In [ ]:
```
# Hello Mr. Biddulph! This file serves to show how data could be easily appe
# existing data set. For this project, I am using Jupyter Notebook. This is
# used for AI and ML applications. I will add comments throughout to show yo
```

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
# Importing cool Python tools for visualization.
```

In [ ]:
```
originalData = pd.read_csv("/Users/alexhanley/MF/simulated_business_data.csv
# Importing our data file. This is randomly generated business data from Cha
```

In [ ]:
```
originalData.head()
# Displaying the first 5 lines of data for testing purposes.
```

Out[ ]:

	Business Name	Address	Owner Age	Credit Score	Occupation	Taxable Value	Business Age	Mortg
0	Smith-Obrien	756 Cruz Creek Apt. 228, East Michaelstad, DC ...	65	414	Toxicologist	76225	24	
1	Cruz-Juarez	521 Quinn Avenue Suite 544, New Michaelmouth, ...	40	528	Engineer, control and instrumentation	196316	24	
2	Richardson PLC	PSC 8796, Box 4913, APO AE 77742	68	389	Medical physicist	669176	14	
3	Owen LLC	106 Jamie Mission, North William, AK 99899	26	395	Programmer, multimedia	279258	8	
4	Anderson Ltd	86642 Ellis Flat Suite 369, West Regina, KY 54296	60	503	Human resources officer	800800	21	

In [ ]:
```
# Uh oh! Looks like we forgot to add the data for each businesses' revenue a
# structure. Dont worry! Well append that easily.
```

In [ ]:
```
newDataToAppend = pd.read_csv("/Users/alexhanley/MF/business_structure_rever
# Our new data, it contains the business structure and its revenue.
```

In [ ]:
```
newDataToAppend.head()
# Displaying the first 5 lines of data for testing purposes.
```

Out[ ]:

	Business Structure	Revenue
0	Limited Liability Company (LLC)	1877825
1	Corporation	429610
2	Partnership	4624226
3	Sole Proprietorship	4118603
4	Corporation	3754854

In [ ]:
```
import pandas as pd

# Ensure the new data is in the same order and has the same number of rows
assert len(originalData) == len(newDataToAppend), "Datasets do not match in

# Concatenate the new data to the original data as new columns
combinedData = pd.concat([originalData, newDataToAppend], axis=1)

# Print first 5 lines
combinedData.head()
```

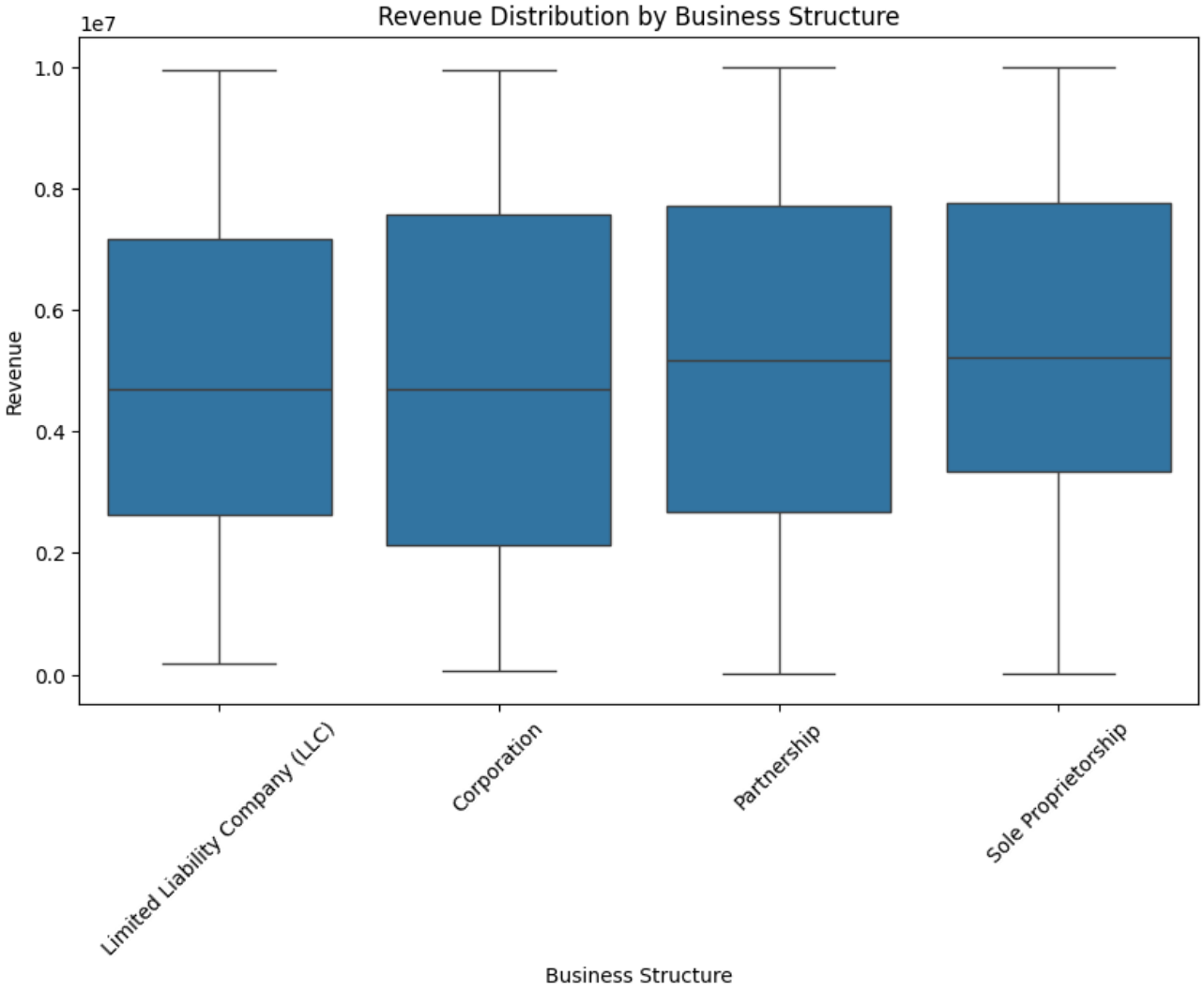
Out[ ]:

	Business Name	Address	Owner Age	Credit Score	Occupation	Taxable Value	Business Age	Mortg
0	Smith-Obrien	756 Cruz Creek Apt. 228, East Michaelstad, DC ...	65	414	Toxicologist	76225	24	
1	Cruz-Juarez	521 Quinn Avenue Suite 544, New Michaelmouth, ...	40	528	Engineer, control and instrumentation	196316	24	
2	Richardson PLC	PSC 8796, Box 4913, APO AE 77742	68	389	Medical physicist	669176	14	
3	Owen LLC	106 Jamie Mission, North William, AK 99899	26	395	Programmer, multimedia	279258	8	
4	Anderson Ltd	86642 Ellis Flat Suite 369, West Regina, KY 54296	60	503	Human resources officer	800800	21	

In [ ]:
```
# The next step would be to gain access to the Experian business data.
```

In [ ]:
```
# Here is a visualization of the AI potentially significant data we appended
```

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Business Structure', y='Revenue', data=combinedData)
plt.title('Revenue Distribution by Business Structure')
plt.xlabel('Business Structure')
plt.ylabel('Revenue')
plt.xticks(rotation=45)
plt.show()
# Business Structure to Revenue
```



In [ ]:
```
# Next, for fun, I created a small linear regression machine learning model
# how easy it would be to gain AI insight after recieving data. Any data tha
# to be appended could be very easily. You can even switch out which data ty
# with a simple copy and paste.
```

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import OneHotEncoder

import numpy as np

one_hot_encoder = OneHotEncoder(sparse=False)
X = one_hot_encoder.fit_transform(combinedData[['Business Structure']])  # C
Y = combinedData['Revenue']
xName = "Business Structure"
yName = "Revenue"
# Splitting the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, rar
```

```
# Initializing and training the linear regression model
model = LinearRegression()
model.fit(X_train, Y_train)

# Making predictions
Y_pred = model.predict(X_test)

# Evaluating the model
meanSquaredError = mean_squared_error(Y_test, Y_pred)
r2Score = r2_score(Y_test, Y_pred)

print("Mean Squared Error:", meanSquaredError)
print("Coefficient of Determination (R^2):", r2Score)

if r2Score < 0.4:
    print("The correlation between {} and {} is not statistically significar
elif r2Score > 0.4 and r2Score <= 1:
        print("The correlation between {} and {} is statistically significar
```

```
Mean Squared Error: 8416218597912.085
Coefficient of Determination (R^2): -0.010416722198247852
The correlation between Business Structure and Revenue is not statistically
significant. There is no correlation between the two.
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packa
ges/sklearn/preprocessing/_encoders.py:975: FutureWarning: `sparse` was rena
med to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_ou
tput` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [ ]:
```
# Wow, these numbers are horrific ;). This is beacuse this was randomly gene
# odds that 1000 lines of randomly generated data lead to some correlation i
```