

```
In [1]: # importing needed liabraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: # Loading the data
data=pd.read_csv(r"C:\Users\kc\Downloads\Car_Purchasing_Data.csv")
```

```
In [3]: # to check the number of rows and column in the data set
data.shape
```

```
Out[3]: (500, 9)
```

```
In [4]: # to view or print the loaded data
data
```

```
Out[4]:
```

	Customer Name	Customer e-mail	Country	Gender	Age
0	Martina Avila	cubilia.Curae.Phasellus@quisaccumsanconvallis.edu	USA	0	42
1	Harlan Barnes	eu.dolor@diam.co.uk	USA	0	41
2	Naomi Rodriquez	vulputate.mauris.sagittis@ametconsectetueradip...	USA	1	43
3	Jade Cunningham	malesuada@dignissim.com	USA	1	58
4	Cedric Leach	felis.ullamcorper.viverra@egetmollislectus.net	USA	1	57
...	...	...	...	...	...
495	Walter	ligula@Cumsociis.ca	USA	0	41
496	Vanna	Cum.sociis.natoque@Sedmolestie.edu	USA	1	38
497	Pearl	penatibus.et@massanonante.com	USA	1	54
498	Nell	Quisque.varius@arcuVivamussit.net	USA	1	59
499	Marla	Camaron.marla@hotmail.com	USA	1	47

500 rows × 9 columns

```
In [5]: # to check the data type of the data set
type(data)
```

```
Out[5]: pandas.core.frame.DataFrame
```

```
In [6]: # to check the the null values and find the data type of the columns in the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Customer Name         500 non-null   object
1   Customer e-mail       500 non-null   object
2   Country               500 non-null   object
3   Gender                500 non-null   int64
4   Age                   500 non-null   int64
5   Annual Salary         500 non-null   float64
6   Credit Card Debt      500 non-null   float64
7   Net Worth             500 non-null   float64
8   Car Purchase Amount   500 non-null   float64
dtypes: float64(4), int64(2), object(3)
memory usage: 35.3+ KB
```

In [7]: *# to check top five rows of data*  
data.head()

Out[7]:

	Customer Name	Customer e-mail	Country	Gender	Age
0	Martina Avila	cubilia.Curae.Phasellus@quisaccumsanconvallis.edu	USA	0	42
1	Harlan Barnes	eu.dolor@diam.co.uk	USA	0	41
2	Naomi Rodriquez	vulputate.mauris.sagittis@ametconsectetueradip...	USA	1	43
3	Jade Cunningham	malesuada@dignissim.com	USA	1	58
4	Cedric Leach	felis.ullamcorper.viverra@egetmollislectus.net	USA	1	57

In [8]: *# to check the bottom 5 rows of the dataset*  
data.tail()

Out[8]:

	Customer Name	Customer e-mail	Country	Gender	Age	Annual Salary
495	Walter	ligula@Cumsociis.ca	USA	0	41	71942.40291
496	Vanna	Cum.sociis.natoque@Sedmolestie.edu	USA	1	38	56039.49793
497	Pearl	penatibus.et@massanonante.com	USA	1	54	68888.77805
498	Nell	Quisque.varius@arcuVivamussit.net	USA	1	59	49811.99062
499	Marla	Camaron.marla@hotmail.com	USA	1	47	61370.67766

In [9]: *# it describes the count , mean , std dev, min, max, percentile of all the numer*  
data.describe()

Out[9]:

	Gender	Age	Annual Salary	Credit Card Debt	Net Worth	Car Purchase Amount
<b>count</b>	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000
<b>mean</b>	0.506000	46.224000	62127.239608	9607.645049	431475.713625	44209.7992
<b>std</b>	0.500465	7.990339	11703.378228	3489.187973	173536.756340	10773.1787
<b>min</b>	0.000000	20.000000	20000.000000	100.000000	20000.000000	9000.0000
<b>25%</b>	0.000000	41.000000	54391.977195	7397.515792	299824.195900	37629.8960
<b>50%</b>	1.000000	46.000000	62915.497035	9655.035568	426750.120650	43997.7833
<b>75%</b>	1.000000	52.000000	70117.862005	11798.867487	557324.478725	51254.7095
<b>max</b>	1.000000	70.000000	100000.000000	20000.000000	1000000.000000	80000.0000

In [10]: *# there is no null value in the data set*  
`data.isnull().sum()`

Out[10]: Customer Name            0  
 Customer e-mail            0  
 Country                    0  
 Gender                     0  
 Age                         0  
 Annual Salary              0  
 Credit Card Debt           0  
 Net Worth                  0  
 Car Purchase Amount       0  
 dtype: int64

In [11]: `data.duplicated().sum()`

Out[11]: 0

In [12]: *# to check the count of the customer to see how many repeated customers we have.*  
`data['Customer Name'].value_counts()`

Out[12]: Customer Name  
 Seth                        2  
 Walter                     2  
 Martina Avila              1  
 Sexton, Shaeleigh H.      1  
 Holloway, Brennan Q.     1  
 ..  
 Holmes Irwin               1  
 Hector Price               1  
 Sebastian Marks           1  
 Marvin Garner               1  
 Marla                      1  
 Name: count, Length: 498, dtype: int64

In [13]: *#to count the email*  
`data['Customer e-mail'].value_counts()`

```
Out[13]: Customer e-mail
cubilia.Curae.Phasellus@quisaccumsanconvallis.edu    1
mi.eleifend.egestas@cursuset.net                    1
ut@Etiamvestibulum.ca                                1
nunc.sed.pede@Quisqueporttitor.net                   1
Cras.eu@vitaavelitegestas.net                        1
..
Nunc.sed.orci@Namligulaelit.net                      1
Aliquam.nisl@semegetmassa.co.uk                     1
et.eros@feugiatmetussit.net                          1
in@sed.org                                             1
Cameron.marla@hotmail.com                            1
Name: count, Length: 500, dtype: int64
```

```
In [14]: data['Country'].value_counts()
```

```
Out[14]: Country
USA      500
Name: count, dtype: int64
```

```
In [15]: data.drop(columns='Country' , inplace = True)
```

```
In [16]: data
```

```
Out[16]:
```

	Customer Name	Customer e-mail	Gender	Age	An Si
0	Martina Avila	cubilia.Curae.Phasellus@quisaccumsanconvallis.edu	0	42	62812.0
1	Harlan Barnes	eu.dolor@diam.co.uk	0	41	66646.8
2	Naomi Rodriquez	vulputate.mauris.sagittis@ametconsectetueradip...	1	43	53798.5
3	Jade Cunningham	malesuada@dignissim.com	1	58	79370.0
4	Cedric Leach	felis.ullamcorper.viverra@egetmollislectus.net	1	57	59729.1
...	...	...	...	...	...
495	Walter	ligula@Cumsociis.ca	0	41	71942.4
496	Vanna	Cum.sociis.natoque@Sedmolestie.edu	1	38	56039.4
497	Pearl	penatibus.et@massanonante.com	1	54	68888.7
498	Nell	Quisque.varius@arcuVivamussit.net	1	59	49811.9
499	Marla	Cameron.marla@hotmail.com	1	47	61370.6

500 rows × 8 columns



```
In [17]: data['Age'].value_counts()
```

```
Out[17]: Age
43      34
51      29
42      26
47      25
48      25
40      24
44      23
45      23
53      20
46      19
50      19
41      19
57      17
55      16
52      15
49      14
37      14
39      14
56      11
35      10
54      10
38      10
33      10
32       9
36       9
58       7
34       6
59       6
63       6
62       5
61       5
60       5
30       3
29       2
70       2
20       1
31       1
27       1
22       1
65       1
64       1
28       1
25       1
Name: count, dtype: int64
```

```
In [18]: # converted all the column into numeric column
df_numeric = data.apply(pd.to_numeric, errors='coerce')
```

```
In [19]: # select only numeric columns out of it
df_numeric_only = df_numeric.select_dtypes(include=['int64', 'float64'])
```

```
In [20]: # calculate the correlation of that columns
correlation_matrix = df_numeric_only.corr()
correlation_matrix
```

Out[20]:

	Customer Name	Customer e-mail	Gender	Age	Annual Salary	Credit Card Debt	Net Worth
Customer Name	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Customer e-mail	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gender	NaN	NaN	1.000000	-0.066488	-0.036499	0.024193	-0.008395
Age	NaN	NaN	-0.066488	1.000000	0.000361	0.031748	0.021794
Annual Salary	NaN	NaN	-0.036499	0.000361	1.000000	0.049599	0.014767
Credit Card Debt	NaN	NaN	0.024193	0.031748	0.049599	1.000000	-0.049378
Net Worth	NaN	NaN	-0.008395	0.021794	0.014767	-0.049378	1.000000
Car Purchase Amount	NaN	NaN	-0.066408	0.633273	0.617862	0.028882	0.488580

```
In [21]: # calculate the average annual salary
average_annualsalary= data["Annual Salary"].mean()
print(average_annualsalary)
```

62127.23960756

```
In [22]: # calculate the average credit card debt
average_CreditCardDebt= data["Credit Card Debt"].mean()
print(average_CreditCardDebt)
```

9607.645048629198

```
In [23]: # calculate the average net worth
average_Net_Worth= data["Net Worth"].mean()
print(average_Net_Worth)
```

431475.71362506005

```
In [24]: # calculate the average car purchase amount
average_Car_Purchase_Amount= data["Car Purchase Amount"].mean()
print(average_Car_Purchase_Amount)
```

44209.79921842

```
In [25]: data.sort_values(by="Customer Name")
```

Out[25]:

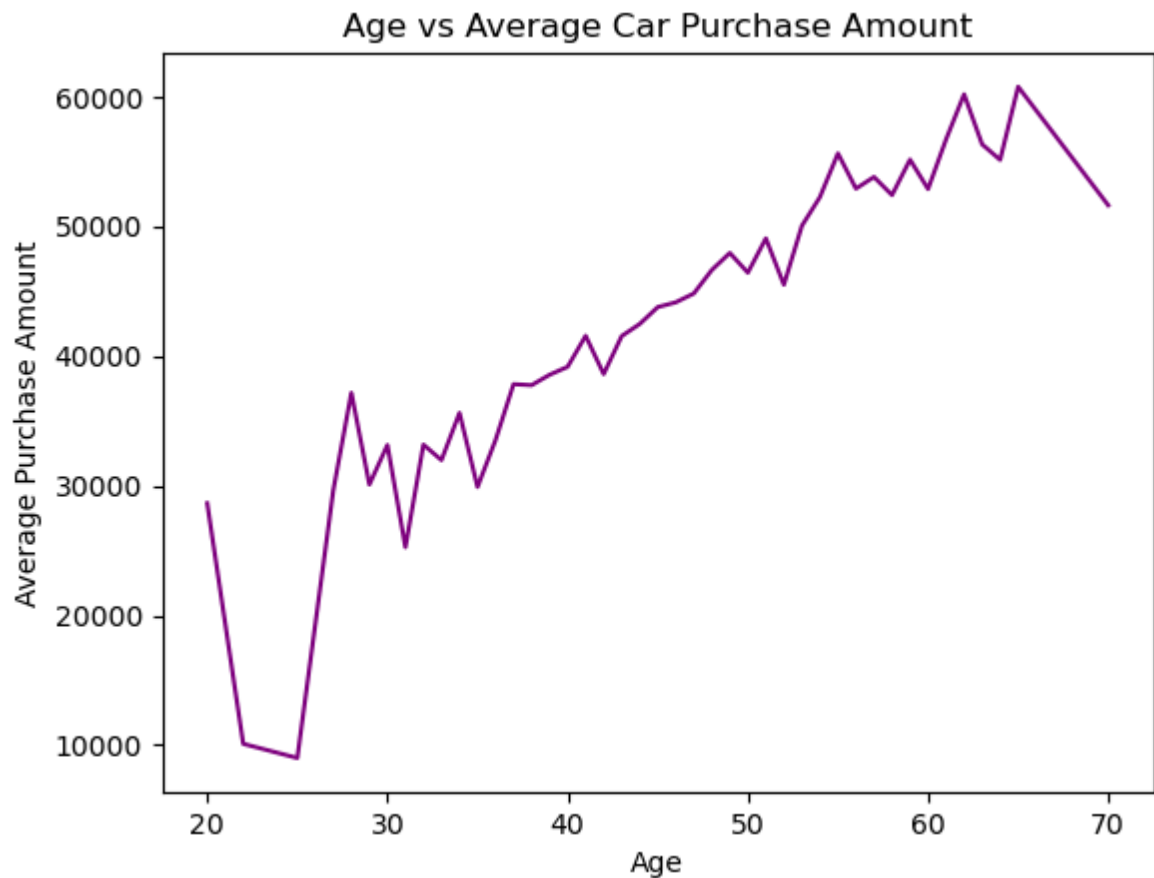
	Customer Name	Customer e-mail	Gender	Age	Annual Salary	Credit Car Del
82	Abel Stanton	eu.lacus.Quisque@congue.edu	1	40	48567.07462	9724.03164
290	Abigail X. Lindsey	dui@nondui.ca	0	63	46549.16329	640.04537
204	Abra D. Golden	odio@Duis.com	0	54	65834.56889	15353.25774
77	Adria Mathis	Aliquam.rutrum.lorem@Donec.net	1	57	53450.90036	8740.72309
124	Adrian Brock	Cras.lorem@nonvestibulumnec.net	0	54	60991.82443	7329.22857
...	...	...	...	...	...	...
239	Zane I. Boone	blandit@Cum.edu	1	45	62939.12851	632.05285
21	Zelenia Byers	auctor.non@sapien.co.uk	0	48	64347.34531	10905.36628
237	Zelenia L. Lowe	Nunc.mauris.elit@Curabiturvel.edu	0	51	60404.38394	4198.83912
295	Zenia H. Patel	dis@tortor.com	1	40	57455.76090	12186.02793
472	Zephania	montes@sedsem.ca	1	39	65019.15701	4931.56016

500 rows × 8 columns



```
In [26]: # Average car purchase according to age
age_purchase = data.groupby('Age')['Car Purchase Amount'].mean()

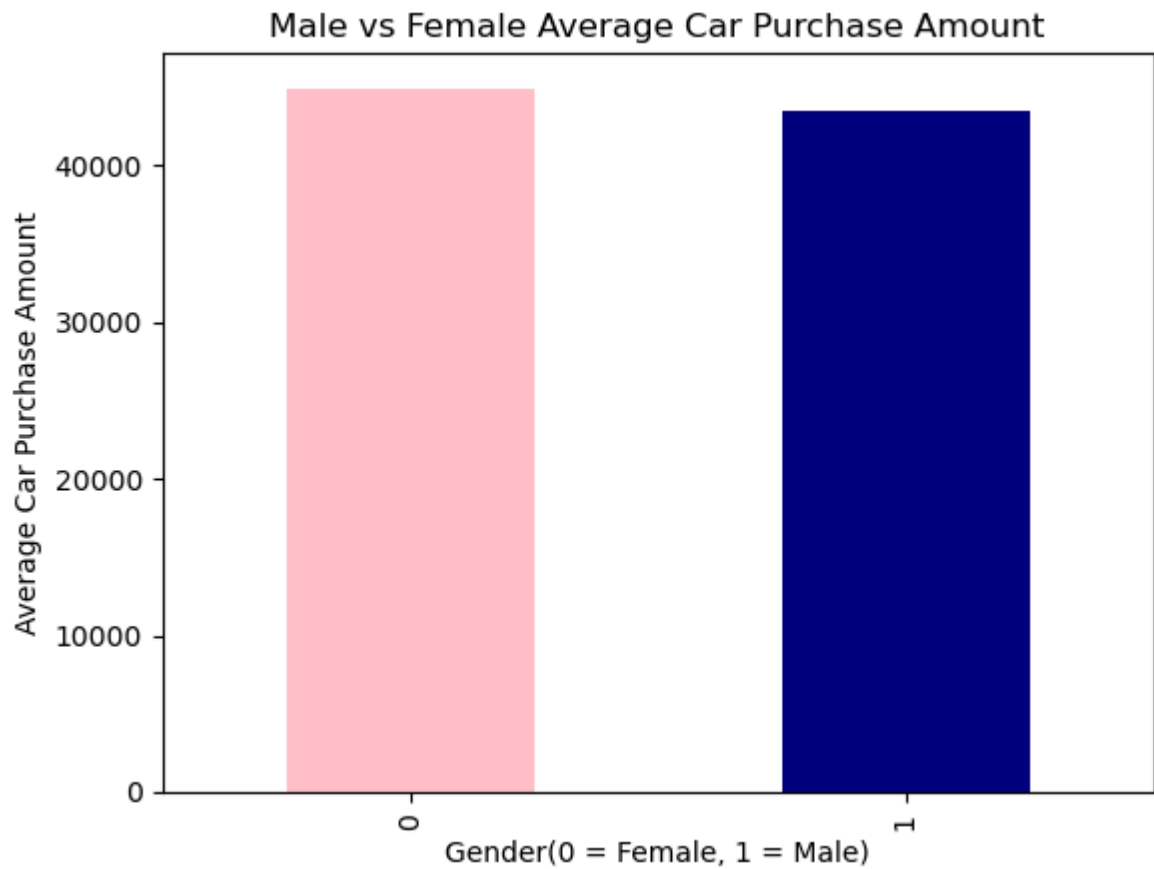
plt.plot(age_purchase.index, age_purchase.values , color='Purple')
plt.xlabel("Age")
plt.ylabel("Average Purchase Amount")
plt.title("Age vs Average Car Purchase Amount")
plt.show()
```



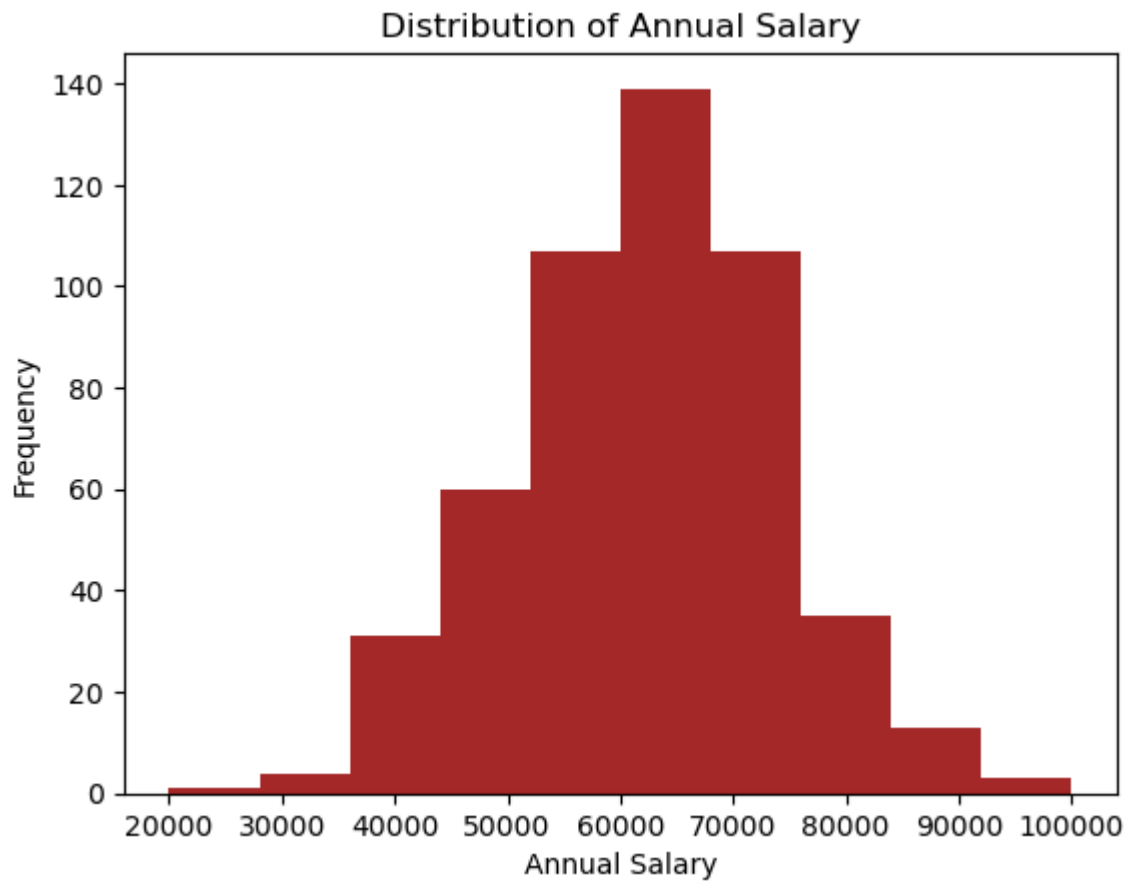
```
In [27]: # Male vs Female Average Car Purchase Amount bar chart
gender_wise_carpurchase = data.groupby('Gender')['Car Purchase Amount'].mean()

gender_wise_carpurchase.plot(kind='bar' , color = ['pink', 'navy'] )

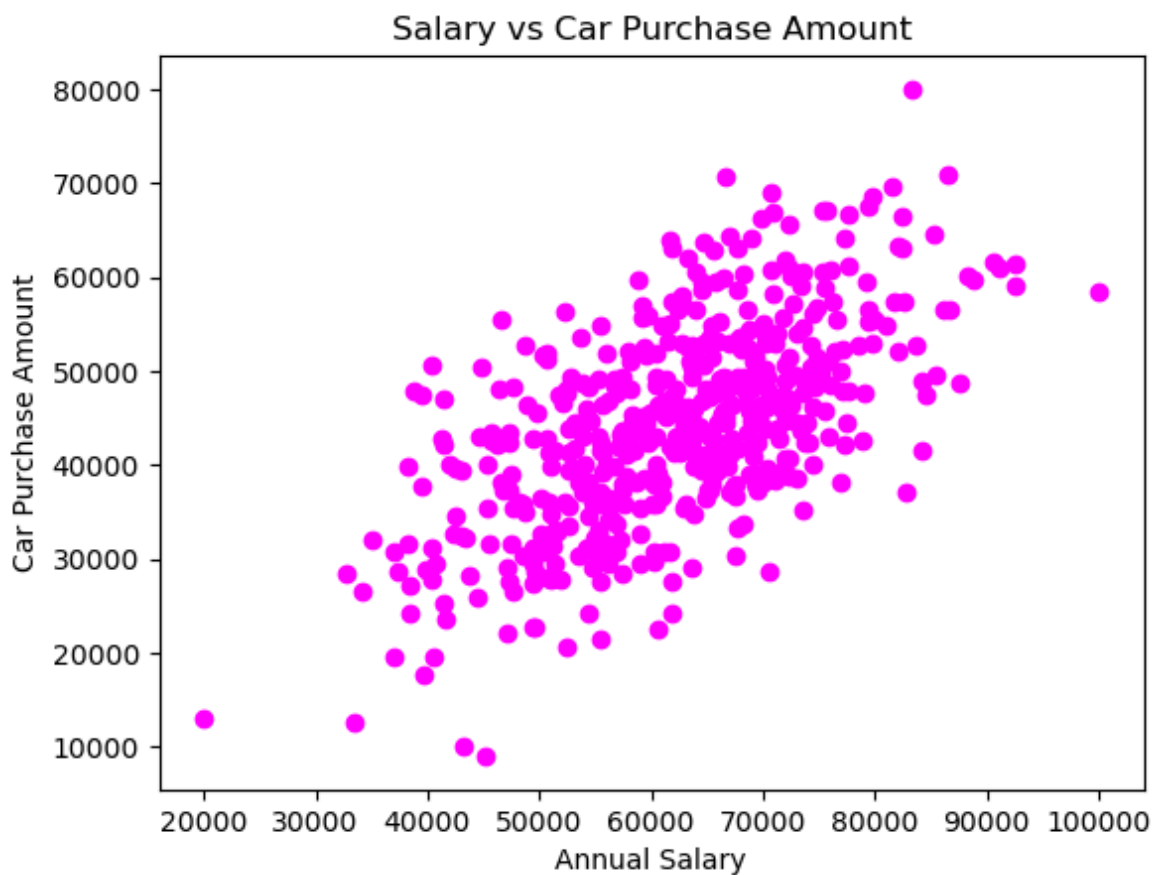
plt.xlabel("Gender(0 = Female, 1 = Male)")
plt.ylabel("Average Car Purchase Amount")
plt.title("Male vs Female Average Car Purchase Amount")
plt.show()
```



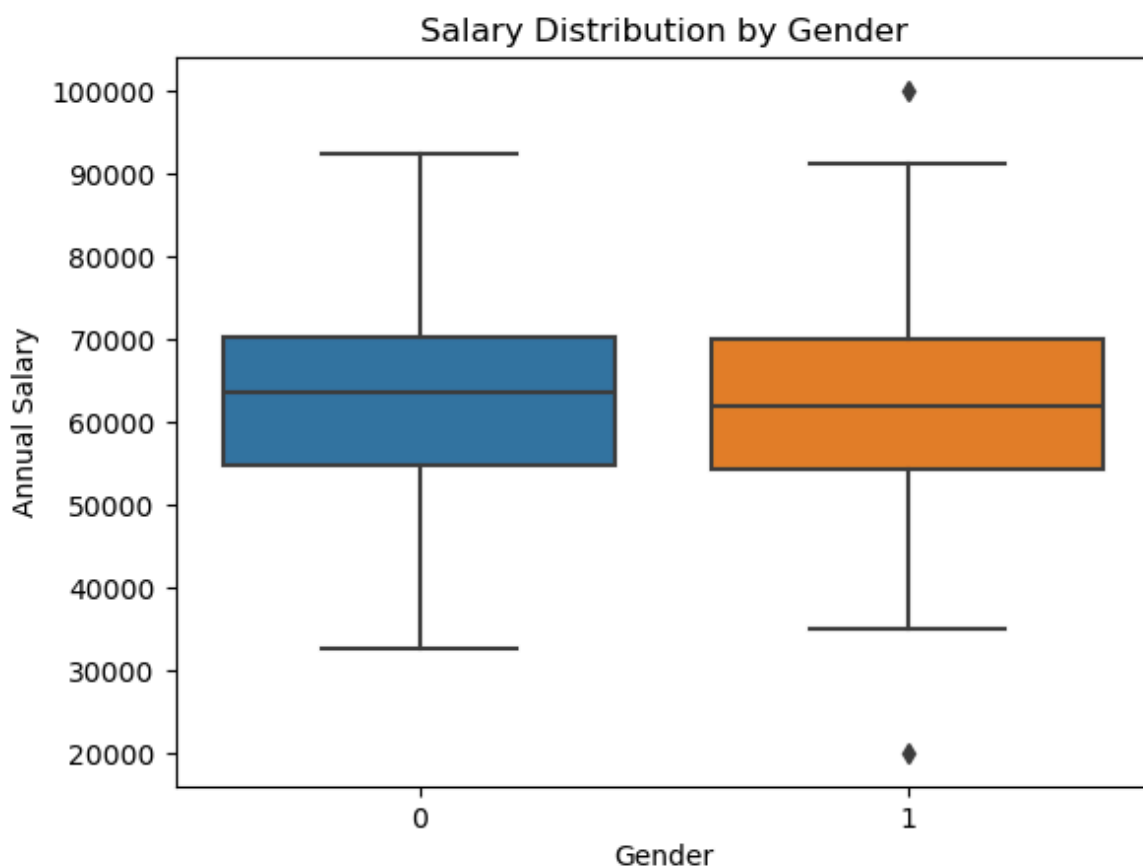
```
In [28]: # distribution frequency of annual salary
plt.hist(data['Annual Salary'], bins=10, color='brown')
plt.xlabel("Annual Salary")
plt.ylabel("Frequency")
plt.title("Distribution of Annual Salary")
plt.show()
```



```
In [29]: # scatter plot for how salary is affecting the car purchase amount
plt.scatter(data['Annual Salary'], data['Car Purchase Amount'], color = 'magenta')
plt.xlabel("Annual Salary")
plt.ylabel("Car Purchase Amount")
plt.title("Salary vs Car Purchase Amount")
plt.show()
```

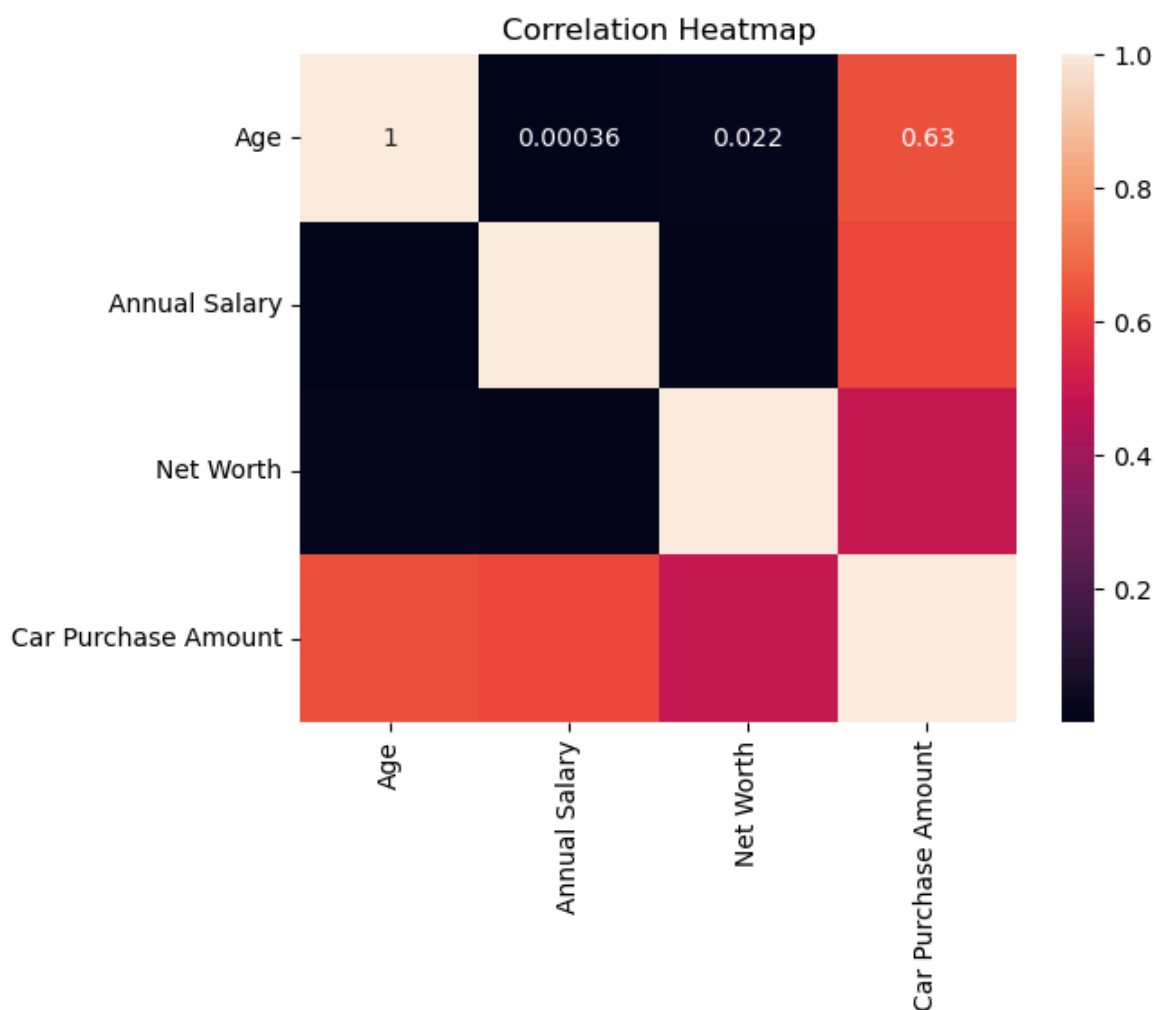


```
In [30]: # distribution of salary according to gender
sns.boxplot(x='Gender', y='Annual Salary', data=data)
plt.title("Salary Distribution by Gender")
plt.show()
```



```
In [31]: # Correlation between different columns in the data set
corr = data[['Age', 'Annual Salary', 'Net Worth', 'Car Purchase Amount']].corr()

sns.heatmap(corr, annot=True)
plt.title("Correlation Heatmap")
plt.show()
```



```
In [32]: # encoding customer name
from sklearn.preprocessing import LabelEncoder
# create a label encoder object
label_encoder = LabelEncoder()
# fit and transform customer column
data["Customer Name encoded"] = label_encoder.fit_transform(data["Customer Name"])
print(data[["Customer Name", "Customer Name encoded"]])
```

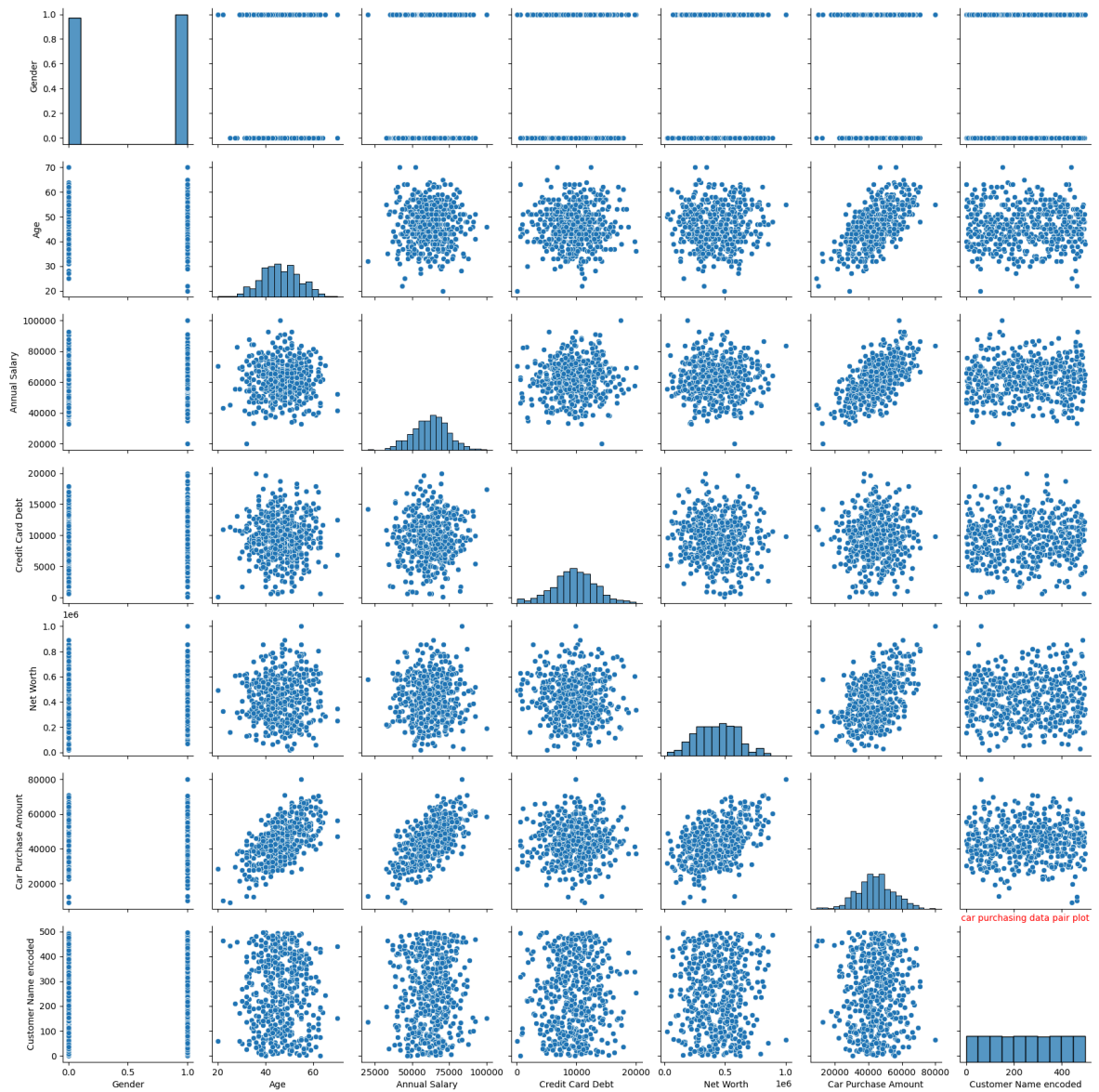
	Customer Name	Customer Name encoded
0	Martina Avila	288
1	Harlan Barnes	168
2	Naomi Rodriquez	311
3	Jade Cunningham	204
4	Cedric Leach	70
..	...	...
495	Walter	475
496	Vanna	466
497	Pearl	352
498	Nell	315
499	Marla	284

[500 rows x 2 columns]

```
In [33]: sns.pairplot(data)
plt.title("car purchasing data pair plot", fontsize = 10 , color = 'r')
```

```
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
Out[33]: Text(0.5, 1.0, 'car purchasing data pair plot')
```



```
In [34]: # encoding customer name
from sklearn.preprocessing import LabelEncoder
# create a label encoder object
label_encoder = LabelEncoder()
# fit and transform customer column
data["Customer e-mail encoded"] = label_encoder.fit_transform(data["Customer e-mail"])
print(data[["Customer e-mail", "Customer e-mail encoded"]])
```

```

Customer e-mail \
0 cubilia.Curae.Phasellus@quisaccumsanconvallis.edu
1 eu.dolor@diam.co.uk
2 vulputate.mauris.sagittis@ametconsectetueradip...
3 malesuada@dignissim.com
4 felis.ullamcorper.viverra@egetmollislectus.net
.. ...
495 ligula@Cumsociis.ca
496 Cum.sociis.natoque@Sedmolestie.edu
497 penatibus.et@massanonante.com
498 Quisque.varius@arcuVivamussit.net
499 Camaron.marla@hotmail.com

```

```

Customer e-mail encoded
0 139
1 224
2 498
3 307
4 241
.. ...
495 289
496 21
497 381
498 72
499 11

```

[500 rows x 2 columns]

In [35]: `data.head()`

Out[35]:

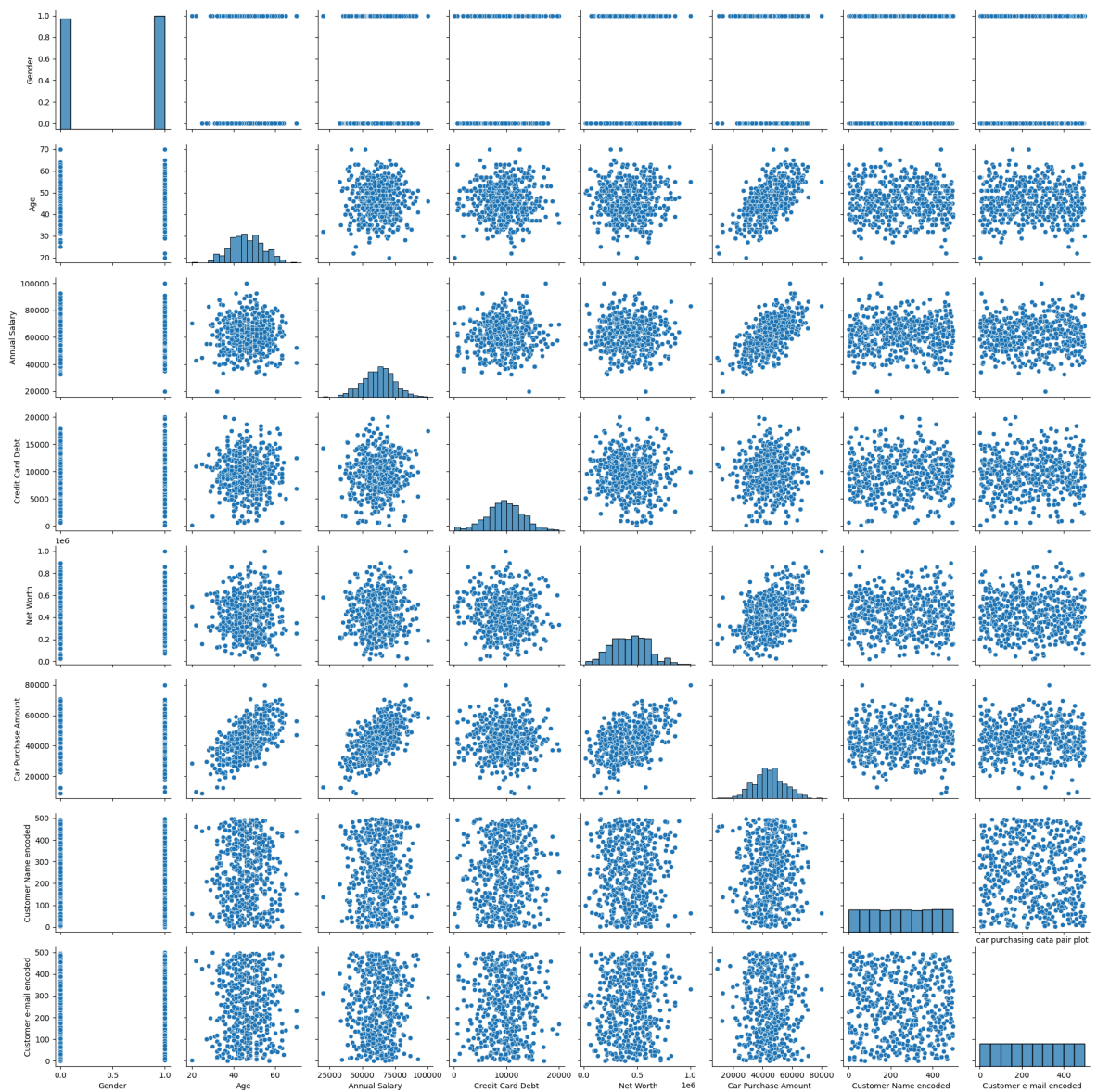
	Customer Name	Customer e-mail	Gender	Age	Annual Salary
0	Martina Avila	cubilia.Curae.Phasellus@quisaccumsanconvallis.edu	0	42	62812.0930
1	Harlan Barnes	eu.dolor@diam.co.uk	0	41	66646.8929
2	Naomi Rodriquez	vulputate.mauris.sagittis@ametconsectetueradip...	1	43	53798.5511
3	Jade Cunningham	malesuada@dignissim.com	1	58	79370.0379
4	Cedric Leach	felis.ullamcorper.viverra@egetmollislectus.net	1	57	59729.1511

In [36]: `data.drop(columns=['Customer Name' , 'Customer e-mail' ] , inplace = True)`

In [37]: `sns.pairplot(data)`  
`plt.title("car purchasing data pair plot", fontsize = 10)`

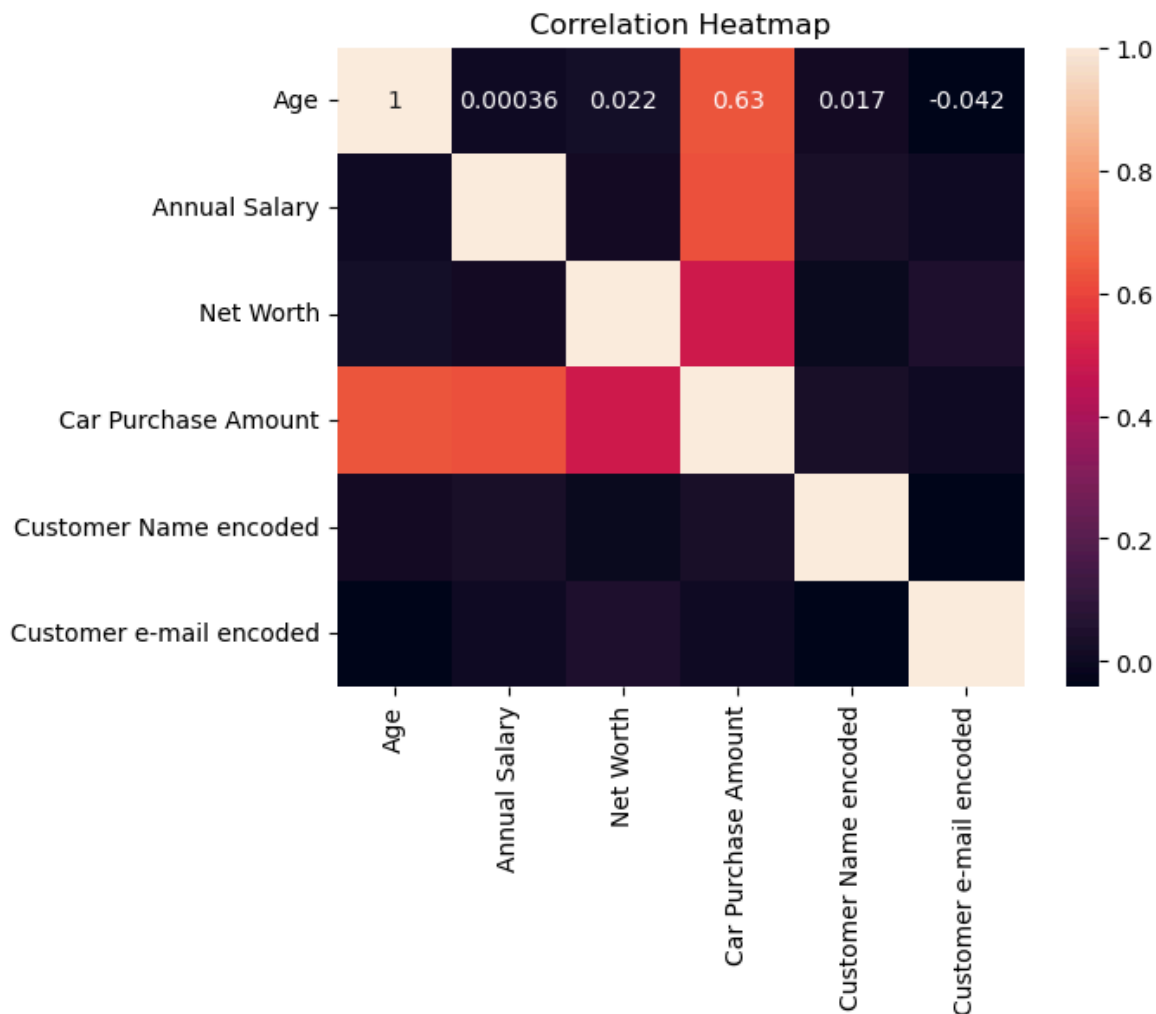
```
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\kc\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
```

```
Out[37]: Text(0.5, 1.0, 'car purchasing data pair plot')
```



```
In [38]: corr = data[['Age', 'Annual Salary', 'Net Worth', 'Car Purchase Amount', 'Customer Name encoded', 'Customer e-mail encoded']]

sns.heatmap(corr, annot = True)
plt.title("Correlation Heatmap")
plt.show()
```



```
In [40]: from sklearn.preprocessing import StandardScaler
```

```
In [41]: #Independent and dependent features
x=data[["Age","Annual Salary","Net Worth","Gender","Customer Name encoded","Customer e-mail encoded"]]
y=data["Car Purchase Amount"]
```

```
In [42]: #Train test split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=42)
```

```
In [43]: print(x_train)
```

	Age	Annual Salary	Net Worth	Gender	Customer Name encoded \
227	39	63675.93263	74257.82785	0	67
417	42	64412.43101	355175.36770	0	199
203	57	50241.48985	607395.01830	0	248
126	59	66905.47644	651215.64350	0	72
329	44	45504.74866	374777.69290	1	250
..	...	...	...	...	...
106	42	51075.46118	450402.29320	0	438
270	35	33422.99683	211168.62930	0	464
348	45	63561.04525	608019.63080	0	478
435	58	41409.29390	421318.97640	1	45
102	30	68289.18229	404457.30990	1	49

	Customer e-mail encoded
227	300
417	386
203	480
126	299
329	317
..	...
106	147
270	185
348	106
435	308
102	499

[375 rows x 6 columns]

In [44]: `print(x_test)`

	Age	Annual Salary	Net Worth	Gender	Customer Name encoded \
361	36	74420.10254	551344.33650	0	32
73	48	59139.21080	473845.85460	1	293
374	57	67752.38329	657178.41350	1	266
155	50	55381.53225	20000.00000	0	98
104	48	63975.06090	891439.87610	0	487
..	...	...	...	...	...
220	59	77662.11090	331460.47270	1	274
176	45	62043.16623	357639.03340	1	10
320	45	79781.90126	427287.62770	0	484
153	34	55576.84068	475126.12520	0	113
231	49	77435.46545	48620.32123	0	131

	Customer e-mail encoded
361	133
73	238
374	378
155	253
104	384
..	...
220	167
176	394
320	215
153	431
231	169

[125 rows x 6 columns]

In [45]: `print(y_train)`

```

227    29002.05665
417    39439.45349
203    51683.60859
126    64391.68906
329    31526.04931

```

...

```

106    34803.82395
270    12536.93842
348    49348.88394
435    42139.64528
102    33640.73697

```

Name: Car Purchase Amount, Length: 375, dtype: float64

In [46]: `print(y_test)`

```

361    46082.80993
73     45058.89690
374    63079.84329
155    31837.22537
104    60461.24268

```

...

```

220    61118.46947
176    41265.52929
320    52983.89411
153    31300.54347
231    44432.71747

```

Name: Car Purchase Amount, Length: 125, dtype: float64

In [47]: `scaler=StandardScaler()  
x_train=scaler.fit_transform(x_train)  
x_test=scaler.transform(x_test)`

In [49]: `#MODEL  
from sklearn.linear_model import LinearRegression  
regression=LinearRegression()`

In [50]: `regression.fit(x_train,y_train)`

Out[50]: `LinearRegression()  
LinearRegression()`

In [51]: `y_predict=regression.predict(x_test)`

In [52]: `intercept=regression.intercept_  
coeff=regression.coef_  
print("intercept: ",intercept)  
print("coefficients:",coeff)`

intercept: 43869.88850223999

coefficients: [6708.46389993 6680.74515679 5079.84924006 19.90834877 9.56190054  
11.21199336]

In [53]: `y_predict`

```
Out[53]: array([45969.46639797, 45260.68016017, 62966.68428016, 31643.34485874,
60045.8040296 , 63016.05915267, 52181.47148584, 54513.84946652,
52398.05814327, 48300.79808563, 37965.32134801, 56145.1290642 ,
44593.74532085, 39324.39769324, 40206.20442889, 55252.90630441,
48530.81560208, 17256.19213604, 60918.14167233, 50039.58960197,
41584.85473091, 52646.65741169, 51782.28967467, 38442.26754124,
41067.12299165, 38680.60927542, 64201.46972015, 47974.16322561,
23059.62306678, 52473.50433823, 54949.81825693, 46110.80659427,
41463.48726346, 57332.05950948, 42846.17627769, 39657.4098392 ,
61922.88571494, 30719.6317484 , 41856.40457464, 39949.56600076,
57304.96606001, 61059.91175685, 47061.51401904, 36395.87079077,
53040.53143544, 44806.11218622, 35452.23135952, 42153.85789762,
52102.59051469, 47649.19731534, 41975.09574332, 32750.58967993,
38260.9244851 , 41539.59043671, 44801.80359564, 47956.32500177,
60593.22749841, 44769.67915061, 44699.21548188, 38398.57389194,
64351.97995963, 43470.83549935, 22817.80279723, 55042.07218041,
41014.53236152, 54791.17692193, 60385.72886039, 34059.52536331,
43178.05030904, 48312.91437048, 59998.50655711, 28692.67793469,
59338.98230232, 55100.04459692, 60016.71736623, 21818.63388345,
41199.44053603, 49356.1785621 , 32642.07892662, 61019.90146201,
42356.02751026, 30660.97402359, 36072.77231634, 43636.38714344,
50589.33588809, 39033.07188874, 38559.12516724, 30744.93017696,
39831.27129183, 35784.75642974, 29494.3958701 , 37207.36425325,
31373.96012871, 40849.75447134, 32323.49574674, 49107.97774095,
44773.27454028, 50964.76527498, 43685.38482958, 52780.11612819,
46067.99077117, 29735.7196947 , 54993.25001094, 59415.77107125,
25050.73090643, 47613.79864216, 42255.3471654 , 41960.61246235,
45019.3351696 , 39438.83612413, 39439.25940447, 49882.04602238,
39611.8540284 , 44718.6681137 , 52206.8917398 , 53312.3407611 ,
65886.08514165, 49609.1534576 , 41754.40120293, 28391.64759259,
60780.77085182, 41009.60306325, 52981.78351876, 31515.00447287,
44035.08385415])
```

```
In [54]: from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute
mae=mean_absolute_error(y_test, y_predict)
mse=mean_squared_error(y_test, y_predict)
mape=mean_absolute_percentage_error(y_test, y_predict)
rmse=np.sqrt(mse)
print("mae=", mae)
print("mse=", mse)
print("mape=", mape)
print("rmse=", rmse)
```

```
mae= 215.10880700433532
mse= 62065.04454697324
mape= 0.00506632391395219
rmse= 249.12857031455312
```

```
In [55]: from sklearn.metrics import r2_score
score=r2_score(y_test, y_predict)
print(score)
#display adjusted R-squared
print(1 - (1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1))
```

```
0.9994158693590651
0.9993861678010515
```

```
In [56]: ## cross validation
from sklearn.model_selection import cross_val_score
```

```
validation_score=cross_val_score(regression,x_train,y_train,scoring='neg_mean_sq
                                cv=3)
```

```
In [57]: np.mean(validation_score)
```

```
Out[57]: -59618.8494544694
```

```
In [58]: ## prediction
y_pred=regression.predict(x_test)
```

```
In [59]: y_pred
```

```
Out[59]: array([45969.46639797, 45260.68016017, 62966.68428016, 31643.34485874,
                60045.8040296 , 63016.05915267, 52181.47148584, 54513.84946652,
                52398.05814327, 48300.79808563, 37965.32134801, 56145.1290642 ,
                44593.74532085, 39324.39769324, 40206.20442889, 55252.90630441,
                48530.81560208, 17256.19213604, 60918.14167233, 50039.58960197,
                41584.85473091, 52646.65741169, 51782.28967467, 38442.26754124,
                41067.12299165, 38680.60927542, 64201.46972015, 47974.16322561,
                23059.62306678, 52473.50433823, 54949.81825693, 46110.80659427,
                41463.48726346, 57332.05950948, 42846.17627769, 39657.4098392 ,
                61922.88571494, 30719.6317484 , 41856.40457464, 39949.56600076,
                57304.96606001, 61059.91175685, 47061.51401904, 36395.87079077,
                53040.53143544, 44806.11218622, 35452.23135952, 42153.85789762,
                52102.59051469, 47649.19731534, 41975.09574332, 32750.58967993,
                38260.9244851 , 41539.59043671, 44801.80359564, 47956.32500177,
                60593.22749841, 44769.67915061, 44699.21548188, 38398.57389194,
                64351.97995963, 43470.83549935, 22817.80279723, 55042.07218041,
                41014.53236152, 54791.17692193, 60385.72886039, 34059.52536331,
                43178.05030904, 48312.91437048, 59998.50655711, 28692.67793469,
                59338.98230232, 55100.04459692, 60016.71736623, 21818.63388345,
                41199.44053603, 49356.1785621 , 32642.07892662, 61019.90146201,
                42356.02751026, 30660.97402359, 36072.77231634, 43636.38714344,
                50589.33588809, 39033.07188874, 38559.12516724, 30744.93017696,
                39831.27129183, 35784.75642974, 29494.3958701 , 37207.36425325,
                31373.96012871, 40849.75447134, 32323.49574674, 49107.97774095,
                44773.27454028, 50964.76527498, 43685.38482958, 52780.11612819,
                46067.99077117, 29735.7196947 , 54993.25001094, 59415.77107125,
                25050.73090643, 47613.79864216, 42255.3471654 , 41960.61246235,
                45019.3351696 , 39438.83612413, 39439.25940447, 49882.04602238,
                39611.8540284 , 44718.6681137 , 52206.8917398 , 53312.3407611 ,
                65886.08514165, 49609.1534576 , 41754.40120293, 28391.64759259,
                60780.77085182, 41009.60306325, 52981.78351876, 31515.00447287,
                44035.08385415])
```

```
In [60]: ## Performance Metrics
from sklearn.metrics import mean_absolute_error,mean_squared_error
mse=mean_squared_error(y_test,y_pred)
mae=mean_absolute_error(y_test,y_pred)
rmse=np.sqrt(mse)
print(mse)
print(mae)
print(rmse)
```

```
62065.04454697324
215.10880700433532
249.12857031455312
```

```
In [61]: from sklearn.metrics import r2_score
score=r2_score(y_test,y_pred)
```

```
print(score)
#display adjusted R-squared
print(1 - (1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1))
```

0.9994158693590651

0.9993861678010515

In [ ]:

In [ ]:

In [ ]: