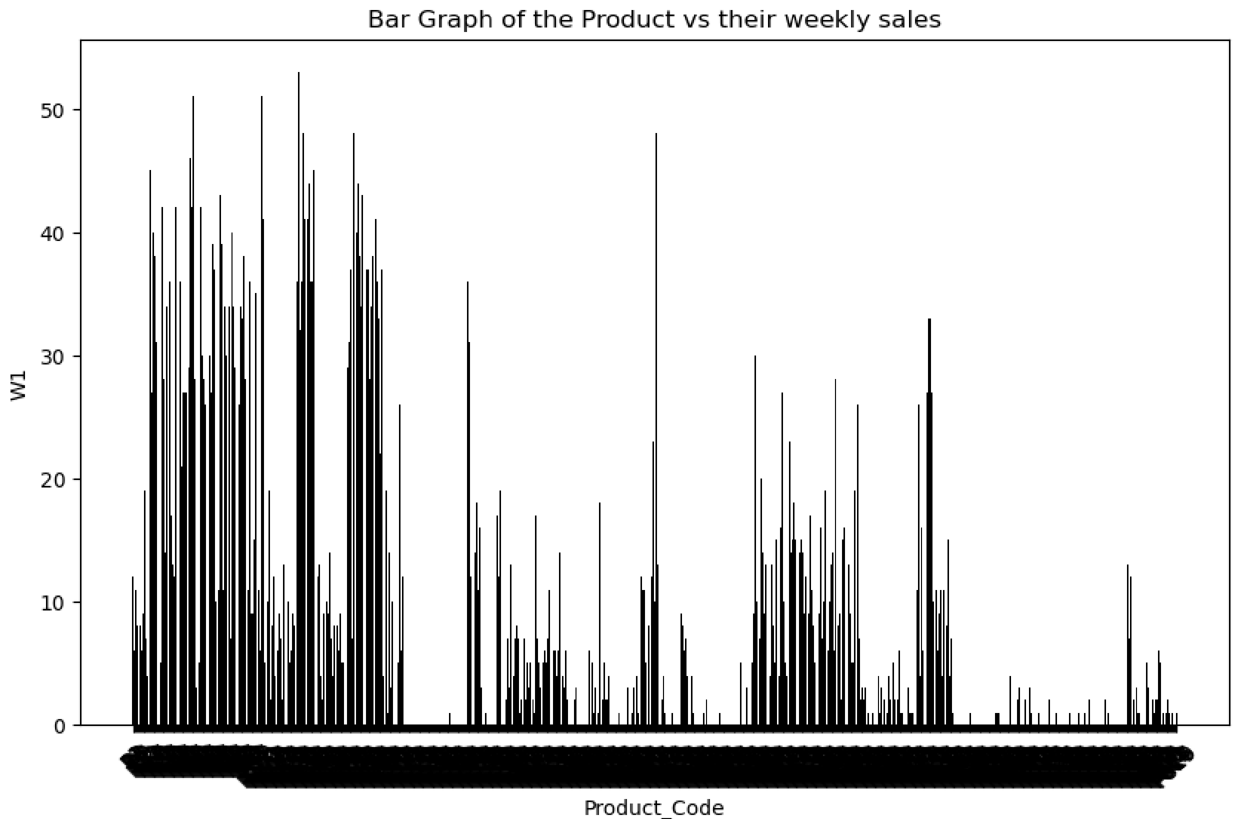
```
std          0.256630       0.263396       0.262226       0.275203
min          0.000000       0.000000       0.000000       0.000000
25%          0.000000       0.000000       0.020000       0.085000
50%          0.270000       0.300000       0.310000       0.330000
75%          0.450000       0.500000       0.500000       0.500000
max          1.000000       1.000000       1.000000       1.000000

       Normalized 47  Normalized 48  Normalized 49  Normalized 50  \
count     811.000000      811.00000     811.000000     811.000000
mean        0.314636        0.33815       0.358903       0.373009
std         0.266029        0.27569       0.286665       0.295197
min         0.000000        0.00000       0.000000       0.000000
25%         0.000000        0.10500       0.100000       0.110000
50%         0.310000        0.33000       0.330000       0.350000
75%         0.500000        0.50000       0.550000       0.560000
max         1.000000        1.00000       1.000000       1.000000

       Normalized 51
count     811.000000
mean        0.427941
std         0.342360
min         0.000000
25%         0.090000
50%         0.430000
75%         0.670000
max         1.000000

[8 rows x 106 columns]
```

```
sales_df['Sales'].fillna(sales_df['Sales'].mean(),inplace=True)
sales_df.dropna(subset=['Product','Quantity','Region'],inplace=True)
```

```
----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3802,
in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:138, in
pandas._libs.index.IndexEngine.get_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:165, in
pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in
pandas._libs.hashtable.PyObjectHashTable.get_item()
```

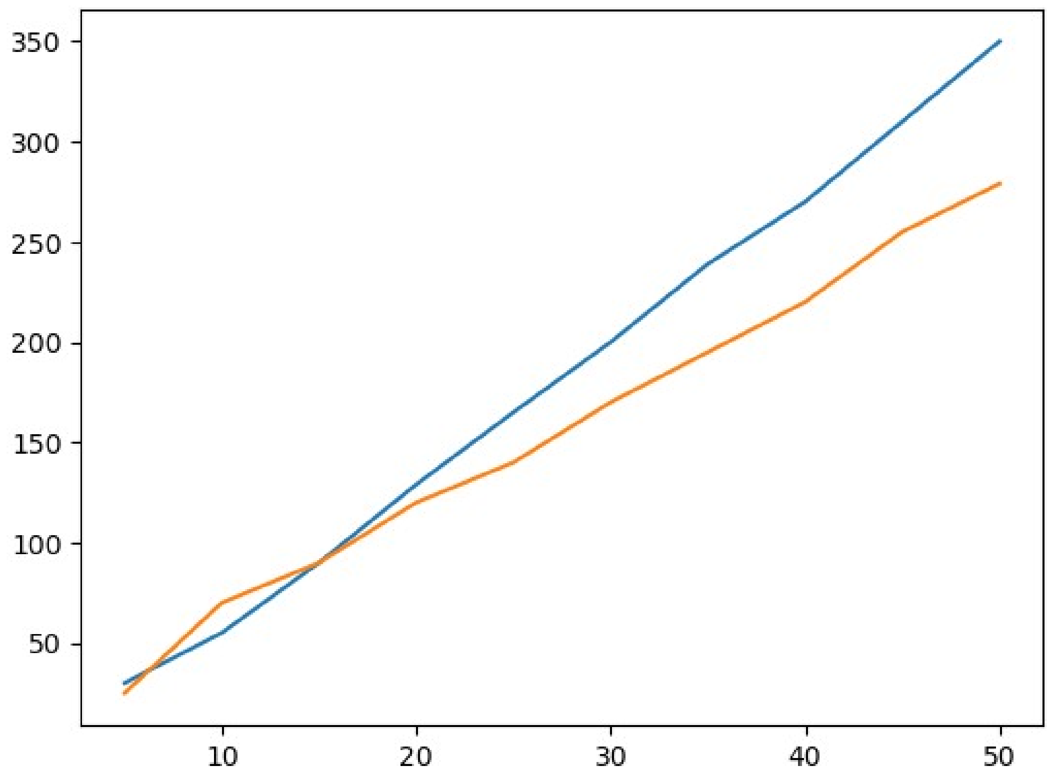
```
File pandas\_libs\hashtable_class_helper.pxi:5753, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Sales'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call
last)
Cell In[7], line 1
----> 1
sales_df['Sales'].fillna(sales_df['Sales'].mean(),inplace=True)
      2
sales_df.dropna(subset=['Product','Quantity','Region'],inplace=True)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3807, in
DataFrame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3804,
in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will
raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-
raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'Sales'
```

```
plt.figure(figsize=(10, 6))
plt.bar(sales_df['Product_Code'],sales_df['W1'],color='black')
plt.xlabel('Product_Code')
plt.ylabel('W1')
plt.title('Bar Graph of the Product vs their weekly sales')
plt.xticks(rotation=45)
plt.show()
```

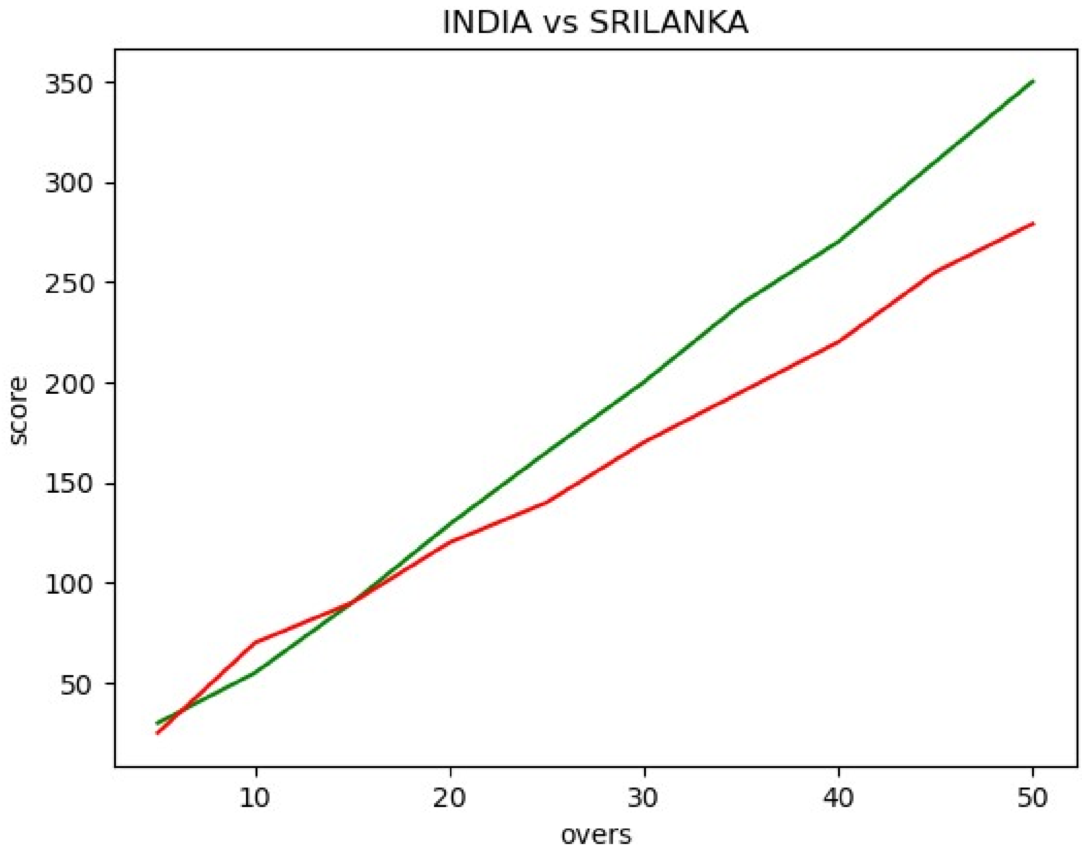


Bar Graph of the Product vs their weekly sales
W1
50
40
30
20
10
0
Product_Code

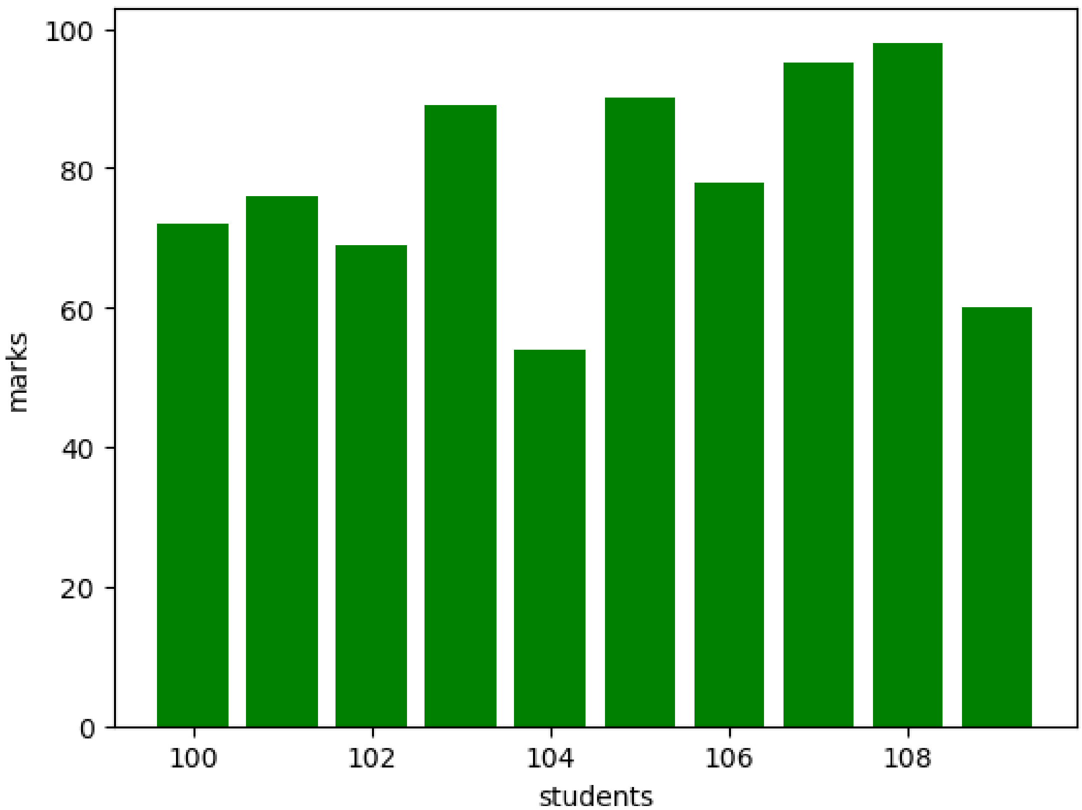
```python
import matplotlib.pyplot as cricket
overs=list(range(5,51,5))
Indian_score=[30,55,90,129,165,200,239,270,310,350]
Srilankan_score=[25,70,90,120,140,170,195,220,255,279]
cricket.plot(overs,Indian_score)
cricket.plot(overs,Srilankan_score)
cricket.show()
cricket.title("INDIA vs SRILANKA")
cricket.xlabel("overs")
cricket.ylabel("score")
cricket.plot(overs,Indian_score,color="green",label="INDIA")
cricket.plot(overs,Srilankan_score,color="red",label="SRILANKA")
```




```
[<matplotlib.lines.Line2D at 0x1f746e08640>]
```




```
import matplotlib.pyplot as plt
student_id=list(range(100,110))
student_marks=[72,76,69,89,54,90,78,95,98,60]
student_remarks=[3,3,2,4,2,5,5,3,5,5,3]
plt.bar(student_id,student_marks,color="green")
plt.xlabel("students")
plt.ylabel("marks")
plt.show()
```




```
election=pd.read_csv("governors.csv")
print(election.head(25))
election.info()
election.describe
```

```
       state                county  current_votes  total_votes  percent
0   Delaware           Kent County          85415        87025      100
1   Delaware     New Castle County         280039       287633      100
2   Delaware         Sussex County         127181       129352      100
3    Indiana          Adams County          14154        14209      100
4    Indiana          Allen County         168312       169082      100
5    Indiana    Bartholomew County          36037        36235      100
6    Indiana         Benton County           4100         4114      100
7    Indiana      Blackford County           5283         5350      100
8    Indiana          Boone County          38492        38520      100
9    Indiana          Brown County           8957         8981      100
10   Indiana        Carroll County           9510         9510      100
11   Indiana           Cass County          15146        15198      100
12   Indiana          Clark County          57426        57869      100
13   Indiana           Clay County          12186        12267      100
14   Indiana        Clinton County          12891        12949      100
15   Indiana       Crawford County           4859         4944      100
16   Indiana        Daviess County          11860        11954      100
17   Indiana       Dearborn County          25295        25383      100
```

```
18   Indiana      Decatur County          12260        12235       95
19   Indiana       DeKalb County          19493        19628      100
20   Indiana     Delaware County          47949        48191      100
21   Indiana       Dubois County          21588        21770      100
22   Indiana      Elkhart County          74425        74425      100
23   Indiana      Fayette County          10054        10136      100
24   Indiana        Floyd County          41589        41802      100
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   state          1025 non-null   object
 1   county         1025 non-null   object
 2   current_votes  1025 non-null   int64 
 3   total_votes    1025 non-null   int64 
 4   percent        1025 non-null   int64 
dtypes: int64(3), object(2)
memory usage: 40.2+ KB

<bound method NDFrame.describe of               state
county  current_votes  total_votes  percent
0          Delaware        Kent County         85415        87025
100
1          Delaware  New Castle County        280039       287633
100
2          Delaware      Sussex County        127181       129352
100
3           Indiana       Adams County         14154        14209
100
4           Indiana       Allen County        168312       169082
100
...             ...                ...           ...          ...
...
1020  West Virginia     Webster County          3339         3402
100
1021  West Virginia      Wetzel County          6553         6667
100
1022  West Virginia        Wirt County          2544         2653
100
1023  West Virginia        Wood County         38435        38762
100
1024  West Virginia     Wyoming County          8320         8592
100

[1025 rows x 5 columns]>
```

```python
election['county'].unique().any()
```

```
'Kent County'
```

```python
l=list(election.current_votes)
l.sort()
print(l)
```

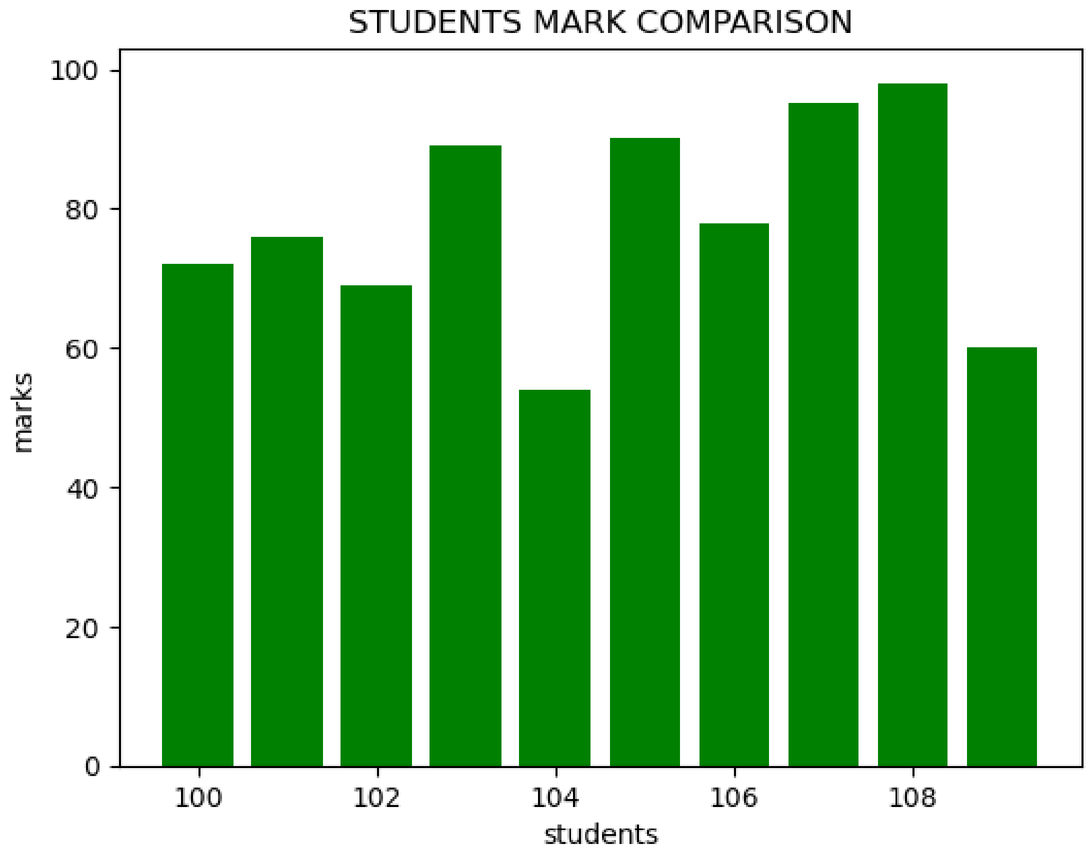
```
[5, 20, 22, 43, 43, 50, 55, 62, 65, 76, 106, 112, 115, 127, 129, 129,
135, 136, 137, 139, 140, 150, 164, 171, 171, 176, 180, 184, 190, 191,
191, 210, 216, 222, 229, 233, 233, 233, 244, 248, 249, 258, 259, 272,
272, 279, 292, 293, 295, 296, 305, 326, 328, 329, 330, 334, 340, 343,
349, 350, 351, 351, 357, 361, 364, 368, 368, 371, 375, 379, 385, 386,
389, 393, 397, 408, 411, 416, 417, 421, 421, 431, 432, 437, 439, 439,
445, 447, 447, 451, 454, 460, 471, 475, 476, 476, 476, 477, 482, 483,
488, 488, 491, 503, 504, 505, 509, 516, 517, 519, 521, 524, 527, 534,
535, 536, 536, 537, 540, 541, 547, 551, 552, 562, 562, 574, 577, 579,
585, 588, 591, 593, 601, 604, 607, 612, 613, 618, 619, 621, 622, 627,
629, 633, 635, 638, 638, 643, 643, 644, 657, 659, 660, 664, 665, 667,
669, 678, 678, 696, 700, 700, 707, 708, 709, 722, 724, 726, 730, 730,
738, 739, 742, 747, 748, 748, 748, 756, 763, 763, 768, 776, 778, 780,
780, 783, 785, 788, 790, 792, 794, 796, 806, 806, 813, 818, 820, 821,
822, 823, 825, 830, 834, 836, 844, 844, 847, 848, 852, 857, 858, 862,
874, 876, 877, 880, 893, 902, 907, 908, 908, 909, 918, 919, 919, 919,
924, 926, 943, 943, 945, 946, 949, 950, 950, 954, 954, 957, 993, 1001,
1009, 1020, 1032, 1032, 1043, 1051, 1056, 1060, 1061, 1068, 1069,
1071, 1078, 1080, 1083, 1084, 1089, 1095, 1097, 1106, 1111, 1112,
1117, 1117, 1122, 1125, 1129, 1129, 1130, 1132, 1133, 1134, 1136,
1141, 1141, 1144, 1154, 1158, 1164, 1178, 1188, 1191, 1193, 1195,
1203, 1204, 1206, 1214, 1218, 1228, 1230, 1243, 1248, 1249, 1250,
1252, 1270, 1270, 1273, 1275, 1276, 1279, 1288, 1298, 1302, 1304,
1306, 1320, 1328, 1334, 1346, 1351, 1374, 1391, 1400, 1400, 1401,
1413, 1448, 1458, 1462, 1466, 1474, 1475, 1478, 1478, 1481, 1487,
1527, 1536, 1541, 1555, 1564, 1567, 1569, 1581, 1587, 1629, 1639,
1647, 1654, 1662, 1662, 1665, 1683, 1689, 1690, 1698, 1711, 1716,
1717, 1718, 1719, 1726, 1727, 1736, 1741, 1742, 1742, 1756, 1762,
1765, 1770, 1786, 1802, 1805, 1815, 1833, 1834, 1846, 1850, 1851,
1859, 1867, 1888, 1890, 1931, 1949, 1955, 1956, 1965, 1976, 2015,
2016, 2016, 2019, 2023, 2038, 2056, 2060, 2079, 2087, 2090, 2094,
2107, 2110, 2115, 2126, 2129, 2142, 2157, 2172, 2184, 2196, 2209,
2236, 2270, 2322, 2330, 2333, 2342, 2365, 2373, 2393, 2393, 2405,
2412, 2446, 2448, 2448, 2450, 2456, 2462, 2462, 2466, 2466, 2471,
2475, 2483, 2490, 2511, 2544, 2552, 2580, 2583, 2628, 2645, 2659,
2724, 2737, 2750, 2764, 2770, 2783, 2794, 2826, 2829, 2850, 2861,
2870, 2870, 2882, 2886, 2899, 2904, 2907, 2918, 2922, 2964, 2964,
2964, 2970, 2972, 2977, 2997, 3013, 3019, 3054, 3096, 3098, 3115,
3117, 3123, 3167, 3182, 3186, 3187, 3194, 3202, 3218, 3218, 3220,
3230, 3292, 3314, 3329, 3334, 3339, 3350, 3372, 3388, 3392, 3408,
3425, 3440, 3459, 3469, 3489, 3577, 3583, 3596, 3611, 3623, 3635,
3685, 3720, 3727, 3744, 3747, 3751, 3766, 3772, 3778, 3800, 3851,
3876, 3880, 3883, 3886, 3887, 3933, 3953, 4007, 4009, 4020, 4025,
4052, 4064, 4072, 4080, 4082, 4089, 4099, 4100, 4105, 4118, 4158,
4248, 4272, 4302, 4306, 4316, 4368, 4375, 4405, 4426, 4427, 4438,
4458, 4498, 4531, 4537, 4556, 4568, 4644, 4650, 4688, 4695, 4729,
```

```
4731, 4757, 4772, 4794, 4816, 4818, 4826, 4859, 4865, 4873, 4886,
4974, 5052, 5063, 5066, 5077, 5115, 5118, 5156, 5157, 5181, 5214,
5283, 5314, 5346, 5358, 5437, 5496, 5506, 5512, 5524, 5545, 5599,
5633, 5642, 5643, 5656, 5658, 5660, 5684, 5691, 5726, 5736, 5744,
5749, 5754, 5768, 5813, 5829, 5840, 5843, 5869, 5902, 5905, 5913,
5932, 5933, 5948, 5992, 6022, 6032, 6084, 6088, 6093, 6120, 6141,
6162, 6191, 6223, 6280, 6400, 6475, 6499, 6552, 6553, 6556, 6580,
6709, 6829, 6838, 6858, 6897, 6971, 6972, 7002, 7029, 7029, 7076,
7113, 7152, 7176, 7181, 7192, 7260, 7325, 7386, 7471, 7495, 7534,
7578, 7586, 7626, 7657, 7664, 7765, 7797, 7837, 7870, 7871, 7880,
7952, 7978, 8076, 8095, 8169, 8246, 8320, 8347, 8357, 8405, 8503,
8527, 8648, 8666, 8677, 8735, 8753, 8759, 8879, 8880, 8916, 8957,
8967, 8992, 9001, 9023, 9118, 9123, 9361, 9373, 9417, 9456, 9458,
9494, 9499, 9510, 9684, 9733, 9742, 9788, 9827, 9845, 9846, 9878,
9930, 9970, 10054, 10084, 10121, 10147, 10183, 10215, 10295, 10309,
10318, 10409, 10433, 10517, 10568, 10575, 10608, 10659, 10702, 10714,
10824, 10932, 10994, 11122, 11138, 11206, 11276, 11374, 11412, 11553,
11580, 11704, 11718, 11822, 11860, 11869, 12020, 12073, 12153, 12186,
12260, 12299, 12326, 12454, 12507, 12513, 12589, 12721, 12745, 12751,
12826, 12891, 12902, 13051, 13109, 13230, 13230, 13278, 13282, 13334,
13455, 13511, 13803, 14008, 14018, 14050, 14154, 14207, 14265, 14361,
14389, 14454, 14460, 14511, 14537, 14647, 14668, 14694, 14838, 15072,
15146, 15371, 15527, 15724, 15757, 15777, 15840, 16025, 16085, 16130,
16351, 16520, 16572, 16589, 16602, 16691, 16711, 16770, 17033, 17033,
17184, 17293, 17451, 17623, 17653, 17721, 17731, 17813, 18004, 18281,
18417, 18528, 18696, 18773, 18991, 18992, 19088, 19136, 19493, 19510,
19654, 19804, 19811, 19867, 19900, 20187, 20199, 20325, 20631, 20684,
20868, 20934, 20950, 21061, 21101, 21195, 21224, 21346, 21528, 21588,
21721, 21809, 22640, 22900, 23027, 23521, 24020, 24116, 24702, 24728,
24947, 25278, 25295, 25301, 25449, 25631, 25681, 25698, 26153, 26183,
26232, 26245, 26549, 26818, 27021, 27144, 27620, 27763, 28284, 28348,
28370, 28371, 28503, 28538, 28861, 28905, 30111, 30237, 30649, 30989,
31548, 32040, 32387, 32698, 33524, 33816, 34005, 34276, 35350, 35658,
35947, 36037, 36185, 36394, 36459, 36599, 37168, 37345, 37526, 38307,
38435, 38492, 39573, 39832, 40161, 40547, 40685, 41589, 41757, 42439,
42524, 42673, 42809, 43046, 43335, 44459, 44532, 45249, 45516, 46590,
46660, 47652, 47897, 47949, 48544, 48618, 49088, 50103, 50108, 51124,
51806, 52160, 52194, 52856, 53370, 53537, 55316, 56560, 57131, 57142,
57426, 58082, 58127, 59825, 59859, 62523, 68190, 70956, 71215, 71478,
71580, 72913, 73079, 73219, 74425, 77274, 77390, 82003, 83248, 83789,
84441, 84815, 85415, 85935, 86524, 87388, 87862, 88122, 88430, 89781,
90705, 95545, 101997, 102519, 108567, 111074, 114693, 114919, 115694,
116337, 125878, 127181, 130471, 131276, 133391, 137935, 141624,
146493, 157584, 160442, 166800, 168312, 168353, 179595, 193584,
200324, 217232, 220613, 271895, 280039, 281498, 282592, 293925,
331357, 390854, 436280, 460731, 528966, 535275, 564433, 629673,
1198049]
```

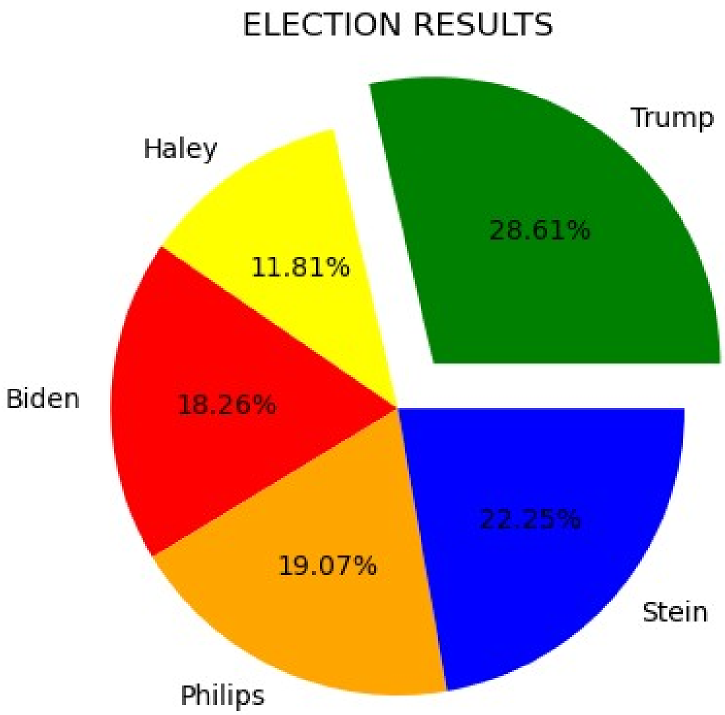
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv("diabetes.csv")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
import matplotlib.pyplot as plt
student_id=list(range(100,110))
student_marks=[72,76,69,89,54,90,78,95,98,60]
student_remarks=[3,3,2,4,2,5,5,3,5,5,3]
plt.bar(student_id,student_marks,color="green")
plt.title("STUDENTS MARK COMPARISON")
plt.xlabel("students")
plt.ylabel("marks")
plt.show()
```


STUDENTS MARK COMPARISON
marks
students
0
20
40
60
80
100
100
102
104
106
108

```python
import matplotlib.pyplot as plt
labels=['Trump','Haley','Biden','Philips','Stein']
votes=[315,130,201,210,245]
colors=['green','yellow','red','orange','blue']
explode=(0.2,0,0,0,0)
plt.pie(votes, labels=labels, colors=colors,
explode=explode,autopct='%0.2f%%')
plt.title('ELECTION RESULTS')
plt.show()
```




```

```

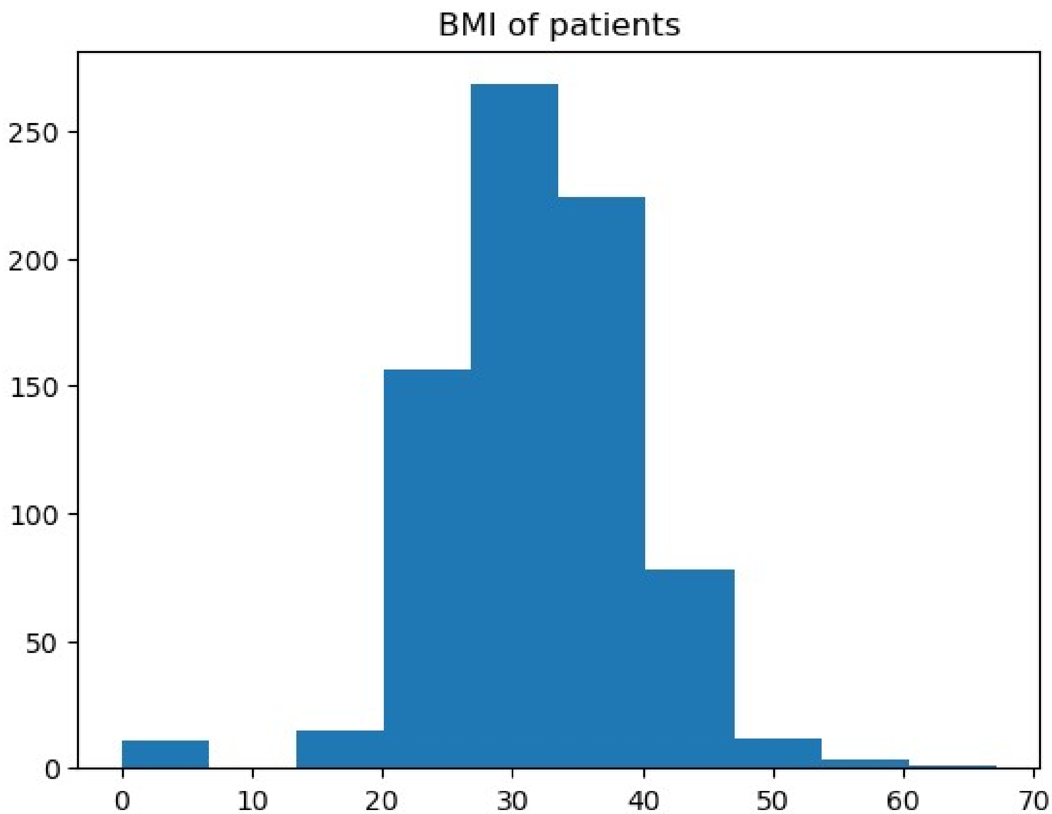
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv("diabetes.csv")

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

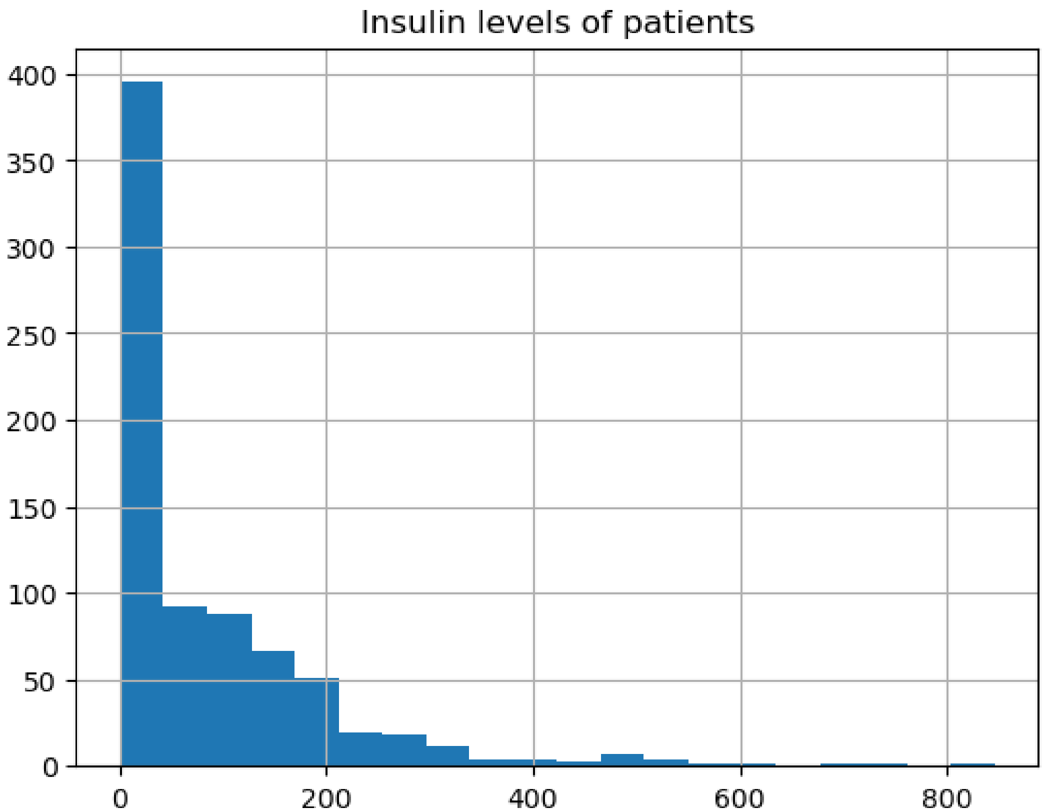
```python
df.describe()
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness
Insulin  \
count   768.000000  768.000000     768.000000     768.000000
768.000000
mean      3.845052  120.894531      69.105469      20.536458
79.799479
std       3.369578   31.972618      19.355807      15.952218
115.244002
min       0.000000    0.000000       0.000000       0.000000
0.000000
25%       1.000000   99.000000      62.000000       0.000000
0.000000
50%       3.000000  117.000000      72.000000      23.000000
30.500000
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
```

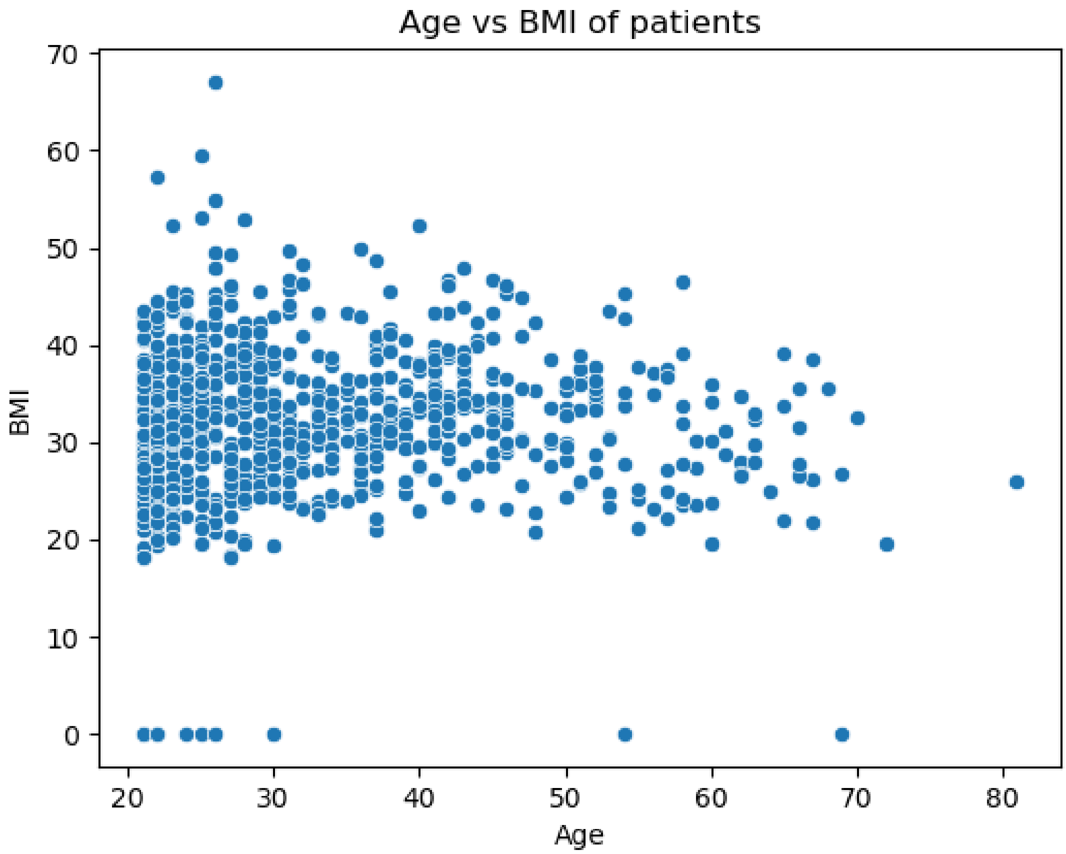
```
min      0.000000                    0.078000   21.000000    0.000000
25%     27.300000                    0.243750   24.000000    0.000000
50%     32.000000                    0.372500   29.000000    0.000000
75%     36.600000                    0.626250   41.000000    1.000000
max     67.100000                    2.420000   81.000000    1.000000

#univariate analysis

plt.hist(df['BMI'])
plt.title("BMI of patients")
plt.show()
```



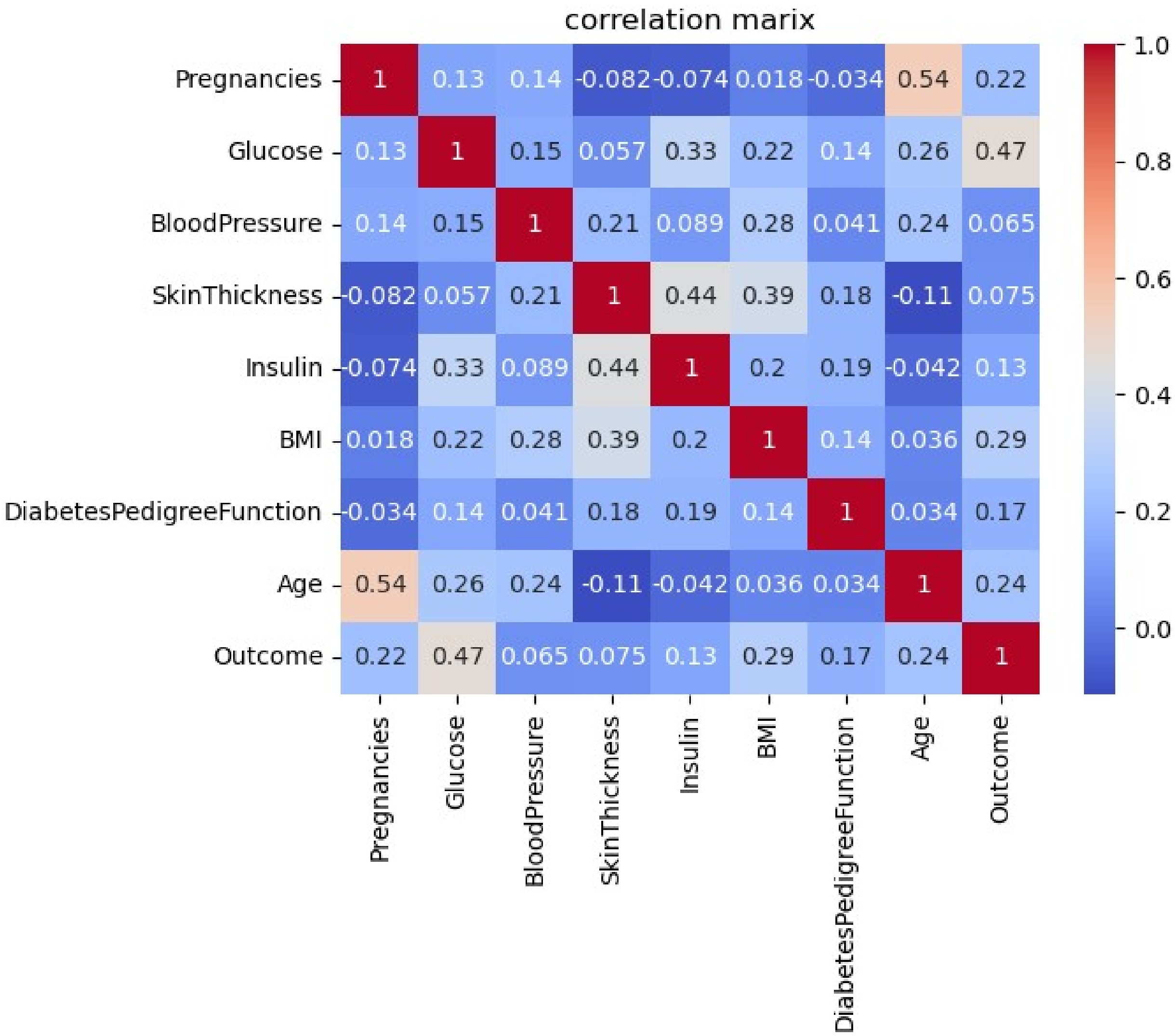

```
df['Insulin'].hist(bins=20)
plt.title("Insulin levels of patients")
plt.show()
```




```
#bivariate analysis

sns.scatterplot(x="Age",y="BMI",data=df)
plt.title("Age vs BMI of patients")
plt.show()
```




```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('correlation marix')
plt.show()
```




```
df["Insulin"].corr(df['BloodPressure'])
```

```
0.08893337837319315
```

```
#it is weak positively correlted
```

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

df=pd.read_csv("Salary_data.csv")
df.describe()
```

```
       YearsExperience         Salary
count        30.000000      30.000000
mean          5.313333   76003.000000
std           2.837888   27414.429785
min           1.100000   37731.000000
25%           3.200000   56720.750000
50%           4.700000   65237.000000
75%           7.700000  100544.750000
max          10.500000  122391.000000
```

```python
df.duplicated().sum()
```

```
0
```

```python
df.isnull().sum()
```

```
YearsExperience    0
Salary             0
dtype: int64
```

```python
features=df.iloc[:,[0]].values
label=df.iloc[:,[1]].values

features
```

```
array([[ 1.1],
       [ 1.3],
       [ 1.5],
       [ 2. ],
       [ 2.2],
       [ 2.9],
       [ 3. ],
       [ 3.2],
       [ 3.2],
       [ 3.7],
       [ 3.9],
       [ 4. ],
       [ 4. ],
       [ 4.1],
       [ 4.5],
       [ 4.9],
       [ 5.1],
       [ 5.3],
```

```
       [ 5.9],
       [ 6. ],
       [ 6.8],
       [ 7.1],
       [ 7.9],
       [ 8.2],
       [ 8.7],
       [ 9. ],
       [ 9.5],
       [ 9.6],
       [10.3],
       [10.5]])
```

```
label
```

```
array([[ 39343],
       [ 46205],
       [ 37731],
       [ 43525],
       [ 39891],
       [ 56642],
       [ 60150],
       [ 54445],
       [ 64445],
       [ 57189],
       [ 63218],
       [ 55794],
       [ 56957],
       [ 57081],
       [ 61111],
       [ 67938],
       [ 66029],
       [ 83088],
       [ 81363],
       [ 93940],
       [ 91738],
       [ 98273],
       [101302],
       [113812],
       [109431],
       [105582],
       [116969],
       [112635],
       [122391],
       [121872]], dtype=int64)
```

```
x_train,x_test,y_train,y_test=train_test_split(features,label,test_siz
e=0.2,random_state=42)
```

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)

LinearRegression()

model.score(x_train,y_train)

0.9645401573418146

model.score(x_test,y_test)

0.9024461774180497

#further applying linear regression for training data set
x_train2, x_test2, y_train2,
y_test2=train_test_split(x_train,y_train,test_size=0.2,random_state=42
)



new_model=LinearRegression()
new_model.fit(x_train2,y_train2)

LinearRegression()

new_model.score(x_train2,y_train2)

0.9581077653380569

new_model.score(x_test2,y_test2)

0.9836657383524982

model.coef_

array([[9423.81532303]])

model.intercept_

array([25321.58301178])

new_model.coef_

array([[9306.04618233]])

new_model.intercept_

array([26241.34560505])

import pickle
pickle.dump(model,open('SalaryPred.model','wb'))

model=pickle.load(open('SalaryPred.model','rb'))
```

```
yr_of_exp=float(input("Enter Years of Experience: "))
yr_of_exp_NP=np.array([[yr_of_exp]])
Salary=model.predict(yr_of_exp_NP)
```

```
Enter Years of Experience: 25
```

```
print("estimated salary for {} years is {}".format(yr_of_exp,Salary))
```

```
estimated salary for 25.0 years is [[260916.96608755]]
```

```
new_Salary=new_model.predict(yr_of_exp_NP)
print("estimated salary for {} years is
{}".format(yr_of_exp,new_Salary))
```

```
estimated salary for 25.0 years is [[258892.50016338]]
```

```python
#logistic regression

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

df=pd.read_csv("Social_Network_Ads.csv")
df.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```python
df.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

```python
features=df.iloc[:,[2,3]].values
features
```

```
array([[    19,  19000],
       [    35,  20000],
       [    26,  43000],
       [    27,  57000],
       [    19,  76000],
       [    27,  58000],
       [    27,  84000],
       [    32, 150000],
       [    25,  33000],
       [    35,  65000],
       [    26,  80000],
       [    26,  52000],
       [    20,  86000],
       [    32,  18000],
       [    18,  82000],
       [    29,  80000],
       [    47,  25000],
       [    45,  26000],
       [    46,  28000],
```

```
       [    48,  29000],
       [    45,  22000],
       [    47,  49000],
       [    48,  41000],
       [    45,  22000],
       [    46,  23000],
       [    47,  20000],
       [    49,  28000],
       [    47,  30000],
       [    29,  43000],
       [    31,  18000],
       [    31,  74000],
       [    27, 137000],
       [    21,  16000],
       [    28,  44000],
       [    27,  90000],
       [    35,  27000],
       [    33,  28000],
       [    30,  49000],
       [    26,  72000],
       [    27,  31000],
       [    27,  17000],
       [    33,  51000],
       [    35, 108000],
       [    30,  15000],
       [    28,  84000],
       [    23,  20000],
       [    25,  79000],
       [    27,  54000],
       [    30, 135000],
       [    31,  89000],
       [    24,  32000],
       [    18,  44000],
       [    29,  83000],
       [    35,  23000],
       [    27,  58000],
       [    24,  55000],
       [    23,  48000],
       [    28,  79000],
       [    22,  18000],
       [    32, 117000],
       [    27,  20000],
       [    25,  87000],
       [    23,  66000],
       [    32, 120000],
       [    59,  83000],
       [    24,  58000],
       [    24,  19000],
       [    23,  82000],
```

```
       [    22,  63000],
       [    31,  68000],
       [    25,  80000],
       [    24,  27000],
       [    20,  23000],
       [    33, 113000],
       [    32,  18000],
       [    34, 112000],
       [    18,  52000],
       [    22,  27000],
       [    28,  87000],
       [    26,  17000],
       [    30,  80000],
       [    39,  42000],
       [    20,  49000],
       [    35,  88000],
       [    30,  62000],
       [    31, 118000],
       [    24,  55000],
       [    28,  85000],
       [    26,  81000],
       [    35,  50000],
       [    22,  81000],
       [    30, 116000],
       [    26,  15000],
       [    29,  28000],
       [    29,  83000],
       [    35,  44000],
       [    35,  25000],
       [    28, 123000],
       [    35,  73000],
       [    28,  37000],
       [    27,  88000],
       [    28,  59000],
       [    32,  86000],
       [    33, 149000],
       [    19,  21000],
       [    21,  72000],
       [    26,  35000],
       [    27,  89000],
       [    26,  86000],
       [    38,  80000],
       [    39,  71000],
       [    37,  71000],
       [    38,  61000],
       [    37,  55000],
       [    42,  80000],
       [    40,  57000],
       [    35,  75000],
```

```
       [     36,  52000],
       [     40,  59000],
       [     41,  59000],
       [     36,  75000],
       [     37,  72000],
       [     40,  75000],
       [     35,  53000],
       [     41,  51000],
       [     39,  61000],
       [     42,  65000],
       [     26,  32000],
       [     30,  17000],
       [     26,  84000],
       [     31,  58000],
       [     33,  31000],
       [     30,  87000],
       [     21,  68000],
       [     28,  55000],
       [     23,  63000],
       [     20,  82000],
       [     30, 107000],
       [     28,  59000],
       [     19,  25000],
       [     19,  85000],
       [     18,  68000],
       [     35,  59000],
       [     30,  89000],
       [     34,  25000],
       [     24,  89000],
       [     27,  96000],
       [     41,  30000],
       [     29,  61000],
       [     20,  74000],
       [     26,  15000],
       [     41,  45000],
       [     31,  76000],
       [     36,  50000],
       [     40,  47000],
       [     31,  15000],
       [     46,  59000],
       [     29,  75000],
       [     26,  30000],
       [     32, 135000],
       [     32, 100000],
       [     25,  90000],
       [     37,  33000],
       [     35,  38000],
       [     33,  69000],
       [     18,  86000],
```

```
       [     22,  55000],
       [     35,  71000],
       [     29, 148000],
       [     29,  47000],
       [     21,  88000],
       [     34, 115000],
       [     26, 118000],
       [     34,  43000],
       [     34,  72000],
       [     23,  28000],
       [     35,  47000],
       [     25,  22000],
       [     24,  23000],
       [     31,  34000],
       [     26,  16000],
       [     31,  71000],
       [     32, 117000],
       [     33,  43000],
       [     33,  60000],
       [     31,  66000],
       [     20,  82000],
       [     33,  41000],
       [     35,  72000],
       [     28,  32000],
       [     24,  84000],
       [     19,  26000],
       [     29,  43000],
       [     19,  70000],
       [     28,  89000],
       [     34,  43000],
       [     30,  79000],
       [     20,  36000],
       [     26,  80000],
       [     35,  22000],
       [     35,  39000],
       [     49,  74000],
       [     39, 134000],
       [     41,  71000],
       [     58, 101000],
       [     47,  47000],
       [     55, 130000],
       [     52, 114000],
       [     40, 142000],
       [     46,  22000],
       [     48,  96000],
       [     52, 150000],
       [     59,  42000],
       [     35,  58000],
       [     47,  43000],
```

```
       [    60, 108000],
       [    49,  65000],
       [    40,  78000],
       [    46,  96000],
       [    59, 143000],
       [    41,  80000],
       [    35,  91000],
       [    37, 144000],
       [    60, 102000],
       [    35,  60000],
       [    37,  53000],
       [    36, 126000],
       [    56, 133000],
       [    40,  72000],
       [    42,  80000],
       [    35, 147000],
       [    39,  42000],
       [    40, 107000],
       [    49,  86000],
       [    38, 112000],
       [    46,  79000],
       [    40,  57000],
       [    37,  80000],
       [    46,  82000],
       [    53, 143000],
       [    42, 149000],
       [    38,  59000],
       [    50,  88000],
       [    56, 104000],
       [    41,  72000],
       [    51, 146000],
       [    35,  50000],
       [    57, 122000],
       [    41,  52000],
       [    35,  97000],
       [    44,  39000],
       [    37,  52000],
       [    48, 134000],
       [    37, 146000],
       [    50,  44000],
       [    52,  90000],
       [    41,  72000],
       [    40,  57000],
       [    58,  95000],
       [    45, 131000],
       [    35,  77000],
       [    36, 144000],
       [    55, 125000],
       [    35,  72000],
```

```
       [    48,  90000],
       [    42, 108000],
       [    40,  75000],
       [    37,  74000],
       [    47, 144000],
       [    40,  61000],
       [    43, 133000],
       [    59,  76000],
       [    60,  42000],
       [    39, 106000],
       [    57,  26000],
       [    57,  74000],
       [    38,  71000],
       [    49,  88000],
       [    52,  38000],
       [    50,  36000],
       [    59,  88000],
       [    35,  61000],
       [    37,  70000],
       [    52,  21000],
       [    48, 141000],
       [    37,  93000],
       [    37,  62000],
       [    48, 138000],
       [    41,  79000],
       [    37,  78000],
       [    39, 134000],
       [    49,  89000],
       [    55,  39000],
       [    37,  77000],
       [    35,  57000],
       [    36,  63000],
       [    42,  73000],
       [    43, 112000],
       [    45,  79000],
       [    46, 117000],
       [    58,  38000],
       [    48,  74000],
       [    37, 137000],
       [    37,  79000],
       [    40,  60000],
       [    42,  54000],
       [    51, 134000],
       [    47, 113000],
       [    36, 125000],
       [    38,  50000],
       [    42,  70000],
       [    39,  96000],
       [    38,  50000],
```

```
       [    49, 141000],
       [    39,  79000],
       [    39,  75000],
       [    54, 104000],
       [    35,  55000],
       [    45,  32000],
       [    36,  60000],
       [    52, 138000],
       [    53,  82000],
       [    41,  52000],
       [    48,  30000],
       [    48, 131000],
       [    41,  60000],
       [    41,  72000],
       [    42,  75000],
       [    36, 118000],
       [    47, 107000],
       [    38,  51000],
       [    48, 119000],
       [    42,  65000],
       [    40,  65000],
       [    57,  60000],
       [    36,  54000],
       [    58, 144000],
       [    35,  79000],
       [    38,  55000],
       [    39, 122000],
       [    53, 104000],
       [    35,  75000],
       [    38,  65000],
       [    47,  51000],
       [    47, 105000],
       [    41,  63000],
       [    53,  72000],
       [    54, 108000],
       [    39,  77000],
       [    38,  61000],
       [    38, 113000],
       [    37,  75000],
       [    42,  90000],
       [    37,  57000],
       [    36,  99000],
       [    60,  34000],
       [    54,  70000],
       [    41,  72000],
       [    40,  71000],
       [    42,  54000],
       [    43, 129000],
       [    53,  34000],
```

```
       [    47,  50000],
       [    42,  79000],
       [    42, 104000],
       [    59,  29000],
       [    58,  47000],
       [    46,  88000],
       [    38,  71000],
       [    54,  26000],
       [    60,  46000],
       [    60,  83000],
       [    39,  73000],
       [    59, 130000],
       [    37,  80000],
       [    46,  32000],
       [    46,  74000],
       [    42,  53000],
       [    41,  87000],
       [    58,  23000],
       [    42,  64000],
       [    48,  33000],
       [    44, 139000],
       [    49,  28000],
       [    57,  33000],
       [    56,  60000],
       [    49,  39000],
       [    39,  71000],
       [    47,  34000],
       [    48,  35000],
       [    48,  33000],
       [    47,  23000],
       [    45,  45000],
       [    60,  42000],
       [    39,  59000],
       [    46,  41000],
       [    51,  23000],
       [    50,  20000],
       [    36,  33000],
       [    49,  36000]])

label=df.iloc[:,-1].values
label

array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0,
```

```
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1,
0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0,
1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1,
1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1,
0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
1,
       1, 1, 0, 1])
```

```python
for i in range(1,401):

x_train,x_test,y_train,y_test=train_test_split(features,label,test_siz
e=0.2,random_state=42)
    model=LogisticRegression()
    model.fit(x_train,y_train)
    train_score=model.score(x_train,y_train)
    test_score=model.score(x_test,y_test)
    if test_score > train_score:
        print("test {} train {} random state
{}".format(test_score,train_score,i))
```

```
test 0.8875 train 0.8375 random state 1
test 0.8875 train 0.8375 random state 2
test 0.8875 train 0.8375 random state 3
test 0.8875 train 0.8375 random state 4
test 0.8875 train 0.8375 random state 5
test 0.8875 train 0.8375 random state 6
test 0.8875 train 0.8375 random state 7
test 0.8875 train 0.8375 random state 8
```

```
test 0.8875 train 0.8375 random state 9
test 0.8875 train 0.8375 random state 10
test 0.8875 train 0.8375 random state 11
test 0.8875 train 0.8375 random state 12
test 0.8875 train 0.8375 random state 13
test 0.8875 train 0.8375 random state 14
test 0.8875 train 0.8375 random state 15
test 0.8875 train 0.8375 random state 16
test 0.8875 train 0.8375 random state 17
test 0.8875 train 0.8375 random state 18
test 0.8875 train 0.8375 random state 19
test 0.8875 train 0.8375 random state 20
test 0.8875 train 0.8375 random state 21
test 0.8875 train 0.8375 random state 22
test 0.8875 train 0.8375 random state 23
test 0.8875 train 0.8375 random state 24
test 0.8875 train 0.8375 random state 25
test 0.8875 train 0.8375 random state 26
test 0.8875 train 0.8375 random state 27
test 0.8875 train 0.8375 random state 28
test 0.8875 train 0.8375 random state 29
test 0.8875 train 0.8375 random state 30
test 0.8875 train 0.8375 random state 31
test 0.8875 train 0.8375 random state 32
test 0.8875 train 0.8375 random state 33
test 0.8875 train 0.8375 random state 34
test 0.8875 train 0.8375 random state 35
test 0.8875 train 0.8375 random state 36
test 0.8875 train 0.8375 random state 37
test 0.8875 train 0.8375 random state 38
test 0.8875 train 0.8375 random state 39
test 0.8875 train 0.8375 random state 40
test 0.8875 train 0.8375 random state 41
test 0.8875 train 0.8375 random state 42
test 0.8875 train 0.8375 random state 43
test 0.8875 train 0.8375 random state 44
test 0.8875 train 0.8375 random state 45
test 0.8875 train 0.8375 random state 46
test 0.8875 train 0.8375 random state 47
test 0.8875 train 0.8375 random state 48
test 0.8875 train 0.8375 random state 49
test 0.8875 train 0.8375 random state 50
test 0.8875 train 0.8375 random state 51
test 0.8875 train 0.8375 random state 52
test 0.8875 train 0.8375 random state 53
test 0.8875 train 0.8375 random state 54
test 0.8875 train 0.8375 random state 55
test 0.8875 train 0.8375 random state 56
test 0.8875 train 0.8375 random state 57
```

```
test 0.8875 train 0.8375 random state 58
test 0.8875 train 0.8375 random state 59
test 0.8875 train 0.8375 random state 60
test 0.8875 train 0.8375 random state 61
test 0.8875 train 0.8375 random state 62
test 0.8875 train 0.8375 random state 63
test 0.8875 train 0.8375 random state 64
test 0.8875 train 0.8375 random state 65
test 0.8875 train 0.8375 random state 66
test 0.8875 train 0.8375 random state 67
test 0.8875 train 0.8375 random state 68
test 0.8875 train 0.8375 random state 69
test 0.8875 train 0.8375 random state 70
test 0.8875 train 0.8375 random state 71
test 0.8875 train 0.8375 random state 72
test 0.8875 train 0.8375 random state 73
test 0.8875 train 0.8375 random state 74
test 0.8875 train 0.8375 random state 75
test 0.8875 train 0.8375 random state 76
test 0.8875 train 0.8375 random state 77
test 0.8875 train 0.8375 random state 78
test 0.8875 train 0.8375 random state 79
test 0.8875 train 0.8375 random state 80
test 0.8875 train 0.8375 random state 81
test 0.8875 train 0.8375 random state 82
test 0.8875 train 0.8375 random state 83
test 0.8875 train 0.8375 random state 84
test 0.8875 train 0.8375 random state 85
test 0.8875 train 0.8375 random state 86
test 0.8875 train 0.8375 random state 87
test 0.8875 train 0.8375 random state 88
test 0.8875 train 0.8375 random state 89
test 0.8875 train 0.8375 random state 90
test 0.8875 train 0.8375 random state 91
test 0.8875 train 0.8375 random state 92
test 0.8875 train 0.8375 random state 93
test 0.8875 train 0.8375 random state 94
test 0.8875 train 0.8375 random state 95
test 0.8875 train 0.8375 random state 96
test 0.8875 train 0.8375 random state 97
test 0.8875 train 0.8375 random state 98
test 0.8875 train 0.8375 random state 99
test 0.8875 train 0.8375 random state 100
test 0.8875 train 0.8375 random state 101
test 0.8875 train 0.8375 random state 102
test 0.8875 train 0.8375 random state 103
test 0.8875 train 0.8375 random state 104
test 0.8875 train 0.8375 random state 105
test 0.8875 train 0.8375 random state 106
```

```
test 0.8875 train 0.8375 random state 107
test 0.8875 train 0.8375 random state 108
test 0.8875 train 0.8375 random state 109
test 0.8875 train 0.8375 random state 110
test 0.8875 train 0.8375 random state 111
test 0.8875 train 0.8375 random state 112
test 0.8875 train 0.8375 random state 113
test 0.8875 train 0.8375 random state 114
test 0.8875 train 0.8375 random state 115
test 0.8875 train 0.8375 random state 116
test 0.8875 train 0.8375 random state 117
test 0.8875 train 0.8375 random state 118
test 0.8875 train 0.8375 random state 119
test 0.8875 train 0.8375 random state 120
test 0.8875 train 0.8375 random state 121
test 0.8875 train 0.8375 random state 122
test 0.8875 train 0.8375 random state 123
test 0.8875 train 0.8375 random state 124
test 0.8875 train 0.8375 random state 125
test 0.8875 train 0.8375 random state 126
test 0.8875 train 0.8375 random state 127
test 0.8875 train 0.8375 random state 128
test 0.8875 train 0.8375 random state 129
test 0.8875 train 0.8375 random state 130
test 0.8875 train 0.8375 random state 131
test 0.8875 train 0.8375 random state 132
test 0.8875 train 0.8375 random state 133
test 0.8875 train 0.8375 random state 134
test 0.8875 train 0.8375 random state 135
test 0.8875 train 0.8375 random state 136
test 0.8875 train 0.8375 random state 137
test 0.8875 train 0.8375 random state 138
test 0.8875 train 0.8375 random state 139
test 0.8875 train 0.8375 random state 140
test 0.8875 train 0.8375 random state 141
test 0.8875 train 0.8375 random state 142
test 0.8875 train 0.8375 random state 143
test 0.8875 train 0.8375 random state 144
test 0.8875 train 0.8375 random state 145
test 0.8875 train 0.8375 random state 146
test 0.8875 train 0.8375 random state 147
test 0.8875 train 0.8375 random state 148
test 0.8875 train 0.8375 random state 149
test 0.8875 train 0.8375 random state 150
test 0.8875 train 0.8375 random state 151
test 0.8875 train 0.8375 random state 152
test 0.8875 train 0.8375 random state 153
test 0.8875 train 0.8375 random state 154
test 0.8875 train 0.8375 random state 155
```

```
test 0.8875 train 0.8375 random state 156
test 0.8875 train 0.8375 random state 157
test 0.8875 train 0.8375 random state 158
test 0.8875 train 0.8375 random state 159
test 0.8875 train 0.8375 random state 160
test 0.8875 train 0.8375 random state 161
test 0.8875 train 0.8375 random state 162
test 0.8875 train 0.8375 random state 163
test 0.8875 train 0.8375 random state 164
test 0.8875 train 0.8375 random state 165
test 0.8875 train 0.8375 random state 166
test 0.8875 train 0.8375 random state 167
test 0.8875 train 0.8375 random state 168
test 0.8875 train 0.8375 random state 169
test 0.8875 train 0.8375 random state 170
test 0.8875 train 0.8375 random state 171
test 0.8875 train 0.8375 random state 172
test 0.8875 train 0.8375 random state 173
test 0.8875 train 0.8375 random state 174
test 0.8875 train 0.8375 random state 175
test 0.8875 train 0.8375 random state 176
test 0.8875 train 0.8375 random state 177
test 0.8875 train 0.8375 random state 178
test 0.8875 train 0.8375 random state 179
test 0.8875 train 0.8375 random state 180
test 0.8875 train 0.8375 random state 181
test 0.8875 train 0.8375 random state 182
test 0.8875 train 0.8375 random state 183
test 0.8875 train 0.8375 random state 184
test 0.8875 train 0.8375 random state 185
test 0.8875 train 0.8375 random state 186
test 0.8875 train 0.8375 random state 187
test 0.8875 train 0.8375 random state 188
test 0.8875 train 0.8375 random state 189
test 0.8875 train 0.8375 random state 190
test 0.8875 train 0.8375 random state 191
test 0.8875 train 0.8375 random state 192
test 0.8875 train 0.8375 random state 193
test 0.8875 train 0.8375 random state 194
test 0.8875 train 0.8375 random state 195
test 0.8875 train 0.8375 random state 196
test 0.8875 train 0.8375 random state 197
test 0.8875 train 0.8375 random state 198
test 0.8875 train 0.8375 random state 199
test 0.8875 train 0.8375 random state 200
test 0.8875 train 0.8375 random state 201
test 0.8875 train 0.8375 random state 202
test 0.8875 train 0.8375 random state 203
test 0.8875 train 0.8375 random state 204
```

```
test 0.8875 train 0.8375 random state 205
test 0.8875 train 0.8375 random state 206
test 0.8875 train 0.8375 random state 207
test 0.8875 train 0.8375 random state 208
test 0.8875 train 0.8375 random state 209
test 0.8875 train 0.8375 random state 210
test 0.8875 train 0.8375 random state 211
test 0.8875 train 0.8375 random state 212
test 0.8875 train 0.8375 random state 213
test 0.8875 train 0.8375 random state 214
test 0.8875 train 0.8375 random state 215
test 0.8875 train 0.8375 random state 216
test 0.8875 train 0.8375 random state 217
test 0.8875 train 0.8375 random state 218
test 0.8875 train 0.8375 random state 219
test 0.8875 train 0.8375 random state 220
test 0.8875 train 0.8375 random state 221
test 0.8875 train 0.8375 random state 222
test 0.8875 train 0.8375 random state 223
test 0.8875 train 0.8375 random state 224
test 0.8875 train 0.8375 random state 225
test 0.8875 train 0.8375 random state 226
test 0.8875 train 0.8375 random state 227
test 0.8875 train 0.8375 random state 228
test 0.8875 train 0.8375 random state 229
test 0.8875 train 0.8375 random state 230
test 0.8875 train 0.8375 random state 231
test 0.8875 train 0.8375 random state 232
test 0.8875 train 0.8375 random state 233
test 0.8875 train 0.8375 random state 234
test 0.8875 train 0.8375 random state 235
test 0.8875 train 0.8375 random state 236
test 0.8875 train 0.8375 random state 237
test 0.8875 train 0.8375 random state 238
test 0.8875 train 0.8375 random state 239
test 0.8875 train 0.8375 random state 240
test 0.8875 train 0.8375 random state 241
test 0.8875 train 0.8375 random state 242
test 0.8875 train 0.8375 random state 243
test 0.8875 train 0.8375 random state 244
test 0.8875 train 0.8375 random state 245
test 0.8875 train 0.8375 random state 246
test 0.8875 train 0.8375 random state 247
test 0.8875 train 0.8375 random state 248
test 0.8875 train 0.8375 random state 249
test 0.8875 train 0.8375 random state 250
test 0.8875 train 0.8375 random state 251
test 0.8875 train 0.8375 random state 252
test 0.8875 train 0.8375 random state 253
```

```
test 0.8875 train 0.8375 random state 254
test 0.8875 train 0.8375 random state 255
test 0.8875 train 0.8375 random state 256
test 0.8875 train 0.8375 random state 257
test 0.8875 train 0.8375 random state 258
test 0.8875 train 0.8375 random state 259
test 0.8875 train 0.8375 random state 260
test 0.8875 train 0.8375 random state 261
test 0.8875 train 0.8375 random state 262
test 0.8875 train 0.8375 random state 263
test 0.8875 train 0.8375 random state 264
test 0.8875 train 0.8375 random state 265
test 0.8875 train 0.8375 random state 266
test 0.8875 train 0.8375 random state 267
test 0.8875 train 0.8375 random state 268
test 0.8875 train 0.8375 random state 269
test 0.8875 train 0.8375 random state 270
test 0.8875 train 0.8375 random state 271
test 0.8875 train 0.8375 random state 272
test 0.8875 train 0.8375 random state 273
test 0.8875 train 0.8375 random state 274
test 0.8875 train 0.8375 random state 275
test 0.8875 train 0.8375 random state 276
test 0.8875 train 0.8375 random state 277
test 0.8875 train 0.8375 random state 278
test 0.8875 train 0.8375 random state 279
test 0.8875 train 0.8375 random state 280
test 0.8875 train 0.8375 random state 281
test 0.8875 train 0.8375 random state 282
test 0.8875 train 0.8375 random state 283
test 0.8875 train 0.8375 random state 284
test 0.8875 train 0.8375 random state 285
test 0.8875 train 0.8375 random state 286
test 0.8875 train 0.8375 random state 287
test 0.8875 train 0.8375 random state 288
test 0.8875 train 0.8375 random state 289
test 0.8875 train 0.8375 random state 290
test 0.8875 train 0.8375 random state 291
test 0.8875 train 0.8375 random state 292
test 0.8875 train 0.8375 random state 293
test 0.8875 train 0.8375 random state 294
test 0.8875 train 0.8375 random state 295
test 0.8875 train 0.8375 random state 296
test 0.8875 train 0.8375 random state 297
test 0.8875 train 0.8375 random state 298
test 0.8875 train 0.8375 random state 299
test 0.8875 train 0.8375 random state 300
test 0.8875 train 0.8375 random state 301
test 0.8875 train 0.8375 random state 302
```

```
test 0.8875 train 0.8375 random state 303
test 0.8875 train 0.8375 random state 304
test 0.8875 train 0.8375 random state 305
test 0.8875 train 0.8375 random state 306
test 0.8875 train 0.8375 random state 307
test 0.8875 train 0.8375 random state 308
test 0.8875 train 0.8375 random state 309
test 0.8875 train 0.8375 random state 310
test 0.8875 train 0.8375 random state 311
test 0.8875 train 0.8375 random state 312
test 0.8875 train 0.8375 random state 313
test 0.8875 train 0.8375 random state 314
test 0.8875 train 0.8375 random state 315
test 0.8875 train 0.8375 random state 316
test 0.8875 train 0.8375 random state 317
test 0.8875 train 0.8375 random state 318
test 0.8875 train 0.8375 random state 319
test 0.8875 train 0.8375 random state 320
test 0.8875 train 0.8375 random state 321
test 0.8875 train 0.8375 random state 322
test 0.8875 train 0.8375 random state 323
test 0.8875 train 0.8375 random state 324
test 0.8875 train 0.8375 random state 325
test 0.8875 train 0.8375 random state 326
test 0.8875 train 0.8375 random state 327
test 0.8875 train 0.8375 random state 328
test 0.8875 train 0.8375 random state 329
test 0.8875 train 0.8375 random state 330
test 0.8875 train 0.8375 random state 331
test 0.8875 train 0.8375 random state 332
test 0.8875 train 0.8375 random state 333
test 0.8875 train 0.8375 random state 334
test 0.8875 train 0.8375 random state 335
test 0.8875 train 0.8375 random state 336
test 0.8875 train 0.8375 random state 337
test 0.8875 train 0.8375 random state 338
test 0.8875 train 0.8375 random state 339
test 0.8875 train 0.8375 random state 340
test 0.8875 train 0.8375 random state 341
test 0.8875 train 0.8375 random state 342
test 0.8875 train 0.8375 random state 343
test 0.8875 train 0.8375 random state 344
test 0.8875 train 0.8375 random state 345
test 0.8875 train 0.8375 random state 346
test 0.8875 train 0.8375 random state 347
test 0.8875 train 0.8375 random state 348
test 0.8875 train 0.8375 random state 349
test 0.8875 train 0.8375 random state 350
test 0.8875 train 0.8375 random state 351
```

```
test 0.8875 train 0.8375 random state 352
test 0.8875 train 0.8375 random state 353
test 0.8875 train 0.8375 random state 354
test 0.8875 train 0.8375 random state 355
test 0.8875 train 0.8375 random state 356
test 0.8875 train 0.8375 random state 357
test 0.8875 train 0.8375 random state 358
test 0.8875 train 0.8375 random state 359
test 0.8875 train 0.8375 random state 360
test 0.8875 train 0.8375 random state 361
test 0.8875 train 0.8375 random state 362
test 0.8875 train 0.8375 random state 363
test 0.8875 train 0.8375 random state 364
test 0.8875 train 0.8375 random state 365
test 0.8875 train 0.8375 random state 366
test 0.8875 train 0.8375 random state 367
test 0.8875 train 0.8375 random state 368
test 0.8875 train 0.8375 random state 369
test 0.8875 train 0.8375 random state 370
test 0.8875 train 0.8375 random state 371
test 0.8875 train 0.8375 random state 372
test 0.8875 train 0.8375 random state 373
test 0.8875 train 0.8375 random state 374
test 0.8875 train 0.8375 random state 375
test 0.8875 train 0.8375 random state 376
test 0.8875 train 0.8375 random state 377
test 0.8875 train 0.8375 random state 378
test 0.8875 train 0.8375 random state 379
test 0.8875 train 0.8375 random state 380
test 0.8875 train 0.8375 random state 381
test 0.8875 train 0.8375 random state 382
test 0.8875 train 0.8375 random state 383
test 0.8875 train 0.8375 random state 384
test 0.8875 train 0.8375 random state 385
test 0.8875 train 0.8375 random state 386
test 0.8875 train 0.8375 random state 387
test 0.8875 train 0.8375 random state 388
test 0.8875 train 0.8375 random state 389
test 0.8875 train 0.8375 random state 390
test 0.8875 train 0.8375 random state 391
test 0.8875 train 0.8375 random state 392
test 0.8875 train 0.8375 random state 393
test 0.8875 train 0.8375 random state 394
test 0.8875 train 0.8375 random state 395
test 0.8875 train 0.8375 random state 396
test 0.8875 train 0.8375 random state 397
test 0.8875 train 0.8375 random state 398
test 0.8875 train 0.8375 random state 399
test 0.8875 train 0.8375 random state 400
```

```
x_train,x_test,y_train,y_test=train_test_split(features,label,test_siz
e=0.2,random_state=42)
finalmodel=LogisticRegression()
finalmodel.fit(x_train,y_train)

finalmodel.score(x_train,y_train)

0.8375

finalmodel.score(x_test,y_test)

0.8875
```

```python
#z-test
import numpy as np
import scipy.stats as stats

sample_data=np.array([152, 148, 151, 149, 147, 153, 150, 148, 152,
149,151, 150, 149, 152, 151, 148, 150, 152, 149, 150,148, 153, 151,
150, 149, 152, 148, 151, 150, 153])
population_mean=150 #given
sample_mean=np.mean(sample_data)
sample_std=np.std(sample_data, ddof=1)
n=len(sample_data)

z_stats=(sample_mean - population_mean)/(sample_std / np.sqrt(n))
p_value=2 * (1 - stats.norm.cdf(np.abs(z_stats))) #since it is a two
tailed test

print(f"Sample Mean: {sample_mean:.2f}")
print(f"Z-Statistic: {z_stats:.4f}")
print(f"P-Value: {p_value:.4f}")
```

```
Sample Mean: 150.20
Z-Statistic: 0.6406
P-Value: 0.5218
```

```python
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The average weight is
significantly different from 150 grams.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in average weight from 150 grams.")
```

```
Fail to reject the null hypothesis: There is no significant difference
in average weight from 150 grams.
```

```python
#t-test

np.random.seed(42)
sample_size=25
sample_data=np.random.normal(loc=102, scale=15, size=sample_size)
population_mean=150
sample_mean=np.mean(sample_data)
sample_std=np.std(sample_data, ddof=1)
n=len(sample_data)
t_stats,p_value=stats.ttest_1samp(sample_data,population_mean)

print(f"Sample Mean: {sample_mean:.2f}")
print(f"T-Statistic: {t_stats:.4f}")
print(f"P-Value: {p_value:.4f}")
```

```
Sample Mean: 99.55
T-Statistic: -17.5814
P-Value: 0.0000
```

```
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The average IQ score is
significantly different from 100.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in average IQ score from 100.")
```

```
Reject the null hypothesis: The average IQ score is significantly
different from 100.
```

```
#ANOVA test
#To compare the growth rates of plants under three different
fertilizer treatments
#(Treatment A, B, and C) to determine if there is a significant
difference in their mean growth.
n_plants=25
growth_A=np.random.normal(loc=10, scale=2, size=n_plants)
growth_B=np.random.normal(loc=12, scale=3, size=n_plants)
growth_C=np.random.normal(loc=15, scale=2, size=n_plants)

all_data=np.concatenate([growth_A, growth_B, growth_C])
treatment_labels = ['A'] * n_plants + ['B'] * n_plants + ['C'] *
n_plants
f_stats, p_value = stats.f_oneway(growth_A, growth_B, growth_C)

print("Treatment A Mean Growth:", np.mean(growth_A))
print("Treatment B Mean Growth:", np.mean(growth_B))
print("Treatment C Mean Growth:", np.mean(growth_C))
print()
print(f"F-Statistic: {f_stats:.4f}")
print(f"P-Value: {p_value:.4f}")
```

```
Treatment A Mean Growth: 9.425120496291623
Treatment B Mean Growth: 12.318281885794768
Treatment C Mean Growth: 14.858935558008634

F-Statistic: 38.2963
P-Value: 0.0000
```

```
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant
difference in mean growth rates among the three treatments.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in mean growth rates among the three treatments.")
```

```
Reject the null hypothesis: There is a significant difference in mean
growth rates among the three treatments.
```

```python
# Additional: Post-hoc analysis (Tukey's HSD) if ANOVA is
significant
if p_value < alpha:
    from statsmodels.stats.multicomp import pairwise_tukeyhsd
    tukey_results = pairwise_tukeyhsd(all_data, treatment_labels,
alpha=0.05)
    print("\nTukey's HSD Post-hoc Test:")
    print(tukey_results)
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[15], line 2
      1 # Additional: Post-hoc analysis (Tukey's HSD) if ANOVA is
----> 2 significant
      3 if p_value < alpha:
      4     from statsmodels.stats.multicomp import pairwise_tukeyhsd

NameError: name 'significant' is not defined
```

```
import numpy as np
import pandas as pd
df=pd.read_csv('/content/pre-process_datasample.csv')
```

```
df
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	NaN	50.0	83000.0	No
9	France	37.0	67000.0	Yes

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.Country.fillna(df.Country.mode()[0],inplace=True)
features=df.iloc[:,:-1].values
```

```
<ipython-input-5-20665a0bbaa1>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame c
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate ob

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inpla


  df.Country.fillna(df.Country.mode()[0],inplace=True)
```

```
label=df.iloc[:,-1].values
```

Start coding or generate with AI.

```
from sklearn.impute import SimpleImputer
```

```
age=SimpleImputer(strategy="mean",missing_values=np.nan)
```

```
Salary=SimpleImputer(strategy="mean",missing_values=np.nan)
```

```
age.fit(features[:,[1]])
```

```
▾ SimpleImputer
SimpleImputer()
```

```
Salary.fit(features[:,[2]])
```

```
▾ SimpleImputer
SimpleImputer()
```

```
SimpleImputer()
```

```
▾ SimpleImputer
SimpleImputer()
```

```
features[:,[1]]=age.transform(features[:,[1]])
```

```
features[:,[2]]=Salary.transform(features[:,[2]])
```

```
features
```

```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
       ['France', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
oh = OneHotEncoder(sparse_output=False)
```

```
Country=oh.fit_transform(features[:,[0]])
```

```
Country
```

```
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
```

```
       [1., 0., 0.],
       [1., 0., 0.]])
```

```
final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)
```

```
final_set
```

```
array([[1.0, 0.0, 0.0, 44.0, 72000.0],
       [0.0, 0.0, 1.0, 27.0, 48000.0],
       [0.0, 1.0, 0.0, 30.0, 54000.0],
       [0.0, 0.0, 1.0, 38.0, 61000.0],
       [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
       [1.0, 0.0, 0.0, 35.0, 58000.0],
       [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
       [1.0, 0.0, 0.0, 48.0, 79000.0],
       [1.0, 0.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(final_set)
feat_standard_scaler=sc.transform(final_set)
```

```
feat_standard_scaler
```

```
array([[ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
         7.58874362e-01,  7.49473254e-01],
       [-1.00000000e+00, -5.00000000e-01,  1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-1.00000000e+00,  2.00000000e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-1.00000000e+00, -5.00000000e-01,  1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-1.00000000e+00,  2.00000000e+00, -6.54653671e-01,
         1.77608893e-01,  6.63219199e-16],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-1.00000000e+00, -5.00000000e-01,  1.52752523e+00,
         0.00000000e+00, -1.07356980e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
         1.34013983e+00,  1.38753832e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
         1.63077256e+00,  1.75214693e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        -2.58340208e-01,  2.93712492e-01]])
```

```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler(feature_range=(0,1))
mms.fit(final_set)
feat_minmax_scaler=mms.transform(final_set)
feat_minmax_scaler
```

```
array([[1.        , 0.        , 0.        , 0.73913043, 0.68571429],
       [0.        , 0.        , 1.        , 0.        , 0.        ],
       [0.        , 1.        , 0.        , 0.13043478, 0.17142857],
       [0.        , 0.        , 1.        , 0.47826087, 0.37142857],
       [0.        , 1.        , 0.        , 0.56521739, 0.45079365],
       [1.        , 0.        , 0.        , 0.34782609, 0.28571429],
       [0.        , 0.        , 1.        , 0.51207729, 0.11428571],
       [1.        , 0.        , 0.        , 0.91304348, 0.88571429],
       [1.        , 0.        , 0.        , 1.        , 1.        ],
       [1.        , 0.        , 0.        , 0.43478261, 0.54285714]])
```

Start coding or generate with AI.