

DATA220 LAB 1- PART 1

Table of Contents

Preface:	3
1. Identify the dataset columns into nominal, categorical, continuous, etc. categories.	3
2. Insights about the data:	3
Other Insights:.....	10
3. Find the number of null values for each column.	11
4. Know about the patients.....	12
A) Oldest Patient.....	12
B) Youngest Patient	12
C) Average age group	12
D) Median Age	12
E) Relationship between the deaths and ages	12
F) Age groups whose survival rate is the largest	13
G) Relationship between the deaths and blood pressure	13
H) Relationship between the deaths and typea test score:	14
I) Relationship between the deaths and family history:	14
H) Visuals on data distributions.....	15
J) Handling Missing Values.....	18
K) Feature Scaling	18
L) Model Fitting: Logistic Regression.....	20

Table of Figures:

Figure 1: % of Patients with CHD	4
Figure 2: SBP Histogram	4
Figure 3: Box Plot Representation for SBP	4
Figure 4: Tobacco Consumption by Patients	5
Figure 5: Box Plot for Tobacco Consumption	5
Figure 6: LDL Histogram	6
Figure 7: Box Plot for LDL by Patients	6
Figure 8: Adipose Tissue Concentration Histogram.....	6
Figure 9: Boxplot for Adipose Tissue Concentration	7
Figure 10: TypeEA: Score Histogram	7
Figure 11: Boxplot for typeEA:Score	8
Figure 12: Obesity Level Histogram.....	8
Figure 13: Boxplot for Obesity Level.....	8
Figure 14: Alcohol Consumption by Patients.....	9
Figure 15: Box Plot for Alcohol Consumption.....	9
Figure 16: Patient Age Histogram	10
Figure 17: Box Plot for Patient's Age	10
Figure 18: SBP by age by CHD.....	10
Figure 19: LDL by Obesity by CHD	11
Figure 20: Count of Patients by age by CHD	12
Figure 21: Patients count by Age with CHD Status	12
Figure 22: Patients count by SBP by CHD Status	13
Figure 23: Patients Count by typeEA vs CHD status	14
Figure 24: CHD by Family History	15
Figure 25: Correlation matrix for all variables	15
Figure 26: Scatter Matrix for all variables	16
Figure 27: Column Distribution for all variables.....	17
Figure 28: Heatmap for missing values	18
Figure 29: Standard Scaling for all variables.....	19
Figure 30: MinMax Scaling for all variables.....	19

Preface:

In this part we are building a model which will help us build a **response** for patient which has coronary heart disease (**CHD**). We have various variables (9 to be specific) which help us to decide if a patient will be dead or alive, depending on if they will have coronary heart disease or not.

In the dataset we have information of patients which will help us decide if a patient will develop a disease or not. We know patients age, family history, tobacco consumption, obesity, cholesterol levels, alcohol consumption, blood pressure, Adipose tissue concentration, score on test designed to measure type-A behaviour.

First, we will **explore** our data and try to find which variable clearly define which factor plays a major role in a patient developing a coronary heart disease. We will try to come up with a model which help us to **predict the response** for future patients coming with similar problems. We will use the model to predict whether they will have the disease or not based on their characteristics.

1. Identify the dataset columns into nominal, categorical, continuous, etc. categories.

Based on the provided dataset, we can categorize the columns into different data types:

Nominal/Categorical Columns:

- famhist: Family history of coronary heart disease (Present or Absent)
- chd: coronary heart disease (binary, 1 if the individual has CHD, 0 if not)

Continuous Columns:

- sbp: Systolic Blood Pressure
- tobacco: Tobacco use (in pack-years)
- ldl: Low-Density Lipoprotein (LDL) cholesterol level
- adiposity: Adiposity (a measure of body fat)
- typea: Type-A behavior (a personality type)
- obesity: Obesity level
- alcohol: Alcohol consumption (in grams per day)
- age: Age of the individual

In summary, "famhist" and "chd" are nominal/categorical variables, while the rest of the columns are continuous variables. These categorizations are essential when performing statistical analyses or machine learning tasks with the dataset, as the data types of the columns influence the types of analyses that can be performed.

2. Insights about the data:

There is data of about **412 patients** given to us. Among which we have 66.7% patients which have no coronary heart disease and about 33.3% patients have coronary heart disease.

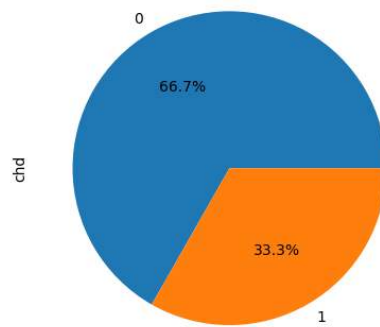


Figure 1: % of Patients with CHD

From our data we can also see that for **blood pressure (sbp)** variable that most of the patients have values in the range of 120 – 140 and very few have in the range of 200 -220. This means that patients that have blood pressure in range of 120-140 are more and patients with blood pressure ranges 200-220 are lesser.

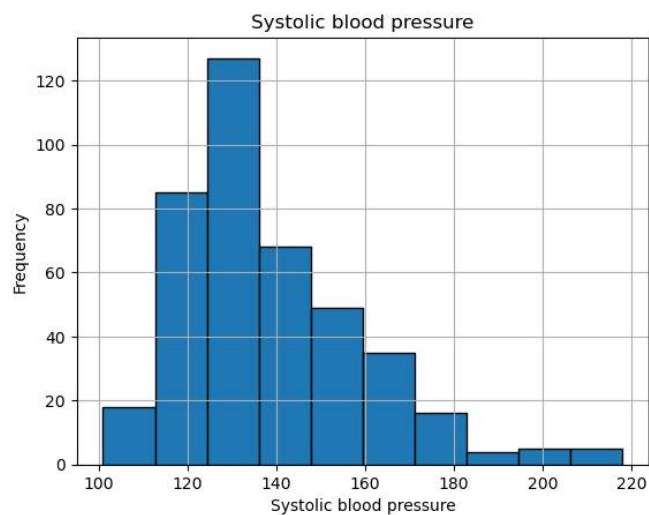


Figure 2: SBP Histogram

Descriptive analysis on blood pressure variable shows the following trends:

Summary Statistics- SBP	
Mean	139.2402913
Standard Error	1.00759292
Median	136
Mode	136
Standard Deviation	20.45190258
Sample Variance	418.2803191
Kurtosis	1.793888904
Skewness	1.153233134
Range	117
Minimum	101
Maximum	218
Sum	57367
Count	412

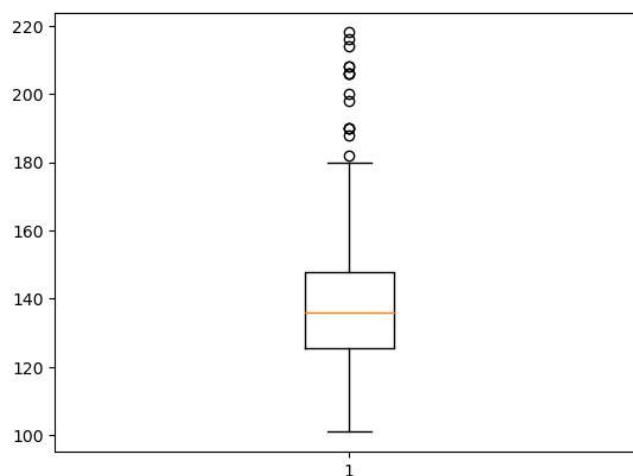


Figure 3: Box Plot Representation for SBP

For Cumulative **tobacco consumption**, most of the patient's consumption in kg is between 0-5kg and Cumulative tobacco consumption is lowest in the range 25-20kg and onwards. That means that patients that consumed 0-5kg of tobacco are more and patients who consumed 20-25kg and more tobacco are less. Tobacco consumption 0 indicates that given patient doesn't eat tobacco.

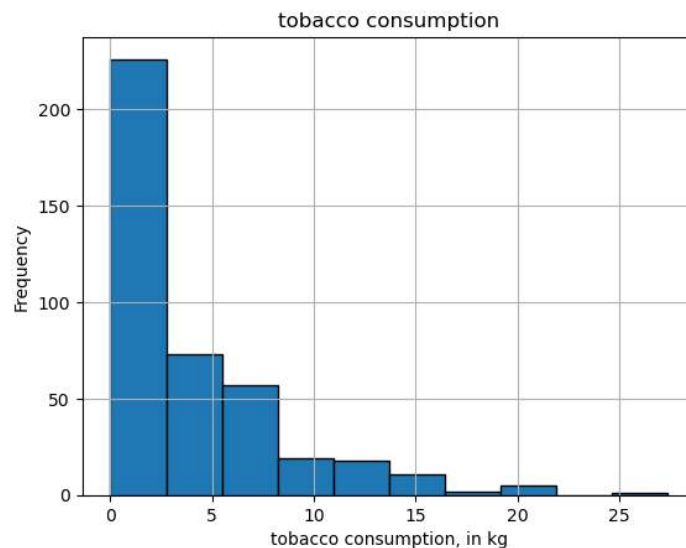


Figure 4: Tobacco Consumption by Patients

When we do descriptive analysis on our Cumulative tobacco consumption variable, we get the following values:

Summary Statistics- Tobacco	
Mean	3.666262136
Standard Error	0.222610587
Median	1.805
Mode	0
Standard Deviation	4.518501421
Sample Variance	20.41685509
Kurtosis	3.245273769
Skewness	1.699621336
Range	27.4
Minimum	0
Maximum	27.4
Sum	1510.5
Count	412

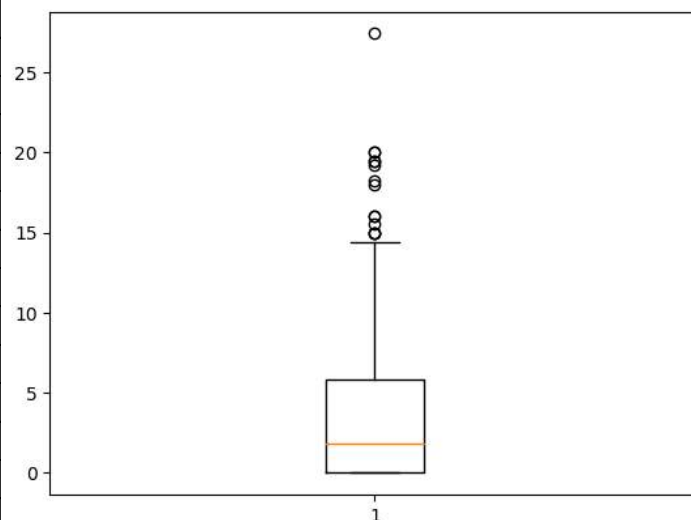


Figure 5: Box Plot for Tobacco Consumption

Next, we have **Low-density lipoprotein (LDL)** cholesterol. for this we are given different cholesterol level of all the patients. For majority of patients cholesterol levels are between 2-6 while very few patients have cholesterol levels between 10-14.

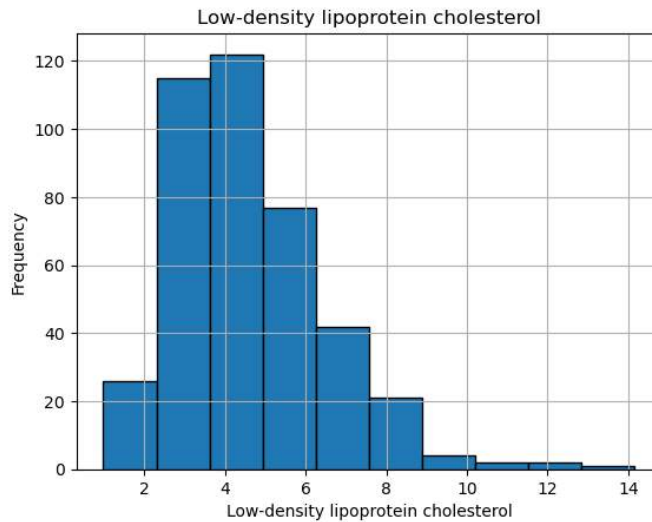


Figure 6: LDL Histogram

When we do descriptive analysis on our **Low-density lipoprotein cholesterol** variable, we get following values:

Summary Statistics- LDL	
Mean	4.589538835
Standard Error	0.0928054
Median	4.225
Mode	3.57
Standard Deviation	1.883743886
Sample Variance	3.548491028
Kurtosis	2.421015963
Skewness	1.16281578
Range	13.18
Minimum	0.98
Maximum	14.16
Sum	1890.89
Count	412

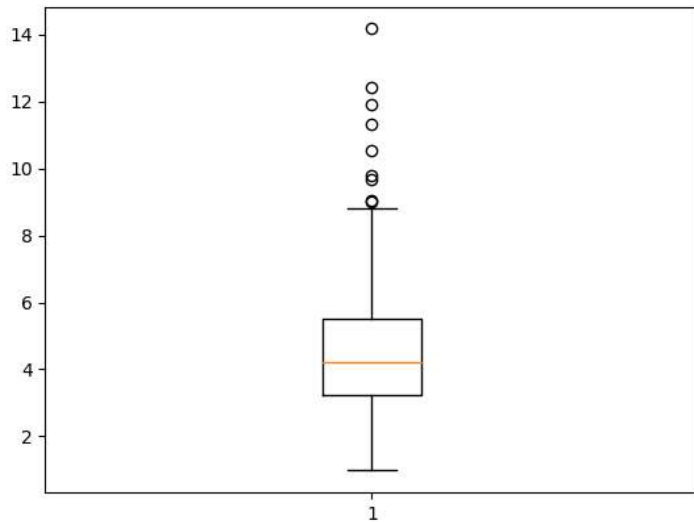


Figure 7: Box Plot for LDL by Patients

Adipose tissue concentration.

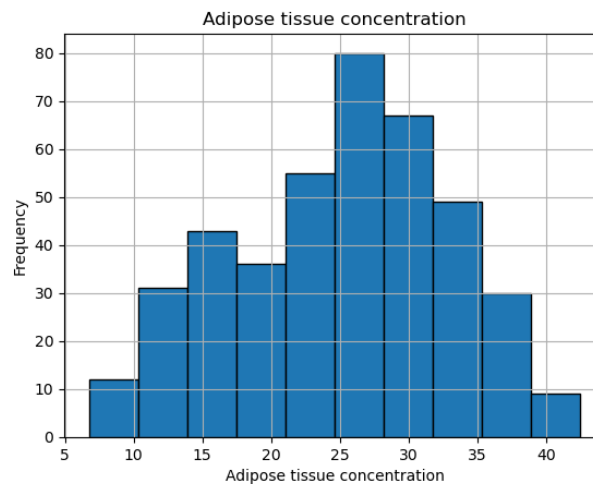


Figure 8: Adipose Tissue Concentration Histogram

Adipose tissue, also known as fat tissue or fatty tissue, is a connective tissue that is mainly composed of fat cells called adipocytes. Major chunk of patients has adipose tissue concentration within the range of 20-35 and we have very few patients which have adipose tissue concentration between 5-10 and very few in 40+ age group.

When we do descriptive analysis on our Adipose tissue concentration variable, we get following values:

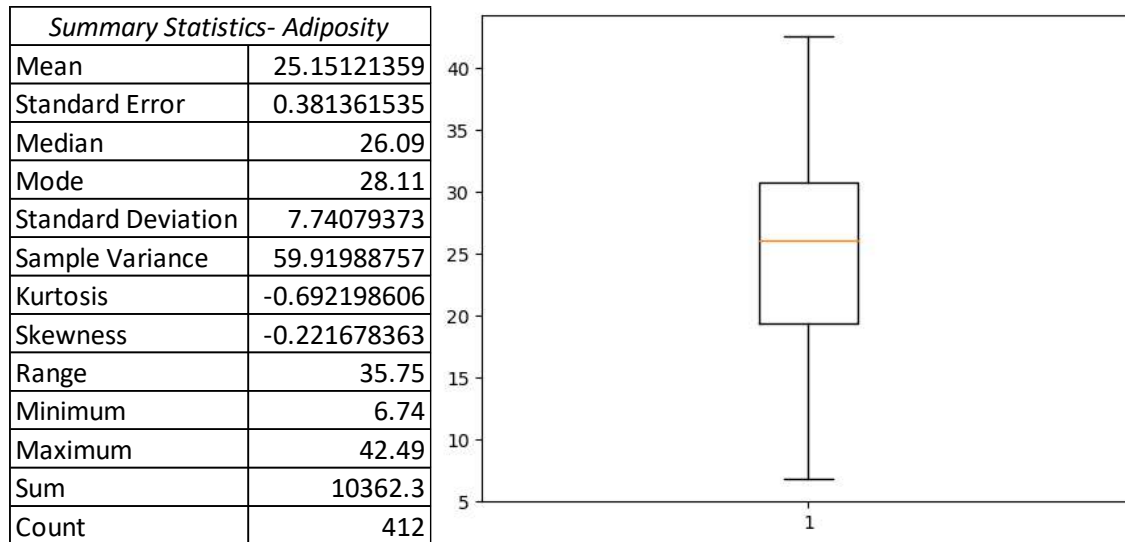


Figure 9: Boxplot for Adipose Tissue Concentration

Next, we have **typea: Score** on test designed to measure type-A behavior which is a score given to patients. Most of the patients scored between 50-60 in a test and there are lesser number of patients that scored between 10-20 in a test. Later we will study the behavior of typea test scores with CHD data.



Figure 10: TypeEA: Score Histogram

When we do descriptive analysis on our **typea: Score** on test variable, we get following values:

Summary Statistics- typeea	
Mean	52.13592233
Standard Error	0.472599734
Median	52
Mode	52
Standard Deviation	9.592726912
Sample Variance	92.02040961
Kurtosis	0.178327441
Skewness	-0.330980251
Range	53
Minimum	20
Maximum	73
Sum	21480
Count	412

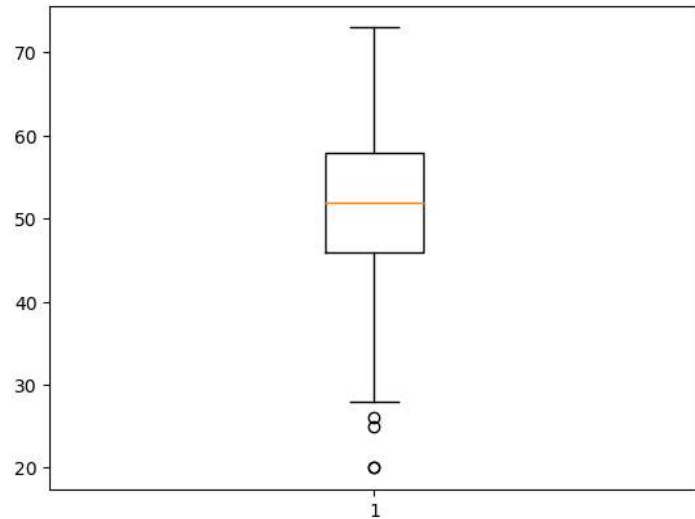


Figure 11: Boxplot for typeEA:Score

Now, we have the **obesity level** of all the patients. Mostly patients have obesity levels in the range of 20-25. But within the obesity range of 40-45 we have very less patients. Later we will analyze if obesity have any effect on patients' health.

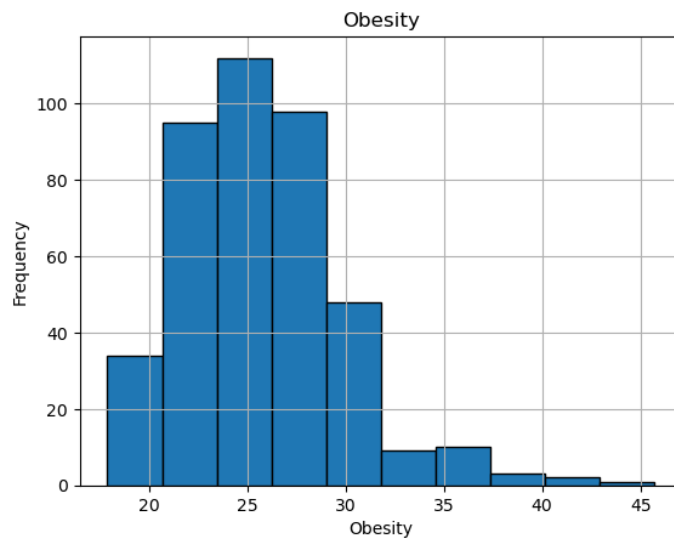


Figure 12: Obesity Level Histogram

Summary Statistics- Obesity	
Mean	25.80211165
Standard Error	0.201093124
Median	25.635
Mode	22.51
Standard Deviation	4.081744613
Sample Variance	16.66063908
Kurtosis	2.019610631
Skewness	0.952887441
Range	27.83
Minimum	17.89
Maximum	45.72
Sum	10630.47
Count	412

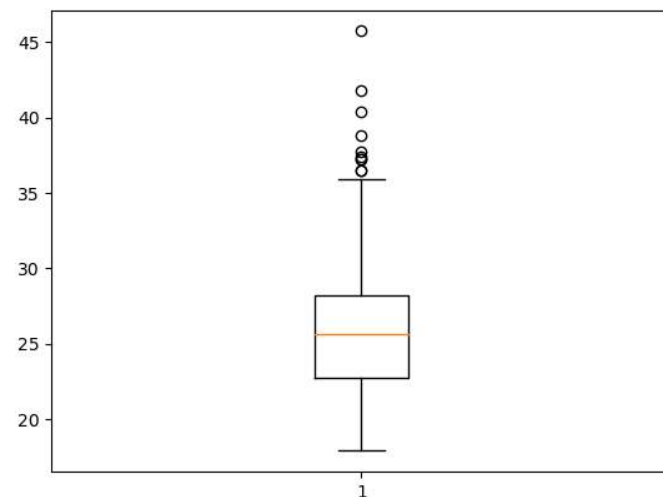


Figure 13: Boxplot for Obesity Level

Next, we have the current **alcohol consumption** of our patients. We observe that majority of patients have alcohol consumption between 0-20. And we have very few patients whose alcohol consumption is between 100-140. Same with alcohol consumption, 0 indicates that person doesn't consume alcohol.

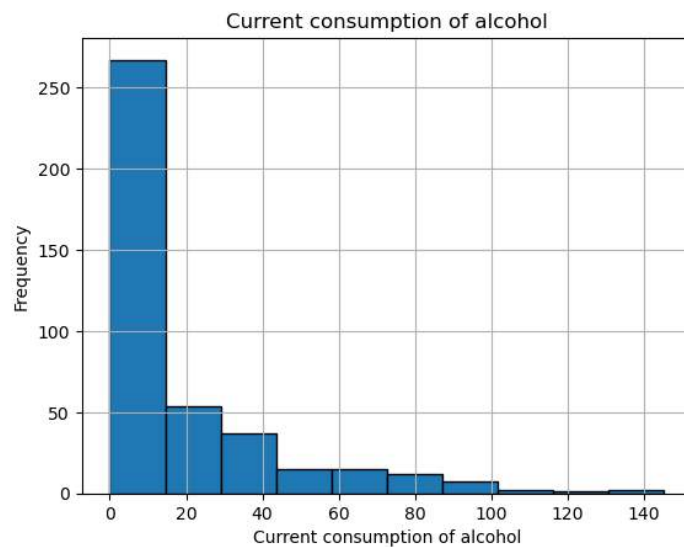


Figure 14: Alcohol Consumption by Patients

When we do descriptive analysis on current alcohol consumption levels, we get following values:

Summary Statistics- Alcohol	
Mean	18.03007282
Standard Error	1.246387792
Median	7.51
Mode	0
Standard Deviation	25.2989091
Sample Variance	640.0348017
Kurtosis	4.389231566
Skewness	2.01190868
Range	145.29
Minimum	0
Maximum	145.29
Sum	7428.39
Count	412

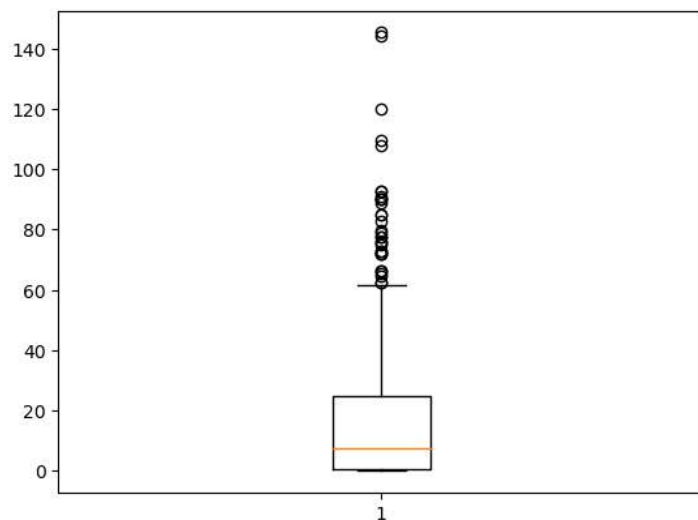


Figure 15: Box Plot for Alcohol Consumption

Next, we have a variable **age** which seems to be uniformly distributed though we have most patients whose ages are between 50-60. And we have comparatively less data for patients whose ages are between 20-30.

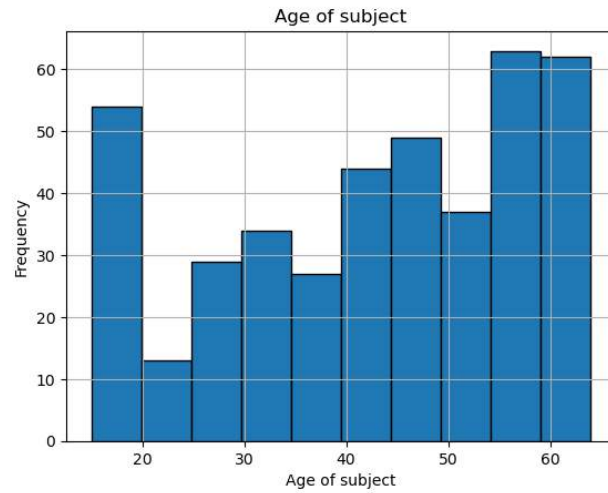


Figure 16: Patient Age Histogram

When we do descriptive analysis on age, we get following values:

Summary Statistics- Age	
Mean	42.6868932
Standard Error	0.74536897
Median	45
Mode	16
Standard Deviation	15.12933771
Sample Variance	228.8968594
Kurtosis	-1.086547324
Skewness	-0.377969109
Range	49
Minimum	15
Maximum	64
Sum	17587
Count	412

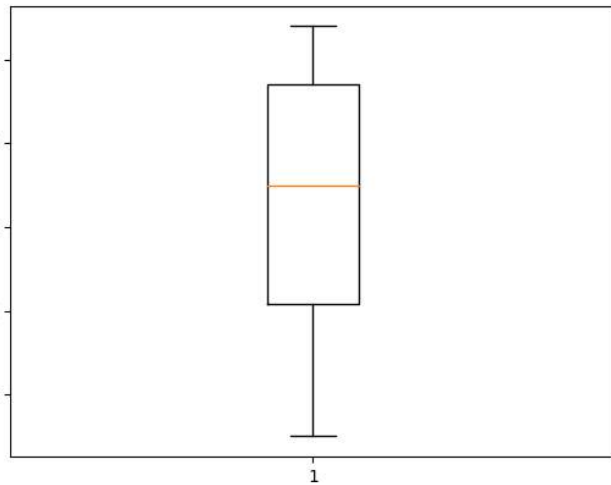


Figure 17: Box Plot for Patient's Age

Other Insights:

a) SBP by age (Hue/color by CHD):

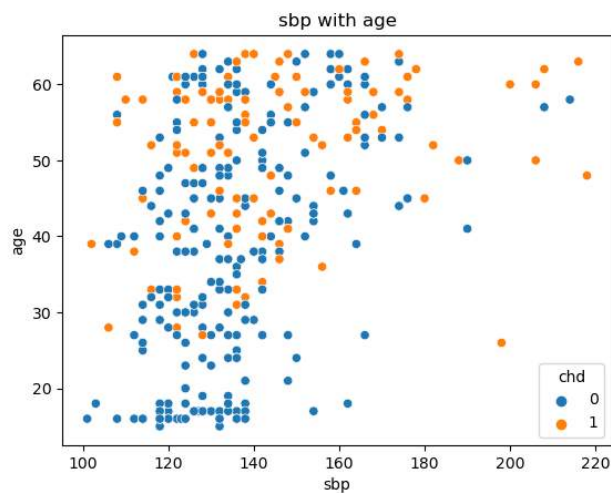


Figure 18: SBP by age by CHD

We can also make such graphs to understand the relationship in the various variables together. Here it looks like many people having disease lies between sbp of 120-160 and age of 30+.

b) LDL by Obesity (Color/Hue by CHD):

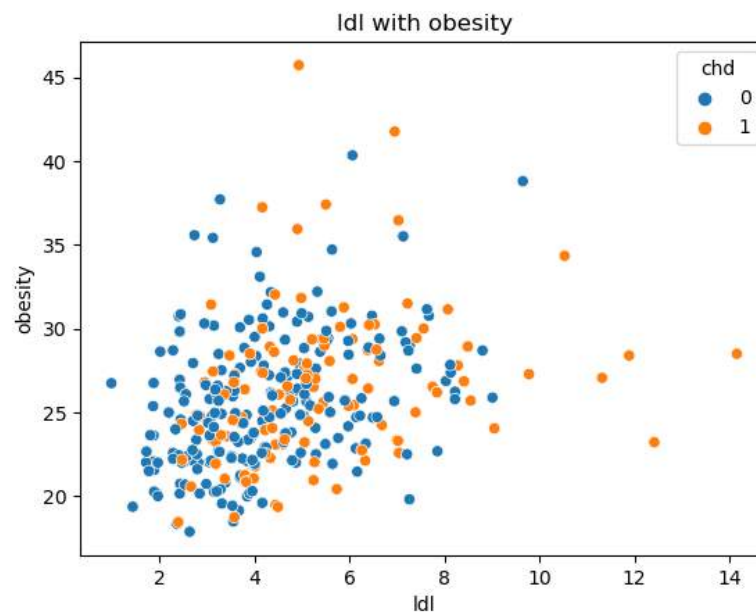


Figure 19: LDL by Obesity by CHD

This graph suggests that maximum of patients which has the disease lies between LDL level of 2 to 8, while having the obesity level of 20 to 35.

3. Find the number of null values for each column.

Now we check for missing values within each column of our dataset, and we observe that there is no missing value. There is value zero in tobacco consumption and alcohol consumption but that's not a missing value it must means that for these patients' consumption of alcohol and tobacco might be zero.

```
In [69]: missing_values = {}

# Iterate through columns
for column in df1.columns:
    missing_count = 0
    # Iterate through rows in the column
    for value in df1[column]:
        if pd.isna(value):
            missing_count += 1
    missing_values[column] = missing_count

# Print the missing values for each column
print("Missing values in the CSV file:")
for column, count in missing_values.items():
    print(f"{column}: {count}")

else:
    print('no missing value')

Missing values in the CSV file:
sbp: 0
tobacco: 0
ldl: 0
adiposity: 0
famhist: 0
typea: 0
obesity: 0
alcohol: 0
age: 0
chd: 0
no missing value
```

4. Know about the patients

A) Oldest Patient

From our dataset we observe that patient with maximum age is of **64 years old**.

B) Youngest Patient

From our dataset we can conclude that patient with minimum age is of **15 years old**.

C) Average age group

From our dataset we can conclude that average age of patients is about **42.7 years**.

D) Median Age

Median age of the patients is **45 years**.

E) Relationship between the deaths and ages

Here I have depicted the relationship between age and chd in the form of **contingency table** in which we can see that among all the ages how many patients have coronary heart disease and how many does not.

Count of chd	Column Labels																																																																
Row Labels	15	16	17	18	19	20	21	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	Grand Total															
0	2	22	18	8	2	3	2	2	6	2	3	13	3	4	6	5	4	8	3	1	1	5	9	4	6	5	7	7	5	10	5	3	7	9	4	3	1	5	3	5	3	6	2	7	12	9	5	4	6	275															
1			2								1	1	2		1	1	2	3	1		1	2	1	3	3	5	3	3	7	3	2	3	3	5	5	5	5	3	11	2	1	13	13	3	8	4	5	6	137																
Grand Total	2	22	20	8	2	3	2	2	6	2	4	14	5	4	7	6	6	11	4	1	2	7	10	7	9	10	10	5	17	8	9	12	7	8	6	10	6	16	5	7	15	20	15	17	9	12	412																		

Figure 20: Count of Patients by age by CHD

Also, I have made a stacked bar graph in which we can clearly see that for what different ages we have coronary heart disease yes or no.

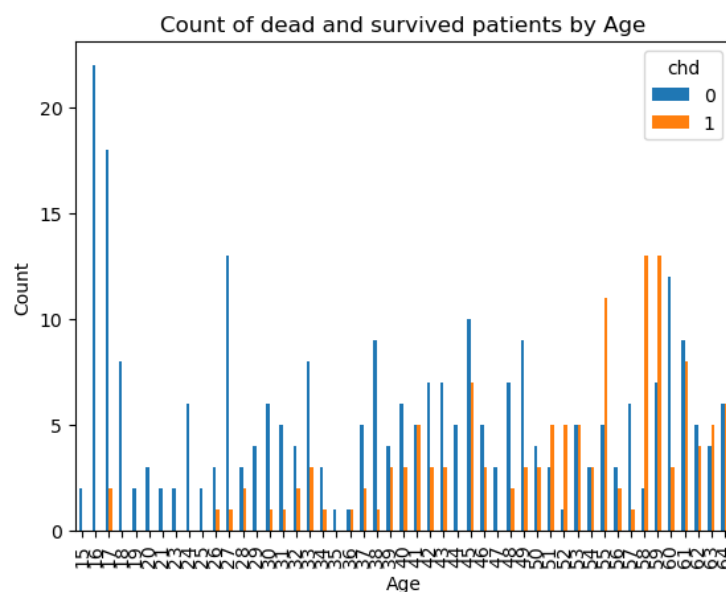


Figure 21: Patients count by Age with CHD Status

F) Age groups whose survival rate is the largest

Survival rate is maximum for those age groups which have no patients with coronary heart disease in that age. From our dataset we can see that age groups with 100% survival rate are 15, 16, 18, 19, 20, 21, 23, 24, 25, 29, 35, 44, 47.

Age group with lowest survival rate is 58 years which have a survival rate of 13.33%.

[illegible][illegible][illegible]

G) Relationship between the deaths and blood pressure

Here I have depicted the relationship between chd and sbp in which we can see that among all the sbp groups how many patients have conorary heart disease and how many did not.

Survival rate is maximum for those patients which have blood pressure reading in these values. From our dataset we can see these blood pressure readings have 100% survival rate are 103, 109, 120, 121, 123, 127, 129, 137, 161, 190, 214.

Blood Pressure readings with lowest survival rate are 102, 110, 145, 156, 168, 178, 180, 182, 188, 198, 200, 206, 216, 218 which have a survival rate of 0%.

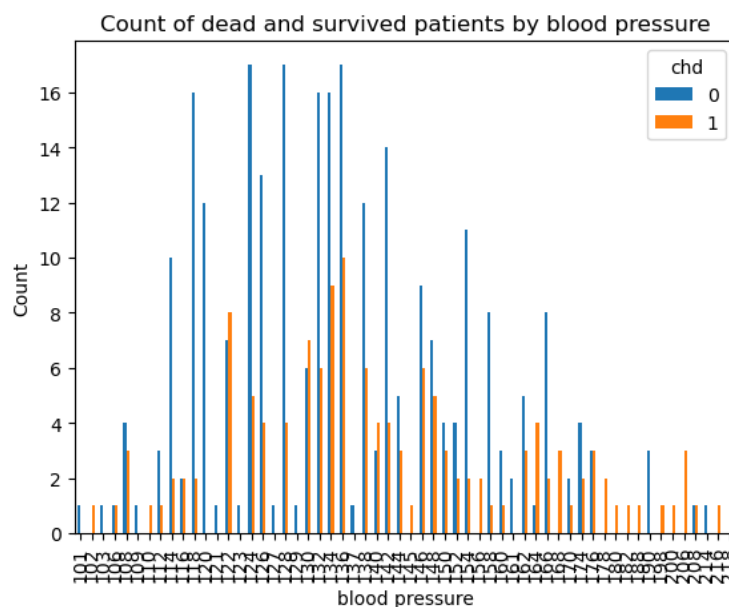


Figure 22: Patients count by SBP by CHD Status

[illegible]

126	127	128	129	130	132	134	136	137	138	140	142	144	145	146	148	150	152	154	156	158	160
76.47%	100.00%	80.95%	100.00%	46.15%	72.73%	64.00%	62.96%	100.00%	66.67%	42.86%	77.78%	62.50%	0.00%	60.00%	58.33%	57.14%	66.67%	84.62%	0.00%	88.89%	75.00%
23.53%	0.00%	19.05%	0.00%	53.85%	27.27%	36.00%	37.04%	0.00%	33.33%	57.14%	22.22%	37.50%	100.00%	40.00%	41.67%	42.86%	33.33%	15.38%	100.00%	11.11%	25.00%
100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

161	162	164	166	168	170	174	176	178	180	182	188	190	198	200	206	208	214	216	218	Grand Total
100.00%	62.50%	20.00%	80.00%	0.00%	66.67%	66.67%	50.00%	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%	50.00%	100.00%	0.00%	0.00%	66.75%
0.00%	37.50%	80.00%	20.00%	100.00%	33.33%	33.33%	50.00%	100.00%	100.00%	100.00%	100.00%	0.00%	100.00%	100.00%	100.00%	50.00%	0.00%	100.00%	100.00%	33.25%
100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

H) Relationship between the deaths and typea test score:

Survival rate is maximum i.e., 100% for those patients which have typea test scores of 25, 26, 29, 30, 32, 34, 39,40 which have no patients with coronary heart disease in these TypeEA test scores.

Typea test scores with lowest survival rate are the patients which have coronary heart disease they are 28, 69, 71 which have a survival rate of 0%.

Count of chd	Column Labels	20	25	26	28	29	30	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
0		0.00%	100.00%	100.00%	0.00%	100.00%	100.00%	100.00%	80.00%	100.00%	83.33%	66.67%	66.67%	40.00%	100.00%	100.00%	87.50%	77.78%	33.33%	25.00%	61.54%	66.67%	60.00%	78.57%
1		100.00%	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%	20.00%	0.00%	16.67%	33.33%	33.33%	60.00%	0.00%	0.00%	12.50%	22.22%	66.67%	75.00%	38.46%	33.33%	40.00%	21.43%
Grand Total		100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	Grand Total
62.50%	77.78%	80.00%	62.96%	61.54%	71.43%	60.00%	70.00%	86.36%	71.43%	50.00%	61.54%	91.67%	75.00%	44.44%	50.00%	76.92%	57.14%	50.00%	75.00%	0.00%	66.67%	0.00%	20.00%	50.00%	66.75%
37.50%	22.22%	20.00%	37.04%	38.46%	28.57%	40.00%	30.00%	13.64%	28.57%	50.00%	38.46%	8.33%	25.00%	55.56%	50.00%	23.08%	42.86%	50.00%	25.00%	100.00%	33.33%	100.00%	80.00%	50.00%	33.25%
100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

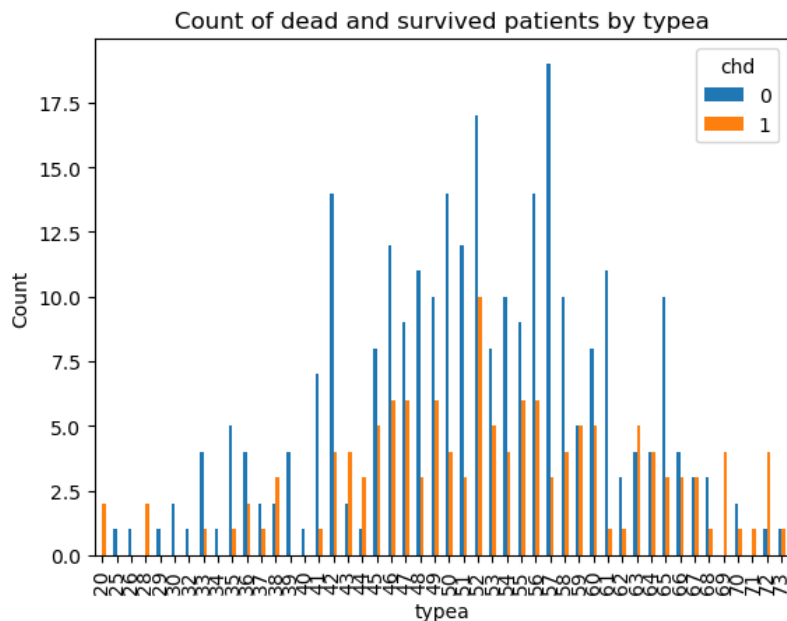


Figure 23: Patients Count by typeEA vs CHD status

I) Relationship between the deaths and family history:

74.90% of patients which have no family history for coronary heart disease i.e., for these patients' family history is absent, have no coronary heart disease.

For patients which have no family history for coronary heart disease have 25.10% patients which have coronary heart disease.

For patients which **have prior history of coronary heart disease** in family have **55.49%** patients which get a coronary heart disease themselves and 44.51% patients which did not a coronary heart disease.

Count of chd	Column Labels		
Row Labels	Absent	Present	Grand Total
0	74.90%	55.49%	66.75%
1	25.10%	44.51%	33.25%
Grand Total	100.00%	100.00%	100.00%

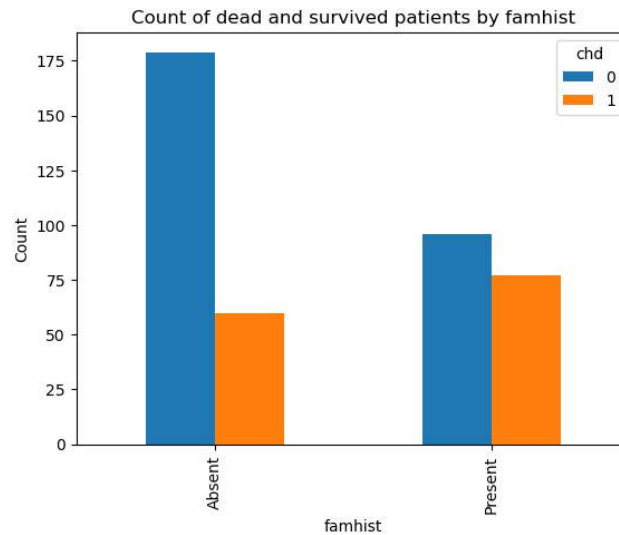


Figure 24: CHD by Family History

H) Visuals on data distributions

i) plot Correlation Matrix

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.

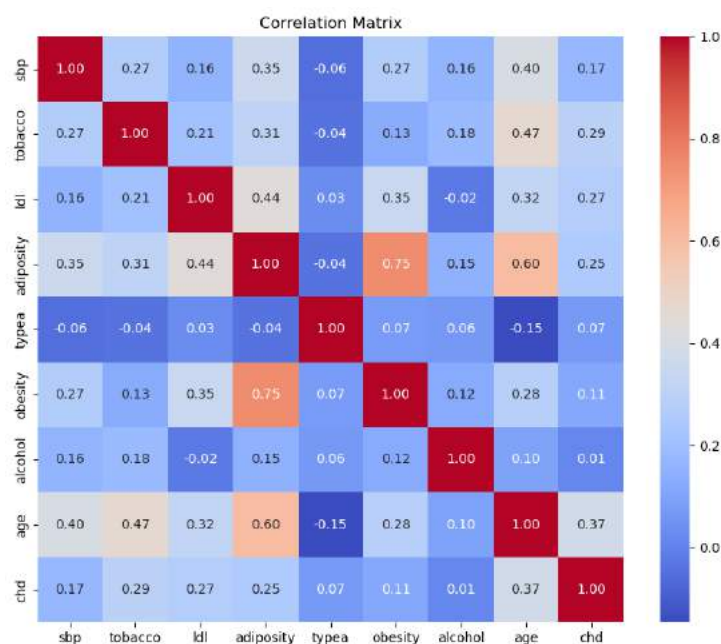


Figure 25: Correlation matrix for all variables

From the correlation matrix above we can make the following observations:

- *obesity* has a strong positive correlation with *adiposity*.
- *sbp* & *typea* has a negative correlation.
- *age* & *adiposity* has a positive correlation.
- *age* & *tobacco* has a positive correlation.
- *age* & *typea* has a negative correlation.
- *idl* & *adiposity* has a positive correlation.

ii) Plot Scatter Matrix:

Scatter plots shows how much one variable is affected by another or the relationship between them with the help of dots in two dimensions. Scatter plots are very much like line graphs in the concept that they use horizontal and vertical axes to plot data points. A scatter plot matrix is a grid (or matrix) of scatter plots used to visualize bivariate relationships between combinations of variables. Each scatter plot in the matrix visualizes the relationship between a pair of variables, allowing many relationships to be explored in one chart.

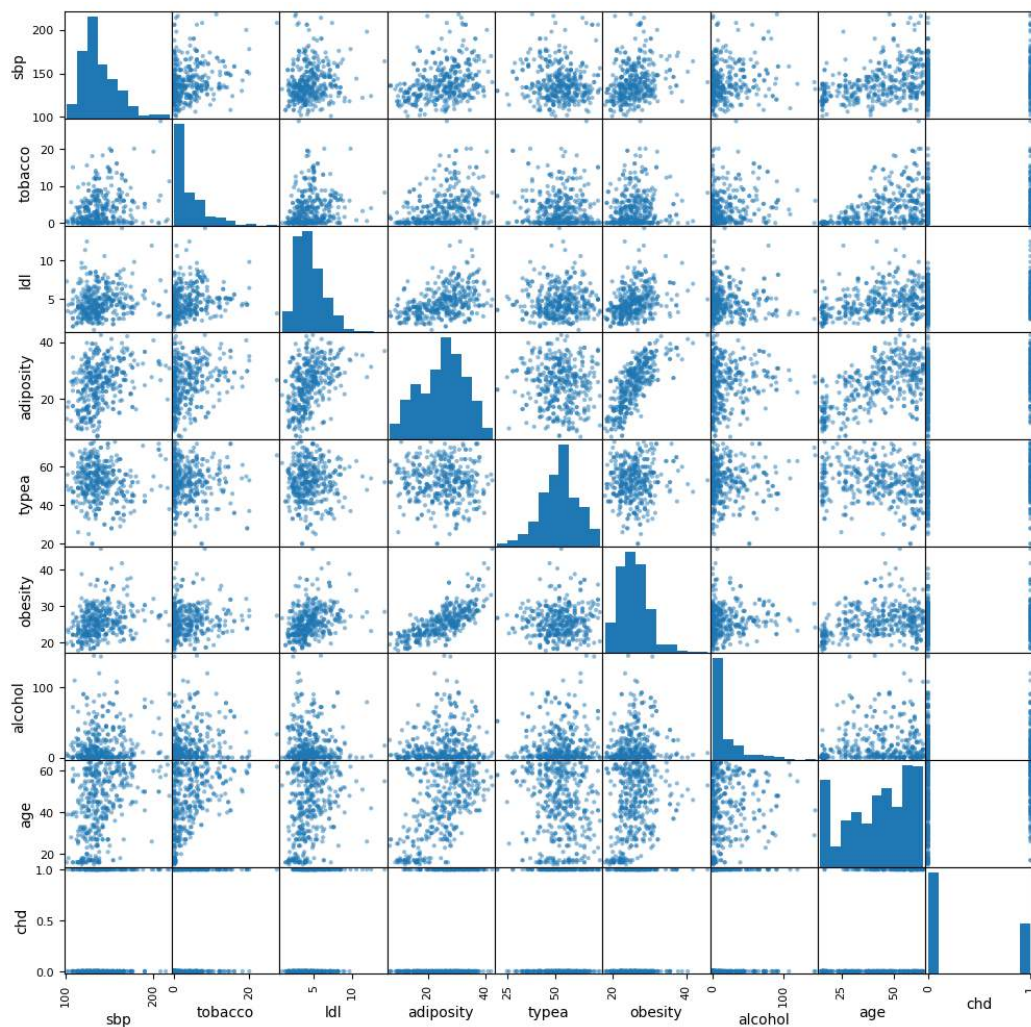


Figure 26: Scatter Matrix for all variables

iii) Plot Per Column Distribution

Column distribution shows the frequency or count of patients in various variables in one place.

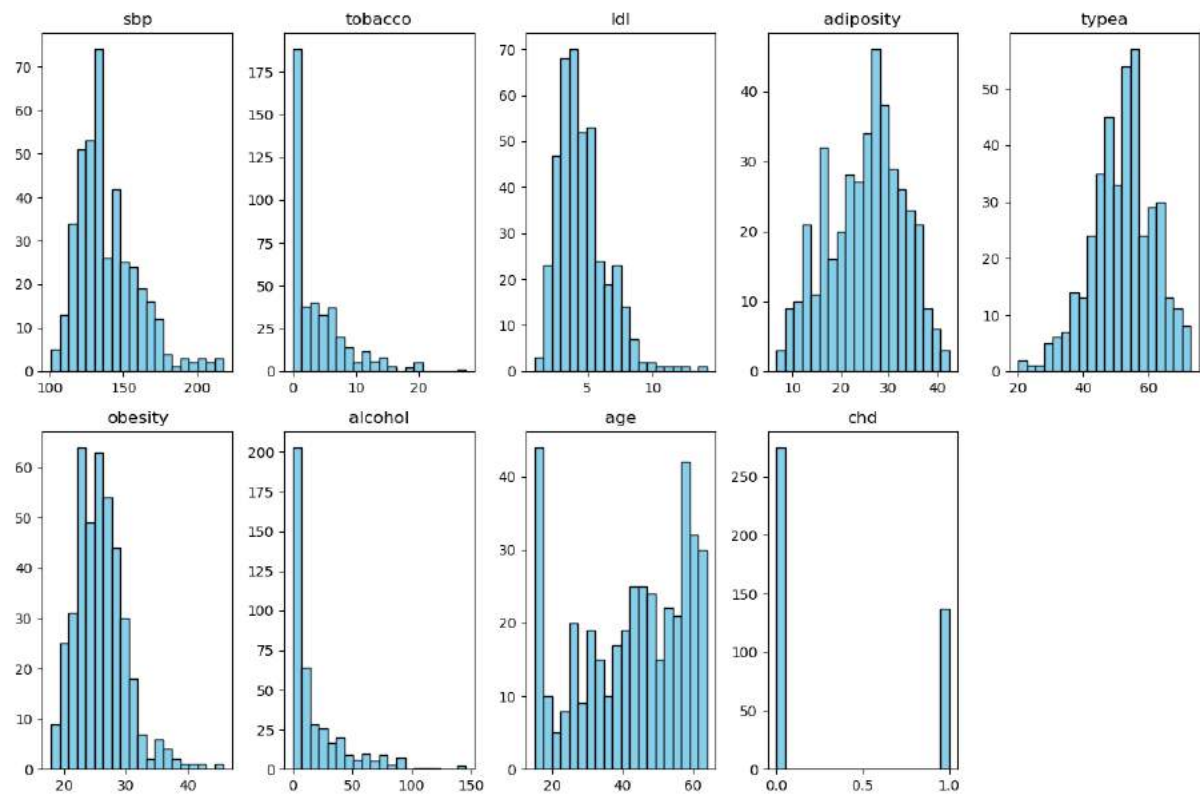


Figure 27: Column Distribution for all variables

Missing Values if Exist.

1. There are no missing values in our dataset.
2. Since there are no missing values in our dataset. Count of missing values in each column is zero.

```
In [69]: missing_values = {}

# Iterate through columns
for column in df1.columns:
    missing_count = 0
    # Iterate through rows in the column
    for value in df1[column]:
        if pd.isna(value):
            missing_count += 1
    missing_values[column] = missing_count

# Print the missing values for each column
print("Missing values in the CSV file:")
for column, count in missing_values.items():
    print(f"{column}: {count}")

else:
    print('no missing value')

Missing values in the CSV file:
sbp: 0
tobacco: 0
ldl: 0
adiposity: 0
fahist: 0
typea: 0
obesity: 0
alcohol: 0
age: 0
chd: 0
no missing value
```

3. Since, there are no missing values in our dataset, creating a heatmap to visualize missing values is not necessary, as there are no missing values to visualize. Heatmaps are typically used to highlight and visualize the presence of missing data. In the absence of missing values, a heatmap would simply show a uniform pattern, which does not provide any meaningful information.

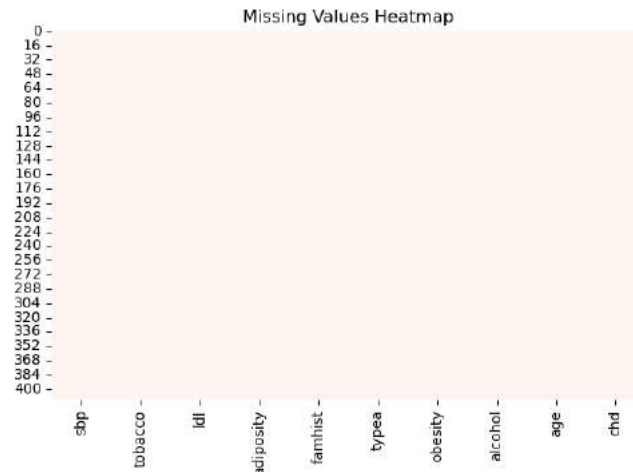


Figure 28: Heatmap for missing values

J) Handling Missing Values

In our dataset, we have thoroughly checked for missing values and found that there are no missing values present. Consequently, we did not employ any specific techniques to handle missing values, such as imputation or interpolation. Our dataset is complete in this regard, which is beneficial for our analysis as we have full data for all variables without the need for additional processing to address missing values.

If we had to, there are multiple ways of handling missing values. For example:

- Imputation: Imputing average value in NAN cells, calculated from the non-missing values
- Imputing average of previous and next values: We could have added average value of previous and next cell for each missing cell.

K) Feature Scaling

For the scaling purpose, there are various techniques possible to use. We will explore some of those, like

- StandardScaler
- MinMaxScaler

StandardScaler:

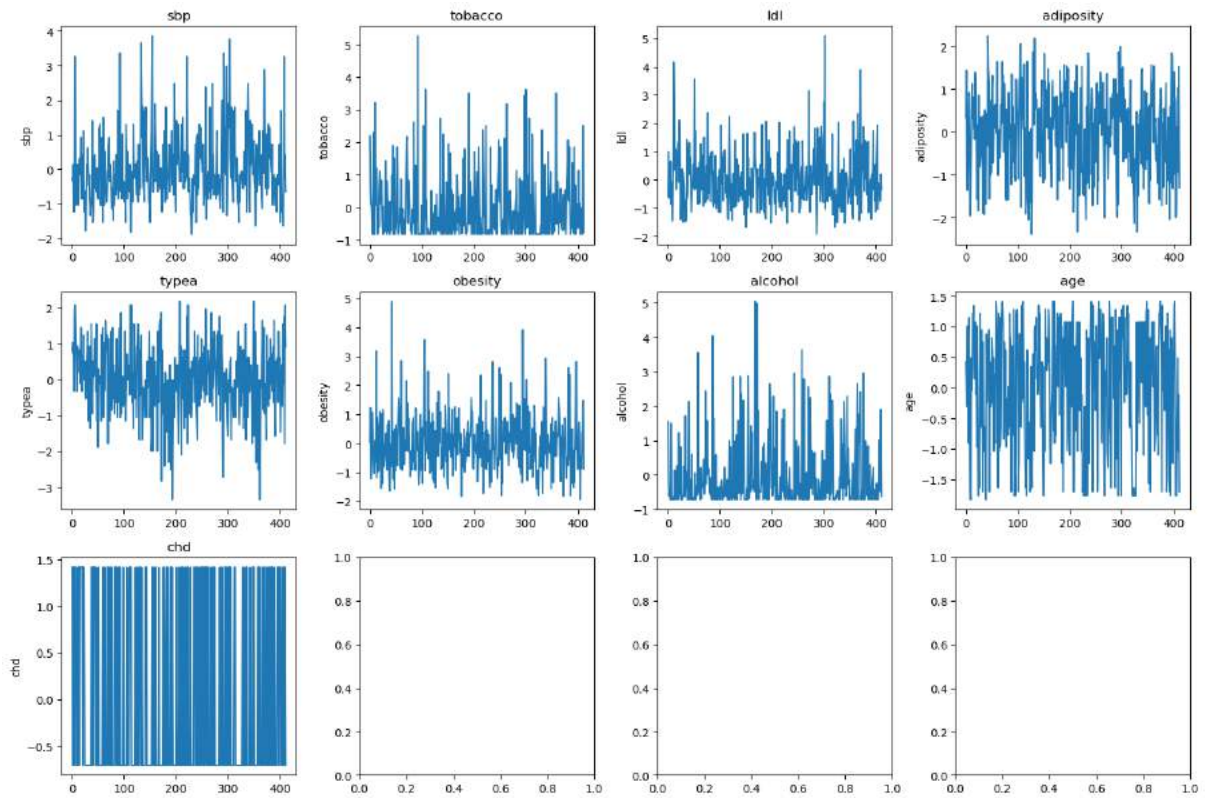


Figure 29: Standard Scaling for all variables

MinMax Scaler:

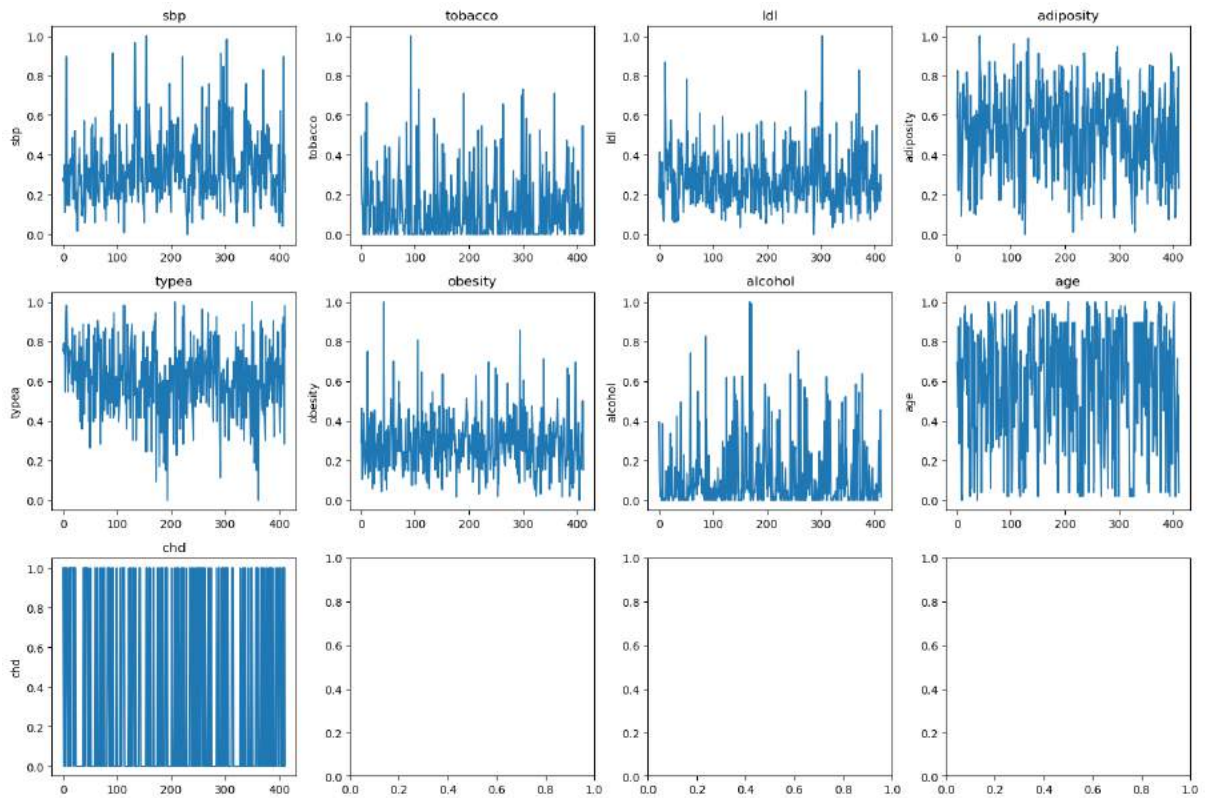


Figure 30: MinMax Scaling for all variables.

L) Model Fitting: Logistic Regression

Now we will fit a classification model on the heart-train data, where we will use the 9 of the variables as features and CHD as the response variable.

The idea is to build a model to predict the CHD response based on the patients' characteristics.

For starting, we are using whole heart-train data to train the model and then we will test the model responses on the train data itself. Although this should not be an ideal setup, because this will create bias.

We should always keep our train and test data separate.

For this exercise, we are fitting and testing the model on the same data. (heart-train),

```
X = Train[['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesity', 'alcohol', 'age']]
Y = Train['chd']

#x = Test[['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesity', 'alcohol', 'age']]

logistic_regression = LogisticRegression()
logistic_regression.fit(X,Y)
print("Train: ", logistic_regression.score(X, Y))
print("Test: ", logistic_regression.score(X, Y))
print("MSE:", mean_squared_error(logistic_regression.predict(X), Y, squared=True))
pred_lreg = logistic_regression.predict(X)
print(pred_lreg)
```

The predictions on the fitted model are,

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

The accuracy of these predictions with actual model CHD values can be calculated, which is 72.3 % in our case.

```
accuracy_a = accuracy_score(Y, pred_lreg)
print("Accuracy:", accuracy_a)
```

Accuracy: 0.7233009708737864

Now we can also use this fitted model to do predictions on the test data.

```
# Define X and Y for training and test data
X_train = Train[['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesity', 'alcohol', 'age']]
Y_train = Train['chd']

X_test = Test[['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesity', 'alcohol', 'age']]
# Y_test can be defined if you have true outcomes in your test data, otherwise you won't have it

# Create and fit the logistic regression model
logistic_regression = LogisticRegression()
logistic_regression.fit(X_train, Y_train)

# Make predictions on the test data (if you have Y_test)
if 'chd' in Test:
    Y_test = Test['chd']
    predictions = logistic_regression.predict(X_test)
    # Evaluate the model's performance using appropriate metrics
else:
    predictions = None # If you don't have true outcomes in your test data
```

The predicted responses are:

```
print(predictions)

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

Here since we do not have the test data CHD responses already given. We cannot calculate the accuracy of the fitted model.

Now for the ideal scenario, it is usually best to keep the test and train data separate. For this, we can use our heart-train data and divide it into two groups.

- a) Train data- randomly selected 80% rows from heart-train.
- b) Test data- randomly selected 20% rows from heart-train.

Using this train data, we can fit the model again and can do the testing on this new test data. As we have the CHD values given in for the test data, we will be able to calculate the accuracy of the predicted responses.

```
# Loading complete dataset into a DataFrame
data = pd.read_csv('heart-train.csv')

# Defining X and Y for the entire dataset
X = data[['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesity', 'alcohol', 'age']]
Y = data['chd']

# Splitting the dataset into a training set and a test set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Creating and fitting the logistic regression model
logistic_regression = LogisticRegression()
logistic_regression.fit(X_train, Y_train)

# Making predictions on the test set
Y_pred = logistic_regression.predict(X_test)

# Evaluating the model's accuracy
accuracy = accuracy_score(Y_test, Y_pred)
print("Accuracy:", accuracy)
```

Accuracy: 0.6506024096385542

Now we can make confusion matrix as well for the last case, which tells us the False positives and false negatives. We can relate this to Type 1 and Type 2 errors of the hypothesis testing as well.

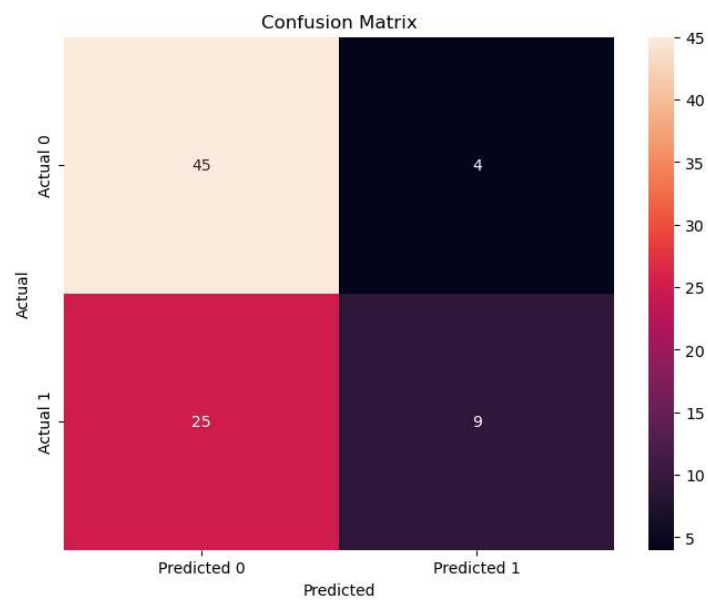


Figure 31: Confusion Matrix

DATA220- LAB 1- PART 2

Contents

Dataset Overview	2
1. Meta data:	2
2. Variable Type	3
3. Insights about the data:	4
4. Missing Values/Null Values Handling	5
5. Summary Statistics for numerical values	6
5.1. Price:.....	6
5.2. Bedrooms:	6
5.3. Bathrooms	6
5.4. sqft_living.....	7
5.5. sqft_lot.....	7
5.6. floors.....	8
5.7. waterfront	8
5.8. view	9
5.9. condition.....	9
5.10. sqft_above.....	9
5.11. sqft_basement	10
5.12. yr_built.....	10
5.13. yr_renovated	11
6. Distribution for Price	11
6.1. Price Histogram:.....	11
7. Observations about Probability Distribution	12
8. Shapiro-Wilk test:.....	12
9. Hypothesis Testing using t-test.....	13
10. Hypothesis test using ANOVA.....	14
11. ANOVA vs t-test Comparison	15
12. Covariance matrix of the numerical features	16
Positive covariances.....	17
Negative covariances	17
13. Heatmap of the covariance matrix	18
13.1. Color Significance:	19
14. Eigen Value, Eigen Vector and Rank of the Covariance Matrix	20
14.1. Eigen values of covariance matrix:	20

14.2.	Eigen Vector	21
14.3.	Rank of Covariance matrix:.....	22
15.	Eigen Values Interpretation	22
16.	Inverse of the covariance matrix	23
17.	Impact of the matrix rank.....	24
18.	Rank and Collinearity	25
19.	Matrix X with Selected Features.....	25
20.	Transpose of Matrix X	26
21.	Solving Linear System of equations	26
22.	Scatter Plot and Regression Line	26

Dataset Overview

The dataset consists of details of properties listed in 2014 for May, June, and July in the Washington Area, USA. Here, we have categorized it into several features, and there are around 4600 properties listed. This dataset can be used to make or train our dataset to predict the price of listed properties which have the similar variables.

1. Meta data:

Each following row contains the information about one real estate properties. The first row is the name of the observed variables. There are 18 variables:

- **Date:** Date of property being listed
- **Price:** Price of real estate property. Some properties have a price of 0, indicating missing or unknown values or severely distressed conditions. These properties could be government-owned, part of a land trust, or considered severely distressed.
- **Bedrooms:** Number of bedrooms a listed property have
- **Bathrooms:** Number of bathrooms a listed property have
- **sqft_living:** carpet area of living room
- **sqft_lot:** carpet area of the lot or land area that the property sits on
- **floors:** Number of floors or levels property has
- **waterfront:** binary indicator of whether the home has a view or direct access to a waterfront. 0 = No, 1 = Yes
- **view:** View indicates the quality of the view from the home.
Some key details about the 'view' variable:
 1. It is an ordinal categorical variable that takes integer values from 0 to 4.
 2. 0 indicates no view.
 3. Higher values indicate better quality views, with 4 being the best possible view.
 4. This is a subjective rating of the aesthetic appeal of the view.
- **Condition:** The condition in your dataset represents the condition of the property listed. A lower condition value may indicate that the property is in poor condition, while a higher condition value suggests that the property is in better condition.

- **sqft_above:** sqft_above measures the total livable floor area on the levels above ground level.
- **sqft_basement:** It measures total floor area for basement
- **yr_built:** year in which listed property is built. Some values may be 0, indicating missing details about the year built.
- **yr_renovated:** year in which if listed property is renovated. Some values may be 0, indicating no renovations were performed.
- **street:** street address of listed property
- **city:** city in which listed property is located
- **statezip:** State and zipcode of given listed property
- **country:** country in which given listed property is located

2. Variable Type

Based on the provided dataset, we can categorize the columns into different data types:

Nominal:

- street
- city
- date
- statezip
- country

Continuous Columns:

- Price
- Bedrooms
- Bathrooms
- sqft_living: Adiposity (a measure of body fat)
- sqft_lot: Type-A behavior (a personality type)
- floors
- waterfront
- view
- Condition
- sqft_above
- sqft_basement
- yr_built
- yr_renovated

We can use DF feature to identify variable type as well.

```
variable_types = data.dtypes

# Categorize the variables based on their types
categorical_vars = variable_types[variable_types == 'object'].index.tolist()
numerical_vars = variable_types[variable_types != 'object'].index.tolist()

print("Categorical Variables:")
print(categorical_vars)

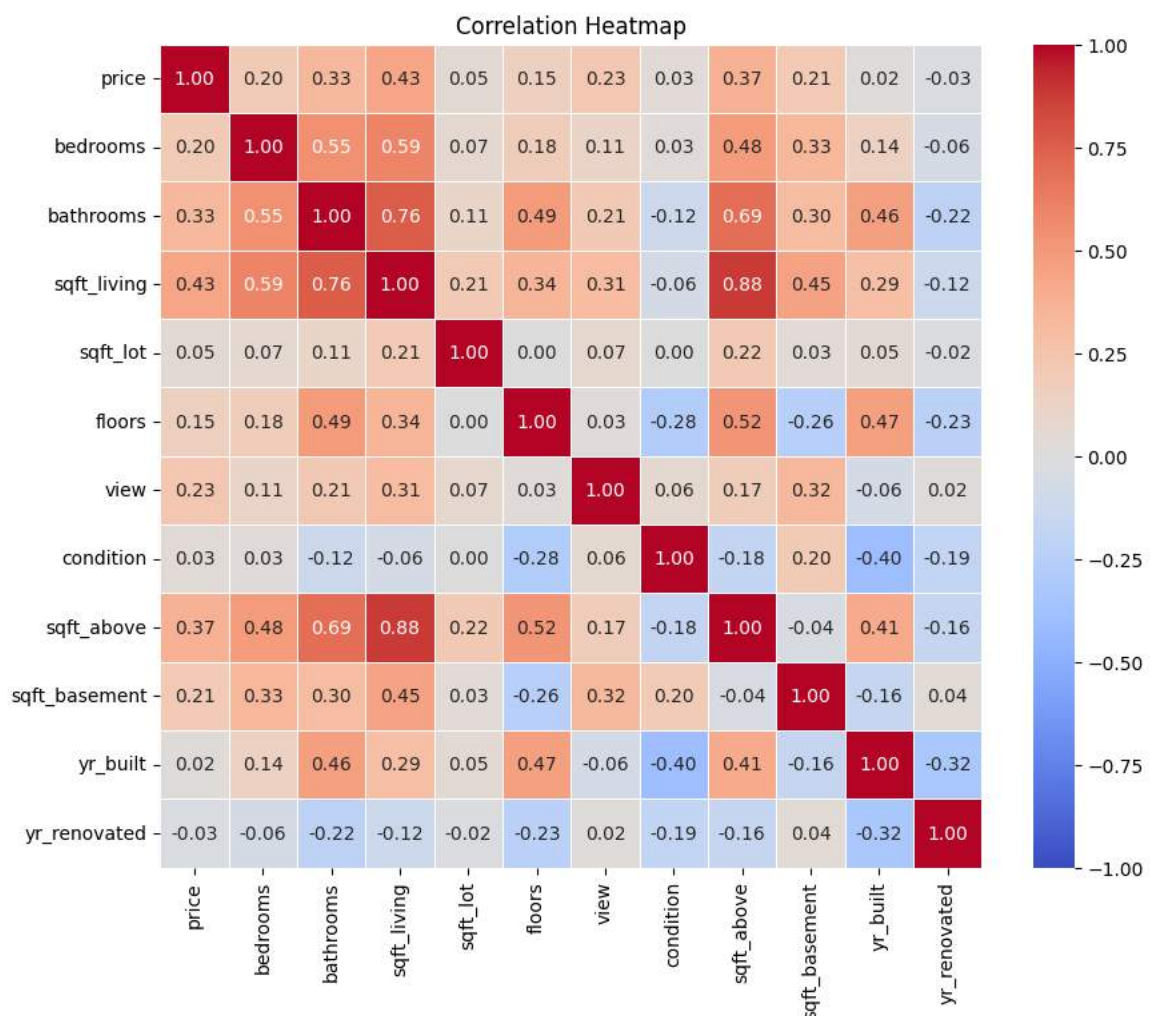
print("Numerical Variables:")
print(numerical_vars)

Categorical Variables:
['date', 'street', 'city', 'statezip', 'country']
Numerical Variables:
['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated']
```

Though date can be considered continuous column as we can perform numerical analysis on it.

3. Insights about the data:

- We have 18 columns, of which 13 are numerical and 5 are string values.
- Property conditions are fair or average; most have three bedrooms and 2.5 bathrooms.
- The heatmap reveals that the pairs of variables are positively correlated (indicated by darker shades), such as price with itself and other attributes with itself. It also shows negative correlations between floors and condition and between condition and year built.



- There are no missing values in any columns, but there are some 0 values, especially in price and year built, as mentioned above.
- The summary statistics provide insights into the distribution and characteristics of the dataset, highlighting the diversity in property prices, living area sizes, lot sizes, and the number of floors, which could impact pricing and property characteristics.
- The presence of properties with 0 bedrooms or 0 bathrooms might require special attention, as these values are unusual in real estate data.
- Most properties do not have a waterfront view, have limited or no views, and are in moderately good condition.
- The dataset includes properties with different sizes of above-ground and basement areas, and the range of years built and renovated provides information about the age and updated status of properties. However, it is crucial to investigate further the potential data quality issues, such as years with values of 0, to ensure accurate analysis.
- Additionally, the above-ground area sizes vary, with most properties having an above-ground area size below 2,300 square feet.
- The dataset includes properties that have been renovated as recently as 2014.
- A significant number of properties with zero basement space suggests that many properties still need basements.
- The year renovated might need to be handled carefully, as it could indicate that renovations were not performed if the year is set to 0.
- Further data exploration or cleaning may be necessary to deal with such cases. These insights provide a better understanding of the basement sizes, ages, and renovation status of the properties in the dataset, which can be valuable for property analysis and potential renovations.

4. Missing Values/Null Values Handling

Now we check for missing values within each column of our dataset, and we observe that there is no missing value.

```
In [40]: missing_values = {}

# Iterate through columns
for column in data.columns:
    missing_count = 0
    # Iterate through rows in the column
    for value in data[column]:
        if pd.isna(value):
            missing_count += 1
    missing_values[column] = missing_count
# Print the missing values for each column
print("Missing values in the CSV file:")
for column, count in missing_values.items():
    print(f"{column}: {count}")
else:
    print('no missing value')
```

Missing values in the CSV file:

```
date: 0
price: 0
bedrooms: 0
bathrooms: 0
sqft_living: 0
sqft_lot: 0
floors: 0
waterfront: 0
view: 0
condition: 0
sqft_above: 0
sqft_basement: 0
yr_built: 0
yr_renovated: 0
street: 0
city: 0
statezip: 0
country: 0
no missing value
```

5. Summary Statistics for numerical values

5.1. Price:

Mean	551962.988
Standard Error	8313.28915
Median	460943.462
Mode	0
Standard Deviation	563834.703
Sample Variance	3.1791E+11
Kurtosis	1044.35215
Skewness	24.7909326
25%	3.228750e+05
50%	4.609435e+05
75%	6.549625e+05
Range	26590000
Minimum	0
Maximum	26590000
Sum	2539029747
Count	4600

5.2. Bedrooms:

Mean	3.40086956521739
Standard Error	0.0134002344000819
Median	3
Mode	3
Standard Deviation	0.908848115525819
Sample Variance	0.826004897094832
Kurtosis	1.2353774286378
Skewness	0.456446633019113
25%	3.000000
50%	3.000000
75%	4.000000
Range	9
Minimum	0
Maximum	9
Sum	15644
Count	4600

5.3. Bathrooms

Mean	2.1608152173913
Standard Error	0.0115562214843625
Median	2.25
Mode	2.5

Standard Deviation	0.78378107465028
Sample Variance	0.614312772979948
Kurtosis	1.86590470973037
Skewness	0.616032723350907
25%	1.750000
50%	2.250000
75%	2.500000
Range	8
Minimum	0
Maximum	8
Sum	9939.75
Count	4600

5.4. sqft_living

Mean	2139.34695652174
Standard Error	14.2017111841707
Median	1980
Mode	1940
Standard Deviation	963.206915760864
Sample Variance	927767.562569557
Kurtosis	8.29168259963044
Skewness	1.7235132706221
25%	1460.000000
50%	1980.000000
75%	2620.000000
Range	13170
Minimum	370
Maximum	13540
Sum	9840996
Count	4600

5.5. sqft_lot

Mean	14852.5160869565
Standard Error	529.087146070623
Median	7683
Mode	5000
Standard Deviation	35884.4361448097
Sample Variance	1287692757.43092
Kurtosis	219.872987410635
Skewness	11.3071387487827
25%	5.000750e+03
50%	7.683000e+03
75%	1.100125e+04

Range	1073580
Minimum	638
Maximum	1074218
Sum	68321574
Count	4600

5.6. floors

Mean	1.5120652173913
Standard Error	0.00793662913241133
Median	1.5
Mode	1
Standard Deviation	0.538288377296989
Sample Variance	0.289754377133025
Kurtosis	-0.538851979546012
Skewness	0.551440646348873
25%	1.000000
50%	1.500000
75%	2.000000
Range	2.5
Minimum	1
Maximum	3.5
Sum	6955.5
Count	4600

5.7. waterfront

Mean	0.00717391304347826
Standard Error	0.00124446572350127
Median	0
Mode	0
Standard Deviation	0.0844037718947432
Sample Variance	0.00712399671005984
Kurtosis	134.548673262581
Skewness	11.682900924984
25%	0.000000
50%	0.000000
75%	0.000000
Range	1
Minimum	0
Maximum	1
Sum	33
Count	4600

5.8. view

Mean	0.240652173913043
Standard Error	0.0114769514186014
Median	0
Mode	0
Standard Deviation	0.778404717212521
Sample Variance	0.605913903778704
Kurtosis	10.4641779184751
Skewness	3.34158638067357
25%	0.000000
50%	0.000000
75%	0.000000
Range	4
Minimum	0
Maximum	4
Sum	1107
Count	4600

5.9. condition

Mean	3.45173913043478
Standard Error	0.00998520816952657
Median	3
Mode	3
Standard Deviation	0.677229767559274
Sample Variance	0.458640158068389
Kurtosis	0.197730205050629
Skewness	0.95906766350085
25%	3.000000
50%	3.000000
75%	4.000000
Range	4
Minimum	1
Maximum	5
Sum	15878
Count	4600

5.10. sqft_above

Mean	1827.26543478261
Standard Error	12.7119880499432
Median	1590
Mode	1200
Standard Deviation	862.168976962598

Sample Variance	743335.344836732
Kurtosis	4.07013826471901
Skewness	1.49421074798294
25%	1190.000000
50%	1590.000000
75%	2300.000000
Range	9040
Minimum	370
Maximum	9410
Sum	8405421
Count	4600

5.11. sqft_basement

Mean	312.08152173913
Standard Error	6.84333008304521
Median	0
Mode	0
Standard Deviation	464.137228066607
Sample Variance	215423.366477353
Kurtosis	4.08238002413812
Skewness	1.64273219221671
25%	0.000000
50%	0.000000
75%	610.000000
Range	4820
Minimum	0
Maximum	4820
Sum	1435575
Count	4600

5.12. yr_built

Mean	1970.78630434783
Standard Error	0.438372188673713
Median	1976
Mode	2006
Standard Deviation	29.7318483900997
Sample Variance	883.982808691876
Kurtosis	-0.670075900398819
Skewness	-0.50215518998789
25%	1951.000000
50%	1976.000000
75%	1997.000000
Range	114

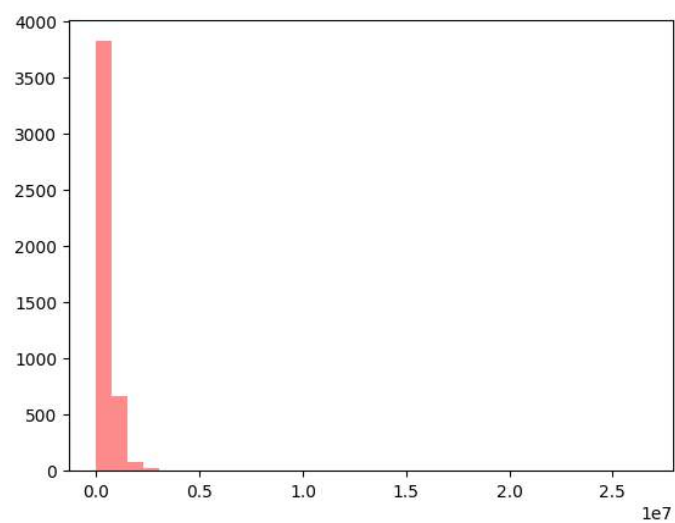
Minimum	1900
Maximum	2014
Sum	9065617
Count	4600

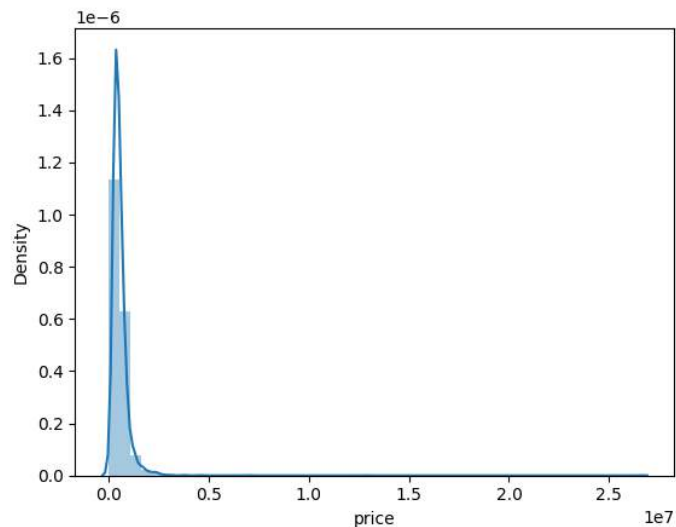
5.13. yr_renovated

Mean	808.608260869565
Standard Error	14.4406795133468
Median	0
Mode	0
Standard Deviation	979.414536400745
Sample Variance	959252.834113087
Kurtosis	-1.85111091339415
Skewness	0.385918700882929
25%	0.000000
50%	0.000000
75%	1999.000000
Range	2014
Minimum	0
Maximum	2014
Sum	3719598
Count	4600

6. Distribution for Price

6.1. Price Histogram:





The histogram reveals that the distribution of property prices is right skewed, meaning that most properties have lower prices, and there are relatively fewer properties with extremely high prices. According to the skewness, the market is characterized by a broader range of property values. The mean price is higher than the median, indicating the influence of higher-priced properties on the distribution. Data quality issues are evident in properties with a recorded price of \$0, which requires further investigation.

7. Observations about Probability Distribution

The histogram being right skewed indicates that the data is not normally distributed. In a right-skewed distribution, the tail of the distribution is extended to the right, which means there are more data points with lower values and relatively fewer data points with higher values. The histogram of the "price" feature does not resemble a bell curve or a normal distribution. It exhibits right-skewness, suggesting that the dataset is characterized by a broader range of property prices with a higher concentration of lower-priced properties.

8. Shapiro-Wilk test:

Shapiro-Wilk test is a hypothesis test that evaluates whether a data set is normally distributed. It evaluates data from a sample with the null hypothesis that the data set is normally distributed. A large p-value indicates the data set is normally distributed, a low p-value indicates that it isn't normally distributed.

For our dataset we have if the p-value is greater than 0.05 then the feature follows the Normal distribution. So, we observe that when we apply Shapiro Wilk test, we get p-value of 0.00 which is less than 0.05 so our distribution doesn't follow Normal distribution for variable "price".

```
In [13]: # Assuming 'data' is your DataFrame with the 'price' column
price_data = data['price']

# Perform the Shapiro-Wilk test
statistic, p_value = stats.shapiro(price_data)

# Set the significance level (alpha)
alpha = 0.05

# Check if the p-value is greater than alpha
if p_value > alpha:
    print("The 'price' feature follows a normal distribution (p-value =", p_value, ")")
else:
    print("The 'price' feature does not follow a normal distribution (p-value =", p_value, ")")

The 'price' feature does not follow a normal distribution (p-value = 0.0 )
```

This also suggests that the distribution is not normal.

9. Hypothesis Testing using t-test

Hypothesis test using a t-test: split into two groups: properties built before 1990 and those built-in or after 1990.

```
data_before_1990 = data[data['yr_built'] < 1990]['price']
data_after_1990 = data[data['yr_built'] >= 1990]['price']
```

We use t-test to determine whether there is a significant difference between the means of two groups. In our case, we performed a t-test to investigate the impact of the year built (specifically, whether a property was built before or after 1990) on the sale price of properties.

Here's what the t-test results tells us:

Null Hypothesis (H0): The null hypothesis in this test is that there is no significant difference in the means of the sale prices between properties built before 1990 and properties built in or after 1990. In other words, the year built has no impact on the sale price.

Alternative Hypothesis (H1): The alternative hypothesis is that there is a significant difference in the means of the sale prices between the two groups. It suggests that the year built does have an impact on the sale price.

```
In [14]: data_before_1990 = data[data['yr_built'] < 1990]['price']
data_after_1990 = data[data['yr_built'] >= 1990]['price']
t_statistic, p_value = ttest_ind(data_before_1990, data_after_1990)

# Print the results
print("Independent t-test results:")
print("T-statistic:", t_statistic)
print("P-value:", p_value)

# Interpret the results based on the p-value and alpha level
alpha = 0.05
if p_value > alpha:
    print("Cannot reject null hypothesis (H0 wins): There is no significant impact of year built on sale price.")
else:
    print("Cannot accept H0 (H1 wins): The year built has a significant impact on sale price.")

Independent t-test results:
T-statistic: -5.014550457612197
P-value: 5.51579911487353e-07
Cannot accept H0 (H1 wins): The year built has a significant impact on sale price.
```

Understanding results:

Data before 1990 have 3094 properties listed in it and mean of prices in this group is 522954.83.

Data after 1990 have 1506 properties listed in it and mean of prices in this group is 611558.76.

Here we are using independent t-test (also known as Student's t-test) to compare the means of two independent groups or samples. It assumes that the two samples are independent of each other.

We observe that p_value is less than α which is 0.05 for our dataset, therefore we can conclude that **we cannot accept null hypothesis**, that's why we accept alternate hypothesis which tells us that year built has a significant impact on sale price of property.

10. Hypothesis test using ANOVA.

Assume that you have 3 groups: groupA has all houses built in 1990, groupB has all houses that were built in 2000 and groupC has all houses built in 2010.

We use ANOVA one way test to determine whether there is a significant difference between the means of three groups. In our case, we performed an ANOVA test to investigate the impact of the year built on the sale price of properties.

Null Hypothesis (H0): The null hypothesis in this context states that the means of sale prices for properties built in 1990, 2000, and 2010 are equal. In other words, the year built on the sale price has no significant impact.

Alternative Hypothesis (H1): The alternative hypothesis suggests that at least one means of sale prices for the three groups is different. This implies that the year built does have a significant impact on the sale price.

```
In [15]: # Assuming 'data' is your DataFrame with 'year_built' and 'sale_price' columns
groupA = data[data['yr_built'] == 1990]['price']
groupB = data[data['yr_built'] == 2000]['price']
groupC = data[data['yr_built'] == 2010]['price']

# Perform ANOVA test
f_statistic, p_value = stats.f_oneway(groupA, groupB, groupC)
print("ANOVA test results:")
print("f-statistic:", f_statistic)
print("P-value:", p_value)

# Set the significance level (alpha)
alpha = 0.05 # 95% confidence interval

# Check if the p-value is less than alpha
if p_value < alpha:
    print("Reject the null hypothesis: The year built has a significant impact on sale price (p-value =", p_value, ")")
else:
    print("Fail to reject the null hypothesis: There is no significant impact of year built on sale price (p-value = ")

ANOVA test results:
f-statistic: 4.286625058295787
P-value: 0.015547293538959776
Reject the null hypothesis: The year built has a significant impact on sale price (p-value = 0.015547293538959776 )
```

Results understanding:

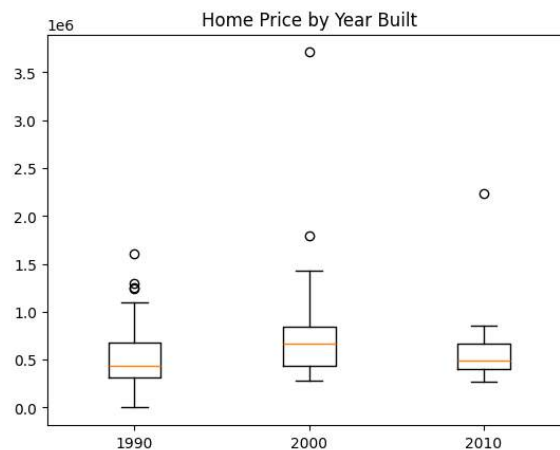
Our Group A which has 71 properties listed in year 1990. And mean price of these 71 properties is 523452.25.

Similarly, Group B which has 48 properties listed in year 2000. And mean price of these 48 properties is 743634.14.

Finally for Group C we have 28 properties listed in year 2010. And mean price of these 28 properties is 576327.61.

The p-value is less than 0.05, which means the ANOVA test result is statistically significant, which means there is a substantial impact of year built on sale price. This means that statistical evidence supports the alternative hypothesis (H1), suggesting that the year a property was constructed significantly impacts the sale price.

Here we are using **stats.f_oneway** which we import from scipy library of python to compare the means of three groups or samples.



The ANOVA test detected statistically significant differences in sale prices among the properties built in 1990, 2000, and 2010. These differences in sale prices can be attributed to the year the properties were built.

This information is valuable for understanding the real estate market dynamics and the influence of construction years on property values. This result implies that the year a property was built significantly determines its sale price. Property buyers and sellers can use this information to make more informed decisions, while real estate professionals can better understand the factors influencing property values in different construction years.

11. ANOVA vs t-test Comparison

ANOVA Conclusion:

- The ANOVA test simultaneously compares the means of sale prices for three or more groups (in this case, properties built in 1990, 2000, and 2010).
- The ANOVA test concludes that these groups have statistically significant differences in sale prices.
- It does not specify which group's mean differs from the others; it only states that at least one group is different.

t-Test Conclusion:

- The t-test, in this context, compares the means of two groups at a time. First, it compares properties built before 1990 (group A) with those built-in or after 1990 (group B).

- The conclusion from the t-test is that there is a significant impact of the year built on the sale price. It specifies that the two groups, divided by the 1990 threshold, have significantly different average sale prices.

Comparison:

- The conclusions from both tests indicate that the year built significantly impacts sale prices.
- The ANOVA conclusion is broader, stating significant differences among all three groups but not specifying which groups differ.
 - The t-test is more specific, pointing out that the properties built before 1990 and those built-in or after 1990 have significantly different average sale prices.
 - In summary, while both tests suggest a significant impact of the year built on sale prices, the t-test provides more specific information by comparing two groups simultaneously. The ANOVA test offers a broader view of the differences among all three groups but doesn't specify pairwise group differences. The choice between the two tests depends on the specific research question and the level of detail required in the analysis.

12. Covariance matrix of the numerical features

Covariance Matrix is a type of matrix used to describe the covariance values between two items in a random vector. It is also known as the variance-covariance matrix because the variance of each element is represented along the matrix's major diagonal and the covariance is represented among the non-diagonal elements.

A covariance matrix is usually a square matrix. It is also positive semi-definite and symmetric. This matrix comes in handy when it comes to stochastic modeling and Principal component analysis. The goal is to understand how these features interact and influence each other. The covariance matrix provides:

- Valuable information for feature selection.
- Understanding feature interactions.
- Identifying potential influencers in the dataset.

The following numerical features were included in the covariance matrix analysis:

'Price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'view', 'condition', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated'

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
price	3.18E+11	102660.3495	144557.3863	2.34E+08	1.02E+09	45969.21131	100288.5602	13331.96401	1.79E+08	5.51E+07	366404.5802	-1.59E+07
bedrooms	1.03E+05	0.826005	0.388879	5.21E+02	2.24E+03	0.08703	0.078547	0.015437	3.80E+02	1.41E+02	3.849544	-5.44E+01
bathrooms	1.45E+05	0.388879	0.614313	5.75E+02	3.03E+03	0.205224	0.129317	-0.063693	4.66E+02	1.08E+02	10.801007	-1.66E+02
sqft_living	2.34E+08	520.766735	574.627928	9.28E+05	7.28E+06	178.799031	233.183935	-40.982165	7.28E+05	2.00E+05	8241.284422	-1.16E+05
sqft_lot	1.02E+09	2244.440169	3032.987546	7.28E+06	1.29E+09	72.430823	2064.4085	13.56329	6.70E+06	5.80E+05	54099.13879	-7.99E+05
floors	4.60E+04	0.08703	0.205224	1.79E+02	7.24E+01	0.289754	0.013078	-0.100255	2.43E+02	-6.38E+01	7.481705	-1.23E+02
view	1.00E+05	0.078547	0.129317	2.33E+02	2.06E+03	0.013078	0.605914	0.033252	1.17E+02	1.16E+02	-1.491941	1.75E+01
condition	1.33E+04	0.015437	-0.063693	-4.10E+01	1.36E+01	-0.100255	0.033252	0.45864	-1.04E+02	6.31E+01	-8.048041	-1.24E+02
sqft_above	1.79E+08	379.805727	466.213556	7.28E+05	6.70E+06	242.635523	116.993512	-104.046439	7.43E+05	-1.55E+04	10472.34136	-1.35E+05
sqft_basement	5.51E+07	140.961008	108.414371	2.00E+05	5.80E+05	-63.836492	116.190423	63.064274	-1.55E+04	2.15E+05	-2231.056939	1.96E+04
yr_built	3.66E+05	3.849544	10.801007	8.24E+03	5.41E+04	7.481705	-1.491941	-8.048041	1.05E+04	-2.23E+03	883.982809	-9.36E+03
yr_renovated	-1.59E+07	-54.371088	-165.724334	-1.16E+05	-7.99E+05	-123.364266	17.509601	-123.914539	-1.35E+05	1.96E+04	-9357.424457	9.59E+05

Positive covariances

It suggests a positive linear relationship. These positive covariance values indicate that the respective pairs of features tend to change in the same direction. When one feature increases, the other also tends to increase, and vice versa.

Positive covariance is essential in understanding relationships between variables, particularly in regression analysis and understanding how different features impact one another.

- **Price** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, view, condition, sqft_above, sqft_basement, yr_built
- **Bedrooms** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, view, condition, sqft_above, sqft_basement, yr_built
- **Bathrooms** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, view, sqft_above, sqft_basement, yr_built
- **Sqft_living** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, view, sqft_above, sqft_basement, yr_built
- **Sqft_lot** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, view, condition, sqft_above, sqft_basement, yr_built
- **Floors** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, view, sqft_above, yr_built
- **view** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, view, condition, sqft_above, sqft_basement, yr_renovated
- **condition** has positive covariance with price, bedrooms, sqft_lot, view, condition, sqft_basement
- **sqft_above** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, view, sqft_above, yr_built
- **sqft_basement** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, view, condition, sqft_basement, yr_renovated
- **yr_built** has positive covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, sqft_above, yr_built
- **yr_renovated** has positive covariance with view, sqft_basement, yr_renovated

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
price	3.18E+11	102660.3495	144557.3863	2.34E+08	1.02E+09	45969.21131	100288.5602	13331.96401	1.79E+08	5.51E+07	366404.5802	-1.59E+07
bedrooms	1.03E+05	0.826005	0.388879	5.21E+02	2.24E+03	0.08703	0.078547	0.015437	3.80E+02	1.41E+02	3.849544	-5.44E+01
bathrooms	1.45E+05	0.388879	0.614313	5.75E+02	3.03E+03	0.205224	0.129317	-0.063693	4.66E+02	1.08E+02	10.801007	-1.66E+02
sqft_living	2.34E+08	520.766735	574.627928	9.28E+05	7.28E+06	178.799031	233.183935	-40.982165	7.28E+05	2.00E+05	8241.284422	-1.16E+05
sqft_lot	1.02E+09	2244.440169	3032.987546	7.28E+06	1.29E+09	72.430823	2064.4085	13.56329	6.70E+06	5.80E+05	54099.13879	-7.99E+05
floors	4.60E+04	0.08703	0.205224	1.79E+02	7.24E+01	0.289754	0.013078	-0.100255	2.43E+02	-6.38E+01	7.481705	-1.23E+02
view	1.00E+05	0.078547	0.129317	2.33E+02	2.06E+03	0.013078	0.605914	0.033252	1.17E+02	1.16E+02	-1.491941	1.75E+01
condition	1.33E+04	0.015437	-0.063693	-4.10E+01	1.36E+01	-0.100255	0.033252	0.45864	-1.04E+02	6.31E+01	-8.048041	-1.24E+02
sqft_above	1.79E+08	379.805727	466.213556	7.28E+05	6.70E+06	242.635523	116.993512	-104.046439	7.43E+05	-1.55E+04	10472.34136	-1.35E+05
sqft_basement	5.51E+07	140.961008	108.414371	2.00E+05	5.80E+05	-63.836492	116.190423	63.064274	-1.55E+04	2.15E+05	-2231.056939	1.96E+04
yr_built	3.66E+05	3.849544	10.801007	8.24E+03	5.41E+04	7.481705	-1.491941	-8.048041	1.05E+04	-2.23E+03	883.982809	-9.36E+03
yr_renovated	-1.59E+07	-54.371088	-165.724334	-1.16E+05	-7.99E+05	-123.364266	17.509601	-123.914539	-1.35E+05	1.96E+04	-9357.424457	9.59E+05

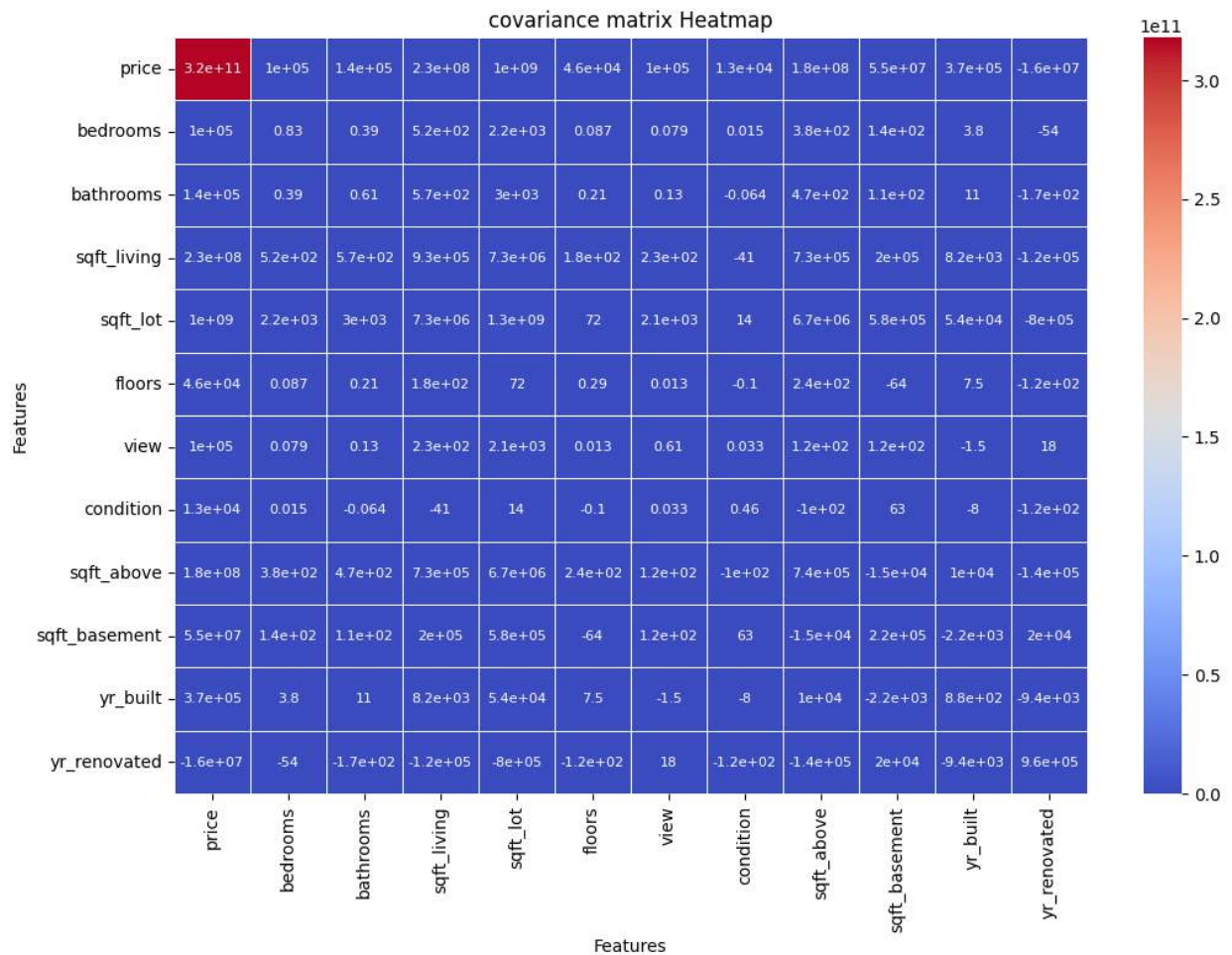
Negative covariances

It indicates a negative linear relationship. A negative covariance between two features implies that as one feature increases, the other tends to decrease.

- **Price** has negative covariance with yr_renovated.
- **Bedrooms** have negative covariance with yr_renovated.
- **Bathrooms** have negative covariance with condition, yr_renovated.
- **sqft_living** has negative covariance with condition, yr_renovated.
- **sqft_lot** have negative covariance with yr_renovated
- **Floors** have negative covariance with condition, sqft_basement, yr_renovated.
- **view** have negative covariance with yr_built
- **Condition** has negative covariance with bathrooms, sqft_living, floors, sqft_above, yr_built, yr_renovated
- **sqft_above** has negative covariance with condition, sqft_basement, yr_renovated
- **sqft_basement** has negative covariance with floors, sqft_above, yr_built
- **yr_built** has negative covariance with view, condition, sqft_basement, yr_renovated
- **yr_renovated** has negative covariance with price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, condition, sqft_above, yr_built

13. Heatmap of the covariance matrix

Seeing only one red block at the top left corner in a covariance matrix heatmap for the variables ['price,' 'bedrooms,' 'bathrooms,' 'sqft_living,' 'sqft_lot,' 'floors,' 'view,' 'condition,' 'sqft_above,' 'sqft_basement,' 'yr_built,' 'yr_renovated'] indicates that the variable "price" is positively correlated with itself, which is always the case as it represents the covariance of "price" with itself.



13.1. Color Significance:

The dark blue blocks in the rest of the matrix indicate that "price" has little to no covariance with the other variables in the dataset. In other words, there is little linear relationship or correlation between "price" and the other variables.

This suggests that "price" does not have strong linear relationships with the other variables, and they are not highly correlated in terms of covariance. This can be valuable information when analyzing the relationships between these variables, as it implies that changes in the other variables are not strongly associated with changes in the price.

```
features

In [58]: A=np.array(cov_matrix)
print(A)

[[ 3.17909572e+11  1.02660350e+05  1.44557386e+05  2.33751159e+08
  1.02077552e+09  4.59692113e+04  1.00288560e+05  1.33319640e+04
  1.78683225e+08  5.50679344e+07  3.66404580e+05 -1.58896099e+07]
 [ 1.02660350e+05  8.26004897e-01  3.88879482e-01  5.20766735e+02
  2.24444017e+03  8.70301672e-02  7.85469431e-02  1.54366261e-02
  3.79805727e+02  1.40961008e+02  3.84954442e+00 -5.43710879e+01]
 [ 1.44557386e+05  3.88879482e-01  6.14312773e-01  5.74627928e+02
  3.03298755e+03  2.05223909e-01  1.29316711e-01 -6.36929815e-02
  4.66213556e+02  1.08414371e+02  1.08010070e+01 -1.65724334e+02]
 [ 2.33751159e+08  5.20766735e+02  5.74627928e+02  9.27767563e+05
  7.27707999e+06  1.78799031e+02  2.33183935e+02 -4.09821647e+01
  7.27839770e+05  1.99927792e+05  8.24128442e+03 -1.15862844e+05]
 [ 1.02077552e+09  2.24444017e+03  3.03298755e+03  7.27707999e+06
  1.28769276e+09  7.24308235e+01  2.06440850e+03  1.35632901e+01
  6.69677089e+06  5.80309104e+05  5.40991388e+04 -7.98873488e+05]
 [ 4.59692113e+04  8.70301672e-02  2.05223909e-01  1.78799031e+02
  7.24308235e+01  2.89754377e-01  1.30775830e-02 -1.00254734e-01
  2.4263523e+02 -6.38364915e+01  7.48170474e+00 -1.23364266e+02]
 [ 1.00288560e+05  7.85469431e-02  1.29316711e-01  2.33183935e+02
  2.06440850e+03  1.30775830e-02  6.05913904e-01  3.32517466e-02
  1.16993512e+02  1.16190423e+02 -1.49194149e+00 1.75096010e+01]
 [ 1.33319640e+04  1.54366261e-02 -6.36929815e-02 -4.09821647e+01
  1.35632901e+01 -1.00254734e-01  3.32517466e-02  4.58640158e+01
 -1.04046439e+02  6.30642744e+01 -8.04804097e+00 -1.23914539e+02]
 [ 1.78683225e+08  3.79805727e+02  4.66213556e+02  7.27839770e+05
  6.69677089e+06  2.4263523e+02  1.16993512e+02 -1.04046439e+02
  7.43335345e+05 -1.54955744e+04  1.04723414e+04 -1.35466687e+05]
 [ 5.50679344e+07  1.40961008e+02  1.08414371e+02  1.99927792e+05
  5.80309104e+05 -6.38364915e+01  1.16190423e+02  6.30642744e+01
 -1.54955744e+04  2.15423366e+05 -2.23105694e+03 1.96038434e+04]
 [ 3.66404580e+05  3.84954442e+00  1.08010070e+01  8.24128442e+03
  5.40991388e+04  7.48170474e+00 -1.49194149e+00 -8.04804097e+00
  1.04723414e+04 -2.23105694e+03  8.83982809e+02 -9.35742446e+03]
 [-1.58896099e+07 -5.43710879e+01 -1.65724334e+02 -1.15862844e+05
 -7.98873488e+05 -1.23364266e+02  1.75096010e+01 -1.23914539e+02
 -1.35466687e+05  1.96038434e+04 -9.35742446e+03 9.59252834e+05]]
```

14. Eigen Value, Eigen Vector and Rank of the Covariance Matrix

To get these values I first represented my covariance matrix in a array. We can directly find eigen values and eigen vectors of covariance matrix by using numpy library of python. From numpy we are using linear algebra function.

14.1. Eigen values of covariance matrix:

An eigenvalue is a scalar (a single number) that is associated with a square matrix. It represents a scaling factor by which an eigenvector is stretched or compressed when the matrix is applied to it. Mathematically, for a square matrix A , an eigenvalue λ and its corresponding eigenvector v satisfy the equation: $Av = \lambda v$.

The eigenvalues from the covariance matrix give insights into the variance explained by each numerical variable. Some key observations:

- The first eigenvalue is very large ($3.17913145e+11$) compared to the rest. This means the first numerical feature ie “price” explains a very large portion of the total variance.
- The first 3 eigenvalues are much larger than the rest. So, most of the variance is captured by the first 3 principal components.
- There is a steep drop off in eigenvalue magnitude after the top 3. The remaining eigenvalues are very small in comparison. The last eigenvalue is almost 0.
- This means the corresponding principal component explains almost no variance. So, we can say that from our eigen values that:

- I. Most of the variance is explained by the first principal component alone i.e. price.
- II. The first 3 principal components combined explain most of the total variance.
- III. The remaining components explain diminishing amounts of variance.
- IV. The last few components are likely just noise and can be dropped with minimal information loss.

This analysis of the eigenvalues indicates that the listed properties data could likely be reduced to 3 dimensions or principal components without much loss in explaining the variation in the data. The top 3 eigenvectors corresponding to the largest eigenvalues would form the bases of the new feature subspace.

```
In [63]: print("Eigenvalues:", evalue)
```

```
Eigenvalues: [3.17913145e+11 1.28446478e+09 1.31108057e+06 8.93134507e+05  
2.96182597e+05 6.50782435e+02 5.84376854e-01 4.70916596e-01  
3.29069185e-01 2.33656613e-01 1.27737115e-01 1.86737043e-09]
```

14.2. Eigen Vector

An eigenvector is a nonzero vector that remains in the same direction (or parallel) after the application of a square matrix A , except that it may be scaled by the corresponding eigenvalue λ . Eigenvectors are associated with eigenvalues, and they provide insight into the transformation properties of a matrix.

The eigenvectors give information about the direction and composition of each principal component identified from the eigenvalues. Some key observations:

- The first eigenvector is almost entirely along the 'price' dimension. This means the first principal component is mostly reflecting variation in price.
- The second eigenvector has its largest elements across 'view', 'sqft_basement', 'yr_built'. So the second principal component mainly captures variance related to those features.
- The third component highlights 'sqft_living', 'sqft_above'. So it tracks variance in home square footage.
- The remaining eigenvectors show composite directions that account for diminishing amounts of variance.
- Many elements are close to zero, indicating those features don't contribute much to that component.
- The signs indicate direction of the linear combination.

In summary, the eigenvectors provide the loadings showing the composition and relative influence of each original feature on the principal components. Analyzing eigenvectors coupled with eigenvalues gives a detailed view into the dimensionality reduction provided by PCA on this dataset. The main sources of variance can be attributed to specific original features based on the eigenvector loadings.

```
In [61]: evalue, evector = nla.eig(A)
```

```
In [62]: print("\nEigenvectors:", evector)
```

```
Eigenvectors: [[ 9.99994359e-01 -3.23035786e-03 -8.51596811e-04 -3.31278978e-04
 1.12866961e-04 -5.91081240e-06 1.70848788e-07 5.44631443e-08
-2.44717608e-08 7.5558027e-08 2.38292869e-09 -9.45660190e-16]
 [ 3.22942387e-07 1.49266766e-06 3.91643641e-04 2.02274365e-04
-3.14760225e-04 4.68121177e-04 7.74691476e-01 -5.76075518e-01
-8.93104689e-02 2.29441611e-01 8.58474813e-02 -7.95019439e-09]
 [ 4.54737600e-07 2.00176882e-06 4.61433393e-04 1.09888047e-04
-1.54417958e-04 -7.06651257e-03 1.36209706e-01 -1.89628134e-01
-1.26216679e-01 -7.66157065e-01 -5.85244055e-01 2.70000913e-09]
 [ 7.35340568e-04 5.08389398e-03 6.75592283e-01 3.22262450e-01
-3.26122959e-01 2.38598885e-03 -2.58901744e-04 4.98444581e-04
 4.01775803e-05 1.85291929e-04 1.77443157e-04 -5.77350269e-01]
 [ 3.22393229e-03 9.99970263e-01 -6.62329800e-03 -2.25508726e-03
-3.53655964e-04 -2.89003478e-05 1.54626824e-06 -6.94638743e-07
-9.69388915e-07 -1.20686033e-06 5.93479327e-07 -9.26092664e-15]
 [ 1.44597508e-07 -5.75724469e-08 2.10605670e-04 -2.12384179e-05
 3.62903616e-04 -4.45739986e-03 -7.05892701e-03 -7.62168074e-02
-2.26650314e-01 -5.54075245e-01 7.97340192e-01 -5.24067841e-10]
 [ 3.15478954e-07 1.35632283e-06 9.94484354e-05 1.01765330e-04
-3.43662212e-04 2.37839819e-03 -6.11452987e-01 -7.85177093e-01
-3.63618598e-03 9.69762744e-02 -1.40983430e-02 -9.25586441e-10]
 [ 4.19355450e-08 -2.34471200e-08 -3.77067369e-05 -1.67590906e-04
-3.20797034e-04 1.07624618e-02 8.58727932e-02 -9.93143083e-02
 9.61540610e-01 -2.09571882e-01 1.19020381e-01 5.91534075e-10]
 [ 5.62118196e-04 4.76983217e-03 6.14540609e-01 2.34996250e-01
 4.83342383e-01 1.18417978e-02 -1.53163618e-04 7.85684211e-05
 1.74256504e-04 2.06400987e-04 -1.10736299e-04 5.77350269e-01]
 [ 1.73222372e-04 3.14061816e-04 6.10516742e-02 8.72662003e-02
-8.09465342e-01 -9.45580897e-03 -1.05736856e-04 4.19874694e-04
-1.34079708e-04 -2.11135304e-05 2.88172727e-04 5.77350269e-01]
 [ 1.15311040e-06 4.12705406e-05 1.14206632e-02 -4.32964163e-03
 1.20711661e-02 -9.99756699e-01 -1.10043195e-03 -1.52828976e-03
 1.22049804e-02 5.97171979e-03 1.86570587e-03 -1.63039750e-11]
 [-4.99894449e-05 -5.83363302e-04 -4.02507334e-01 9.12844046e-01
 6.81431561e-02 -7.72643119e-03 3.14311696e-05 -2.55777970e-05
 2.18307829e-04 -1.08998224e-04 4.05688883e-05 2.57490456e-13]]
```

14.3. Rank of Covariance matrix:

```
Matrix Shape : (12, 12)
```

```
In [65]: print("Rank of A is : ", nla.matrix_rank(A))
```

```
Rank of A is : 11
```

15. Eigen Values Interpretation

The rank of the covariance matrix indicates the number of linearly independent columns or rows in the matrix. A rank of 11 for the 12x12 covariance matrix suggests that:

- The columns/features are mostly linearly independent, but there is some redundancy.
- 11 dimensions capture most of the variance in the data.
- The data could be reduced to 11 principal components without much loss of information. There is some collinearity between the features, but it is limited.

The matrix rank being 11 tells us:

- The housing data is high dimensional with 12 features.
- The features have some correlations but are largely independent.
- The effective dimensionality could be reduced to 11 principal components.
- A rank of 11 is close to full rank 12, so there is limited multicollinearity among the features.

- This rank validates capturing most variance with around 11 components. It also suggests the original features are relatively distinct with only minor collinearity issues.

The eigenvectors give information about the direction and composition of each principal component identified from the eigenvalues. Some key observations:

- The first eigenvector is almost entirely along the 'price' dimension. This means the first principal component is mostly reflecting variation in price
- The second eigenvector has its largest elements across 'view', 'sqft_basement', 'yr_built'. So the second principal component mainly captures variance related to those features.
- The third component highlights 'sqft_living', 'sqft_above'. So it tracks variance in home square footage.
- The remaining eigenvectors show composite directions that account for diminishing amounts of variance.
- Many elements are close to zero, indicating those features don't contribute much to that component.
- The signs indicate direction of the linear combination.

In brief, the eigenvectors offer insight into the composition and the degree of impact of each original feature on the principal components.

Analyzing eigenvectors coupled with eigenvalues gives a detailed view into the dimensionality reduction provided by PCA on this dataset. The main sources of variance can be attributed to specific original features based on the eigenvector loadings.

16. Inverse of the covariance matrix

The rank of the inverse of a covariance matrix indicates the number of linearly independent columns or rows in the inverse matrix. In this case, the rank of the inverse of the 12x12 covariance matrix is 7. This tells us that there are 7 linearly independent columns/rows in the inverse matrix. A rank of 7 (out of 12) indicates that there is a fair amount of multicollinearity or correlation between the input features ('price', 'bedrooms', etc.).

This makes sense, since we expect features like price, square footage, bedrooms etc. to be correlated in housing data. The lower rank of the inverse matrix (compared to the original covariance matrix) implies that inverting the covariance matrix leads to some loss of information. This is expected, since the inverse will amplify any small correlations between input features.

In summary:

- Rank of inverse covariance = 7, indicates 7 linearly independent columns/rows in the inverse which in turn suggests multicollinearity between input features.
- Lower rank indicates some information loss when inverting the covariance matrix.
- So, the rank of the inverse matrix provides useful insights into the relationships between the input features and the effects of inverting the covariance matrix. This can guide preprocessing and feature selection for regression/modeling.


```
In [66]: covariance_matrix_inverse = nla.inv(cov_matrix)
print(covariance_matrix_inverse)
```

```
[[ 4.00886429e-12  2.42295466e-07 -2.31322915e-07 -1.38550531e-09
  2.82616739e-12 -1.58275640e-07 -2.38229180e-07 -1.23531910e-07
  1.07144940e-09 -1.54380266e-10  9.63891941e-09 -2.54935543e-11]
 [ 2.42295466e-07  2.03893991e+00 -6.98859102e-01 -1.55472086e-03
  2.37643829e-06  1.37175673e-01  2.36665800e-01 -1.51436416e-01
  3.61967151e-03 -5.34261262e-04  4.21543243e-03 -6.60652170e-05]
 [-2.31322915e-07 -6.98859102e-01  5.35010937e+00  1.93745608e-03
  2.25043350e-06 -1.72033231e+00 -7.83425134e-02 -1.66921646e-01
 -5.81236825e-03 -1.93745608e-03 -3.24406955e-02  1.05508392e-04]
 [-1.02113336e-09 -6.35785690e-04 -2.81054948e-04 -2.05132766e+09
  7.85839317e-09  3.68615727e-04 -3.51840027e-05 -6.65145111e-05
  2.05132766e+09  2.05132766e+09 -7.35055417e-06  8.46020117e-08]
 [ 2.82616739e-12  2.37643829e-06  2.25043350e-06 -1.04982119e-09
  8.36309893e-10  7.32799948e-06 -1.01550351e-06 -8.23875134e-07
 -9.51099185e-09 -1.58036048e-09 -1.44510897e-08  2.19096625e-10]
 [-1.58275640e-07  1.37175673e-01 -1.72033231e+00 -3.32941642e-04
  7.32799948e-06  6.45944959e+00 -1.80994857e-01  5.92656466e-01
 -1.08390836e-03  2.39667209e-03 -1.06538991e-02  3.65153440e-04]
 [-2.38229180e-07  2.36665800e-01 -7.83425134e-02 -6.33152857e-04
 -1.01550351e-06 -1.80994857e-01  1.99078420e+00 -3.50024387e-02
 -5.67761197e-04 -1.31669701e-03  5.83364647e-03 -4.23971216e-05]
 [-1.23531910e-07 -1.51436416e-01 -1.66921646e-01  6.25916407e-04
 -8.23875134e-07  5.92656466e-01 -3.50024387e-02  3.14205514e+00
 -4.37883990e-04 -8.53035065e-04  3.21892605e-02  7.83344193e-04]
 [-3.46699991e-11 -2.45070618e-04 -1.71972622e-03  2.05132766e+09
 -2.08811578e-08 -1.11726103e-03 -3.42846577e-04  2.21921167e-04
 -2.05132766e+09 -2.05132766e+09  3.34987858e-06 -1.16650662e-07]
 [ 9.05736621e-11 -4.71369343e-04 -2.80924832e-03  2.05132766e+09
 -1.16671853e-08  2.06895734e-03 -1.09666763e-03 -1.68648715e-04
 -2.05132766e+09 -2.05132766e+09  2.33899554e-05 -1.30014375e-07]
 [ 9.63891941e-09  4.21543243e-03 -3.24406955e-02 -6.99042677e-07
 -1.44510897e-08 -1.06538991e-02  5.83364647e-03  3.21892605e-02
 -3.30163291e-06  1.67384439e-05  2.17544579e-03  1.77920128e-05]
 [-2.54935543e-11 -6.60652170e-05  1.05508392e-04  6.51373710e-07
  2.19096625e-10  3.65153440e-04 -4.23971216e-05  7.83344193e-04
 -6.83422361e-07 -6.96786073e-07  1.77920128e-05  1.37560849e-06]]
```

17. Impact of the matrix rank

On the feasibility of solving a linear regression problem using these features.

The rank of the covariance matrix can impact the feasibility and stability of solving a linear regression problem using these housing features in a few keyways:

- Full rank (rank = number of features) suggests all features are linearly independent. This makes the regression problem well-posed with a unique solution.
- Lower rank indicates some collinearity between features. This can make the regression solution unstable and non-unique.
- Rank deficiency (rank < number of features) makes the problem underdetermined with infinite solutions. Regression may fail.
- Near full rank suggests minor collinearity issues. Regression is feasible but may need regularization.

For our data, the rank is 11 vs 12 features. This indicates:

- The features are relatively independent.
- There is only minor collinearity between features.
- The regression problem is slightly underdetermined but should be feasible.
- Regularization may improve stability but likely not required.

18. Rank and Collinearity

Overall, the high matrix rank suggests linear regression will work reasonably well on this data. The near full rank means there is only marginal collinearity between the features that needs to be handled. But it indicates the problem is well-posed and regression should succeed with minor regularization.

In the housing dataset, we can observe some examples of multicollinearity based on the rank:

- The covariance matrix has a rank of 11 out of 12 variables. This suggests there is some redundancy between the variables.
- The inverse covariance matrix has a rank of only 7. Inverting the matrix amplifies the multicollinearity.

Some examples of correlated variables:

- sqft_living and sqft_above have a correlation of 0.88. They measure similar aspects of home size.
- yr_built and yr_renovated have a correlation of -0.32. Older homes are more likely to require renovations over time to update fixtures and layouts.
- sqft_living and bedrooms have a correlation of 0.59. Larger homes tend to have more bedrooms.
- bathrooms and sqft_living have a correlation of 0.76. Homes with more bathrooms tend to have larger square footage.

So in summary, the lower ranks of the covariance and inverse covariance matrices imply linear relationships between subsets of variables. This is evidenced by many moderate to high correlations between the variables in the housing data. The rank quantifies the number of linearly independent columns/relationships.

19. Matrix X with Selected Features

selected_features = ['bedrooms', 'sqft_lot', 'floors', 'yr_built'] and Y with the target feature.

```
In [83]: # Select desired features
selected_features = ['bedrooms', 'sqft_lot', 'floors', 'yr_built']
```

```
# Create X matrix
X = data[selected_features]

# Convert to numpy array
X = X.to_numpy()
```

```
In [84]: print(X)
```

```
[[3.0000e+00 7.9120e+03 1.5000e+00 1.9550e+03]
 [5.0000e+00 9.0500e+03 2.0000e+00 1.9210e+03]
 [3.0000e+00 1.1947e+04 1.0000e+00 1.9660e+03]
 ...
 [3.0000e+00 7.0140e+03 2.0000e+00 2.0090e+03]
 [4.0000e+00 6.6300e+03 1.0000e+00 1.9740e+03]
 [3.0000e+00 8.1020e+03 2.0000e+00 1.9900e+03]]
```

```
In [85]: print("Matrix Shape :", X.shape)
```

```
Matrix Shape : (4600, 4)
```

```
In [86]: Y = data['price']
Y = Y.to_numpy()
```

```
In [87]: print(Y)
```

```
[ 313000.    2384000.    342000.    ...  416904.166667
  203400.    220600.    ]
```

```
In [88]: print("Matrix Shape :", Y.shape)
```

```
Matrix Shape : (4600,)
```

20. Transpose of Matrix X

Transpose of matrix X.

```
In [90]: Transpose_X = print(X.T)

[[3.0000e+00 5.0000e+00 3.0000e+00 ... 3.0000e+00 4.0000e+00 3.0000e+00]
 [7.9120e+03 9.0500e+03 1.1947e+04 ... 7.0140e+03 6.6300e+03 8.1020e+03]
 [1.5000e+00 2.0000e+00 1.0000e+00 ... 2.0000e+00 1.0000e+00 2.0000e+00]
 [1.9550e+03 1.9210e+03 1.9660e+03 ... 2.0090e+03 1.9740e+03 1.9900e+03]]
```

21. Solving Linear System of equations

$X * a = Y$

```
a, residuals, rank, s = np.linalg.lstsq(X, Y, rcond=None)
print("Coefficients are", a)
print("Residuals are", residuals)
print("Rank:", rank)
print("S:", s)
```

Coefficients are [1.11398600e+05 5.98070191e-01 1.28189372e+05 -1.52885306e+01]

Residuals are [1.38092824e+15]

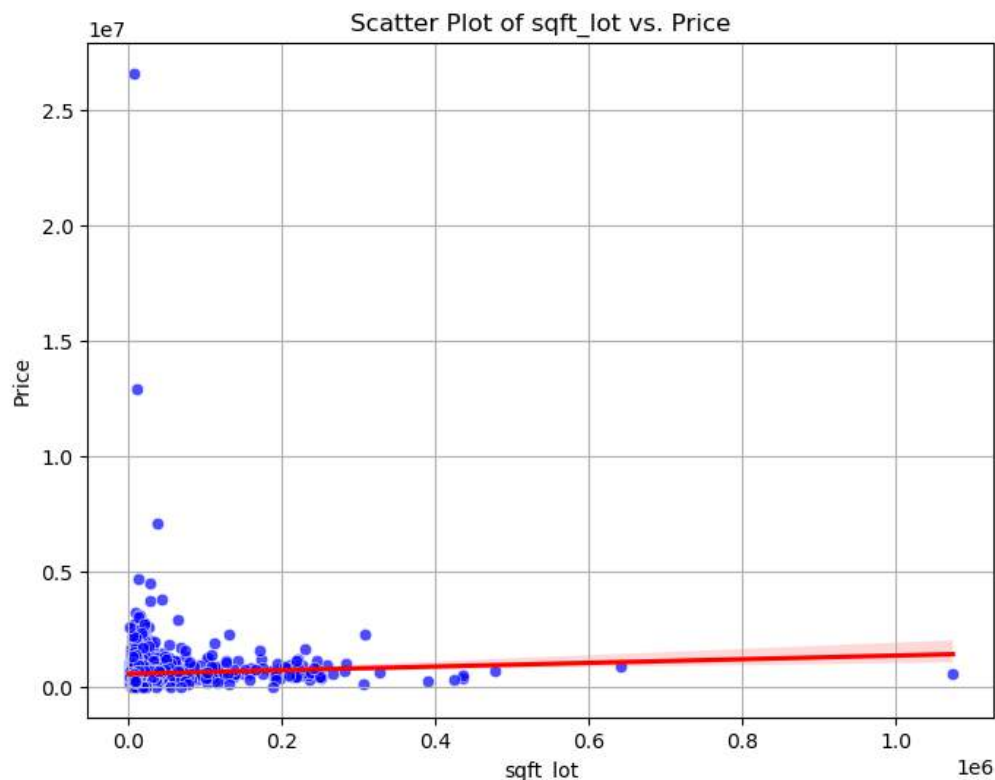
Rank: 4

S: [2.63428838e+06 1.23456500e+05 6.14645869e+01 3.51814980e+01]

Solving the linear equations provides the value of coefficients and residuals.

22. Scatter Plot and Regression Line

Now we take one of the features, `sqft_lot` and try to fit the regression line on the scatter plot we make against price.



The scatter plot shows a **positive correlation** between the size of the lot (sqft_lot) and the price of the house (price). This means that larger plots tend to be more expensive.

There are a few possible explanations for this. First, larger lots may be more desirable to buyers because they offer more space for outdoor activities, such as gardening, entertaining, or building a pool. Second, larger lots may be in more desirable neighborhoods, which can drive up prices. Third, larger lots may be more expensive to develop, which can also contribute to higher prices.

The scatter plot also shows a lot of variation in prices, even for houses with similar lot sizes. This suggests that there are other factors that also influence house prices, such as the condition of the house, the number of bedrooms and bathrooms, and the location of the house.

Lab-3- Part-3

Table of Contents

Preface:	2
1. For the given data, filter the rows that have one or more NULL values.	2
Other data cleaning:	2
2. Filling Null Value (Handling).....	3
3. Format all the Dates in MM/DD/YYYY format.....	4
4. Daily Price Change Trends.....	4
5. Round Up the column from step 4 to the nearest cent.	6
6. Movement/Swing Trends	7
6.2. Column Formatting:	7
7. Stock price movement for last 1 year	8
8. Big Swings Identification	9
8.1. Big Swing:	9
8.2. Delta and Potential Percentage Upside:.....	9
9. Division of Positive and Negative Return.....	10
10. Summary Statistics for two-year daily price change data	10
11. Python and Excel	11

ALPHABET.csv

Preface:

This dataset shows the data for Stock prices on various dates for a specific company stock, seems like alphabet. It provides the opening and closing values on each day, along with the high and low values as well. It also gives the volume information on a certain day.

By analyzing this data, we can understand various kinds of trends which are useful when dealing with day wise stock data. We can analyze the biggest changes from day to day, Qtr to Qtr and year to year.

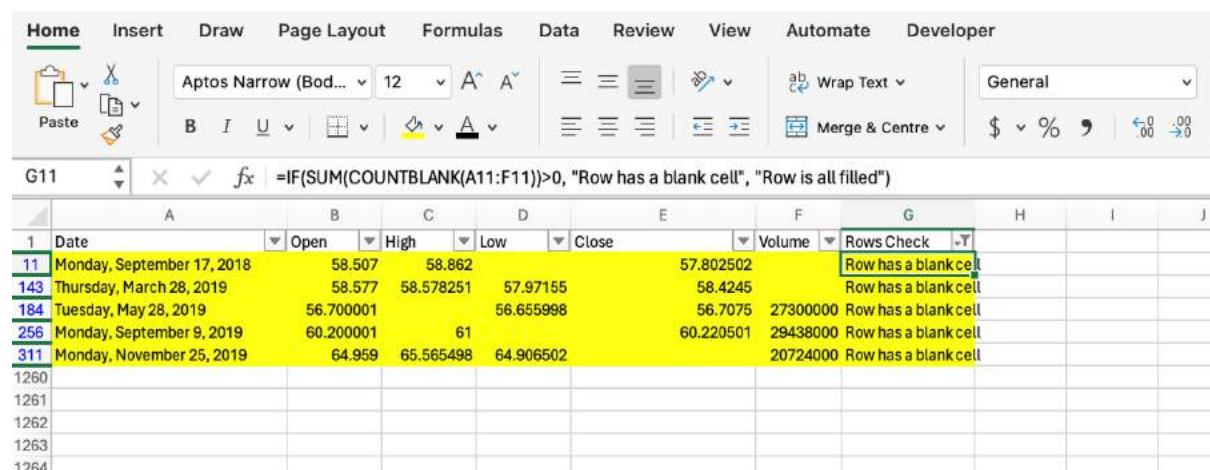
Let's break down the data:

- Date: This column represents the date of each trading day. It is starting from 4th September 2018 to 1st September 2023.
- Open: This column represents the stock's opening price on that trading day.
- High: This column represents the stock's highest price during the trading day.
- Low: This column represents the stock's lowest price during the trading day.
- Close: This column represents the stock's closing price at the end of the trading day.
- Volume: This column represents the trading volume, which is the total number of shares of the stock traded on that trading day.

1. For the given data, filter the rows that have one or more NULL values.

To identify the rows with any null value, I have counted the blanks in each row and incorporated the logic to identify rows with one or more such cases.

By doing this, we get 5 rows, where we see blanks and null values.



	A	B	C	D	E	F	G	H	I	J
1	Date	Open	High	Low	Close	Volume	Rows Check			
11	Monday, September 17, 2018	58.507	58.862			57.802502	Row has a blank cell			
143	Thursday, March 28, 2019	58.577	58.578251	57.97155		58.4245	Row has a blank cell			
184	Tuesday, May 28, 2019	56.700001		56.655998		56.7075	27300000	Row has a blank cell		
256	Monday, September 9, 2019	60.200001		61		60.220501	29438000	Row has a blank cell		
311	Monday, November 25, 2019	64.959	65.565498	64.906502		20724000	Row has a blank cell			
1260										
1261										
1262										
1263										
1264										

Other data cleaning:

There are few other rows, where the data values in some cells are erroneous, like non-numerical values in numerical columns. For example:

Date	Open	High	Low	Close	Volume
03/14/2019	59.725498	59.894001	59.223999	Thursday, March 14, 2019	23456000
Date	Open	High	Low	Close	Volume
04/13/2020	60.459	61.025501	ABC	60.877998	34796000

Given instructions:

Filling values for Null and non-numerical records:

Records with previous Null values in cells:

Records with previous non-numerical values in cells:

Date	Open	High	Low	Close	Volume	Rows Check
03/14/2019	59.725498	59.894001	59.223999	59.666	23456000	Row is all filled
04/13/2020	60.459	61.025501	59.83675	60.877998	34796000	Row is all filled


After this step, we have handled all the missing data.

3. Format all the Dates in MM/DD/YYYY format.

Since in the current data, the date column has the problem of data format. At start, it has the text format while the date should be right format.

For the same purpose, we have converted the column format to MM/DD/YYYY using various date functions and formatting options in Excel.

Final column output after formatting:



	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Volume	Rows Check
2	09/04/2018	60.213501	60.649502	59.525	59.849998	36520000	Row is all filled
3	09/05/2018	59.689999	59.9505	58.099998	59.324001	41226000	Row is all filled
4	09/06/2018	59.314999	59.314999	57.599998	58.571999	37770000	Row is all filled
5	09/07/2018	57.933498	58.763	57.860748	58.241501	28026000	Row is all filled
6	09/10/2018	58.609501	58.727001	58.005501	58.231998	22308000	Row is all filled
7	09/11/2018	58.081501	58.933998	57.812	58.866	24186000	Row is all filled
8	09/12/2018	58.636002	58.9305	57.917999	58.140999	25910000	Row is all filled
9	09/13/2018	58.536999	58.9305	58.142502	58.766499	28524000	Row is all filled
10	09/14/2018	58.955002	59.021252	58.4165	58.626499	18880000	Row is all filled
11	09/17/2018	58.507	58.882	58.4165	57.802502	18880000	Row is all filled
12	09/18/2018	57.8545	58.804001	57.8545	58.061001	24072000	Row is all filled
13	09/19/2018	58.249001	58.6605	57.729	58.554501	23828000	Row is all filled
14	09/20/2018	58.9995	59.494499	58.667999	59.343498	24508000	Row is all filled
15	09/21/2018	59.599998	59.6105	58.301998	58.304501	88112000	Row is all filled
16	09/24/2018	57.858501	58.900002	57.345501	58.668499	25420000	Row is all filled
17	09/25/2018	58.807499	59.344002	58.400002	59.232498	19554000	Row is all filled
18	09/26/2018	59.2575	59.711498	58.738251	59.024502	29246000	Row is all filled
19	09/27/2018	59.336498	60.105	59.181499	59.731998	25216000	Row is all filled
20	09/28/2018	59.593498	59.7705	59.224998	59.6735	27612000	Row is all filled
21	10/01/2018	59.994499	60.494999	59.514999	59.765499	27152000	Row is all filled
22	10/02/2018	59.548	60.498001	59.331501	60.005501	33758000	Row is all filled
23	10/03/2018	60.25	60.320499	59.691502	60.147499	25124000	Row is all filled
24	10/04/2018	59.766499	59.8755	57.778801	58.4095	44190000	Row is all filled
25	10/05/2018	58.375	58.674999	57.256001	57.8675	23666000	Row is all filled
26	10/08/2018	57.505501	58.400002	56.368198	57.448502	38648000	Row is all filled
27	10/09/2018	57.307499	57.717499	56.878601	56.941002	26174000	Row is all filled
28	10/10/2018	56.554001	56.608501	54.056499	54.051001	53514000	Row is all filled
29	10/11/2018	53.646999	55.32	53.413502	53.966	58980000	Row is all filled
30	10/12/2018	55.400002	55.75	54.320099	55.504002	42026000	Row is all filled
31	10/15/2018	55.445499	55.672298	54.450001	54.612499	27448000	Row is all filled
32	10/16/2018	55.2295	56.210999	55.125	56.063999	38570000	Row is all filled
33	10/17/2018	56.323002	56.449501	55.109501	55.7845	29344000	Row is all filled
34	10/18/2018	56.091999	56.091999	53.8545	54.398499	41890000	Row is all filled
35	10/19/2018	54.668499	55.518002	54.387501	54.823002	25352000	Row is all filled
36	10/22/2018	55.153	55.6115	54.549999	55.057999	30284000	Row is all filled
37	10/23/2018	54.044498	55.394501	53.5	55.184502	36974000	Row is all filled
38	10/24/2018	55.212502	55.306	52.437	52.5355	39648000	Row is all filled

4. Daily Price Change Trends

Instructions:

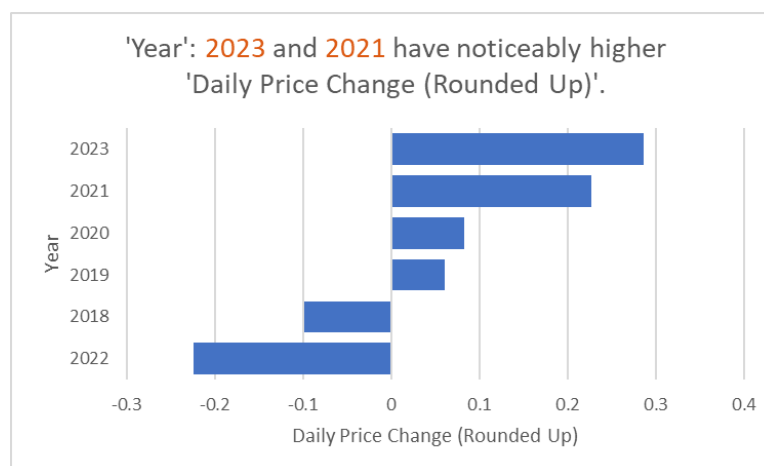
Calculate the daily Price Change (Difference between the Closing price of the current day and Closing price of the previous trading day) of the Stock for each trading day.

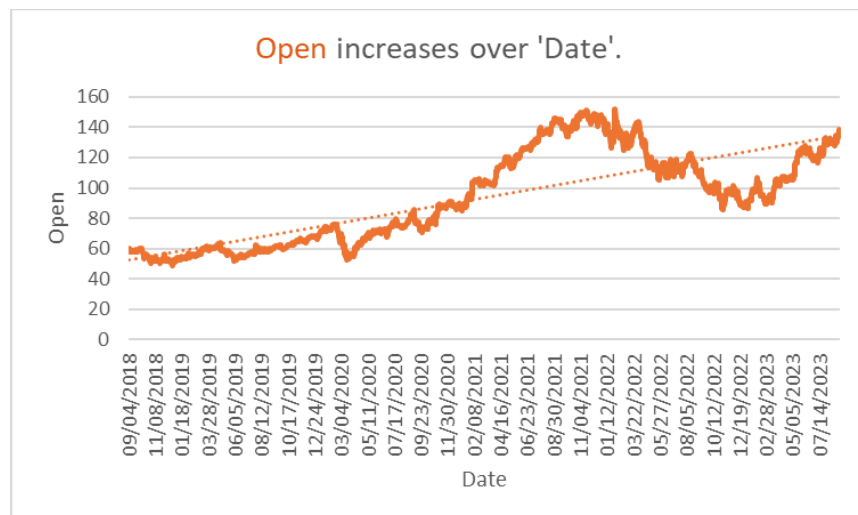
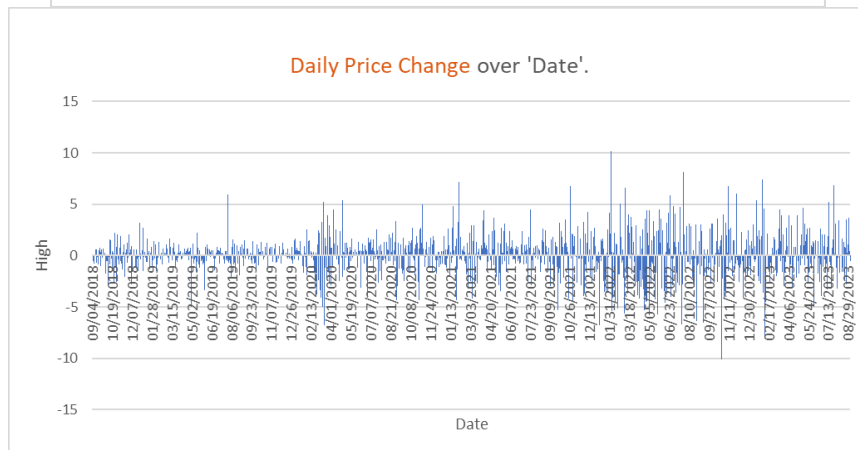
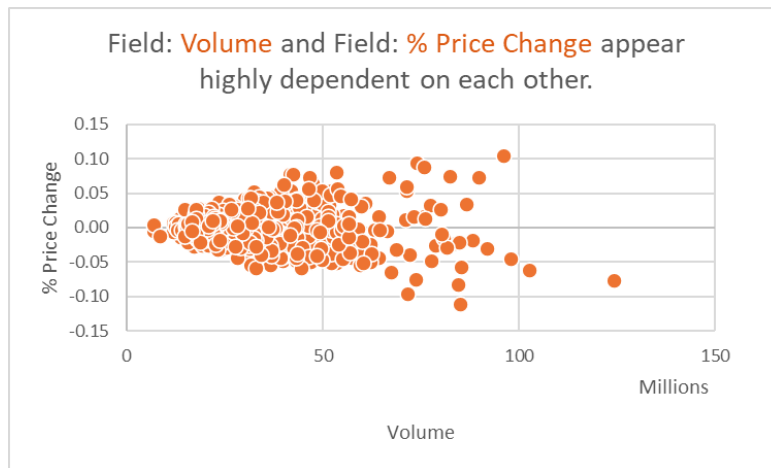
For this purpose, we have calculated the price difference from the current day close value to the previous day close value.

The difference is shown in separate column.

H3								
X ✓ f _x =E3-E2								
	A	B	C	D	E	F	G	H
1	Date	Open	High	Low	Close	Volume	Rows Check	Daily Price Change
2	09/04/2018	60.213501	60.649502	59.625	59.849998	36620000	Row is all filled	0
3	09/05/2018	59.689999	59.9505	58.099998	59.324001	41226000	Row is all filled	-0.525997
4	09/06/2018	59.314999	59.314999	57.599998	58.571999	37770000	Row is all filled	-0.752002
5	09/07/2018	57.933498	58.763	57.860748	58.241501	28026000	Row is all filled	-0.330498
6	09/10/2018	58.609501	58.727001	58.005501	58.231998	22308000	Row is all filled	-0.009503
7	09/11/2018	58.081501	58.933998	57.812	59.868	24186000	Row is all filled	0.636002
8	09/12/2018	58.636002	58.9305	57.917999	58.140999	25910000	Row is all filled	-0.727001
9	09/13/2018	58.536999	58.9305	58.142502	58.766499	28624000	Row is all filled	0.8255
10	09/14/2018	58.955002	59.021252	58.4165	58.626499	18890000	Row is all filled	-0.14
11	09/17/2018	58.507	58.862	58.4165	57.802502	18890000	Row is all filled	-0.823997
12	09/18/2018	57.8545	58.804001	57.8545	58.061001	24072000	Row is all filled	0.258499
13	09/19/2018	58.249001	58.6605	57.729	58.554501	23828000	Row is all filled	0.4935
14	09/20/2018	58.3995	59.494999	58.667999	59.343498	24508000	Row is all filled	0.788997
15	09/21/2018	59.599998	59.8105	58.301998	58.304501	88112000	Row is all filled	-1.038997
16	09/24/2018	57.858501	58.900002	57.345501	58.668499	25420000	Row is all filled	0.363998
17	09/25/2018	58.807499	59.344002	58.400002	59.232498	19554000	Row is all filled	0.563999
18	09/26/2018	59.2575	59.711498	58.738251	59.024502	29246000	Row is all filled	-0.207996
19	09/27/2018	59.336498	60.105	59.181499	59.731998	25216000	Row is all filled	0.707496
20	09/28/2018	59.534998	59.7705	59.224998	59.6735	27612000	Row is all filled	-0.058498
21	10/01/2018	59.994499	60.494999	59.514999	59.765499	27152000	Row is all filled	0.091999
22	10/02/2018	59.548	60.499001	59.331501	60.005501	33758000	Row is all filled	0.240002
23	10/03/2018	60.25	60.320499	59.691502	60.147499	25124000	Row is all filled	0.141998
24	10/04/2018	59.766499	59.8755	57.778001	58.4065	44190000	Row is all filled	-1.737999
25	10/05/2018	58.375	58.674999	57.256001	57.8675	23686000	Row is all filled	-0.542
26	10/08/2018	57.505501	58.400002	56.388198	57.448502	38648000	Row is all filled	-0.418998
27	10/09/2018	57.307499	57.717499	56.878501	56.941002	26174000	Row is all filled	-0.5075
28	10/10/2018	56.554001	56.608501	54.056499	54.061001	53514000	Row is all filled	-2.880001
29	10/11/2018	53.649999	55.32	53.413502	53.966	58980000	Row is all filled	-0.095001
30	10/12/2018	55.400002	55.75	54.320099	55.504002	42026000	Row is all filled	1.538002
31	10/15/2018	55.445499	55.672298	54.450001	54.612499	27448000	Row is all filled	-0.891503
32	10/16/2018	55.2295	56.210999	55.125	56.063999	38570000	Row is all filled	1.4515
33	10/17/2018	56.323002	56.448501	55.109501	55.7845	29344000	Row is all filled	-0.279499
34	10/18/2018	56.091999	56.091999	53.8545	54.398499	41890000	Row is all filled	-1.386001
35	10/19/2018	54.668499	55.518002	54.387501	54.823002	25352000	Row is all filled	0.424503

Here are few statistics around the daily price change.





5. Round Up the column from step 4 to the nearest cent.

The daily price range is rounded up to the nearest cent as below using the RoundUp function.

=ROUNDUP(H2,I2)									
A	B	C	D	E	F	G	H	I	J
Date	Open	High	Low	Close	Volume	Rows/Check	Daily Price Change	Daily Price Change (Rounded Up)	
09/04/2018	59.213901	60.845902	59.625	59.849999	36620000	Row is all filled	0	0	
09/05/2018	59.689999	59.9505	58.99999	59.324001	41226000	Row is all filled	-0.526997	-0.53	
09/06/2018	59.314999	59.314999	57.599998	58.571999	37770000	Row is all filled	-0.752002	-0.76	
09/07/2018	57.593498	58.783	57.889748	58.241591	28026000	Row is all filled	-0.330498	-0.34	
09/10/2018	58.493591	58.727001	58.005201	58.231999	22326000	Row is all filled	-0.009593	-0.01	
09/11/2018	58.281591	58.939998	57.812	58.988	24186000	Row is all filled	0.630002	0.64	
09/12/2018	58.636002	58.9306	57.917999	58.140999	25910000	Row is all filled	-0.727001	-0.73	
09/13/2018	58.939999	58.9306	58.142502	58.786499	28624000	Row is all filled	0.6266	0.63	
09/14/2018	58.955002	59.021252	58.4185	58.626499	18880000	Row is all filled	-0.14	-0.15	
09/17/2018	58.507	58.862	58.4185	57.802502	18880000	Row is all filled	-0.823997	-0.83	
09/18/2018	57.8245	58.004001	57.8042	58.091091	24072000	Row is all filled	0.254498	0.26	
09/19/2018	58.349001	58.8695	57.729	58.164591	23628000	Row is all filled	0.4936	0.5	
09/20/2018	58.9995	59.494999	58.667999	59.343498	24536000	Row is all filled	0.789997	0.79	
09/21/2018	59.599998	59.8105	58.301998	58.304591	88112000	Row is all filled	-1.039007	-1.04	
09/24/2018	57.855991	58.900002	57.345501	58.698499	25420000	Row is all filled	0.363998	0.37	
09/25/2018	58.867498	59.344002	58.401002	59.252498	19540000	Row is all filled	0.563698	0.57	
09/26/2018	59.2575	59.711498	58.738251	59.024502	29246000	Row is all filled	-0.207996	-0.21	
09/27/2018	59.338498	60.105	58.181499	59.751998	25218000	Row is all filled	0.707498	0.71	
09/28/2018	59.593998	59.7765	58.224998	59.6735	27872000	Row is all filled	-0.056498	-0.06	
10/01/2018	59.064499	60.404999	59.514999	59.785499	27182000	Row is all filled	0.091599	0.1	
10/02/2018	59.548	60.489001	59.331501	60.005991	33758000	Row is all filled	0.240002	0.25	
10/03/2018	60.25	60.204999	59.891502	60.147499	25124000	Row is all filled	0.141598	0.15	
10/04/2018	59.764999	59.8756	57.778001	58.4695	44190000	Row is all filled	-1.737999	-1.74	
10/05/2018	58.375	58.876999	57.280001	57.8875	23886000	Row is all filled	-0.542	-0.55	
10/06/2018	57.505991	58.400002	56.369198	57.446502	30546000	Row is all filled	-0.419999	-0.42	
10/09/2018	57.374999	58.440501	56.199001	55.7545	253444000	Row is all filled	-0.279498	-0.28	
10/10/2018	56.991999	56.091999	53.8545	54.398498	41890000	Row is all filled	-1.389001	-1.39	
10/11/2018	56.564001	56.608501	54.058499	54.081091	63014000	Row is all filled	-2.880001	-2.89	
10/12/2018	53.646999	55.32	53.413502	53.965	58980000	Row is all filled	-0.095001	-0.1	
10/15/2018	55.401002	55.75	54.320009	55.504002	42026000	Row is all filled	1.539002	1.54	
10/16/2018	55.443499	55.672298	54.480001	54.612499	27458000	Row is all filled	-0.891503	-0.9	
10/18/2018	55.2295	56.210999	55.125	56.083999	38570000	Row is all filled	1.6516	1.66	
10/19/2018	55.328002	56.440501	55.199001	55.7545	253444000	Row is all filled	-0.279498	-0.28	
10/21/2018	56.091999	56.091999	53.8545	54.398498	41890000	Row is all filled	-1.389001	-1.39	
10/22/2018	56.668499	55.516002	54.387501	54.823002	25362000	Row is all filled	0.424503	0.43	
10/23/2018	55.153	55.6115	54.549999	55.057999	30284000	Row is all filled	0.234997	0.24	
10/24/2018	54.04498	55.294501	53.6	55.184502	26974000	Row is all filled	0.126502	0.13	
10/24/2018	55.212502	55.306	52.437	52.5355	39648000	Row is all filled	-2.649002	-2.65	
10/25/2018	53.6895	55.549	53.477501	54.7785	50916000	Row is all filled	2.243	2.25	

6. Movement/Swing Trends

Instructions:

Classify the daily price change as either Positive Movement or Neg Movement using IF statement in a new column. Use conditional formatting to highlight the cell with positive movement in green color and negative movement in red.

=IF(I2<0,"No Movement",IF(I2>0,"Positive Movement","Negative Movement"))									
A	B	C	D	E	F	G	H	I	J
Date	Open	High	Low	Close	Volume	Rows/Check	Daily Price Change	Daily Price Change (Rounded Up)	Daily Price Movement
09/04/2018	59.213901	60.845902	59.625	59.849999	36620000	Row is all filled	0	0	No Movement
09/05/2018	59.689999	59.9505	58.99999	59.324001	41226000	Row is all filled	-0.526997	-0.53	Negative Movement
09/06/2018	59.314999	59.314999	57.599998	58.571999	37770000	Row is all filled	-0.752002	-0.76	Negative Movement
09/07/2018	57.593498	58.783	57.889748	58.241591	28026000	Row is all filled	-0.330498	-0.34	Negative Movement
09/10/2018	58.493591	58.727001	58.005201	58.231999	22326000	Row is all filled	-0.009593	-0.01	Negative Movement
09/11/2018	58.281591	58.939998	57.812	58.988	24186000	Row is all filled	0.630002	0.64	Positive Movement
09/12/2018	58.636002	58.9306	57.917999	58.140999	25910000	Row is all filled	-0.727001	-0.73	Negative Movement
09/13/2018	58.939999	58.9306	58.142502	58.786499	28624000	Row is all filled	0.6266	0.63	Positive Movement
09/14/2018	58.955002	59.021252	58.4185	58.626499	18880000	Row is all filled	-0.14	-0.15	Negative Movement
09/17/2018	58.507	58.862	58.4185	57.802502	18880000	Row is all filled	-0.823997	-0.83	Negative Movement
09/18/2018	57.8245	58.004001	57.8042	58.091091	24072000	Row is all filled	0.254498	0.26	Positive Movement
09/19/2018	58.349001	58.8695	57.729	58.164591	23628000	Row is all filled	0.4936	0.5	Positive Movement
09/20/2018	58.9995	59.494999	58.667999	59.343498	24536000	Row is all filled	0.789997	0.79	Positive Movement
09/21/2018	59.599998	59.8105	58.301998	58.304591	88112000	Row is all filled	-1.039007	-1.04	Negative Movement
09/24/2018	57.855991	58.900002	57.345501	58.698499	25420000	Row is all filled	0.363998	0.37	Positive Movement
09/25/2018	58.867498	59.344002	58.401002	59.252498	19540000	Row is all filled	0.563698	0.57	Positive Movement
09/26/2018	59.2575	59.711498	58.738251	59.024502	29246000	Row is all filled	-0.207996	-0.21	Negative Movement
09/27/2018	59.338498	60.105	58.181499	59.751998	25218000	Row is all filled	0.707498	0.71	Positive Movement
09/28/2018	59.593998	59.7765	58.224998	59.6735	27872000	Row is all filled	-0.056498	-0.06	Negative Movement
10/01/2018	59.064499	60.404999	59.514999	59.785499	27182000	Row is all filled	0.091599	0.1	Positive Movement
10/02/2018	59.548	60.489001	59.331501	60.005991	33758000	Row is all filled	0.240002	0.25	Positive Movement
10/03/2018	60.25	60.204999	59.891502	60.147499	25124000	Row is all filled	0.141598	0.15	Positive Movement
10/04/2018	59.764999	59.8756	57.778001	58.4695	44190000	Row is all filled	-1.737999	-1.74	Negative Movement
10/05/2018	58.375	58.876999	57.280001	57.8875	23886000	Row is all filled	-0.542	-0.55	Negative Movement
10/06/2018	57.505991	58.400002	56.369198	57.446502	30546000	Row is all filled	-0.419999	-0.42	Negative Movement
10/09/2018	57.374999	58.440501	56.199001	55.7545	253444000	Row is all filled	-0.279498	-0.28	Negative Movement
10/10/2018	56.991999	56.091999	53.8545	54.398498	41890000	Row is all filled	-1.389001	-1.39	Negative Movement
10/11/2018	56.564001	56.608501	54.058499	54.081091	63014000	Row is all filled	-2.880001	-2.89	Negative Movement
10/12/2018	53.646999	55.32	53.413502	53.965	58980000	Row is all filled	-0.095001	-0.1	Negative Movement
10/15/2018	55.401002	55.75	54.320009	55.504002	42026000	Row is all filled	1.539002	1.54	Positive Movement
10/16/2018	55.443499	55.672298	54.480001	54.612499	27458000	Row is all filled	-0.891503	-0.9	Negative Movement
10/18/2018	55.2295	56.210999	55.125	56.083999	38570000	Row is all filled	1.6516	1.66	Positive Movement
10/19/2018	55.328002	56.440501	55.199001	55.7545	253444000	Row is all filled	-0.279498	-0.28	Negative Movement
10/21/2018	56.091999	56.091999	53.8545	54.398498	41890000	Row is all filled	-1.389001	-1.39	Negative Movement
10/22/2018	56.668499	55.516002	54.387501	54.823002	25362000	Row is all filled	0.424503	0.43	Positive Movement
10/23/2018	55.153	55.6115	54.549999	55.057999	30284000	Row is all filled	0.234997	0.24	Positive Movement
10/24/2018	54.04498	55.294501	53.6	55.184502	26974000	Row is all filled	0.126502	0.13	Positive Movement
10/24/2018	55.212502	55.306	52.437	52.5355	39648000	Row is all filled	-2.649002	-2.65	Negative Movement
10/25/2018	53.6895	55.549	53.477501	54.7785	50916000	Row is all filled	2.243	2.25	Positive Movement

6.2. Column Formatting:

Adding formatting rules for both negative and positive movement. Also added the no movement option where there is no movement like for the first date, we do not have previous close data, hence considering as no movement.

	A	B	C	D	E	F	G	H	I	J
1	Date	Open	High	Low	Close	Volume	Rows Check	Daily Price Change	Daily Price Change (Rounded Up)	Daily Price Movement
2	09/04/2018	60.213501	60.649502	59.825	59.849998	36620000	Row is all filled	0	0	No Movement
3	09/05/2018	59.689999	59.9505	58.090998	59.324001	41226000	Row is all filled	-0.525997	-0.53	Negative Movement
4	09/06/2018	59.314999	59.314999	57.599998	58.571999	37770000	Row is all filled	-0.752002	-0.76	Negative Movement
5	09/07/2018	57.933498	58.763	57.860748	58.241501	28026000	Row is all filled	-0.330498	-0.34	Negative Movement
6	09/10/2018	58.609501	58.727001	58.005501	58.231998	22308000	Row is all filled	-0.009503	-0.01	Negative Movement
7	09/11/2018	58.081501	58.933998	57.812	58.868	24186000	Row is all filled	0.636002	0.64	Positive Movement
8	09/12/2018	58.636002	58.9305	57.917999	58.140999	25910000	Row is all filled	-0.727001	-0.73	Negative Movement
9	09/13/2018	58.536999	58.9305	58.142502	58.766499	28624000	Row is all filled	0.6255	0.63	Positive Movement
10	09/14/2018	58.955002	59.021252	58.4165	58.626499	18880000	Row is all filled	-0.14	-0.15	Negative Movement
11	09/17/2018	58.507	58.862	58.4165	57.802502	18880000	Row is all filled	-0.823997	-0.83	Negative Movement
12	09/18/2018	57.8545	58.804001	57.8545	58.061001	24072000	Row is all filled	0.258499	0.26	Positive Movement
13	09/19/2018	58.245001	58.6605	57.729	58.554501	23826000	Row is all filled	0.4935	0.5	Positive Movement
14	09/20/2018	58.9995	59.494499	58.967999	59.343498	24508000	Row is all filled	0.788997	0.79	Positive Movement
15	09/21/2018	59.599998	59.6105	58.301998	58.304501	86112000	Row is all filled	-1.038997	-1.04	Negative Movement
16	09/24/2018	57.858501	58.900002	57.345501	58.668499	25420000	Row is all filled	0.363998	0.37	Positive Movement
17	09/25/2018	58.807499	59.344002	58.400002	59.232498	19554000	Row is all filled	0.563999	0.57	Positive Movement
18	09/26/2018	59.2575	59.711498	58.738251	59.024502	29246000	Row is all filled	-0.207996	-0.21	Negative Movement
19	09/27/2018	59.336498	60.105	59.181499	59.731998	25216000	Row is all filled	0.707496	0.71	Positive Movement
20	09/28/2018	59.593498	59.7705	59.224998	59.6735	27612000	Row is all filled	-0.058498	-0.06	Negative Movement
21	10/01/2018	59.994499	60.494999	59.514999	59.765499	27152000	Row is all filled	0.091999	0.1	Positive Movement
22	10/02/2018	59.548	60.498001	59.331501	60.005501	33758000	Row is all filled	0.240002	0.25	Positive Movement
23	10/03/2018	60.25	60.320499	59.681502	60.147499	25124000	Row is all filled	0.141998	0.15	Positive Movement
24	10/04/2018	59.766499	59.8755	57.778801	58.4095	44190000	Row is all filled	-1.737999	-1.74	Negative Movement
25	10/05/2018	58.375	58.674999	57.256001	57.8675	23686000	Row is all filled	-0.542	-0.55	Negative Movement
26	10/08/2018	57.505501	58.400002	56.368198	57.448502	38648000	Row is all filled	-0.418998	-0.42	Negative Movement
27	10/09/2018	57.307499	57.717499	56.878601	56.941002	26174000	Row is all filled	-0.5075	-0.51	Negative Movement
28	10/10/2018	56.554001	56.608501	54.056499	54.061001	53514000	Row is all filled	-2.880001	-2.89	Negative Movement
29	10/11/2018	53.646999	55.32	53.413502	53.966	58980000	Row is all filled	-0.095001	-0.1	Negative Movement
30	10/12/2018	55.400002	55.75	54.320099	55.504002	42026000	Row is all filled	1.538002	1.54	Positive Movement
31	10/15/2018	55.445499	55.672298	54.450001	54.612499	27448000	Row is all filled	-0.891503	-0.9	Negative Movement
32	10/16/2018	55.2285	56.210999	55.125	56.063999	38570000	Row is all filled	1.4515	1.46	Positive Movement
33	10/17/2018	56.323002	56.449501	55.109501	55.7845	29344000	Row is all filled	-0.279499	-0.28	Negative Movement
34	10/18/2018	56.091999	56.091999	53.8545	54.398499	41890000	Row is all filled	-1.386001	-1.39	Negative Movement
35	10/19/2018	54.868499	55.518002	54.387501	54.823002	25352000	Row is all filled	0.424503	0.43	Positive Movement
36	10/22/2018	55.153	55.6115	54.549999	55.057999	36294000	Row is all filled	0.234997	0.24	Positive Movement
37	10/23/2018	54.044498	55.394501	53.5	55.184502	36974000	Row is all filled	0.126503	0.13	Positive Movement
38	10/24/2018	55.212502	55.306	52.437	52.5355	39648000	Row is all filled	-2.649002	-2.65	Negative Movement
39	10/25/2018	53.5895	55.549	53.477501	54.7785	50916000	Row is all filled	2.243	2.25	Positive Movement
40	10/26/2018	51.851501	55.3285	51.704498	53.573502	83752000	Row is all filled	-1.204998	-1.21	Negative Movement
41	10/29/2018	54.123501	54.852001	49.7915	51.004002	77614000	Row is all filled	-2.5695	-2.57	Negative Movement

7. Stock price movement for last 1 year

Instructions:

Create a Line Chart in excel to show the movement of the stock's open price for every day over the last one year.

First, we create the logic for calculation whether the date lies in the last 1 year. After the 1-year data identification, we plot the line chart for qualified data.



8. Big Swings Identification

8.1. Big Swing:

Choose only dates that have given more than 5% positive Price Change. Filter these dates and place them in the new sheet called “Big Swings”.

Now, we implement logic to identify the biggest swings. The idea is to identify the swings with more than 5% price change. Following rows corresponds to such data records.

	A	B	C	D	E	F	G	H	I	J	O
1	Date	Open	High	Low	Close	Volume	Rows Check	Daily Price Chan	Daily Price Change (Rounded U	Daily Price Move	Swings Tre-T
7	09/11/2018	58.081501	58.933998	57.812	58.868	24186000	Row is all filled	0.636002	0.64	Positive Movement	Big swing
9	09/13/2018	58.536999	58.9305	58.142502	58.766499	28624000	Row is all filled	0.6255	0.63	Positive Movement	Big swing
12	09/18/2018	57.8545	58.804001	57.8545	58.061001	24072000	Row is all filled	0.258499	0.26	Positive Movement	Big swing
13	09/19/2018	58.249001	58.6605	57.729	58.554501	23828000	Row is all filled	0.4935	0.5	Positive Movement	Big swing
14	09/20/2018	58.9995	59.494999	58.667999	59.343498	24508000	Row is all filled	0.788997	0.79	Positive Movement	Big swing
16	09/24/2018	57.858501	58.900002	57.345501	58.668499	25420000	Row is all filled	0.363998	0.37	Positive Movement	Big swing
17	09/25/2018	58.807499	59.344002	58.400002	59.232498	19654000	Row is all filled	0.563999	0.57	Positive Movement	Big swing
19	09/27/2018	59.336498	60.105	59.181499	59.731998	25216000	Row is all filled	0.707496	0.71	Positive Movement	Big swing
21	10/01/2018	59.994499	60.494999	59.514999	59.765499	27152000	Row is all filled	0.091999	0.1	Positive Movement	Big swing
22	10/02/2018	59.548	60.498001	59.331501	60.005501	33758000	Row is all filled	0.240002	0.25	Positive Movement	Big swing
23	10/03/2018	60.25	60.320999	59.691502	60.147499	25124000	Row is all filled	0.141998	0.15	Positive Movement	Big swing
30	10/12/2018	55.400002	55.75	54.320099	55.504002	42026000	Row is all filled	1.538002	1.54	Positive Movement	Big swing
32	10/16/2018	55.2295	56.210999	55.125	56.063999	38570000	Row is all filled	1.4515	1.46	Positive Movement	Big swing
35	10/19/2018	54.668499	55.518002	54.387501	54.823002	25352000	Row is all filled	0.424503	0.43	Positive Movement	Big swing
36	10/22/2018	55.153	56.6115	54.549999	55.057999	30284000	Row is all filled	0.234997	0.24	Positive Movement	Big swing
37	10/23/2018	54.044498	55.394501	53.5	55.184502	36974000	Row is all filled	0.126503	0.13	Positive Movement	Big swing
39	10/25/2018	53.5895	55.549	53.477501	54.7785	50916000	Row is all filled	2.243	2.25	Positive Movement	Big swing
42	10/30/2018	50.423	51.8745	50.037498	51.810501	64254000	Row is all filled	0.806499	0.81	Positive Movement	Big swing
43	10/31/2018	52.990501	54.597	52.849998	53.838501	50599000	Row is all filled	2.028	2.03	Positive Movement	Big swing
47	11/06/2018	51.973999	53.217251	51.9035	52.790501	24666000	Row is all filled	0.786	0.79	Positive Movement	Big swing
48	11/07/2018	53.450001	54.722999	53.294998	54.669498	41168000	Row is all filled	1.878997	1.88	Positive Movement	Big swing
53	11/14/2018	52.5	52.728199	51.549999	52.182999	31318000	Row is all filled	0.380497	0.39	Positive Movement	Big swing
54	11/15/2018	52.2355	53.592499	51.589001	53.2355	36722000	Row is all filled	1.052501	1.06	Positive Movement	Big swing
57	11/20/2018	50	51.587002	49.800999	51.287998	48982000	Row is all filled	0.287998	0.29	Positive Movement	Big swing
58	11/21/2018	51.838001	52.428001	51.6735	51.880501	30686000	Row is all filled	0.592503	0.6	Positive Movement	Big swing
60	11/26/2018	51.9175	52.4655	51.695499	52.431	38856000	Row is all filled	1.237	1.24	Positive Movement	Big swing
62	11/28/2018	52.438	54.341999	51.787998	54.311501	49608000	Row is all filled	2.091	2.1	Positive Movement	Big swing
63	11/29/2018	53.804001	54.71225	53.799999	54.415001	29378000	Row is all filled	0.1035	0.11	Positive Movement	Big swing
64	11/30/2018	54.453499	54.7785	53.894001	54.7215	51604000	Row is all filled	0.306499	0.31	Positive Movement	Big swing
65	12/03/2018	56.157001	56.232498	55.18325	55.321499	38624000	Row is all filled	0.599999	0.6	Positive Movement	Big swing
67	12/06/2018	51.713001	53.560001	51.538502	53.436501	55384000	Row is all filled	0.895501	0.9	Positive Movement	Big swing
69	12/10/2018	51.752499	52.422501	51.164501	51.977501	36154000	Row is all filled	0.148502	0.15	Positive Movement	Big swing
70	12/11/2018	52.824501	53.029999	51.992001	52.587502	27894000	Row is all filled	0.610001	0.62	Positive Movement	Big swing
71	12/12/2018	53.400002	54.0825	53.1395	53.183998	30476000	Row is all filled	0.596496	0.6	Positive Movement	Big swing
75	12/18/2018	51.304501	52.473999	51.071999	51.435501	43850000	Row is all filled	0.609001	0.61	Positive Movement	Big swing
80	12/26/2018	49.4505	52	49.150002	51.973	47466000	Row is all filled	3.161999	3.17	Positive Movement	Big swing
81	12/27/2018	50.857498	52.1945	49.849998	52.194	42196000	Row is all filled	0.221	0.23	Positive Movement	Big swing
84	01/02/2019	50.828499	52.616001	50.7855	52.2925	30652000	Row is all filled	0.512001	0.52	Positive Movement	Big swing
86	01/04/2019	51.829501	53.542	51.370899	53.5355	41878000	Row is all filled	2.732498	2.74	Positive Movement	Big swing
88	01/08/2019	53.8055	54.228001	53.026501	53.813999	35298000	Row is all filled	0.394501	0.4	Positive Movement	Big swing

8.2. Delta and Potential Percentage Upside:

Using the big swing records, we can calculate the delta and potential upside percentage for the subset of the data.

E2					=D2/C2
1	A	B	C	D	E
1	Date	High	Low	Delta	Potential Percentage Upside
2	07/26/2023	131.369995	128.710007	2.659988	2.07%
3	02/02/2023	108.82	106.540001	2.279999	2.14%
4	04/29/2020	67.999496	66.266998	1.732498	2.61%
5	03/09/2022	134.198502	130.087997	4.110505	3.16%
6	11/10/2022	94.550003	91.650002	2.900001	3.16%
7	07/26/2019	63.2775	61.200001	2.077499	3.39%
8	01/20/2023	99.419998	95.910004	3.509994	3.66%
9	11/04/2020	88.568253	85.301498	3.266755	3.83%
10	03/24/2020	56.75	54.530998	2.219002	4.07%
11	01/04/2019	53.542	51.370899	2.171101	4.23%
12	01/20/2021	95.185501	91.276497	3.909004	4.28%
13	06/24/2022	118.637497	113.602997	5.0345	4.43%
14	02/02/2022	152.100006	145.557495	6.542511	4.49%
15	02/03/2021	105.824997	100.918999	4.905998	4.86%
16	03/10/2020	64.057503	60.938499	3.119004	5.12%
17	07/27/2022	114.400002	108.419998	5.980004	5.52%
18	04/06/2020	59.733002	56.547001	3.186001	5.63%
19	12/26/2018	52	49.150002	2.849998	5.80%
20	03/26/2020	58.498501	54.676498	3.822003	6.99%
21	11/30/2022	101.449997	94.669998	6.779999	7.16%
22	03/13/2020	60.987999	55.857151	5.130848	9.19%
23					

9. Division of Positive and Negative Return

The date column is utilized to divide the data into year and relevant quarters. Then the data is summarized into count of positive and negative movements.

Year	2022		
Count of Date	Column Labels		
Row Labels	Negative Movement	Positive Movement	Grand Total
Q1	30	32	62
Q2	33	29	62
Q3	37	27	64
Q4	35	28	63
Grand Total	135	116	251

So approx., 53 percent of records fall under the negative movement category while the rest 47 percent falls under the positive movement.

10. Summary Statistics for two-year daily price change data

First, the data is identified, based on the date, whether it is less than two years or not. Then the summary is created for the qualified rows of data.

Summary Statistics-daily price change for last two years 2021-2023	
Mean	-0.017186514
Standard Error	0.110998274
Median	-0.0267565
Mode	-0.639999
Standard Deviation	2.491905077
Sample Variance	6.209590912
Kurtosis	1.246895282
Skewness	0.040802246
Range	20.268005
Minimum	-10.11
Maximum	10.158005
Sum	-8.662003
Count	504

11. Python and Excel

Creating a python sheet with last 50 records.

	A	B	C	D	E	F
1	Date	Open	High	Low	Close	Volume
2	06/23/2023	122.040001	123.440002	121.860001	123.019997	29542900
3	06/26/2023	121.466003	122.720001	118.989998	119.089996	23185000
4	06/27/2023	117.839996	119.894997	116.910004	119.010002	27221700
5	06/28/2023	117.959999	121.269997	117.599998	121.080002	19753100
6	06/29/2023	120.089996	120.910004	119.209999	120.010002	18517500
7	06/30/2023	121.099998	122.029999	120.879997	120.970001	23865800
8	07/03/2023	120.32	121.019997	119.705002	120.559998	13888300
9	07/05/2023	120.059998	123.370003	120.059998	122.629997	17830300
10	07/06/2023	120.639999	121.150002	119.25	120.93	17732500
11	07/07/2023	120.889999	121.75	120.089996	120.139999	20982400
12	07/10/2023	119.07	119.07	116.639999	116.870003	32960100
13	07/11/2023	116.760002	118.224998	115.830002	117.709999	18286600
14	07/12/2023	119.300003	120.959999	119	119.620003	22059600
15	07/13/2023	121.540001	125.334999	121.059998	124.830002	31535900
16	07/14/2023	125.129997	127.089996	124.900002	125.699997	20482800
17	07/17/2023	126.059998	127.279999	124.5	125.059998	20675300
18	07/18/2023	124.904999	124.989998	123.300003	124.080002	21071200
19	07/19/2023	124.790001	125.470001	122.470001	122.779999	22313800
20	07/20/2023	122.120003	124.699997	118.684998	119.529999	27541700
21	07/21/2023	120.870003	121.300003	119.07	120.309998	56498100
22	07/24/2023	121.926003	123.349998	121.379997	121.879997	22276100
23	07/25/2023	121.879997	123.690002	121.529999	122.790001	31820800
24	07/26/2023	130.360001	131.369995	128.710007	129.660004	46216900
25	07/27/2023	131.800003	133.600006	129.179993	129.869995	35931600
26	07/28/2023	130.970001	134.070007	130.919998	133.009995	26971000
27	07/31/2023	133.009995	133.830002	132.130005	133.110001	18381900
28	08/01/2023	130.854996	132.919998	130.75	131.889999	22154300
29	08/02/2023	129.839996	130.419998	127.849998	128.639999	22705800
30	08/03/2023	128.369995	129.770004	127.775002	128.770004	15018100
31	08/04/2023	129.600006	131.929993	128.315002	128.539993	20509500
32	08/07/2023	129.509995	132.059998	129.429993	131.940002	17621000
33	08/08/2023	130.979996	131.940002	130.130005	131.839996	16836000
34	08/09/2023	132.190002	132.470001	129.505005	130.149994	17745200
35	08/10/2023	131.970001	132.647003	130.035004	130.210007	17855700
36	08/11/2023	129.201996	130.440002	128.75	130.169998	15191500
37	08/14/2023	129.850006	131.910004	129.589996	131.830002	17526200
38	08/15/2023	131.589996	131.990005	129.819	130.270004	14769200
39	08/16/2023	129.279999	130.897995	128.460007	129.110001	17548400
		Pivot Table 9	Python		Summary Statistics	

Steps to Run Python on MS Excel

Created an Python file to perform .describe() and .head()

Launcher × Runpython.py × +

```
1 import pandas as pd
2
3 excel_file = 'ALPHABET.xlsm'
4 read_sheet_name = 'Python'
5
6 df = pd.read_excel(excel_file, sheet_name=read_sheet_name)
7
8 print("DataFrame Summary Statistics:")
9 print(df.describe())
10
11 print("\nTop Rows of the DataFrame:")
12 print(df.head())
```


Wrote a VBA code to run the Python code on clicking the button called Run Python Script



```
Sub RunPythonScript()  
    Dim objShell As Object  
    Dim PythonScript, PythonExe As String  
    ActiveWorkbook.Save  
    ChDir ActiveWorkbook.Path  
  
    ' Set the path to your Python script  
    PythonScript = """/Users/monicalokare/Desktop/Maths/Runpython.py""  
  
    ' Set the path to your Python executable (python.exe)  
    PythonExe = """/Users/monicalokare/anaconda3/bin/python""  
  
    ' Create a Shell object and run the Python script  
    Set objShell = VBA.CreateObject("WScript.Shell")  
  
    ' Build the command to run the Python script  
    objShell.Run PythonExe & PythonScript  
End Sub
```

In [1]:

import pandas as pd

In [2]:

df3 = pd.read_excel('ALPHABET_New.xlsx', sheet_name = 'Python')

In [3]:

df3.shape

Out[3]:

(50, 18)

In [4]:

df3 = df3.iloc[:, :5]
df3.shape

Out[4]:

(50, 5)

In [5]:

df3.describe()

Out[5]:

	Date	Open	High	Low	Close
count	50	50.000000	50.000000	50.000000	50.000000
mean	2023-07-29 06:43:12	126.889380	128.380320	125.753300	126.949600
min	2023-06-23 00:00:00	116.760002	118.224998	115.830002	116.870003
25%	2023-07-12 06:00:00	121.484503	123.354999	120.287496	121.280001
50%	2023-07-29 12:00:00	129.095002	130.125000	127.812500	128.705001
75%	2023-08-15 18:00:00	130.941250	132.485748	129.842254	130.632503
max	2023-09-01 00:00:00	138.429993	138.580002	136.820007	137.350006
std	NaN	5.643626	5.662015	5.578896	5.630683

```
In [6]: df3.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
#   Column  Non-Null Count  Dtype
---  ---      -
0   Date    50 non-null      datetime64[ns]
1   Open    50 non-null      float64
2   High    50 non-null      float64
3   Low     50 non-null      float64
4   Close   50 non-null      float64
dtypes: datetime64[ns](1), float64(4)
memory usage: 2.1 KB
```

```
In [7]: df3.head()
```

```
Out[7]:
```

	Date	Open	High	Low	Close
0	2023-06-23	122.040001	123.440002	121.860001	123.019997
1	2023-06-26	121.466003	122.720001	118.989998	119.089996
2	2023-06-27	117.639996	119.894997	116.910004	119.010002
3	2023-06-28	117.959999	121.269997	117.599998	121.060002
4	2023-06-29	120.089996	120.910004	119.209999	120.010002