```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
```

Question 1

```
s_p = pd.read_csv("/S&P500.csv")
```

```
s_p.head()
```

	Symbol	Name	Sector	Price	52 Week High	52 Week Low	Dividend Yield	Earnings per Share	Sales per Share	Book Value per Share	EBITDA	Market Cap
0	LB	L Brands Inc.	Consumer Discretionary	47.77	63.10	35.00	4.886988	3.98	27.999569	0.034039	2.329000e+09	1.386204e+10
1	PM	Philip Morris International	Consumer Staples	100.39	123.55	96.66	4.328479	4.48	36.406581	0.076128	1.180200e+10	1.540000e+11
2	HRB	Block H&R	Financials	25.19	31.80	19.85	3.729604	1.92	12.477428	0.122633	8.947540e+08	5.381434e+09
3	CL	Colgate-Palmolive	Consumer Staples	68.95	77.91	66.26	2.280177	2.28	17.234325	0.291642	4.064000e+09	6.161664e+10

Next steps: Generate code with s_p View recommended plots

```
s_p.shape
```

```
(505, 12)
```

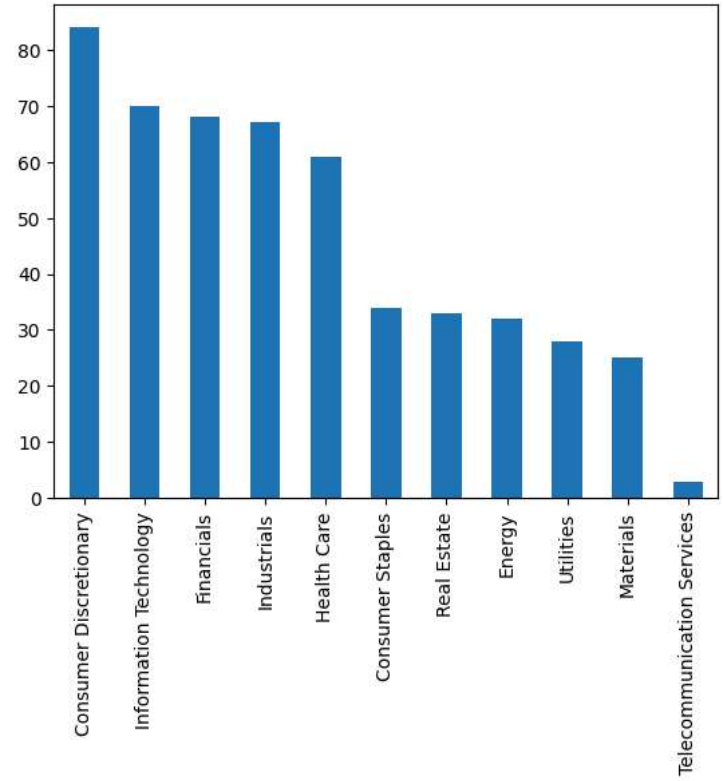
1. How many unique levels of Sector variable? Which sector has the most companies (using a bar graph to show)?

```
s_p['Sector'].unique()
```

```
array(['Consumer Discretionary', 'Consumer Staples', 'Financials',
       'Information Technology', 'Health Care', 'Energy', 'Industrials',
       'Real Estate', 'Utilities', 'Materials',
       'Telecommunication Services'], dtype=object)
```

```
s_p['Sector'].value_counts().plot(kind = 'bar')
```

```
<Axes: >
```



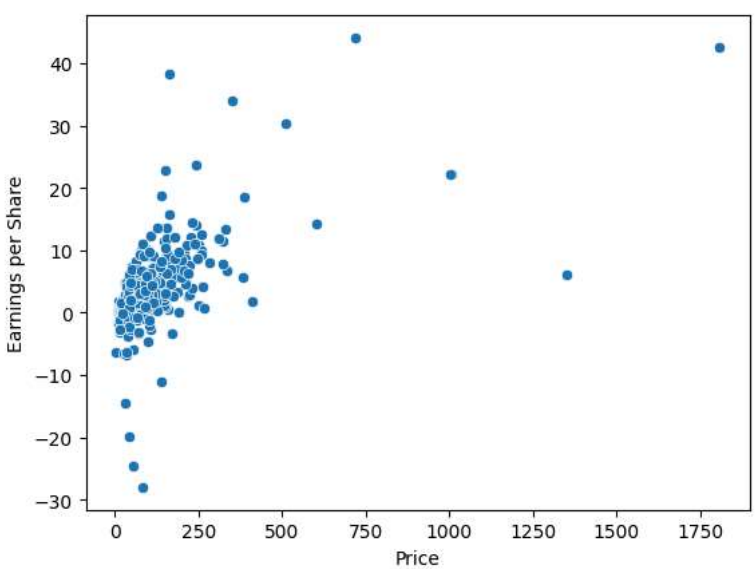
2. Explore the relationship between price and earnings per share using scatterplot. Is there any correlation between the two?

Generate | Explore the relationship between price and earnings per share using scatterplot. | Close

1 of 4 Undo Changes Use code with caution

```
# prompt: Explore the relationship between price and earnings per share using scatterplot.

import matplotlib.pyplot as plt
sns.scatterplot(x = s_p['Price'], y = s_p['Earnings per Share'])
plt.show()
```




```
correlation = s_p['Earnings per Share'].corr(s_p['Price'])
correlation
```

```
0.5910611927275284
```

3. Fit a multiple linear regression model to Price as a function of the logged variables: log Dividend Yield, log Sales per share, log Book value per share, log Market cap, Earning per share and Sector.

```
# Create log-transformed variables
import math
s_p['log_Dividend Yield'] = np.log(s_p['Dividend Yield'])
s_p['log_Sales per Share'] = np.log(s_p['Sales per Share'])
s_p['log_Book value per Share'] = np.log(s_p['Book Value per Share'])
s_p['log_Market cap'] = np.log(s_p['Market Cap'])

#dropping null values
s_p.replace([np.inf, -np.inf], np.nan, inplace=True)
s_p.dropna(inplace=True)

#creating independant variables
X = s_p[['log_Dividend Yield', 'log_Sales per Share', 'log_Book value per Share',
       'log_Market cap', 'Earnings per Share','Sector']]
X = pd.get_dummies(X, columns = ['Sector'], drop_first = True, dtype= int )
X_with_intercept = sm.add_constant(X)

# Define the dependent variable
y = s_p['Price']

# Fit the multiple linear regression model
model = sm.OLS(y, X_with_intercept).fit()

# Print the model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.570
Model:                            OLS   Adj. R-squared:                  0.553
Method:                 Least Squares   F-statistic:                     34.97
Date:                Sat, 02 Mar 2024   Prob (F-statistic):           3.99e-63
Time:                        23:35:33   Log-Likelihood:                -2126.5
No. Observations:                 412   AIC:                             4285.
Df Residuals:                     396   BIC:                             4349.
Df Model:                          15
Covariance Type:            nonrobust
==============================================================================================
                                       coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                            -380.0799     56.148     -6.769      0.000    -490.466    -269.694
log_Dividend Yield                -22.5729      3.290     -6.862      0.000     -29.040     -16.106
log_Sales per Share                26.3911      3.069      8.599      0.000      20.357      32.425
log_Book value per Share            2.4850      2.511      0.990      0.323      -2.451       7.421
log_Market cap                     14.7058      2.338      6.289      0.000      10.109      19.303
Earnings per Share                  4.8595      0.545      8.909      0.000       3.787       5.932
Sector_Consumer Staples             3.3636      9.505      0.354      0.724     -15.323      22.050
Sector_Energy                      14.7853     10.538      1.403      0.161      -5.932      35.502
Sector_Financials                  11.8186      8.286      1.426      0.155      -4.472      28.109
Sector_Health Care                 10.3921      9.568      1.086      0.278      -8.418      29.203
Sector_Industrials                 22.2050      7.960      2.789      0.006       6.555      37.855
Sector_Information Technology      21.1346      8.702      2.429      0.016       4.026      38.243
Sector_Materials                   19.1842     10.314      1.860      0.064      -1.092      39.461
Sector_Real Estate                 82.2831     11.437      7.194      0.000      59.797     104.769
Sector_Telecommunication Services  -21.5675     26.071     -0.827      0.409     -72.823      29.688
Sector_Utilities                   21.4223     10.727      1.997      0.047       0.333      42.512
==============================================================================
Omnibus:                      155.928   Durbin-Watson:                   1.811
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              910.737
Skew:                           1.504   Prob(JB):                    1.72e-198
Kurtosis:                       9.634   Cond. No.                         657.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

4. What is the coefficient of Earnings per share? p-value? Is it significant? How would you interpret the impact of it on stock price?

coefficient of Earnings per share: 4.8595 p = 0.000 (Significant) One unit of earnings per share will increase the price by 4.8595 keeping all the other variables constant.

5. Check the dummy variables of Sector. Which is the base case? What is the coefficient of Sector of Information Technology? p-value? Is it significant? How would you interpret

Base case: *Consumer Discretionary*
coeff of section of IT: 21.1346
p value: 0.016 (significant)
Sector IT have 21.1346 price more than sector Consumer discretionary keeping all other variables constant.

6. How well does the model fit the data?

With an R-squared value of 0.570, it indicates that approximately 57% of the variance in the dependent variable (Y) can be explained by the independent variable (X) in your model. This suggests a moderate level of fit; however, there is still some unexplained variance in the data. It's essential to consider the context of your analysis and the specific requirements of your model when interpreting the R-squared value

7. Make predictions of stock price.

```
price_predicted = model.predict(X_with_intercept)
s_p['price_predicted'] = price_predicted
s_p.head()
```

	Symbol	Name	Sector	Price	52 Week High	52 Week Low	Dividend Yield	Earnings per Share	Sales per Share	Book Value per Share	EBITDA	Market Cap	log_Dividend Yield
0	LB	L Brands Inc.	Consumer Discretionary	47.77	63.10	35.00	4.886988	3.98	27.999569	0.034039	2.329000e+09	1.386204e+10	1.586576
1	PM	Philip Morris International	Consumer Staples	100.39	123.55	96.66	4.328479	4.48	36.406581	0.076128	1.180200e+10	1.540000e+11	1.465216
2	HRB	Block H&R	Financials	25.19	31.80	19.85	3.729604	1.92	12.477428	0.122633	8.947540e+08	5.381434e+09	1.316302
3	CL	Colgate-Palmolive	Consumer Staples	68.95	77.91	66.26	2.280177	2.28	17.234325	0.291642	4.064000e+09	6.161664e+10	0.824253
4	YUM	Yum! Brands Inc	Consumer Discretionary	76.30	86.93	62.85	1.797080	4.07	12.084953	0.359770	2.289000e+09	2.700330e+10	0.586163

Next steps: Generate code with s_p View recommended plots

Question 2

Double-click (or enter) to edit

```
churn = pd.read_csv('/TelcoChurn.csv')
churn.head()
```

rCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	S
0	Yes	No	1	No	No phone service	DSL	No	...	No	No	
0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	
0	No	No	2	Yes	No	DSL	Yes	...	No	No	
0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	
0	No	No	2	Yes	No	Fiber optic	No	...	No	No	

```
churn.shape
```

```
(7043, 21)
```

1. Explore the relationship between “gender” and “MonthlyCharges” using boxplots across gender categories. What is your finding?

```
boolean_data = churn.gender
continuous_data = churn.MonthlyCharges
```

Generate | Explore the relationship between “gender” and “MonthlyCharges” using boxplots across gender categories | Close

1 of 4 Undo Changes Use code with caution

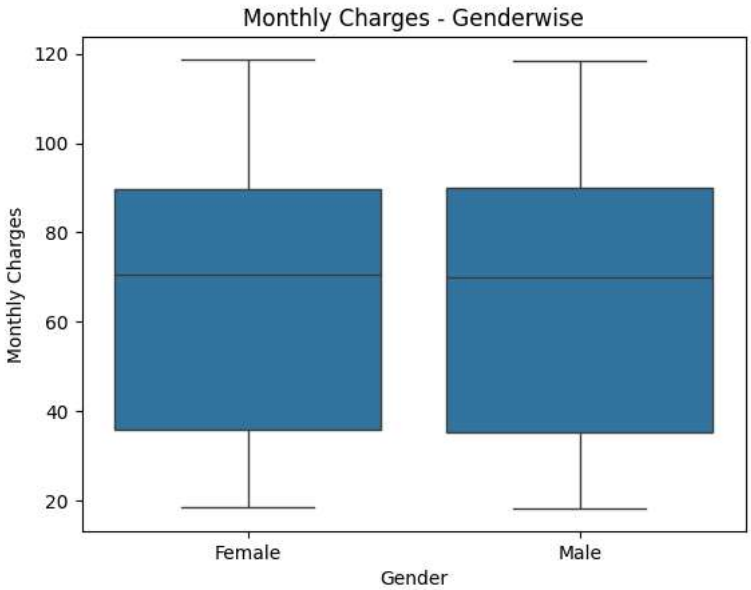
```
# prompt: Explore the relationship between “gender” and “MonthlyCharges” using boxplots across gender categories

import seaborn as sns
import matplotlib.pyplot as plt

# Create a boxplot for each gender
sns.boxplot(x="gender", y="MonthlyCharges", data=churn)

# Add title and labels
plt.title("Monthly Charges - Genderwise")
plt.xlabel("Gender")
plt.ylabel("Monthly Charges")

# Show the plot
plt.show()
```



```
churn[churn.gender=='Male'].MonthlyCharges.describe(),churn[churn.gender=='Female'].MonthlyCharges.describe()
```

```
(count    3555.000000
 mean       64.327482
 std        30.116093
 min        18.250000
 25%        35.225000
 50%        70.100000
 75%        89.875000
 max       118.350000
 Name: MonthlyCharges, dtype: float64,
 count    3488.000000
 mean       65.204243
 std        30.061341
 min        18.400000
 25%        35.900000
 50%        70.650000
 75%        89.850000
 max       118.750000
 Name: MonthlyCharges, dtype: float64)
```

Mean, maximum and minimum monthly charges for both male and female customers are almost same.

2. Fit a logistic regression model to predict “Churn” using the following variables: a. Gender b. SeniorCitizen c. Dependents d. tenure e. Contract f. MonthlyCharges

```
churn.columns
```

```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

```
churn.isna().sum()
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

```
import statsmodels.api as sm
import patsy

y_log, X_log = patsy.dmatrices('Churn ~ gender + SeniorCitizen + Dependents + tenure + Contract + MonthlyCharges',
                               data = churn,
                               return_type="dataframe")
```

```
y_log.head()
```

	Churn[No]	Churn[Yes]
0	1.0	0.0
1	1.0	0.0
2	0.0	1.0
3	1.0	0.0
4	0.0	1.0

Next steps: Generate code with y_log View recommended plots

```
X_log.head()
```

	Intercept	gender[T.Male]	Dependents[T.Yes]	Contract[T.One year]	Contract[T.Two year]	SeniorCitizen	tenure	MonthlyCharges
0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	29.85
1	1.0	1.0	0.0	1.0	0.0	0.0	34.0	56.95
2	1.0	1.0	0.0	0.0	0.0	0.0	2.0	53.85
3	1.0	1.0	0.0	1.0	0.0	0.0	45.0	42.30
4	1.0	0.0	0.0	0.0	0.0	0.0	2.0	70.70

Next steps: Generate code with X_log View recommended plots

```
X_log.columns
```

```
Index(['Intercept', 'gender[T.Male]', 'Dependents[T.Yes]',
       'Contract[T.One year]', 'Contract[T.Two year]', 'SeniorCitizen',
       'tenure', 'MonthlyCharges'],
      dtype='object')
```

```
import statsmodels.api as sm
logit_model = sm.Logit(y_log['Churn[Yes]'], X_log[['Intercept', 'gender[T.Male]', 'Dependents[T.Yes]',
       'Contract[T.One year]', 'Contract[T.Two year]', 'SeniorCitizen',
       'tenure', 'MonthlyCharges']])
logit_results = logit_model.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.433272
         Iterations 8
```

3. Display the results.

```
logit_results.summary()
```

Logit Regression Results

Dep. Variable:	Churn[Yes]	**No. Observations:**	7043
Model:	Logit	**Df Residuals:**	7035
Method:	MLE	**Df Model:**	7
Date:	Sun, 03 Mar 2024	**Pseudo R-squ.:**	0.2512
Time:	01:28:55	**Log-Likelihood:**	-3051.5
converged:	True	**LL-Null:**	-4075.1
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-1.5281	0.098	-15.598	0.000	-1.720	-1.336
gender[T.Male]	-0.0132	0.063	-0.209	0.834	-0.137	0.110
Dependents[T.Yes]	-0.2442	0.079	-3.081	0.002	-0.399	-0.089
Contract[T.One year]	-0.9845	0.102	-9.699	0.000	-1.183	-0.786
Contract[T.Two year]	-1.8909	0.168	-11.280	0.000	-2.219	-1.562
SeniorCitizen	0.4178	0.081	5.135	0.000	0.258	0.577
tenure	-0.0363	0.002	-17.572	0.000	-0.040	-0.032
MonthlyCharges	0.0272	0.001	19.853	0.000	0.024	0.030

4. How does gender impact Churn?

Gender data consists of male and female category. From the results it is interpreted that, male have e^-0.0132 times more odds of churn than females.

5. What is the impact of gender on Churn? Use odds ratio to interpret.

odds ratio = e^-0.0132. Refer above for impact

6. Get the predicted probabilities and make the classification based on probabilities. You can specify the cutoff probability yourself.

```
# assuming cutoff threshold as 0.5
predicted_churn = logit_results.predict(X_log)
predicted_churn
```

```
0       0.319956
1       0.098452
2       0.462112
3       0.046893
4       0.579044
          ...
7038    0.207627
7039    0.071093
7040    0.202920
7041    0.679454
7042    0.049220
Length: 7043, dtype: float64
```

```
predicted_class = (predicted_churn >= 0.5).astype(int)
predicted_class.value_counts()
```

```
0    5582
1    1461
dtype: int64
```