PROBLEM STATEMENT

Need to develop a model to predict the total dollar amount that customers are willing to pay given the following attributes:

- Customer Name
- Customer e-mail
- Country
- Gender
- Age
- Annual Salary
- Credit Card Debt
- Net Worth

The model should predict:

- Car Purchase Amount

STEP #0: LIBRARIES IMPORT

In [21]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

STEP #1: IMPORT DATASET

In [22]:
```
car_df = pd.read_csv('Car_Purchasing_Data.csv', encoding='ISO-8859-1')
```

In [23]:
```
car_df
```

Out[23]:

	Customer Name	Customer e-mail	Country	Gender	Age	
0	Martina Avila	cubilia.Curae.Phasellus@quisaccumsanconvallis.edu	Bulgaria	0	41.851720	628
1	Harlan Barnes	eu.dolor@diam.co.uk	Belize	0	40.870623	666
2	Naomi Rodriquez	vulputate.mauris.sagittis@ametconsectetueradip...	Algeria	1	43.152897	537
3	Jade Cunningham	malesuada@dignissim.com	Cook Islands	1	58.271369	793
4	Cedric Leach	felis.ullamcorper.viverra@egetmollislectus.net	Brazil	1	57.313749	597
...	...	...	...	...	...	
495	Walter	ligula@Cumsociis.ca	Nepal	0	41.462515	719

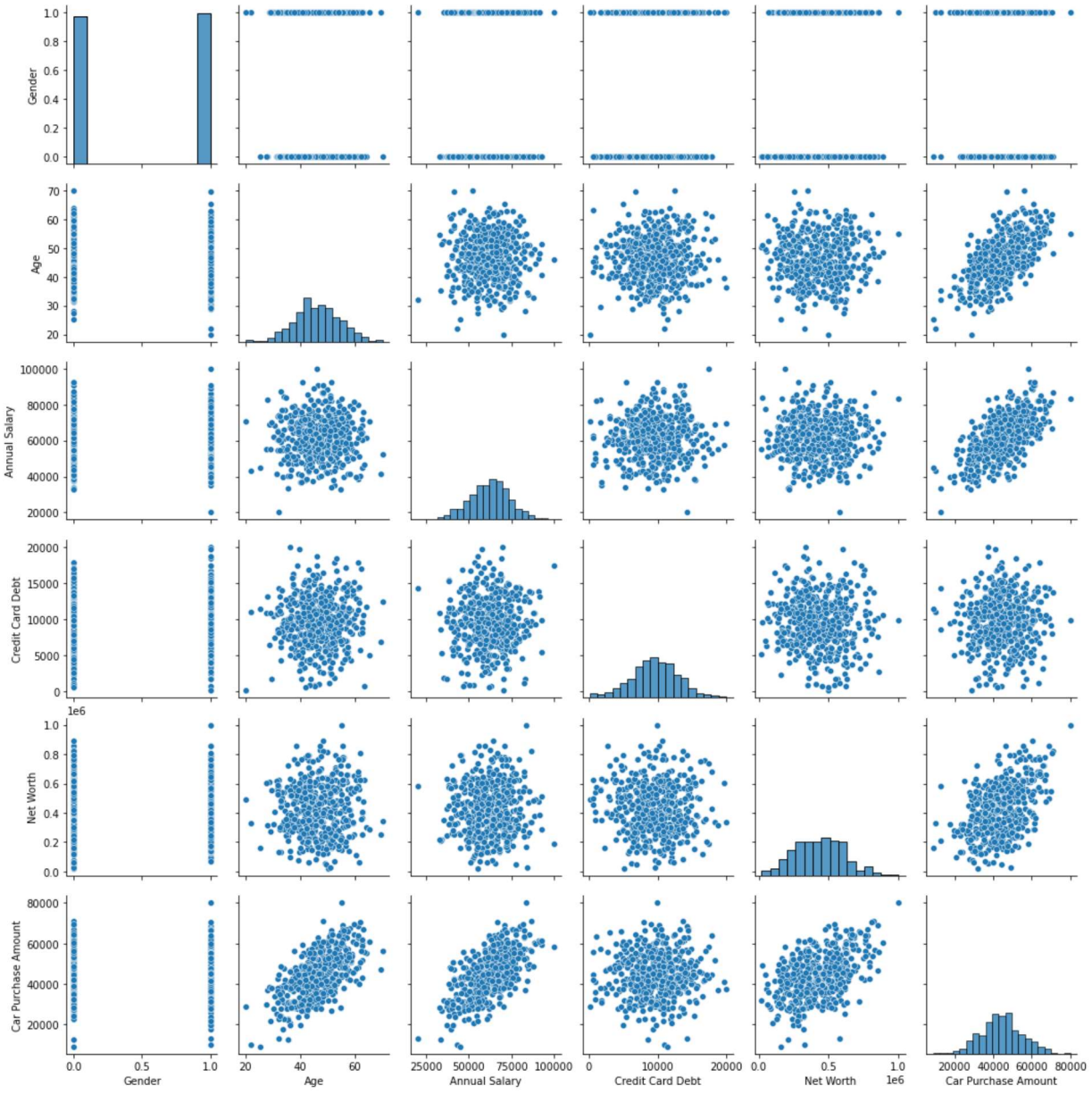
	Customer Name	Customer e-mail	Country	Gender	Age	
496	Vanna	Cum.sociis.natoque@Sedmolestie.edu	Zimbabwe	1	37.642000	56(
497	Pearl	penatibus.et@massanonante.com	Philippines	1	53.943497	68
498	Nell	Quisque.varius@arcuVivamussit.net	Botswana	1	59.160509	49
499	Marla	Camaron.marla@hotmail.com	marlal	1	46.731152	61

500 rows × 9 columns

STEP #2: VISUALIZE DATASET

In [24]:
```
sns.pairplot(car_df)
```

Out[24]: <seaborn.axisgrid.PairGrid at 0x270a67a0490>



STEP #3: CREATE TESTING AND TRAINING DATASET/DATA CLEANING

In [25]:
```
X = car_df.drop(['Customer Name', 'Customer e-mail', 'Country', 'Car Purchase Amount
```

In [ ]:
```

```

In [26]:
```
X
```

Out[26]:

	Gender	Age	Annual Salary	Credit Card Debt	Net Worth
0	0	41.851720	62812.09301	11609.380910	238961.2505
1	0	40.870623	66646.89292	9572.957136	530973.9078
2	1	43.152897	53798.55112	11160.355060	638467.1773
3	1	58.271369	79370.03798	14426.164850	548599.0524
4	1	57.313749	59729.15130	5358.712177	560304.0671
...	...	...	...	...	...
495	0	41.462515	71942.40291	6995.902524	541670.1016
496	1	37.642000	56039.49793	12301.456790	360419.0988
497	1	53.943497	68888.77805	10611.606860	764531.3203
498	1	59.160509	49811.99062	14013.034510	337826.6382
499	1	46.731152	61370.67766	9391.341628	462946.4924

500 rows × 5 columns

In [27]:
```
y = car_df['Car Purchase Amount']
y.shape
```

Out[27]:
```
(500,)
```

In [28]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

In [29]:
```
scaler.data_max_
```

Out[29]:
```
array([1.e+00, 7.e+01, 1.e+05, 2.e+04, 1.e+06])
```

In [30]:
```
scaler.data_min_
```

Out[30]:
```
array([    0.,    20., 20000.,   100., 20000.])
```

In [31]:
```
print(X_scaled[:,0])
```

```
[0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1.
 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0.
 1. 0. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 0.
 0. 1. 1. 0. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1.
 0. 1. 0. 0. 1. 1. 1. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 0.
 1. 0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1. 0. 1.
 0. 0. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1.
 0. 1. 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 1. 0. 1. 1. 0. 1. 0. 0. 1. 1. 1. 1.
 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0.
 1. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 0. 1.
 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 0. 0. 1. 1. 0. 1. 1. 1. 1. 0. 0. 0. 1. 1.
 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0.
 0. 0. 0. 1. 1. 1. 1. 1. 1. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 1. 0. 0. 1. 0.
 0. 1. 0. 1. 0. 1. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0.
 0. 1. 0. 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 1. 0. 0. 1. 0. 0. 1. 1. 1. 0. 1.
 1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 1. 1.
 0. 1. 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1.
 0. 1. 0. 1. 1. 0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0.
 1. 1. 1. 1. 0. 0. 0. 0. 1. 0. 1. 0. 1. 0. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1.
 0. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 1. 0.
 0. 1. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 1. 1. 1.]
```

In [32]:
```
y.shape
```

Out[32]:
```
(500,)
```

In [33]:
```
y = y.values.reshape(-1,1)
```

In [34]:
```
y.shape
```

Out[34]:
```
(500, 1)
```

In [35]:
```
y_scaled = scaler.fit_transform(y)
```

In [36]:
```
y_scaled
```

Out[36]:
```
array([[0.37072477],
       [0.50866938],
       [0.47782689],
       [0.82285018],
       [0.66078116],
       [0.67059152],
       [0.28064374],
       [0.54133778],
       [0.54948752],
       [0.4111198 ],
       [0.70486638],
       [0.46885649],
       [0.27746526],
       [0.56702642],
       [0.57056385],
       [0.61996151],
       [0.46217916],
       [0.49157341],
       [0.50188722],
       [0.64545808],
       [0.59339372],
       [0.48453965],
       [0.53860366],
       [0.53007738],
```

```
[0.50814651],
[0.49841668],
[0.3966416 ],
[0.56467566],
[0.6950749 ],
[0.49287831],
[0.12090943],
[0.50211776],
[0.80794216],
[0.62661214],
[0.43394857],
[0.60017103],
[0.42223485],
[0.01538345],
[0.37927499],
[0.64539707],
[0.51838974],
[0.45869677],
[0.26804521],
[0.2650104 ],
[0.84054134],
[0.84401542],
[0.35515157],
[0.406246  ],
[0.40680623],
[0.55963883],
[0.2561583 ],
[0.77096325],
[0.55305289],
[0.5264948 ],
[0.3236476 ],
[0.55070832],
[0.54057623],
[0.45669016],
[0.41053254],
[0.33433524],
[0.39926954],
[0.5420261 ],
[0.57366948],
[0.43793831],
[0.46897896],
[0.61908354],
[0.55076214],
[0.48846357],
[0.61560519],
[0.56891394],
[0.30761974],
[0.56863909],
[0.46343171],
[0.50787179],
[0.61959897],
[0.59095889],
[0.45573492],
[0.49908055],
[0.4155271 ],
[0.45382041],
[0.41407692],
[0.45713635],
[0.30133556],
[0.47019791],
[0.42256447],
[0.14863766],
[0.50939895],
[0.38056275],
[0.59067519],
[0.05486922],
[0.4219045 ],
[0.59466257],
[0.23904164],
```

```
[0.72775122],
[0.63486085],
[0.43668833],
[0.74075381],
[0.42962519],
[0.61231998],
[0.46743215],
[0.68227386],
[0.19270023],
[0.34705263],
[0.31747576],
[0.72480623],
[0.51744133],
[0.36343414],
[0.36116341],
[0.26178477],
[0.64750739],
[0.56538749],
[0.70197943],
[0.68037083],
[0.60481233],
[0.29916352],
[0.81860421],
[0.47053558],
[0.45706646],
[0.47313925],
[0.35945319],
[0.46779854],
[0.46357095],
[0.71008706],
[0.59721993],
[0.64444254],
[0.53723155],
[0.78016463],
[0.39792327],
[0.61500514],
[0.49297475],
[0.59931944],
[0.4118826 ],
[0.43333307],
[0.43771229],
[0.33988067],
[0.55806565],
[0.54497514],
[0.42831636],
[0.344062  ],
[0.33380674],
[0.75865395],
[0.28768775],
[0.49885366],
[0.3893741 ],
[0.62895125],
[0.52080458],
[0.41555485],
[0.54839351],
[0.72143981],
[0.42155708],
[0.26493265],
[0.54372318],
[0.46981253],
[0.31409216],
[0.46999496],
[0.32165106],
[0.24646921],
[0.41088501],
[0.42863953],
[0.40442699],
[0.67782275],
[0.52751195],
```

```
[0.49091635],
[0.65623527],
[0.47160596],
[0.44900269],
[0.16412978],
[0.37237754],
[0.38187033],
[0.41101838],
[0.45107076],
[0.26605566],
[0.48907732],
[0.68212351],
[0.45217002],
[0.56409653],
[0.45444407],
[0.78232624],
[0.28242388],
[0.3059129 ],
[0.41919878],
[0.42720002],
[0.33251345],
[0.69078257],
[0.63925743],
[0.38927052],
[0.42988917],
[0.47856826],
[0.73050372],
[0.52648517],
[0.67013948],
[0.47569515],
[0.40675569],
[0.50997782],
[0.665203  ],
[0.58387492],
[0.57353959],
[0.31967601],
[0.56647918],
[0.52378641],
[0.38150608],
[0.48259225],
[0.44592089],
[0.60117759],
[0.50035241],
[0.55660425],
[0.51838459],
[0.6457801 ],
[0.33068236],
[0.46773647],
[0.64966102],
[0.54907635],
[0.48459001],
[0.50109082],
[0.40485272],
[0.5465529 ],
[0.5048829 ],
[0.53018684],
[0.66991531],
[0.46018938],
[0.73406295],
[0.39864179],
[0.53369389],
[0.66841888],
[0.51422109],
[0.26233016],
[0.52661271],
[0.28171911],
[0.5965593 ],
[0.46494647],
[0.61485077],
```

```
[0.49905236],
[0.52189581],
[0.69345654],
[0.47873651],
[0.58735466],
[0.53534616],
[0.56901367],
[0.47883335],
[0.49908428],
[0.5257114 ],
[0.53305254],
[0.66900144],
[0.47568675],
[0.61005611],
[0.3540794 ],
[0.72905982],
[0.80505204],
[0.31289637],
[0.523032  ],
[0.67567861],
[0.2138602 ],
[0.56450168],
[0.39568902],
[0.4845291 ],
[0.28298776],
[0.55421359],
[0.34170423],
[0.45531219],
[0.56811431],
[0.5972613 ],
[0.31338011],
[0.48730944],
[0.55352142],
[0.63399262],
[0.41795296],
[0.39544824],
[0.40771621],
[0.45521229],
[0.81533714],
[0.04981603],
[0.43026944],
[0.61562087],
[0.6268025 ],
[0.60728039],
[0.41838955],
[0.54964825],
[0.71104101],
[0.37903726],
[0.45118709],
[0.60183344],
[0.62002621],
[0.33560612],
[0.28757249],
[0.6825565 ],
[0.58368485],
[0.45880771],
[0.52693631],
[0.54380447],
[0.8715253 ],
[0.65554063],
[0.63167261],
[0.4352791 ],
[0.50332973],
[0.5343264 ],
[0.27381426],
[0.41053523],
[0.47531745],
[0.2911385 ],
[0.76110147],
```

```
[0.76406707],
[0.46931782],
[0.49948317],
[0.81820046],
[0.18438195],
[0.43693203],
[0.66298048],
[0.56960734],
[0.47179899],
[0.39556023],
[0.60375334],
[0.37628608],
[0.43511174],
[0.37719946],
[0.47698891],
[1.        ],
[0.72573208],
[0.71491172],
[0.43107389],
[0.69917902],
[0.61949147],
[0.58685749],
[0.71302854],
[0.19197549],
[0.4526464 ],
[0.62671101],
[0.38794277],
[0.48597146],
[0.3119255 ],
[0.3172683 ],
[0.31103807],
[0.51220225],
[0.22891454],
[0.43505067],
[0.60902435],
[0.43537481],
[0.51912329],
[0.30740999],
[0.42849005],
[0.36167369],
[0.20447774],
[0.27793926],
[0.70558126],
[0.58012487],
[0.3754806 ],
[0.52673735],
[0.32804493],
[0.56449711],
[0.56829414],
[0.45672673],
[0.21439436],
[0.49893066],
[0.72380375],
[0.47597174],
[0.53430602],
[0.69953618],
[0.40905353],
[0.42634618],
[0.64234067],
[0.4237175 ],
[0.54907212],
[0.5222931 ],
[0.30935321],
[0.37643596],
[0.56429808],
[0.56275857],
[0.39694511],
[0.53113416],
[0.61816285],
```

```
[0.29231919],
[0.73184274],
[0.4362737 ],
[0.41613807],
[0.67273869],
[0.76168793],
[0.65775822],
[0.3854533 ],
[0.61236387],
[0.58164331],
[0.39802597],
[0.54529522],
[0.28930803],
[0.72629843],
[0.38204913],
[0.68033622],
[0.60453629],
[0.37815253],
[0.47470876],
[0.65035196],
[0.24788603],
[0.63371047],
[0.54888405],
[0.48791067],
[0.46027877],
[0.76253593],
[0.30644588],
[0.79707352],
[0.40664378],
[0.47773971],
[0.19154167],
[0.86759109],
[0.48228989],
[0.41040247],
[0.30168732],
[0.77280198],
[0.50863303],
[0.49805458],
[0.14824364],
[0.57734658],
[0.7427002 ],
[0.4615406 ],
[0.52679628],
[0.39967091],
[0.34882593],
[0.3051776 ],
[0.60642838],
[0.30614901],
[0.4287247 ],
[0.41091372],
[0.44492764],
[0.74688327],
[0.5558489 ],
[0.43795845],
[0.571387  ],
[0.31561446],
[0.54534475],
[0.37724541],
[0.47754279],
[0.55657526],
[0.51540401],
[0.32481193],
[0.32687852],
[0.37288737],
[0.28900791],
[0.6538108 ],
[0.46675557],
[0.58506904],
[0.36510463],
```

```
[0.49152498],
[0.42443705],
[0.45278122],
[0.21316327],
[0.4747199 ],
[0.42114943],
[0.27669569],
[0.60775236],
[0.81194719],
[0.47759537],
[0.56687473],
[0.25779114],
[0.54746234],
[0.71808681],
[0.51086565],
[0.        ],
[0.52129727],
[0.33756622],
[0.56035434],
[0.51865585],
[0.43805795],
[0.3726359 ],
[0.2895323 ],
[0.41187412],
[0.499023  ],
[0.59220313],
[0.61366712],
[0.73808769],
[0.27413582],
[0.26177748],
[0.54900684],
[0.26992761],
[0.85449963],
[0.55003734],
[0.39949626],
[0.50001154],
[0.36815842],
[0.6502447 ],
[0.55469987],
[0.37779655],
[0.38757338],
[0.62128001],
[0.62041473],
[0.17564949],
[0.50726309],
[0.65320953],
[0.6691568 ],
[0.54146119],
[0.45760058],
[0.33173992],
[0.46457217],
[0.7118085 ],
[0.4556686 ],
[0.61669253],
[0.71996731],
[0.54592485],
[0.77729956],
[0.56199216],
[0.31678049],
[0.77672238],
[0.51326977],
[0.50855247]])
```

STEP#4: TRAINING THE MODEL

In [37]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size =
```

In [38]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.preprocessing import MinMaxScaler

model = Sequential()
model.add(Dense(25, input_dim=5, activation='relu'))
model.add(Dense(25, activation='relu'))
model.add(Dense(1, activation='linear'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 25)                150
_________________________________________________________________
dense_1 (Dense)              (None, 25)                650
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 26
=================================================================
Total params: 826
Trainable params: 826
Non-trainable params: 0
_________________________________________________________________
```

In [39]:
```
model.compile(optimizer='adam', loss='mean_squared_error')
```

In [40]:
```
epochs_hist = model.fit(X_train, y_train, epochs=20, batch_size=25,  verbose=1, vali
```

```
Epoch 1/20
12/12 [==============================] - 0s 16ms/step - loss: 0.0670 - val_loss: 0.0
464
Epoch 2/20
12/12 [==============================] - 0s 3ms/step - loss: 0.0301 - val_loss: 0.02
51
Epoch 3/20
12/12 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.01
82
Epoch 4/20
12/12 [==============================] - 0s 3ms/step - loss: 0.0137 - val_loss: 0.01
47
Epoch 5/20
12/12 [==============================] - 0s 4ms/step - loss: 0.0107 - val_loss: 0.01
19
Epoch 6/20
12/12 [==============================] - 0s 4ms/step - loss: 0.0084 - val_loss: 0.00
99
Epoch 7/20
12/12 [==============================] - 0s 3ms/step - loss: 0.0067 - val_loss: 0.00
84
Epoch 8/20
12/12 [==============================] - 0s 3ms/step - loss: 0.0053 - val_loss: 0.00
67
Epoch 9/20
12/12 [==============================] - 0s 3ms/step - loss: 0.0041 - val_loss: 0.00
52
Epoch 10/20
12/12 [==============================] - 0s 3ms/step - loss: 0.0031 - val_loss: 0.00
44
Epoch 11/20
12/12 [==============================] - 0s 4ms/step - loss: 0.0026 - val_loss: 0.00
37
```

```
Epoch 12/20
12/12 [==============================] - 0s 3ms/step - loss: 0.0020 - val_loss: 0.00
30
Epoch 13/20
12/12 [==============================] - 0s 3ms/step - loss: 0.0017 - val_loss: 0.00
26
Epoch 14/20
12/12 [==============================] - ETA: 0s - loss: 0.001 - 0s 3ms/step - loss:
0.0015 - val_loss: 0.0023
Epoch 15/20
12/12 [==============================] - 0s 4ms/step - loss: 0.0013 - val_loss: 0.00
20
Epoch 16/20
12/12 [==============================] - 0s 4ms/step - loss: 0.0011 - val_loss: 0.00
18
Epoch 17/20
12/12 [==============================] - 0s 3ms/step - loss: 9.9379e-04 - val_loss:
0.0016
Epoch 18/20
12/12 [==============================] - 0s 3ms/step - loss: 9.0831e-04 - val_loss:
0.0016
Epoch 19/20
12/12 [==============================] - 0s 4ms/step - loss: 8.3466e-04 - val_loss:
0.0014
Epoch 20/20
12/12 [==============================] - 0s 3ms/step - loss: 7.6295e-04 - val_loss:
0.0013
```

STEP#5: EVALUATING THE MODEL

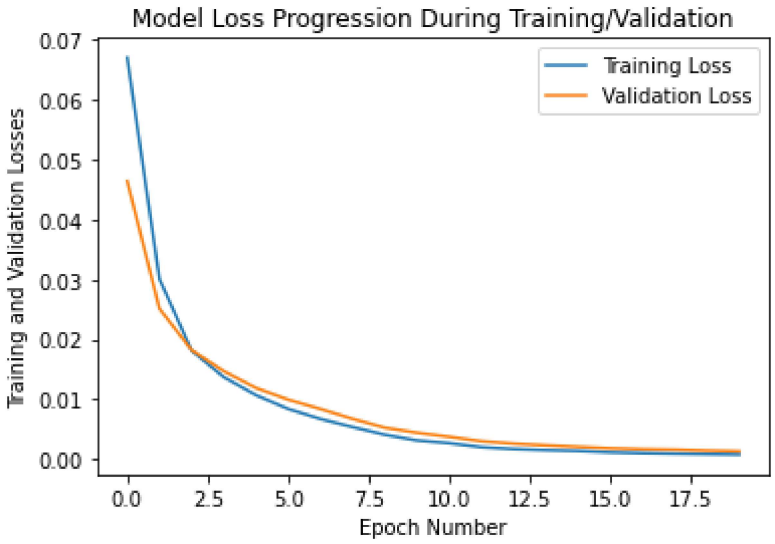
In [41]:
```
print(epochs_hist.history.keys())
```

```
dict_keys(['loss', 'val_loss'])
```

In [42]:
```
plt.plot(epochs_hist.history['loss'])
plt.plot(epochs_hist.history['val_loss'])

plt.title('Model Loss Progression During Training/Validation')
plt.ylabel('Training and Validation Losses')
plt.xlabel('Epoch Number')
plt.legend(['Training Loss', 'Validation Loss'])
```

Out[42]:
```
<matplotlib.legend.Legend at 0x270b4f92c10>
```



In [43]:
```
# Gender, Age, Annual Salary, Credit Card Debt, Net Worth
```

```
X_Testing = np.array([[1, 50, 50000, 10985, 629312]])
```

In [44]:
```
y_predict = model.predict(X_Testing)
y_predict.shape
```

Out[44]: (1, 1)

In [45]:
```
print('Expected Purchase Amount=', y_predict[:,0])
```

```
Expected Purchase Amount= [160163.5]
```