Assignment 1

First, type your name in the code cell below and run the cell.

Full Name

Name: " Edem Faith Dotse "

Show code

```
Name: Edem Faith Dotse
```

Directions & Business Case:

Directions

In this assignment, you will perform preprocessing and EDA classification.

Include your code and any required written responses immediately under the relevant question prompt in the space provided.

You will include your answers in this .ipynb template notebook in the space provided under the numbered questions below. Under the numbered questions there is either a code cell or text cell provided for your response.

There are 3 deliverables for this assignment:

> 1. ***The link to your Google Colab notebook file***: Submit the link to your notebook file. To do so, click **Share** on the top right-hand side. Then a box will pop-up. You need to change **"restricted"** to **"anyone with the link."** Then, copy the link and paste it as a comment when submitting the assignment on Canvas.
> 2. ***The IPYNB notebook file***: Download the same file as ipynb. To do so, Go to **File**, select **Download**. Then Click on **ipynb** on the menu box.
> 3. ***The PDF version of your notebook file***: Download the same file as pdf. To do so, Go to **File**, select **Print**,A menu box will pop up. Then Click on **PDF** on the menu box. This will convert the file into a PDF file, instead of printing it using a printer.

All written responses must be in your own words. If using AI in any capacity to aid with written responses to assignment question prompts, there is 1 additional required deliverable for the assignment.

> 1. ***A PDF of all AI prompts and responses used***: Submit this information aggregated as a PDF.

If this is not included and responses appear to be AI generated you will receive a 0 for the assignment. If this is included and your written responses are plagiarized, you will receive a 0 for the assignment.

Rename this template file - **LastnameFirstname_A#.ipynb**, where # is the assignment number. As an example, my Assignment #1 would be named **HillChelsey_A1.ipynb**. Your .pdf file should follow the same file naming format (for example, my Assignment #1 .pdf file would be named **HillChelsey_A1.pdf**).

Note: Points will be deducted for extraneous code in the submissions, inefficient code, incorrect file naming, and if your answers are not in the space provided for answers.

Business Case

You work for an analytics consulting firm hired by Southwest Airlines. You are given a dataset for airline routes and are tasked with preprocessing and exploring the data.

A basic description of variables in the [Airfares2.csv]() dataset is below:

Variable	Description
S_CODE	starting airport code
S_CITY	starting airport city
E_CODE	ending airport code
E_CITY	ending airport city

Variable	Description
COUPON	average number of coupons for the route
NEW	number of new carriers entering the route
VACATION	indicates if the route is a vacation route (Yes) or not (No)
SW	indicates if Southwest Airlines serves the route (Yes) or not (No)
S_INCOME	starting city's average personal income
E_INCOME	ending city's average personal income
S_POP	starting city's population
E_POP	ending city's population
SLOT	indicates if either endpoint airport is slot controlled (Controlled) or not (Free)
GATE	indicates if either endpoint airport has gate constraints(Constrained) or not (Free)
DISTANCE	distance (miles) between two endpoint airports
PAX	the number of passengers on the route
FARE	the average fare for the route

Import Packages:

```
import numpy as np
import pandas as pd
from sklearn import set_config
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler, PowerTransformer
from sklearn.preprocessing import KBinsDiscretizer
from sklearn.preprocessing import LabelBinarizer
from sklearn.impute import SimpleImputer, KNNImputer, MissingIndicator
from sklearn.decomposition import PCA
import matplotlib.pylab as plt
import seaborn as sns
set_config(transform_output = "pandas")
```

Questions:

```
data = pd.read_csv('Airfares2.csv') #read csv
```

> 1. **(a) (10) Use the 'Airfares2.csv' file to create a dataframe named data. Then, view the first 5 observations in the `data` dataframe.**

```
data.head() #data preview
```

	S_CODE	S_CITY	E_CODE	E_CITY	COUPON	NEW	VACATION	SW	HI	S_INCOME	E_INCOME	S_POP	E_POP	SLOT	GATE
0	*	Dallas/Fort Worth TX	*	Amarillo TX	1.00	3.0	No	Yes	5291.99	28637	21112	3036732	205711.0	Free	Free
1	*	Atlanta GA	*	Baltimore/Wash Intl MD	1.06	3.0	No	No	5419.16	26993	29838	3532657	7145897.0	Free	Free
2	*	Boston MA	*	Baltimore/Wash Intl MD	1.06	3.0	No	No	9185.28	30124	29838	5787293	7145897.0	Free	Free
3	ORD	Chicago IL	*	Baltimore/Wash Intl MD	1.06	3.0	No	Yes	2657.35	29260	29838	7830332	7145897.0	Controlled	Free
4	MDW	Chicago IL	*	Baltimore/Wash Intl MD	1.06	3.0	No	Yes	2657.35	29260	29838	7830332	7145897.0	Free	Free

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

> 1. **(b) (10) View the dataframe information. Are missing values present?**

```
data.info() #data overview
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 638 entries, 0 to 637
Data columns (total 18 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   S_CODE    638 non-null    object 
 1   S_CITY    638 non-null    object 
 2   E_CODE    638 non-null    object 
 3   E_CITY    638 non-null    object 
 4   COUPON    637 non-null    float64
 5   NEW       632 non-null    float64
 6   VACATION  636 non-null    object 
 7   SW        637 non-null    object 
 8   HI        638 non-null    float64
 9   S_INCOME  638 non-null    int64  
 10  E_INCOME  638 non-null    int64  
 11  S_POP     638 non-null    int64  
 12  E_POP     637 non-null    float64
 13  SLOT      635 non-null    object 
 14  GATE      638 non-null    object 
 15  DISTANCE  637 non-null    float64
 16  PAX       638 non-null    int64  
 17  FARE      636 non-null    float64
dtypes: float64(6), int64(4), object(8)
memory usage: 89.8+ KB
```

Double-click (or enter) to edit

> 1. **(c) (15) Identify variables by variable type and convert any categorical variables to category types. Then, output arrays of variables by type.**

```
nums = ['S_INCOME','COUPON','NEW','HI','E_INCOME','S_POP','E_POP','DISTANCE', 'PAX', 'FARE'] #numerical variable list
ords = ['SLOT'] # Ordinal variable list
noms = list(data.columns.difference(nums + ords)) #nominal variable list


data[noms + ords] = data[noms + ords].astype('category') #convert all categorcal varibale to category types
data[nums] = data[nums].apply(pd.to_numeric,errors = 'coerce') #numeric variable to  numeric


print('variable by type')
print('Numerical:' , nums) # output numerical values
print('Nominal:', noms) # output nominal values
print('Ordinal:', ords) # output ordinal values
```

```
variable by type
Numerical: ['S_INCOME', 'COUPON', 'NEW', 'HI', 'E_INCOME', 'S_POP', 'E_POP', 'DISTANCE', 'PAX', 'FARE']
Nominal: ['E_CITY', 'E_CODE', 'GATE', 'SW', 'S_CITY', 'S_CODE', 'VACATION']
Ordinal: ['SLOT']
```

> 2. **(a) (5) View descriptive statistic information for the numerical variables in the dataframe.**

```
data[nums].describe()
```

	S_INCOME	COUPON	NEW	HI	E_INCOME	S_POP	E_POP	DISTANCE	PAX	FARE
count	638.000000	637.000000	632.000000	638.000000	638.000000	6.380000e+02	6.370000e+02	637.000000	638.000000	636.000000
mean	27759.860502	1.202512	2.775316	4442.141129	27663.727273	4.557004e+06	3.192518e+06	973.406593	12782.214734	160.749607
std	3596.207837	0.203932	0.723277	1724.267051	4611.325018	3.010985e+06	2.737294e+06	644.251137	13202.228860	75.961846
min	14600.000000	1.000000	0.000000	1230.480000	14600.000000	2.983800e+04	1.117450e+05	114.000000	1504.000000	42.470000
25%	24706.000000	1.040000	3.000000	3090.137500	23903.000000	1.862106e+06	1.228816e+06	455.000000	5328.500000	106.245000
50%	28637.000000	1.150000	3.000000	4208.185000	26409.000000	3.532657e+06	2.195215e+06	846.000000	7792.000000	144.600000
75%	29693.500000	1.300000	3.000000	5480.575000	31981.000000	7.830332e+06	4.549784e+06	1301.000000	14090.500000	209.350000
max	38813.000000	1.940000	3.000000	10000.000000	38813.000000	9.056076e+06	9.056076e+06	2764.000000	73892.000000	402.020000

> 2. **(b) (5) View descriptive statistic information for the categorical variables in the dataframe.**

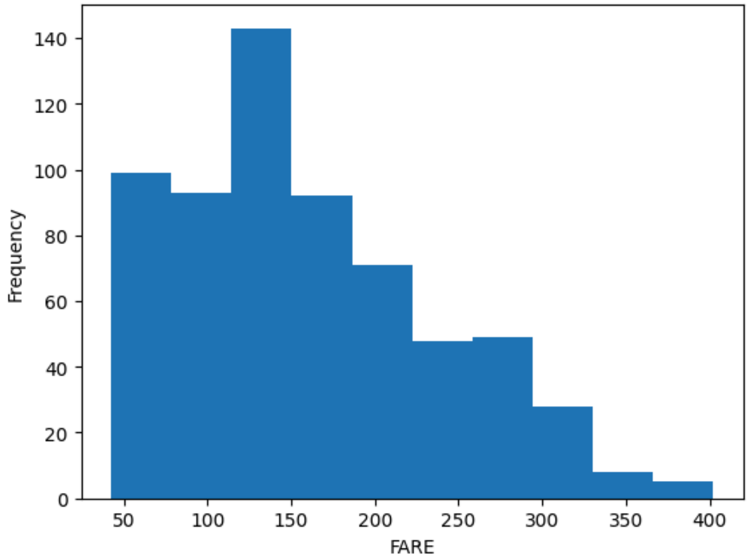
```
cats = noms + ords
data[cats].describe()
```

	E_CITY	E_CODE	GATE	SW	S_CITY	S_CODE	VACATION	SLOT
count	638	638	638	637	638	638	636	635
unique	68	8	2	2	51	8	2	2
top	New York/Newark NY	*	Free	No	Chicago IL	*	No	Free
freq	75	501	514	443	90	454	466	454

> 2. **(c) (5) Visualize the distribution of the `FARE` variable.**

```
data.FARE.plot.hist(xlabel='FARE')
```

```
<Axes: xlabel='FARE', ylabel='Frequency'>
```



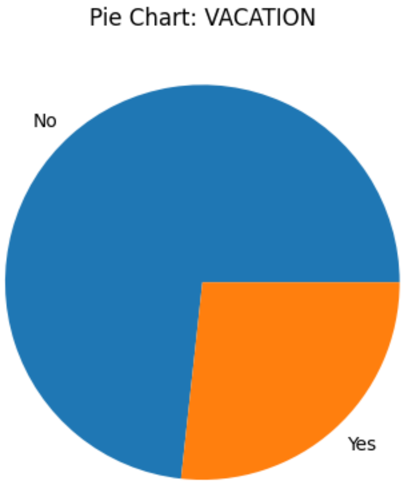
> 2. **(d) (5) Describe the shape of the `FARE` variable.**

The distribution is right skewd since majority of the fares lies between 50 and 250 and the highest fare between 100 and 150.

> 2. **(e) (5) Visualize the distribution of the `VACATION` variable.**

```
data['VACATION'].value_counts().plot(kind= 'pie', # pie chart  output
                                     title= 'Pie Chart: VACATION',
                                     ylabel= '')
```

```
<Axes: title={'center': 'Pie Chart: VACATION'}>
```



> 2. **(f) (5) Describe the distribution of the `VACATION` variable.**

The distribution of the VACATION variable is skewed towards 'No'. A higher number of passengers were not traveling for vacation compared to those who were. This indicates that the dataset contains more business or non-leisure travelers than vacation travelers.

> 3. **(a) (15) Next, you will use imputation to handle class imbalance. For numerical variables, impute the median. For categorical variables, impute the mode. Be sure to apply your changes to the `data` dataframe. Then, output dataframe information to confirm the transformation.**

```
miss = MissingIndicator()
miss_id = miss.fit_transform(data)
miss_df = pd.DataFrame(miss_id, columns=miss.features_)



input_nums = SimpleImputer(strategy = 'median') #impute median
data[nums] = input_nums.fit_transform(data[nums])

input_cats = SimpleImputer(strategy="most_frequent")   # mode
data[cats] = input_cats.fit_transform(data[cats])

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 638 entries, 0 to 637
Data columns (total 18 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   S_CODE    638 non-null    object 
 1   S_CITY    638 non-null    object 
 2   E_CODE    638 non-null    object 
 3   E_CITY    638 non-null    object 
 4   COUPON    638 non-null    float64
 5   NEW       638 non-null    float64
 6   VACATION  638 non-null    object 
 7   SW        638 non-null    object 
 8   HI        638 non-null    float64
 9   S_INCOME  638 non-null    float64
 10  E_INCOME  638 non-null    float64
 11  S_POP     638 non-null    float64
 12  E_POP     638 non-null    float64
 13  SLOT      638 non-null    object 
 14  GATE      638 non-null    object 
 15  DISTANCE  638 non-null    float64
 16  PAX       638 non-null    float64
 17  FARE      638 non-null    float64
dtypes: float64(10), object(8)
```

```
memory usage: 89.8+ KB
```

> 3. **(b) (10) Next, rescale your numerical variables using Min-Max normalization. Then, output the first 5 observations of the `data` dataframe to preview the transformation.**

```
# Create scaler instance
mm_norm = MinMaxScaler()

# Apply Min-Max scaling to numerical variables
data[nums] = mm_norm.fit_transform(data[nums])

# Check dataframe info
data.head()
```

	S_CODE	S_CITY	E_CODE	E_CITY	COUPON	NEW	VACATION	SW	HI	S_INCOME	E_INCOME	S_POP	E_POP	SLOT	GATE
0	*	Dallas/Fort Worth TX	*	Amarillo TX	0.00000	1.0	No	Yes	0.463139	0.579730	0.268946	0.333128	0.010506	Free	Free
1	*	Atlanta GA	*	Baltimore/Wash Intl MD	0.06383	1.0	No	No	0.477641	0.511832	0.629331	0.388071	0.786437	Free	Free
2	*	Boston MA	*	Baltimore/Wash Intl MD	0.06383	1.0	No	No	0.907096	0.641143	0.629331	0.637858	0.786437	Free	Free
3	ORD	Chicago IL	*	Baltimore/Wash Intl MD	0.06383	1.0	No	Yes	0.162708	0.605460	0.629331	0.864202	0.786437	Controlled	Free
4	MDW	Chicago IL	*	Baltimore/Wash Intl MD	0.06383	1.0	No	Yes	0.162708	0.605460	0.629331	0.864202	0.786437	Free	Free

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

> 4. **(10) Finally, transform your categorical variables using one-hot encoding. Create a new dataframe, named `data_ohe`, which contains your transformed numerical variables and the one-hot encoded categorical variables. Then, preview the last 5 rows of the `data_ohe` dataframe to preview the transformation.**

```
data_ohe = pd.get_dummies(data, columns=cats, drop_first=False)
data_ohe.tail()
```

	COUPON	NEW	HI	S_INCOME	E_INCOME	S_POP	E_POP	DISTANCE	PAX	FARE	...	S_CODE_EWR	S_CODE_IAD	S_CODE_JFK
633	0.085106	1.0	0.112460	0.759551	0.94061	0.951812	0.098383	0.345660	0.453390	0.242414	...	False	False	False
634	0.085106	1.0	0.112460	0.759551	0.94061	0.951812	0.098383	0.345660	0.453390	0.242414	...	True	False	False
635	0.180851	1.0	0.634849	0.553174	0.94061	0.544912	0.098383	0.319245	0.062331	0.229175	...	False	False	False
636	0.297872	1.0	0.494434	0.717838	0.94061	0.500756	0.098383	0.280755	0.046596	0.242386	...	False	True	False
637	0.297872	1.0	0.494434	0.717838	0.94061	0.500756	0.098383	0.280755	0.046596	0.242386	...	False	False	False

5 rows × 153 columns