ASSIGNMENT - 1

Assignment-1

October 25, 2024

0.0.1 Imports

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

0.0.2 Question 1

[2]:
```
df = pd.read_excel('./dataset/Flipkart-Laptops.xlsx')
print(df.columns)
```

```
Index(['Product Name', 'ProductID', 'Product image', 'Actual price',
       'Discount price', 'Stars', 'Rating', 'Reviews', 'Description', 'Link'],
      dtype='object')
```

[3]:
```
"""Load the dataset into a pandas Data Frame."""

df = pd.DataFrame(df)
```

[4]:
```
""" Display the first and the last 5 rows of the dataset"""

print(f"\nThe first 5 row of dataset \n{df.head()}")
print(f"\nThe last 5 row of dataset \n{df.tail()}")
```

```
The first 5 row of dataset
                                        Product Name         ProductID  \
0  MSI Cyborg 15 Intel Core i5 12th Gen 12450H - …  COMGZW35W3DSJADN
1  MSI Thin 15 Intel Core i7 12th Gen 12650H - (8…  COMGZW37ZX66DBHF
2  DELL Inspiron 3520 Intel Core i3 12th Gen 1215…  COMGJ75HJGFDJ6JN
3  Acer One (2024) Intel Core i3 11th Gen 1115G4 …  COMGPF5CQ7VDWDT4
4  Lenovo V15 AMD Ryzen 3 Quad Core 7320U - (8 GB…  COMGPYKZAWY8UX6C

   Product image Actual price Discount price Stars         Rating  \
0            NaN        89990          54990   3.9      7 Ratings
1            NaN        83990          67990   NIL            NIL
2            NaN        49240          35660   4.2  1,805 Ratings
3            NaN        43999          26990   4.2  6,977 Ratings
4            NaN        59400          27989   4.2  1,263 Ratings
```

```
       Reviews                                          Description  \
0    1 Reviews  Intel Core i5 Processor (12th Gen)16 GB DDR5 R…
1          NIL  Intel Core i7 Processor (12th Gen)8 GB DDR4 RA…
2  143 Reviews  Intel Core i3 Processor (12th Gen)8 GB DDR4 RA…
3  596 Reviews  Intel Core i3 Processor (11th Gen)8 GB DDR4 RA…
4  113 Reviews  AMD Ryzen 3 Quad Core Processor8 GB LPDDR5 RAM…

                                                Link
0  https://www.flipkart.com/msi-cyborg-15-intel-c…
1  https://www.flipkart.com/msi-thin-15-intel-cor…
2  https://www.flipkart.com/dell-inspiron-3520-in…
3  https://www.flipkart.com/acer-one-2024-intel-c…
4  https://www.flipkart.com/lenovo-v15-amd-ryzen-…

The last 5 row of dataset
                                        Product Name         ProductID  \
955  Acer Swift Go 14 (2024) AI Powered EVO Intel C…  COMGWKF2VKGAVHDU
956  HP Victus Intel Core i5 12th Gen 12450H - (16 …  COMH2DYZHMHZ5UPG
957  Infinix X1 Slim Series (2024) Intel Core i3 10…  COMGEHP5EFEGWZW5
958  Lenovo IdeaPad Slim 3 Intel Core i5 12th Gen 1…  COMGYHP5ZB4AGZH6
959  HP (15s-fq5007TU) Intel Core i3 12th Gen 1215U…  COMGYHP5MCEYZHSV

     Product image Actual price Discount price Stars         Rating  \
955            NaN       129999          79990   4.1    108 Ratings
956            NaN          NIL          82414   NIL            NIL
957            NaN        49999          32990   4.3  3,897 Ratings
958            NaN        69890          53390   3.8     53 Ratings
959            NaN        51134          38990   4.2  5,540 Ratings

         Reviews                                          Description  \
955   16 Reviews  Intel Core Ultra 5 Processor16 GB LPDDR5X RAMW…
956          NIL  Intel Core i5 Processor (12th Gen)16 GB DDR4 R…
957  457 Reviews  Intel Core i3 Processor (10th Gen)8 GB LPDDR4X…
958    5 Reviews  Intel Core i5 Processor (12th Gen)16 GB LPDDR5…
959  485 Reviews  Intel Core i3 Processor (12th Gen)8 GB DDR4 RA…

                                               Link
955  https://www.flipkart.com/acer-swift-go-14-2024…
956  https://www.flipkart.com/hp-victus-intel-core-…
957  https://www.flipkart.com/infinix-x1-slim-serie…
958  https://www.flipkart.com/lenovo-ideapad-slim-3…
959  https://www.flipkart.com/hp-15s-fq5007tu-intel…
```

[5]:
```
"""Custom Data Works"""

# df = df.head()
```

```
df = df.dropna(axis=1,how='all')
df['Actual price'] = pd.to_numeric(df['Actual price'], errors='coerce')

df['Discount price'] = pd.to_numeric(df['Discount price'], errors='coerce')
df['Discount price'] = df['Discount price'].replace({' ': '', ',': ''},␣
 ↪regex=True).astype(float)
df['Rating'] = df['Rating'].str.replace(' Ratings', '')

df['Reviews'] = df['Reviews'].str.replace(' Reviews','')
df['Reviews'] = pd.to_numeric(df['Reviews'].str.replace(',',''),errors='coerce')

df['Stars'] = pd.to_numeric(df['Stars'], errors='coerce')

print(f"Does any columns contain Null value: \n{df.isna().any(axis=0)}")
df = df.fillna(0)
print(f"\nDoes any columns contain Null value: \n\n{df.isna().any(axis=0)}")
```

```
Does any columns contain Null value:
Product Name      False
ProductID         False
Actual price       True
Discount price     True
Stars              True
Rating            False
Reviews            True
Description       False
Link              False
dtype: bool

Does any columns contain Null value:

Product Name      False
ProductID         False
Actual price      False
Discount price    False
Stars             False
Rating            False
Reviews           False
Description       False
Link              False
dtype: bool
```

[6]:
```
"""Calculate the total sales of each company"""

df['Total Sales'] = df['Actual price'] - df['Discount price']
print(f"\nThe total sales of each company \n{df}")
```

The total sales of each company

	Product Name	ProductID
0	MSI Cyborg 15 Intel Core i5 12th Gen 12450H - …	COMGZW35W3DSJADN
1	MSI Thin 15 Intel Core i7 12th Gen 12650H - (8…	COMGZW37ZX66DBHF
2	DELL Inspiron 3520 Intel Core i3 12th Gen 1215…	COMGJ75HJGFDJ6JN
3	Acer One (2024) Intel Core i3 11th Gen 1115G4 …	COMGPF5CQ7VDWDT4
4	Lenovo V15 AMD Ryzen 3 Quad Core 7320U - (8 GB…	COMGPYKZAWY8UX6C
..	…	…
955	Acer Swift Go 14 (2024) AI Powered EVO Intel C…	COMGWKF2VKGAVHDU
956	HP Victus Intel Core i5 12th Gen 12450H - (16 …	COMH2DYZHMHZ5UPG
957	Infinix X1 Slim Series (2024) Intel Core i3 10…	COMGEHP5EFEGWZW5
958	Lenovo IdeaPad Slim 3 Intel Core i5 12th Gen 1…	COMGYHP5ZB4AGZH6
959	HP (15s-fq5007TU) Intel Core i3 12th Gen 1215U…	COMGYHP5MCEYZHSV

	Actual price	Discount price	Stars	Rating	Reviews
0	89990.0	54990.0	3.9	7	1.0
1	83990.0	67990.0	0.0	NIL	0.0
2	49240.0	35660.0	4.2	1,805	143.0
3	43999.0	26990.0	4.2	6,977	596.0
4	59400.0	27989.0	4.2	1,263	113.0
..	…	…	…	…	…
955	129999.0	79990.0	4.1	108	16.0
956	0.0	82414.0	0.0	NIL	0.0
957	49999.0	32990.0	4.3	3,897	457.0
958	69890.0	53390.0	3.8	53	5.0
959	51134.0	38990.0	4.2	5,540	485.0

	Description
0	Intel Core i5 Processor (12th Gen)16 GB DDR5 R…
1	Intel Core i7 Processor (12th Gen)8 GB DDR4 RA…
2	Intel Core i3 Processor (12th Gen)8 GB DDR4 RA…
3	Intel Core i3 Processor (11th Gen)8 GB DDR4 RA…
4	AMD Ryzen 3 Quad Core Processor8 GB LPDDR5 RAM…
..	…
955	Intel Core Ultra 5 Processor16 GB LPDDR5X RAMW…
956	Intel Core i5 Processor (12th Gen)16 GB DDR4 R…
957	Intel Core i3 Processor (10th Gen)8 GB LPDDR4X…
958	Intel Core i5 Processor (12th Gen)16 GB LPDDR5…
959	Intel Core i3 Processor (12th Gen)8 GB DDR4 RA…

	Link	Total Sales
0	https://www.flipkart.com/msi-cyborg-15-intel-c…	35000.0
1	https://www.flipkart.com/msi-thin-15-intel-cor…	16000.0
2	https://www.flipkart.com/dell-inspiron-3520-in…	13580.0
3	https://www.flipkart.com/acer-one-2024-intel-c…	17009.0
4	https://www.flipkart.com/lenovo-v15-amd-ryzen-…	31411.0
..	…	…
955	https://www.flipkart.com/acer-swift-go-14-2024…	50009.0

```
956  https://www.flipkart.com/hp-victus-intel-core-…     -82414.0
957  https://www.flipkart.com/infinix-x1-slim-serie…      17009.0
958  https://www.flipkart.com/lenovo-ideapad-slim-3…      16500.0
959  https://www.flipkart.com/hp-15s-fq5007tu-intel…      12144.0

[960 rows x 10 columns]
```

[7]:
```
"""Display the product name that has price greater than 50000 and less than␣
 ↪80000"""

dfGreaterOrLess = df[(df['Actual price'] > 50000) & (df['Actual price'] <␣
 ↪80000)]
print(f"\nProduct Greater than 50000 and less than 80000␣
 ↪\n{dfGreaterOrLess[['Product Name','Actual price']]}")
```

```
Product Greater than 50000 and less than 80000
                                          Product Name  Actual price
4    Lenovo V15 AMD Ryzen 3 Quad Core 7320U - (8 GB…       59400.0
5    Lenovo AMD Ryzen 3 Quad Core 7330U - (8 GB/512…       63900.0
6    HP FQ Series Intel Core i3 12th Gen 1215U - (8…       50843.0
12   HP AMD Ryzen 5 Hexa Core 5500U - (16 GB/512 GB…       59109.0
19   HP 2023 Intel Core i3 12th Gen 1215U - (8 GB/5…       51266.0
..                                                 …           …
950  DELL Intel Core i3 13th Gen 1305U - (8 GB/512 …       51944.0
951  HP Pavilion AMD Ryzen 5 Hexa Core AMD R5-5600H…       73544.0
952  HP Intel Core i3 12th Gen 1215U - (16 GB/512 G…       52721.0
958  Lenovo IdeaPad Slim 3 Intel Core i5 12th Gen 1…       69890.0
959  HP (15s-fq5007TU) Intel Core i3 12th Gen 1215U…       51134.0

[444 rows x 2 columns]
```

[8]:
```
"""Count total number of products which has more than 3000 ratings"""

df['Rating'] = pd.to_numeric(df['Rating'].str.replace(',', ''),errors='coerce')
print(f"\nTotal number of products which has more than 3000 ratings:␣
 ↪{df[df['Rating'] > 3000]['Product Name'].count()}")
```

```
Total number of products which has more than 3000 ratings: 159
```

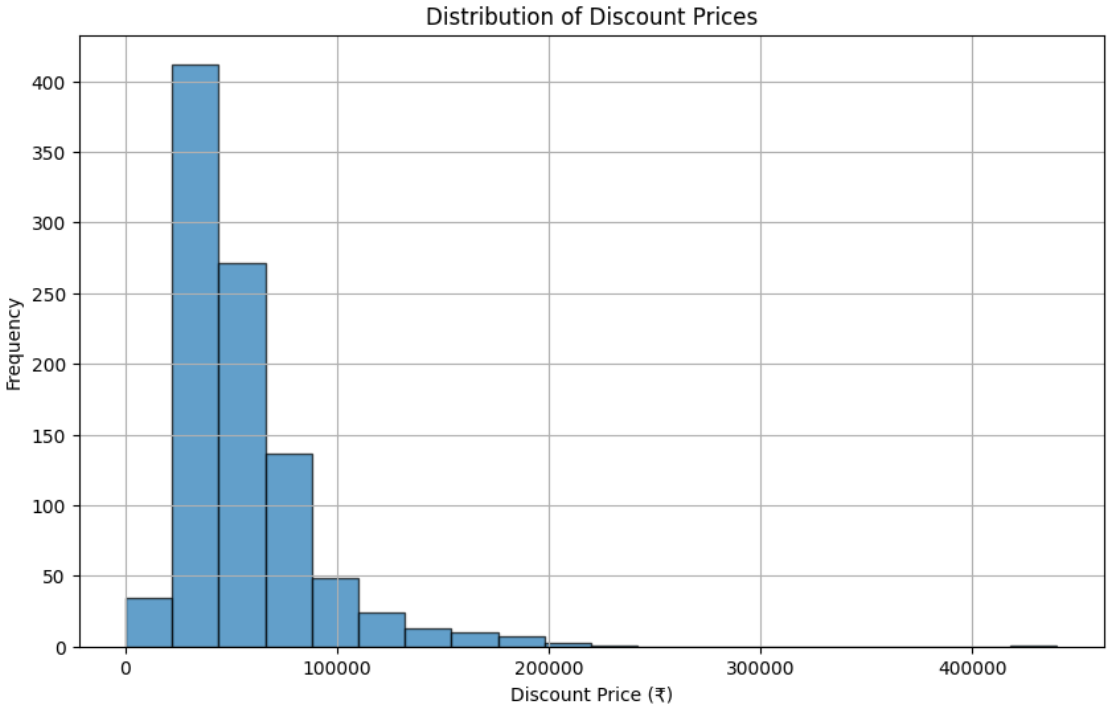
[9]:
```
"""Display the product name which has maximum and minimum review count"""

print(f"\nDisplay the Product having maximum reviews \n{df.loc[df['Reviews'].
 ↪idxmax()]}")
print(f"\nDisplay the Product having minimum reviews \n{df.loc[df['Reviews'].
 ↪idxmin()]}")
```

```
Display the Product having maximum reviews
Product Name      realme Book(Slim) Intel Evo Intel Core i5 11th…
ProductID                                          COMG5YDPM8FZZWMQ
Actual price                                                69999.0
Discount price                                              47999.0
Stars                                                           4.3
Rating                                                       7975.0
Reviews                                                      1042.0
Description       Powered by 11th Gen Intel Evo Core i5 Processo…
Link              https://www.flipkart.com/realme-book-slim-inte…
Total Sales                                                 22000.0
Name: 419, dtype: object

Display the Product having minimum reviews
Product Name      MSI Thin 15 Intel Core i7 12th Gen 12650H - (8…
ProductID                                          COMGZW37ZX66DBHF
Actual price                                                83990.0
Discount price                                              67990.0
Stars                                                           0.0
Rating                                                          NaN
Reviews                                                         0.0
Description       Intel Core i7 Processor (12th Gen)8 GB DDR4 RA…
Link              https://www.flipkart.com/msi-thin-15-intel-cor…
Total Sales                                                 16000.0
Name: 1, dtype: object
```

[10]:
```python
"""Show the statistical analysis of discount prices"""

print(f"Show the statistical analysis of discount prices:\n{np.
  ↪round(df['Discount price'].describe(),decimals=2)}")

# Plot histogram
plt.figure(figsize=(10, 6))
plt.hist(df['Discount price'], bins=20, edgecolor='black', alpha=0.7)
plt.title('Distribution of Discount Prices')
plt.xlabel('Discount Price ( )')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

```
Show the statistical analysis of discount prices:
count       960.00
mean      56030.32
std       33527.08
min           0.00
25%       35990.00
50%       48745.00
75%       67115.00
```

```
max      439990.00
Name: Discount price, dtype: float64
```



[11]:
```
"""Show how many product having poor rating(1-2), Average rating (2-3), Good␣
 ↪rating (3-4) and Excellent rating (4-5)."""

poorRating = df[df['Stars'] < 2].count()
averageRating = df[(df['Stars'] >= 2 ) & (df['Stars'] < 3)].count()
goodRating = df[(df['Stars'] >= 3 ) & (df['Stars'] < 4)].count()
excellentRating = df[(df['Stars'] >= 4 ) & (df['Stars'] <= 5)].count()

print(f"Show how many product having :")
print(f"\tPoor Rating (1-2): {poorRating['Product Name']}")
print(f"\tAverage Rating (2-3): {averageRating['Product Name']}")
print(f"\tGood Rating (3-4): {goodRating['Product Name']}")
print(f"\tExcellent Rating (4-5): {excellentRating['Product Name']}")
```

```
Show how many product having :
        Poor Rating (1-2): 150
        Average Rating (2-3): 9
        Good Rating (3-4): 161
        Excellent Rating (4-5): 640
```

0.0.3 Question - 2

[12]:
```
result = pd.read_csv("./dataset/olympic/Olympic_Results.csv")
result.shape
```

[12]:
```
(7394, 12)
```

[13]:
```
result.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7394 entries, 0 to 7393
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   result_id            7394 non-null   int64 
 1   event_title          7394 non-null   object
 2   edition              7394 non-null   object
 3   edition_id           7394 non-null   int64 
 4   sport                7394 non-null   object
 5   sport_url            7394 non-null   object
 6   result_date          7394 non-null   object
 7   result_location      7393 non-null   object
 8   result_participants  7394 non-null   object
 9   result_format        7394 non-null   object
 10  result_detail        7394 non-null   object
 11  result_description   7394 non-null   object
dtypes: int64(2), object(10)
memory usage: 693.3+ KB
```

[14]:
```
# Remove a row if any value is missing
result = result.dropna()
result.shape
```

[14]:
```
(7393, 12)
```

[15]:
```
# Drop Row if it contains `na` in columns `result_format`, `result_detail` and␣
 ↪`result_description`
for i in ['result_format','result_detail','result_description']:
    result = result[result[i] != 'na']
result.shape
```

[15]:
```
(368, 12)
```

[16]:
```
# Drop Column that we don't required `sport_url`, `edition` and `edition_id`
print(f"Before Drop: {result.columns}")
result = result.drop(columns=['sport_url','edition','edition_id'])
print(f"\nAfter Drop: {result.columns}")
```

```
Before Drop: Index(['result_id', 'event_title', 'edition', 'edition_id',
```

```
'sport',
       'sport_url', 'result_date', 'result_location', 'result_participants',
       'result_format', 'result_detail', 'result_description'],
      dtype='object')

After Drop: Index(['result_id', 'event_title', 'sport', 'result_date',
'result_location',
       'result_participants', 'result_format', 'result_detail',
       'result_description'],
      dtype='object')
```

[17]:
```
result = result.drop_duplicates(subset='event_title',keep='first')
result.shape
```

[17]:
```
(67, 9)
```

[18]:
```
# Sort in ascending order
print(f"Before sorting:\n{result['result_id'].head()}")
result = result.sort_values(by='result_id')
result.head()
```

```
Before sorting:
1     1626
2       76
39    1592
76    1504
91    2403
Name: result_id, dtype: int64
```

[18]:

	result_id	event_title	sport	result_date	result_location	result_participants	result_format
1213	6	Skeleton, Men	Skeleton	3 - 4 February 1948	Cresta Run, St. Moritz / Celerina	15 from 6 countries	Six runs, total time determined placement. Fir…
5908	46	Singles, Men1	Luge	11 - 13 February 1968	Piste de Luge, Villard-de-Lans	50 from 14 countries	Three runs, total time determined placement.
5909	54	Singles, Women1	Luge	11 - 13 February 1968	Piste de Luge, Villard-de-Lans	26 from 10 countries	Three runs, total time determined placement.
2	76	Singles, Men	Luge	4 - 7 February 1976	Kunsteis-Bob- und Rodelbahn, Igls	43 from 15 countries	Four runs, total time determined placement.
1620	154	Singles, Women	Luge	11 - 12 February 1992	Piste de Bobsleigh et Luge, La Plagne	24 from 12 countries	

```
1620        Four runs, total time determined placement.

                                                result_detail  \
1213  Curves: ? / 15Length: 870 m / 1231 mStart Alti…
5908  Curves: 14Length: 1000 mStart Altitude: 1110 m…
5909  Curves: ?Length: ?Start Altitude: ?Vertical Dr…
2     Curves: 14Length: 1220 mStart Altitude: ?Verti…
1620  Curves: 14Length: 1143 mStart Altitude: 1652 m…

                                     result_description
1213  Twenty years after the first Olympic skeleton …
5908  Although the Germans, who dominated the podium…
5909  The women's competition was overshadowed by a …
2     Once more, the competitors from East and West …
1620  German sliders had continued to dominate the w…
```

[19]:
```
medal = pd.read_csv("./dataset/olympic/Olympic_Games_Medal_Tally.csv")
print(medal.shape)
medal.columns
```

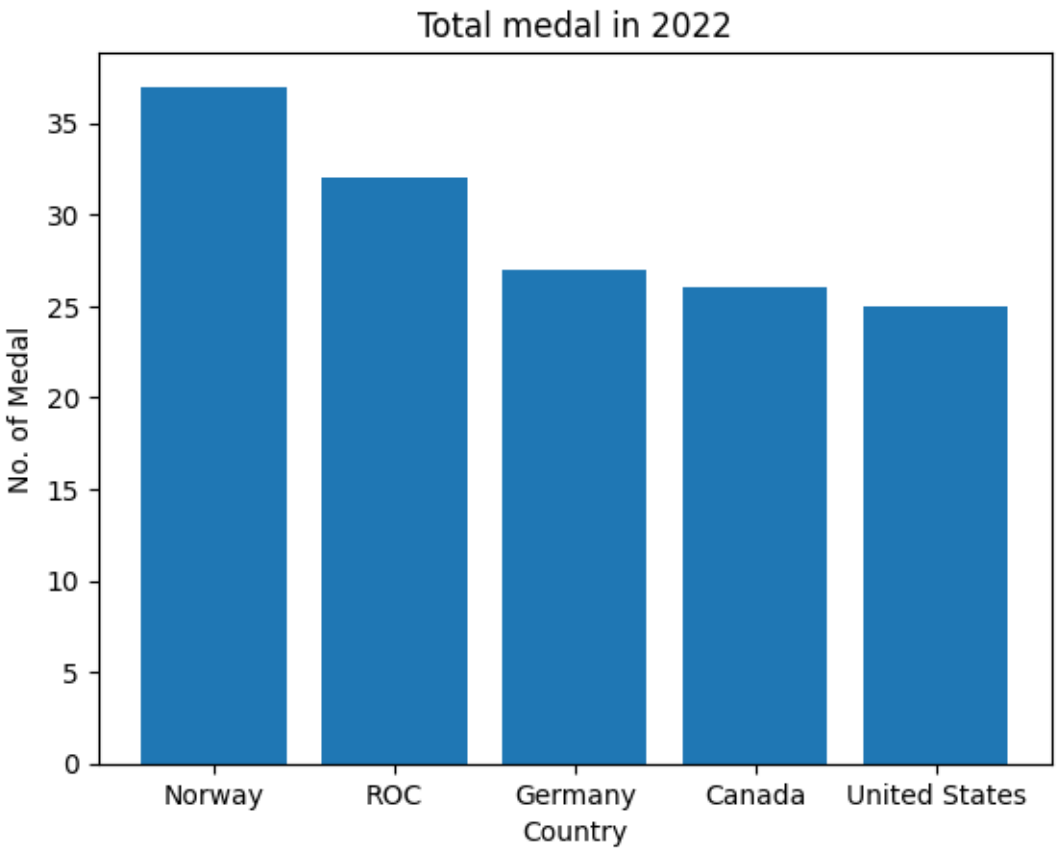
```
(1807, 9)
```

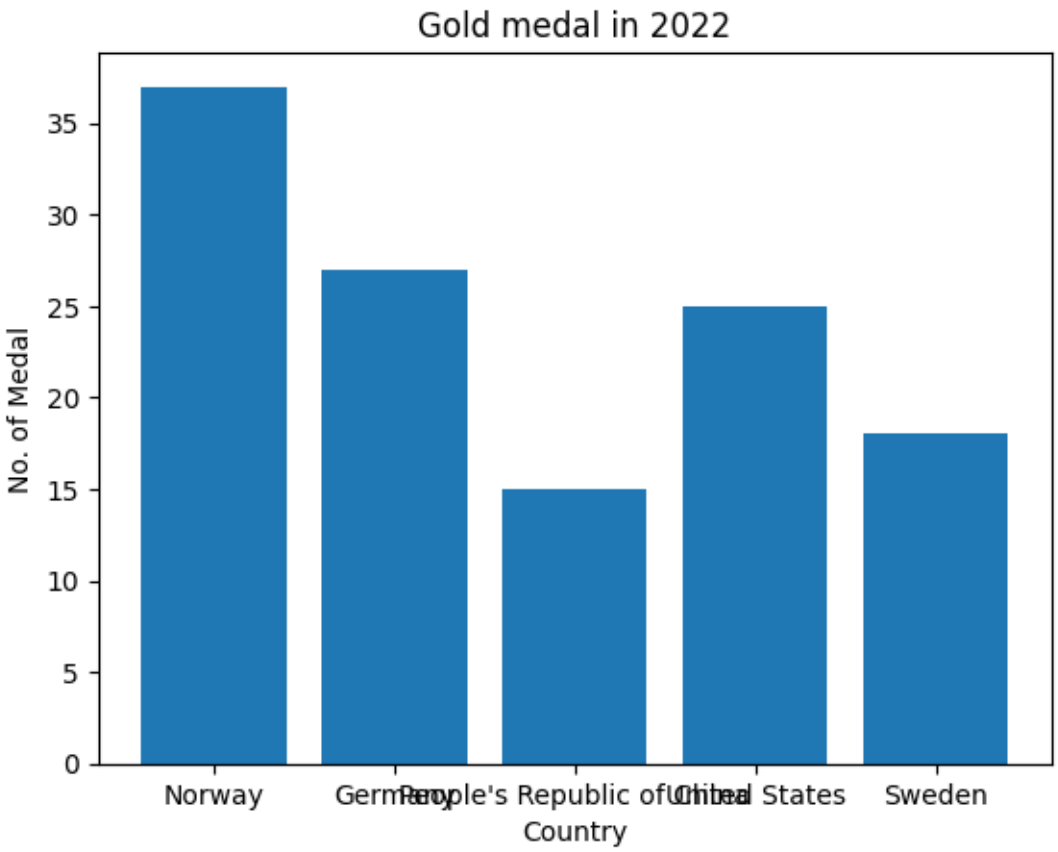
[19]:
```
Index(['edition', 'edition_id', 'year', 'country', 'country_noc', 'gold',
       'silver', 'bronze', 'total'],
      dtype='object')
```

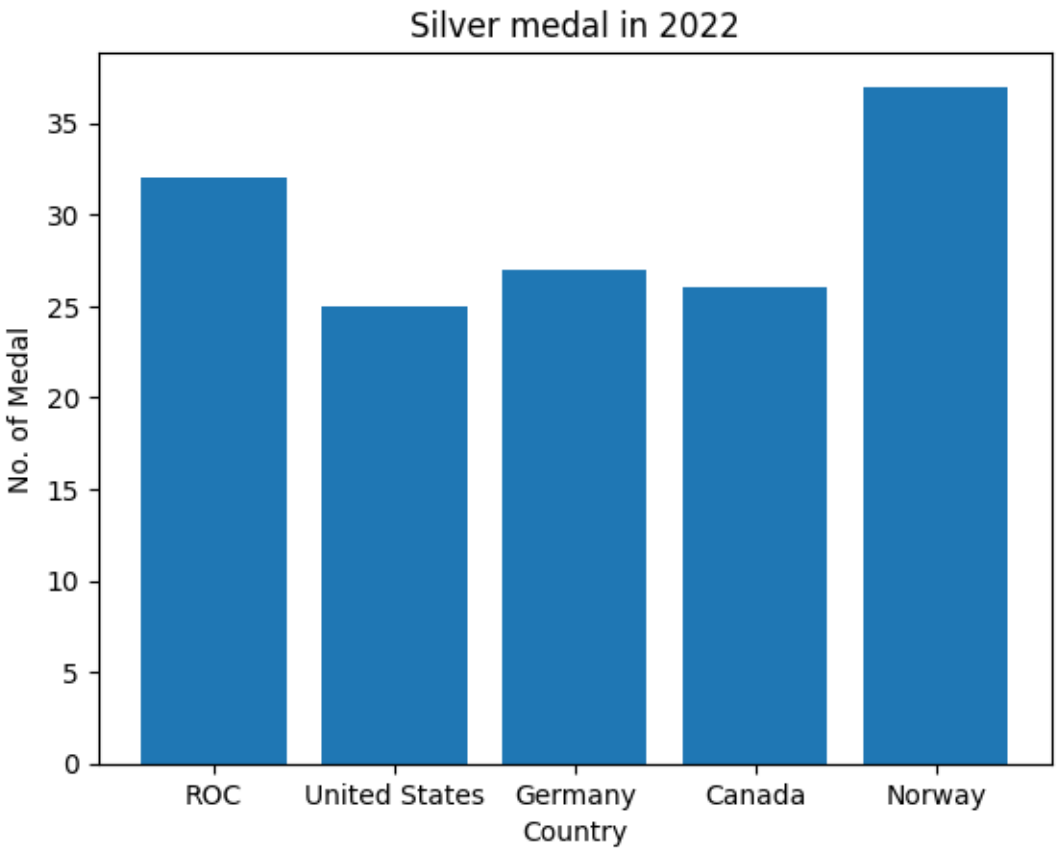
[20]:
```
"""Medal list after 2020"""
medal = medal[medal.year >= 2020]
medal.shape
```

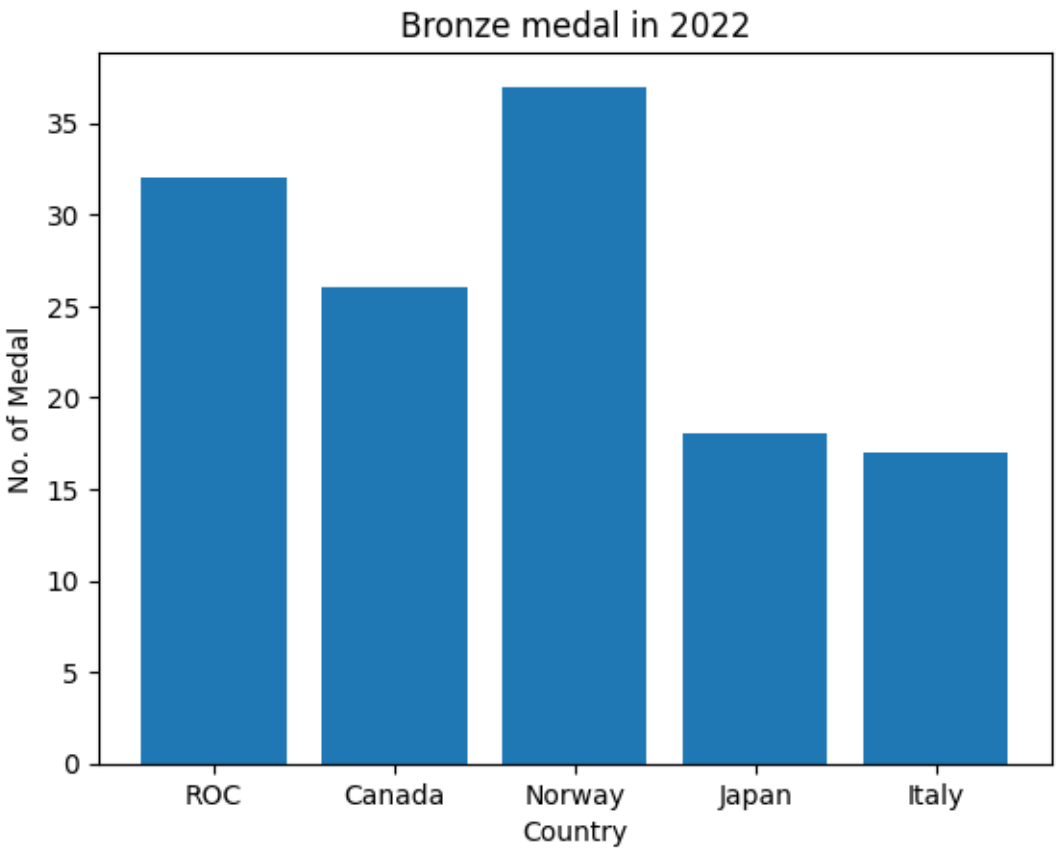
[20]:
```
(122, 9)
```

[21]:
```
"""Total Medal win by the top 5 country"""
medal = medal[medal.year == 2022]
total = medal.sort_values(by='total',ascending=False)
gold = medal.sort_values(by='gold',ascending=False)
silver = medal.sort_values(by='silver',ascending=False)
bronze = medal.sort_values(by='bronze',ascending=False)

for i in [(total,"Total"),(gold,"Gold"),(silver,"Silver"),(bronze,"Bronze")]:
    x = i[0].head(5)['country'].tolist()
    y = i[0].head(5)['total'].tolist()
    plt.title(f"{i[1]} medal in 2022")
    plt.bar(x,y)
    plt.xlabel("Country")
    plt.ylabel("No. of Medal")
    plt.show()
```


Total medal in 2022
No. of Medal
0
5
10
15
20
25
30
35
Norway
ROC
Germany
Canada
United States
Country


Gold medal in 2022
No. of Medal
0
5
10
15
20
25
30
35
Norway
Germany
People's Republic of China
United States
Sweden
Country


Silver medal in 2022
No. of Medal
0
5
10
15
20
25
30
35
ROC
United States
Germany
Canada
Norway
Country


Bronze medal in 2022
No. of Medal
0
5
10
15
20
25
30
35
ROC
Canada
Norway
Japan
Italy
Country

ASSIGNMENT - 2

Assignment-2

October 25, 2024

0.0.1 Imports

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

0.0.2 Question 1 : Take employee_data Dataset and Perform following task:

[36]:
```
empDf = pd.read_csv('./dataset/Employee_data.csv')
empDf.head()
```

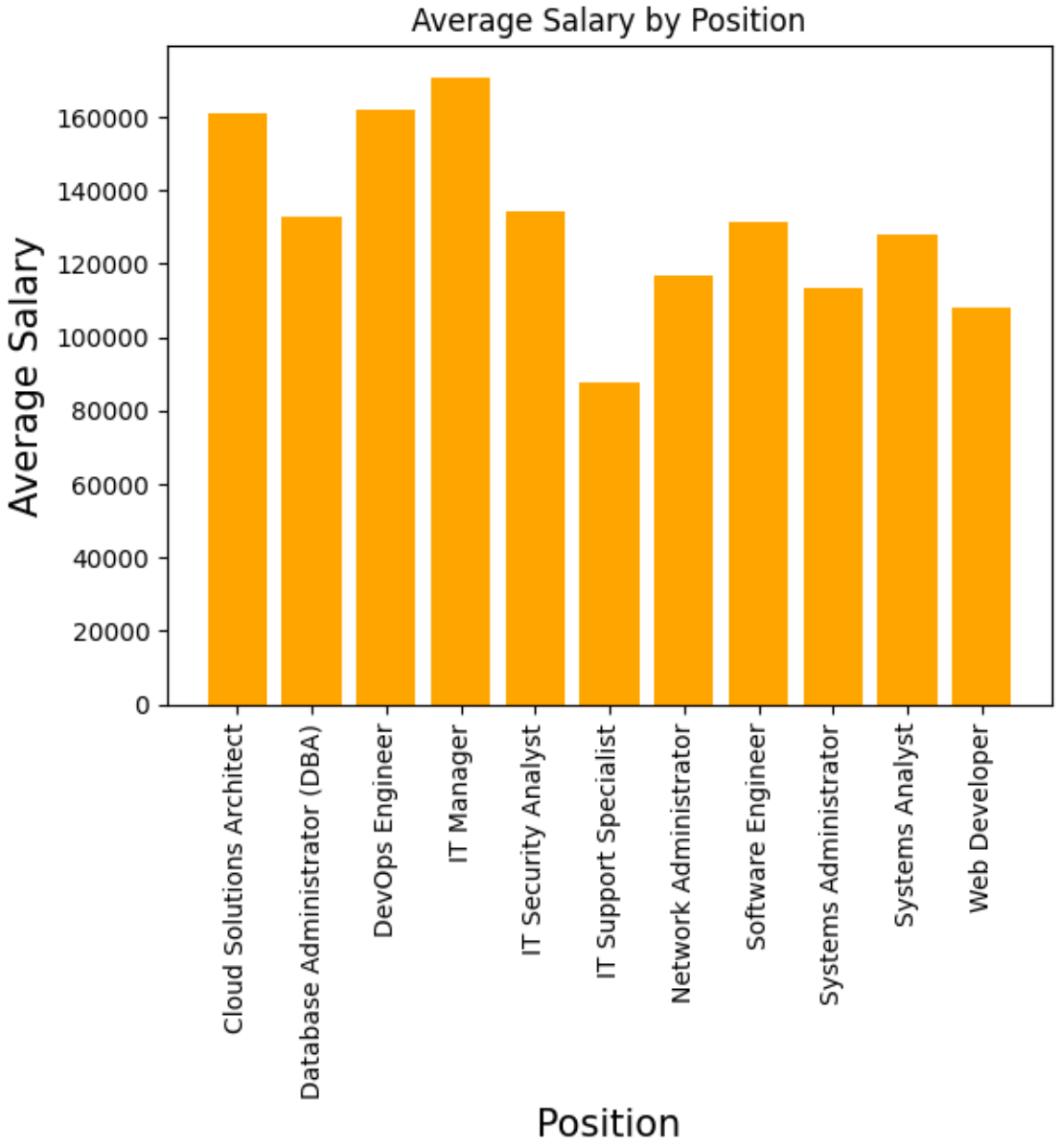
[36]:

	ID	Gender	Experience (Years)	Position	Salary
0	1	F	4	DevOps Engineer	109976
1	2	M	6	DevOps Engineer	120088
2	3	M	17	Web Developer	181301
3	4	M	7	Systems Administrator	77530
4	5	F	13	Systems Administrator	152397

[37]:
```
# Display Average Salary of each Position also plot same on graph
avgSalary = empDf.groupby('Position')['Salary'].mean()
print(avgSalary)

plt.bar(avgSalary.index,avgSalary, color='orange')
plt.title('Average Salary by Position')
plt.xlabel('Position',fontsize=15)
plt.ylabel('Average Salary',fontsize=15)
plt.xticks(rotation=90)
plt.show()
```

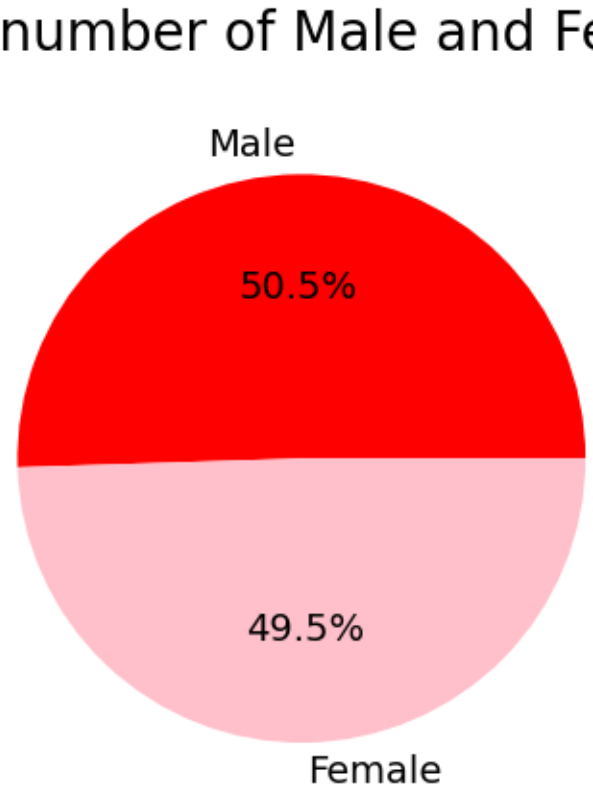
```
Position
Cloud Solutions Architect       160841.633333
Database Administrator (DBA)    132864.552632
DevOps Engineer                 161859.081081
IT Manager                      170711.550000
IT Security Analyst             134440.820513
IT Support Specialist            87683.806452
Network Administrator           116865.064516
Software Engineer               131357.416667
Systems Administrator           113117.447368
```

```
Systems Analyst                 127658.189189
Web Developer                   108238.116279
Name: Salary, dtype: float64
```

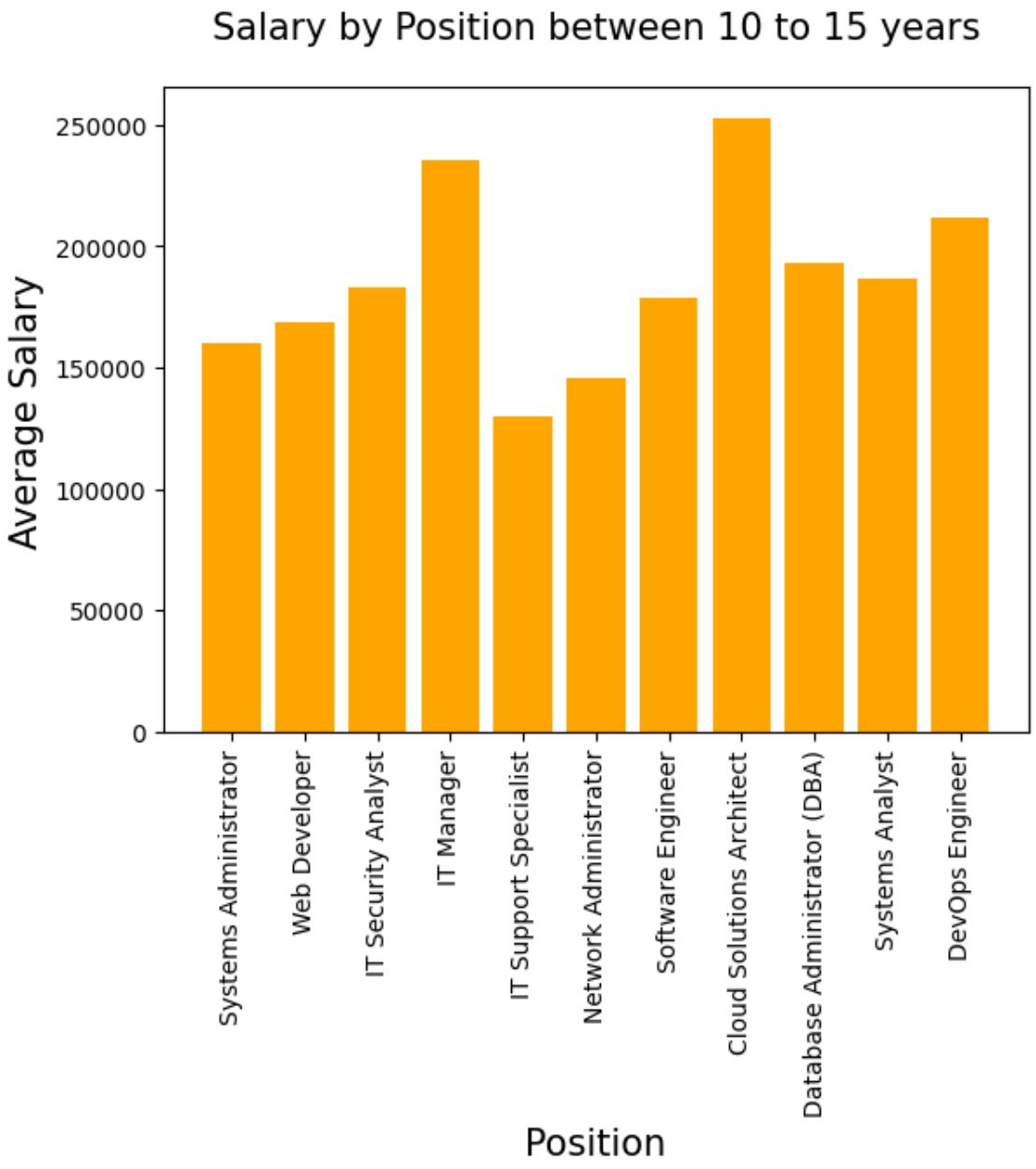


[38]:
```
# Display total number of Male and Female employees and plot same on Chart
male = empDf[empDf['Gender'] == 'M'].count()['Gender']
female = empDf[empDf['Gender'] == 'F'].count()['Gender']

plt.pie([male,female],colors=['red','pink'],  labels=['Male',␣
  ↪'Female'],autopct='%1.1f%%',textprops={'fontsize':14})
```

```
plt.title('Total number of Male and Female',fontsize=20,pad=20)
plt.show()
```

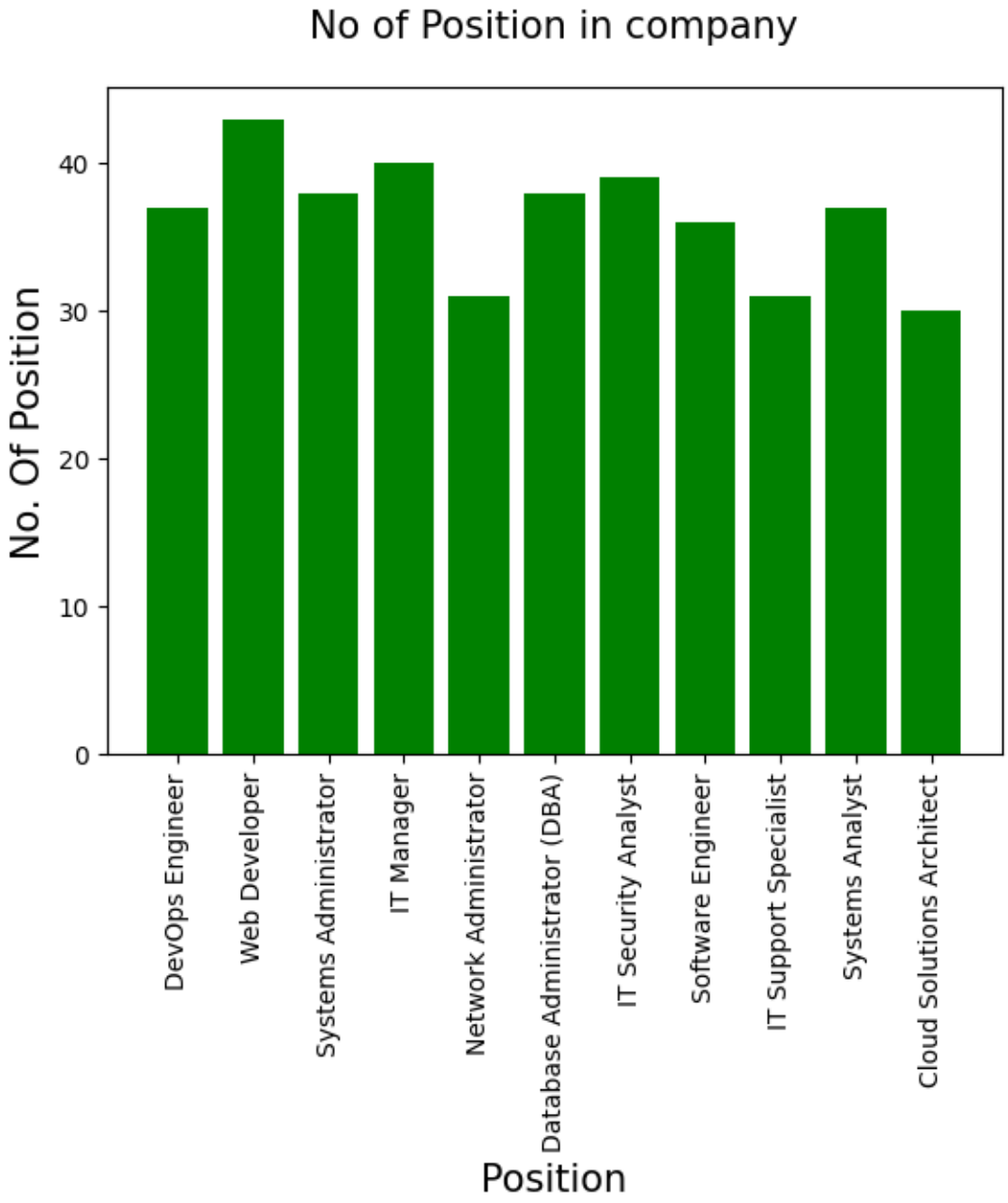


[39]:
```
# Show using chart, how much salary is earn by the employee who having␣
 ↪experience between 10 to 15 years

experienceDf = empDf[(empDf['Experience (Years)'] >= 10) & (empDf['Experience␣
 ↪(Years)'] <= 15)][['Position','Salary']]

plt.bar(experienceDf['Position'],experienceDf['Salary'], color='orange')
plt.title('Salary by Position between 10 to 15 years',fontsize=15,pad=20)
plt.xlabel('Position',fontsize=15)
plt.ylabel('Average Salary',fontsize=15)
plt.xticks(rotation=90)
plt.show()
```



[40]:
```
# Display a bar chart for number of position in company

noOfPosition = dict()
for i in empDf['Position'].unique():
    noOfPosition[i] = int(empDf[empDf['Position'] ==i]['Position'].count())

plt.bar(noOfPosition.keys(),noOfPosition.values(),color=['green'])
plt.title('No of Position in company',fontsize=15,pad=20)
plt.xlabel('Position',fontsize=15)
```

```
plt.ylabel('No. Of Position',fontsize=15)
plt.xticks(rotation=90)
plt.show()
```



[41]:
```
# Analysis which position is better in terms of salary
print(f"The best position in terms of salary is '{avgSalary.idxmax()}' with an␣
 ↪average salary of {avgSalary.max()}.")
```

```
The best position in terms of salary is 'IT Manager' with an average salary of
170711.55.
```

0.0.3 2. Take a Flipkart-Laptops Dataset and do the data preprocessing using pandas and visualize the important data from it using different charts.

[42]:
```
fkDf = pd.read_excel('./dataset/Flipkart-Laptops.xlsx')
print("Shape :",fkDf.shape)
print("Columns :",fkDf.columns)
```

```
Shape : (960, 10)
Columns : Index(['Product Name', 'ProductID', 'Product image', 'Actual price',
       'Discount price', 'Stars', 'Rating', 'Reviews', 'Description', 'Link'],
      dtype='object')
```

[43]:
```
# Remove Column having null value and Duplicate values
fkDf = fkDf.dropna(how='all',axis=1)
fkDf = fkDf.drop_duplicates()
fkDf.shape
```

[43]:
```
(960, 9)
```

[44]:
```
# Remove Row where value is 'NIL'
fkDf.replace('NIL', np.nan, inplace=True)
fkDf.dropna(inplace=True)
fkDf.shape
```

```
/tmp/ipykernel_141098/2738969124.py:2: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  fkDf.replace('NIL', np.nan, inplace=True)
```

[44]:
```
(808, 9)
```

[45]:
```
# Remove unused columns
fkDf.drop(['Description','Link'],axis=1,inplace=True)
fkDf.shape
```

[45]:
```
(808, 7)
```

[46]:
```
# Convert string row into number
print(fkDf.dtypes,'\n')
fkDf['Rating'] = fkDf['Rating'].str.replace(' Ratings','').str.replace(',','').
 ↪astype(float)
fkDf['Reviews'] = fkDf['Reviews'].str.replace(' Reviews','').str.
 ↪replace(',','').astype(float)
fkDf['Discount price'] = fkDf['Discount price'].astype(float)
```

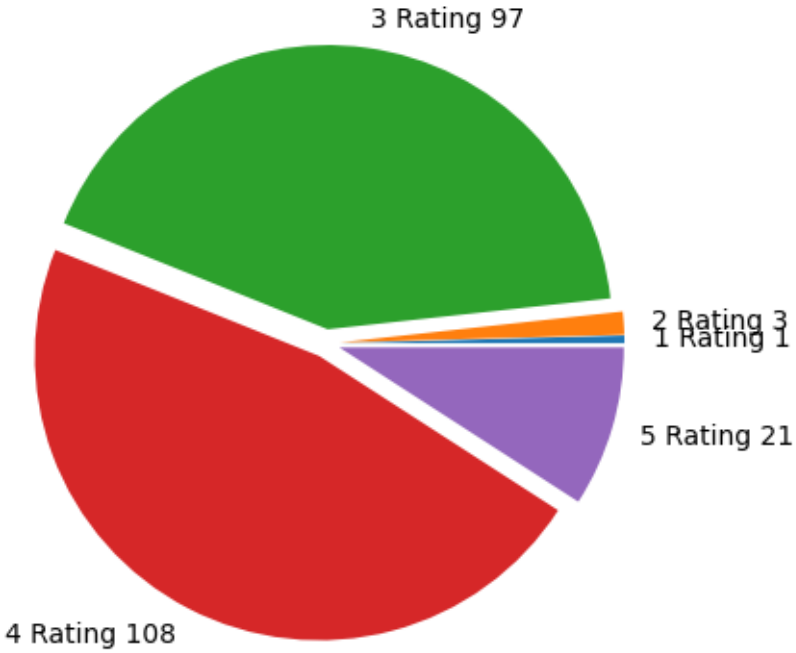
```
print(fkDf.dtypes)
fkDf.columns
```

```
Product Name       object
ProductID          object
Actual price      float64
Discount price     object
Stars             float64
Rating             object
Reviews            object
dtype: object

Product Name       object
ProductID          object
Actual price      float64
Discount price    float64
Stars             float64
Rating            float64
Reviews           float64
dtype: object
```

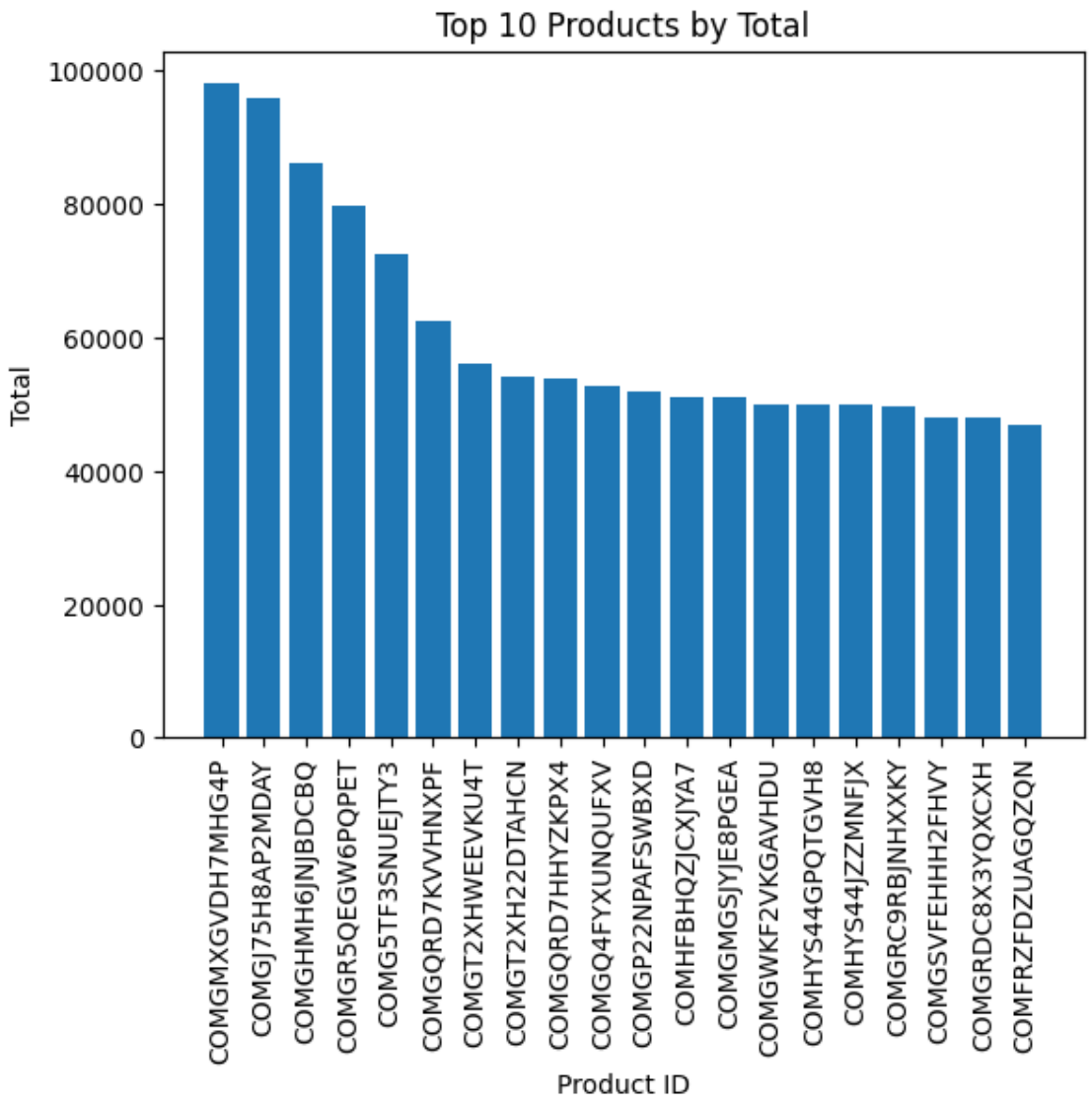
[46]:
```
Index(['Product Name', 'ProductID', 'Actual price', 'Discount price', 'Stars',
       'Rating', 'Reviews'],
      dtype='object')
```

[47]:
```
rate = dict()
label = []
for i in range(5):
    rate[i] = fkDf[(fkDf['Rating'] >= i)&(fkDf['Rating'] <= i+1)].
 ↪count()['Rating']
    label.append(f'{i+1} Rating {rate[list(rate.keys())[-1]]}')

total = sum(rate.values())
percentage = [(i/total)*100 for i in rate.values()]
print(sum(percentage))
plt.pie(percentage,labels=label,explode=tuple(0.05 for _ in range(len(rate)) ))
plt.show()
```

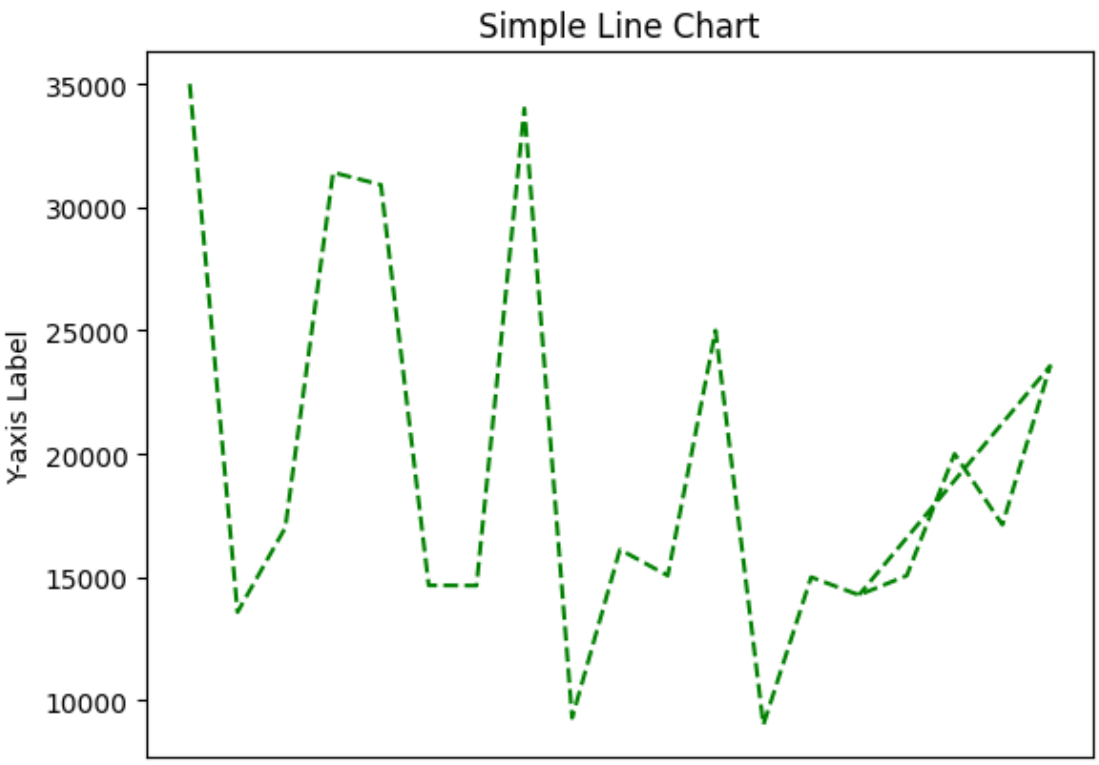
```
100.0
```



[57]:
```
# Total prise data
fkDf['Total'] = fkDf['Actual price'] - fkDf['Discount price']
top10DF= fkDf.sort_values(by='Total', ascending=False).head(20)
plt.bar(top10DF['ProductID'],top10DF['Total'])
plt.xticks(rotation=90)
plt.xlabel('Product ID')
plt.ylabel('Total')
plt.title('Top 10 Products by Total')
plt.show()
```



[69]:
```
plt.plot(fkDf['ProductID'].head(20), fkDf['Total'].head(20), linestyle='--',␣
 ↪color='g')

# Add labels and title
plt.ylabel("Y-axis Label")
plt.title("Simple Line Chart")
plt.xticks([])
# Display the plot
plt.show()
```


Simple Line Chart
Y-axis Label
35000
30000
25000
20000
15000
10000

ASSIGNMENT - 3

==========

app.py

==========

```
from flask import Flask, render_template,request
import pandas as pd
import matplotlib.pyplot as plt
import io
import base64
import numpy as np


app = Flask(__name__)


itemsq1 = ['Average Salary Of Each Position', 'Total number of male and female employee', 'Salary earn by experience example between 10 to 15 year', 'Num of Position in Company','Which position is better in terms of salary']
itemsq2 = ['Rating of Products','Top 10 Products by Total','Price Growth Total']


plt.switch_backend('Agg')
@app.route("/")
def index():
    return render_template("welcome.html",itemsq1=itemsq1,itemsq2=itemsq2)


@app.route("/question-1")
def question1():
    plot_urls = []
    empDf = pd.read_csv("./dataset/Employee_data.csv")
    avgSalary = empDf.groupby('Position')['Salary'].mean()


    # Q1
    img = io.BytesIO()
```

```
    plt.figure(figsize=(12, 8))
    plt.bar(avgSalary.index, avgSalary, color='orange',)
    plt.xlabel('Position', fontsize=20)
    plt.ylabel('Average Salary', fontsize=20)
    plt.xticks(rotation=90, fontsize=12)
    plt.yticks(fontsize=12)
    plt.tight_layout()
    plt.savefig(img,format='png')
    plt.close()
    img.seek(0)
    plot_urls.append(base64.b64encode(img.getvalue()).decode('utf-8'))
    img.truncate(0)
    img.seek(0)

    # Q2
    plt.figure(figsize=(12, 8))
    male = empDf[empDf['Gender'] == 'M'].count()['Gender']
    female = empDf[empDf['Gender'] == 'F'].count()['Gender']
    plt.pie([male,female],colors=['red','pink'],  labels=['Male',
'Female'],autopct='%1.1f%%',textprops={'fontsize':20})
    plt.tight_layout()
    plt.savefig(img,format='png')
    plt.close()
    img.seek(0)
    plot_urls.append(base64.b64encode(img.getvalue()).decode('utf-8'))
    img.truncate(0)
    img.seek(0)

    # Q3
    plt.figure(figsize=(12, 8))
```

```
    experienceDf = empDf[(empDf['Experience (Years)'] >= 10) & (empDf['Experience (Years)'] <= 15)][['Position','Salary']]
    plt.bar(experienceDf['Position'],experienceDf['Salary'], color='mediumorchid')
    plt.xlabel('Position',fontsize=15)
    plt.ylabel('Average Salary',fontsize=15)
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.savefig(img,format='png')
    plt.close()
    img.seek(0)
    plot_urls.append(base64.b64encode(img.getvalue()).decode('utf-8'))
    img.truncate(0)
    img.seek(0)

    # Q4
    noOfPosition = dict()
    for i in empDf['Position'].unique():
        noOfPosition[i] = int(empDf[empDf['Position'] ==i]['Position'].count())
    plt.figure(figsize=(12, 8))
    plt.bar(noOfPosition.keys(),noOfPosition.values(),color=['limegreen'])
    plt.xlabel('Position',fontsize=15)
    plt.ylabel('No. Of Position',fontsize=15)
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.savefig(img,format='png')
    plt.close()
    img.seek(0)
    plot_urls.append(base64.b64encode(img.getvalue()).decode('utf-8'))
    img.truncate(0)
    img.seek(0)
```

```
    # Q5
    plt.figure(figsize=(12, 8))
    avg_salary = empDf.groupby('Position')['Salary'].mean()
    labels = avg_salary.index
    sizes = avg_salary.values
    plt.figure(figsize=(8, 6))
    plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
    plt.axis('equal')
    plt.tight_layout()
    plt.savefig(img,format='png')
    plt.close()
    img.seek(0)
    plot_urls.append(base64.b64encode(img.getvalue()).decode('utf-8'))
    img.truncate(0)
    img.seek(0)
    return render_template("q1.html",image=plot_urls,label=itemsq1)

@app.route("/question-2")
def question2():
    plot_urls = []
    fkDf = pd.read_excel("./dataset/Flipkart-Laptops.xlsx")
    fkDf.replace('NIL', np.nan, inplace=True)
    fkDf.drop_duplicates(inplace=True)
    fkDf.dropna(how='all',axis=1,inplace=True)
    fkDf.dropna(inplace=True)
    # NOTE - Drop Unused columns
    fkDf.drop(['Description','Link'],axis=1,inplace=True)
    # NOTE - Convert String into number
    fkDf['Rating'] = fkDf['Rating'].str.replace(' Ratings','').str.replace(',','').astype(float)
```

```
fkDf['Reviews'] = fkDf['Reviews'].str.replace(' Reviews','').str.replace(',','').astype(float)
fkDf['Discount price'] = fkDf['Discount price'].astype(float)
img = io.BytesIO()

# Q1
rate = dict()
label = []
for i in range(5):
    rate[i] = fkDf[(fkDf['Rating'] >= i)&(fkDf['Rating'] <= i+1)].count()['Rating']
    label.append(f'{i+1} Rating {rate[list(rate.keys())[-1]]}')
total = sum(rate.values())
percentage = [(i/total)*100 for i in rate.values()]
print(sum(percentage))
plt.figure(figsize=(12, 8))
plt.pie(percentage,labels=label,explode=tuple(0.05 for _ in range(len(rate))))
plt.tight_layout()
plt.savefig(img,format='png')
plt.close()
img.seek(0)
plot_urls.append(base64.b64encode(img.getvalue()).decode('utf-8'))
img.truncate(0)
img.seek(0)

# Q2
plt.figure(figsize=(12, 8))
fkDf['Total'] = fkDf['Actual price'] - fkDf['Discount price']
top10DF= fkDf.sort_values(by='Total', ascending=False).head(20)
plt.bar(top10DF['ProductID'],top10DF['Total'])
plt.xticks(rotation=90)
plt.xlabel('Product ID')
```

```
    plt.ylabel('Total')
    plt.title('Top 10 Products by Total')
    plt.tight_layout()
    plt.savefig(img,format='png')
    plt.close()
    img.seek(0)
    plot_urls.append(base64.b64encode(img.getvalue()).decode('utf-8'))
    img.truncate(0)
    img.seek(0)

    # Q3
    plt.figure(figsize=(12, 8))
    plt.plot(fkDf['ProductID'].head(20), fkDf['Total'].head(20), linestyle='--', color='g')
    plt.ylabel("Y-axis Label")
    plt.title("Simple Line Chart")
    plt.xticks([])
    plt.tight_layout()
    plt.savefig(img,format='png')
    plt.close()
    img.seek(0)
    plot_urls.append(base64.b64encode(img.getvalue()).decode('utf-8'))
    img.truncate(0)
    img.seek(0)
    return render_template("q1.html",image=plot_urls,label=itemsq2)

@app.route('/graph', methods=['POST'])
def graph():
    title = request.form.get('title')
    src = request.form.get('src')
    return render_template('graph.html', title=title, src=src)
```

```
if __name__ == "__main__":
    app.run(debug=True)
```

==========

util/footer.html

==========

```
<footer class="bg-indigo-900 text-white mt-10">
    <div class="container mx-auto px-4 py-4 text-center">
        <h2 class="text-lg font-bold">Assignment-3</h2>
        <p class="text-sm">© 2024 Your Name. All rights reserved.</p>
    </div>
</footer>
```

==========

util/header.html

==========

```
<header class="bg-indigo-700 text-white fixed w-full">
    <div class="container mx-auto px-4 py-4 flex justify-between items-center">
        <div class="text-lg font-bold">
            <a href="{{ url_for('index') }}" class="hover:text-gray-400">Assignment-3</a>
        </div>
        <nav>
            <ul class="flex space-x-6">
                <li>
                    <a href="{{ url_for('index') }}" class="hover:text-indigo-400 hover:underline
hover:underline-offset-4">Home</a>
                </li>
                <li>
```

```
                <a href="{{ url_for('question1')}}" class="hover:text-indigo-400 hover:underline
hover:underline-offset-4">Question-1</a>

            </li>

            <li>

                <a href="{{ url_for('question2')}}" class="hover:text-indigo-400 hover:underline
hover:underline-offset-4">Question-2</a>

            </li>

        </ul>

    </nav>

  </div>

</header>
```

==========

util/macros.html

==========

```
{% macro image(title, src) %}

<div class="mb-10">

  <div class="text-2xl font-bold">{{ title }}</div>

  <form action="/graph" method="POST">

    <input type="hidden" name="title" value="{{ title }}">

    <input type="hidden" name="src" value="{{ src }}">

    <button type="submit">

      <img class="h-auto w-full max-w-4xl" src="{{ src }}"
alt="{{ title }}">

    </button>

  </form>

</div>

{% endmacro %}
```

==========

graph.html

==========

```
{% extends "index.html" %}
{% block title %}Graphs {% endblock %}
{% block body %}
    <div class="text-3xl font-bold">{{ title }}</h1>
    <img src="{{ src }}">
{% endblock %}
```

==========

index.html

==========

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://cdn.tailwindcss.com"></script>
        <title>{% block title required %}{% endblock %} - Assignment-3</title>
</head>
<body>
        {% include 'util/header.html' %}
        <div class="p-3 pt-20 min-h-screen">
                {% block body required %} {% endblock %}
        </div>
        {% include 'util/footer.html' %}
</body>
</html>
```

==========

q1.html

==========

```
{% extends "index.html" %}
{% block title %}Q.1{% endblock %}
{% block body %}
{% import 'util/macros.html' as macros %}
<div>
    <div class="grid grid-cols-2 space-x-5">
        {% for img in image %}
        {{ macros.image(label[loop.index-1], img) }}
        {% endfor%}
    </div>

</div>
{% endblock %}
```

==========

q2.html

==========

```
{% extends "index.html" %}
{% block title %}Q.2{% endblock %}
{% block body %}
{% import 'util/macros.html' as macros %}
<div>
    <div class="grid grid-cols-2 space-x-5">
        {% for img in image %}
        {{ macros.image(label[loop.index-1], img) }}
```

```
        {% endfor%}
    </div>


</div>
{% endblock %}
```

==========

welcome.html

==========

```
{% extends "index.html" %}
{% block title %}Welcome{% endblock %}
{% block body %}
<div class="grid grid-cols-2 space-x-10">
    <div>
        <div class="text-3xl mt-5 mb-5">Question - 1</div>
        <div class="font-bold text-lg mb-2">List of Charts</div>
        <ul class="p-5 bg-indigo-100 w-full text-indigo-900 rounded-lg font-mono border-2 border-
indigo-300">
            {% for item in itemsq1 %}
            <li class="mb-4">{{ loop.index }}) {{ item }}.</li>
            {% endfor %}
        </ul>
    </div>
    <div>
        <div class="text-3xl mt-5 mb-5">Question - 2</div>
        <div class="font-bold text-lg mb-2">List of Charts</div>
        <ul class="p-5 bg-indigo-100 w-full text-indigo-900 rounded-lg font-mono border-2 border-
indigo-300">
            {% for item in itemsq2 %}
            <li class="mb-4">{{ loop.index }}) {{ item }}.</li>
```

```
            {% endfor %}
        </ul>
    </div>
</div>
{% endblock %}
```

==========

OUTPUT :

==========

Assignment-3 Home Question-1 Question-2

Question - 1

List of Charts

1) Average Salary Of Each Position.
2) Total number of male and female employee.
3) Salary earn by experience example between 10 to 15 year.
4) Num of Position in Company.
5) Which position is better in terms of salary.

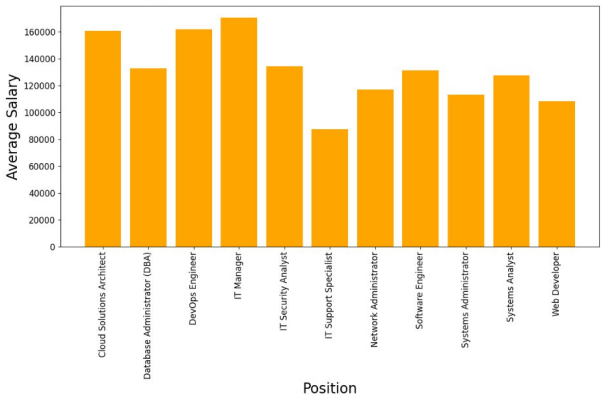
Question - 2

List of Charts

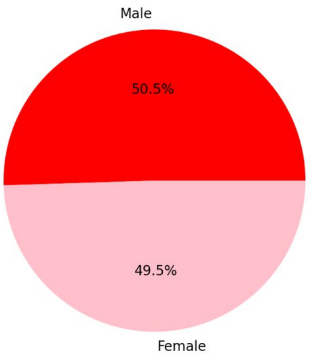
1) Rating of Products.
2) Top 10 Products by Total.
3) Price Growth Total.

Assignment-3 Home Question-1 Question-2

Average Salary Of Each Position



Total number of male and female employee

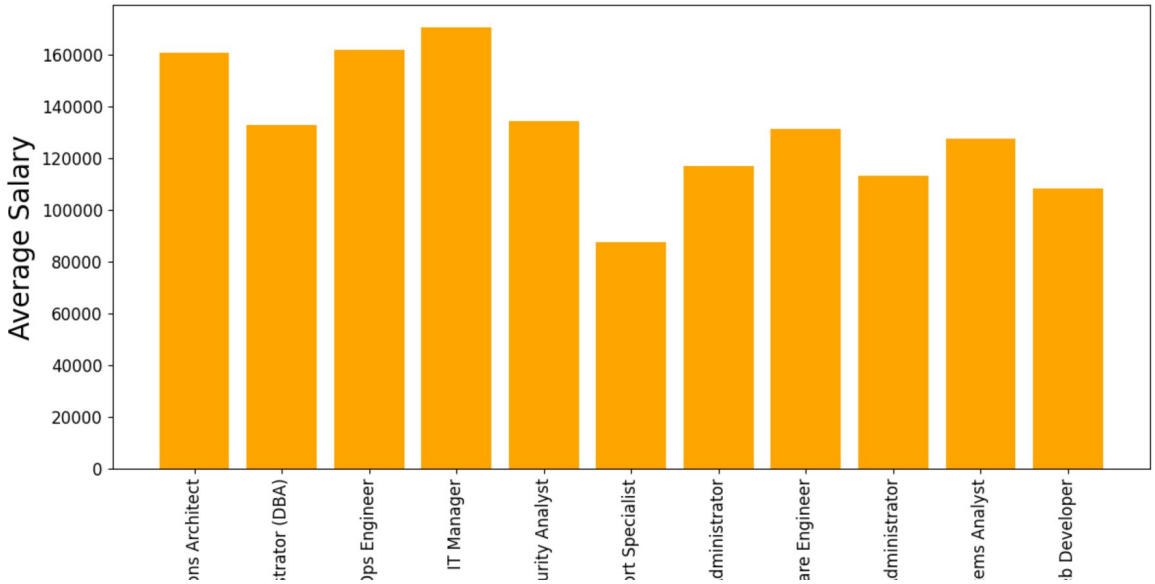


Salary earn by experience example between 10 to 15 year

Num of Position in Company

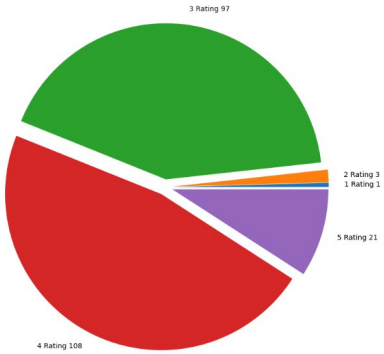
Assignment-3 Home Question-1 Question-2

Average Salary Of Each Position

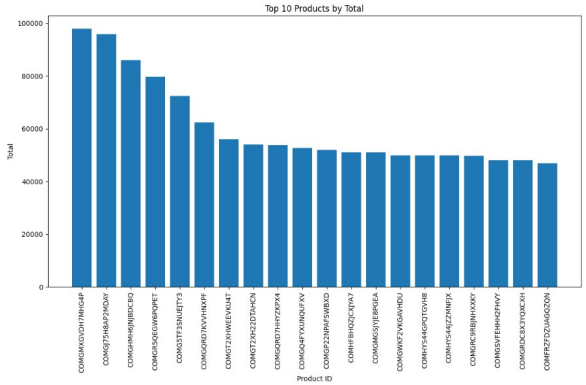


Assignment-3 Home Question-1 Question-2

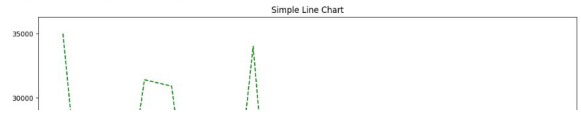
Rating of Products

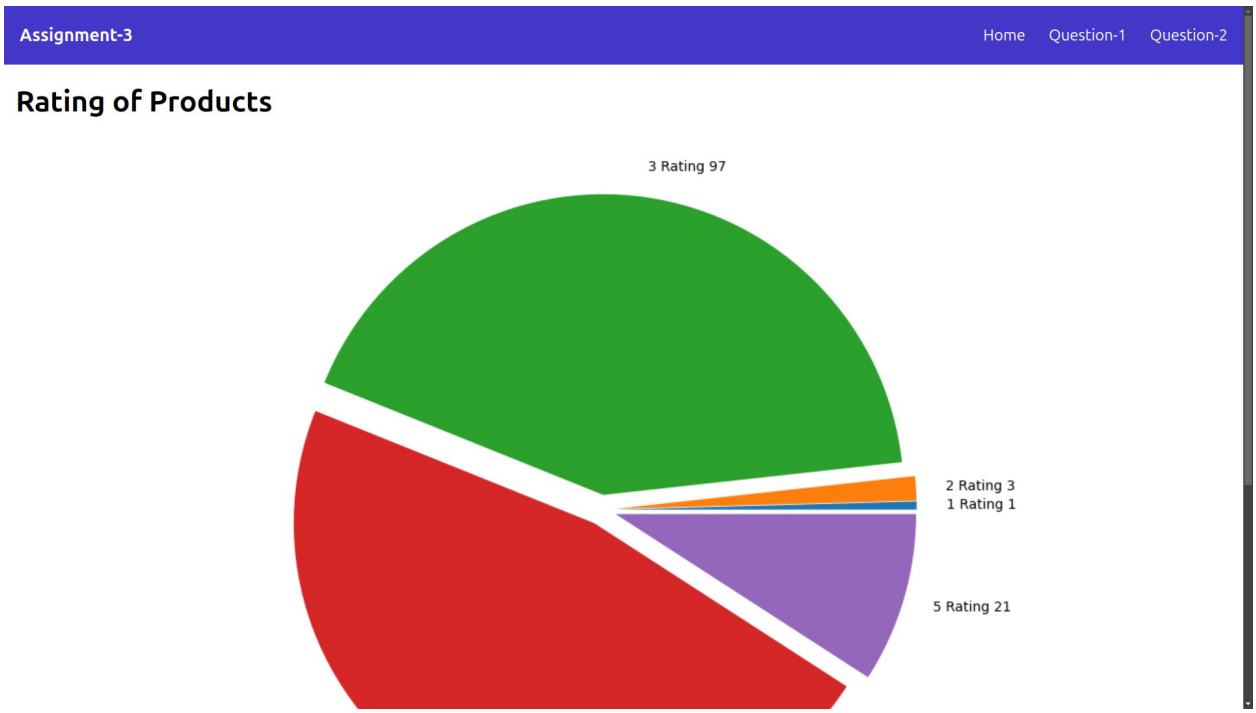


Top 10 Products by Total



Price Growth Total




Assignment-3
Home
Question-1
Question-2
Rating of Products
3 Rating 97
2 Rating 3
1 Rating 1
5 Rating 21

****

THANK YOU ☺

****

ASSIGNMENT - 4

List of Charts

1) Pie Chart for How many students got more than 70 score, between 60 to 70, between 40 to 60 and below 40 in math.

2) Pie Chart for How many students got more than 70 score, between 60 to 70, between 40 to 60 and below 40 in reading.

3) Pie Chart for How many students got more than 70 score, between 60 to 70, between 40 to 60 and below 40 in writing.

4) Bar chart for how many student completed test preparation course and how many student not completed test preparation course.

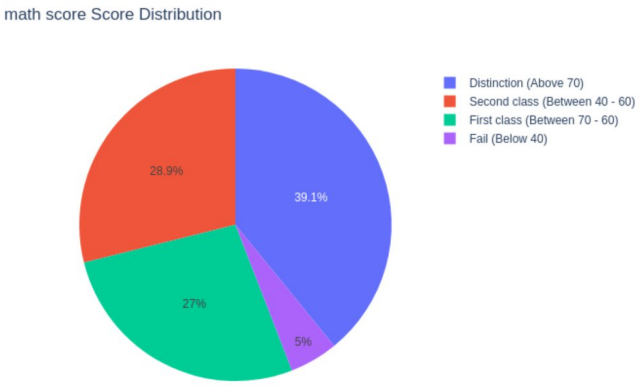
5) Line chart for math score of 20 to 30 roll nos.

6) Display the count with proper design that how many students parent having bachelor and master degree.

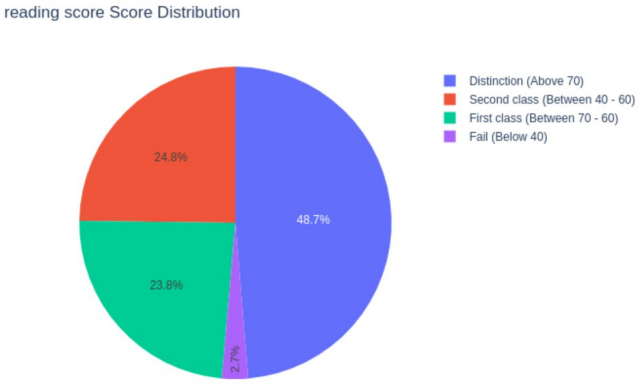
7) In dataset, replace data of lunch for free/reduced to premium and display the count of no of students who choose standard lunch and premium lunch.

Assignment-4 Home Question 1 Question 2 Question 3 Question 4 Question 5 Question 6 Question 7

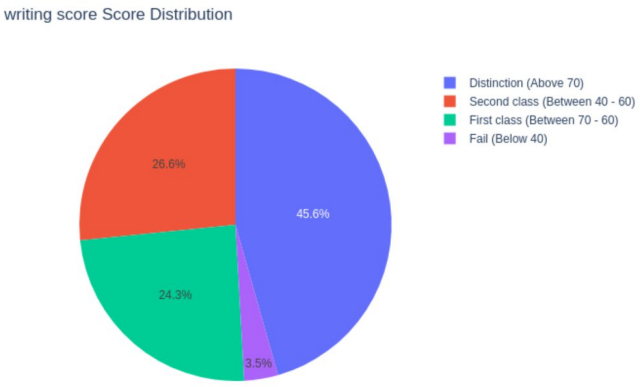
Pie Chart for How many students got more than 70 score, between 60 to 70, between 40 to 60 and below 40 in math.



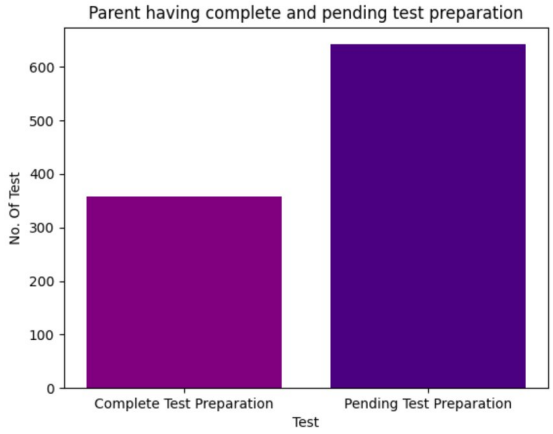
Pie Chart for How many students got more than 70 score, between 60 to 70, between 40 to 60 and below 40 in reading.



Pie Chart for How many students got more than 70 score, between 60 to 70, between 40 to 60 and below 40 in writing.

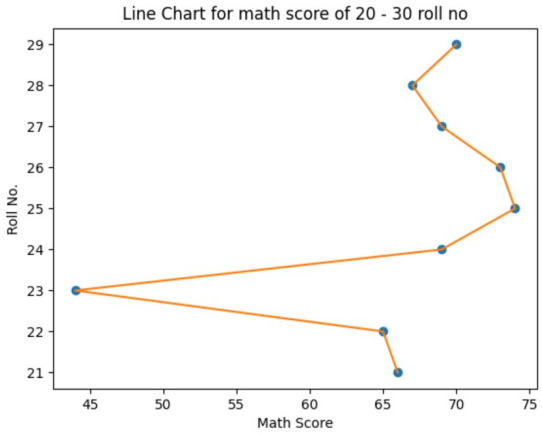


Bar chart for how many student completed test preparation course and how many student not completed test preparation course.

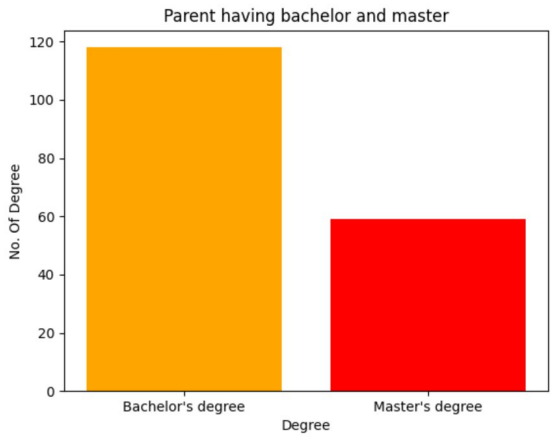


Assignment-4 Home Question 1 Question 2 Question 3 Question 4 Question 5 Question 6 Question 7

Line chart for math score of 20 to 30 roll nos.

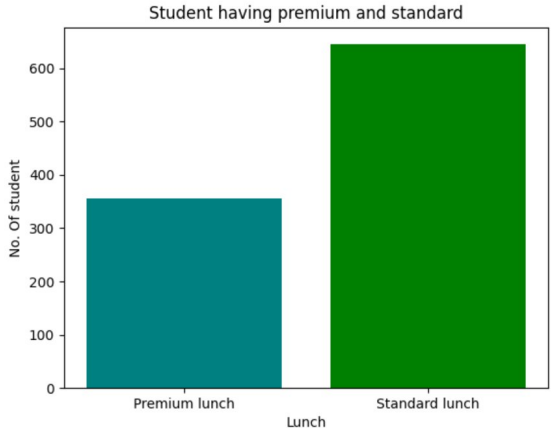


Display the count with proper design that how many students parent having bachelor and master degree.



127.0.0.1:5000/graph/6

In dataset, replace data of lunch for free/reduced to premium and display the count of no of students who choose standard lunch and premium lunch.



127.0.0.1:5000/graph/7

```
================================================================================
                                    app.py
================================================================================
import io
import base64
import matplotlib.pyplot as plt
from flask import Flask, render_template
import pandas as pd
import plotly.express as px
import io
import base64
import matplotlib.pyplot as plt
import plotly.io as pio
import matplotlib

matplotlib.use('Agg')

app = Flask(__name__)

class ImagePlot:
    """
    Class for creating and encoding plots using Matplotlib.

    This class provides methods to generate plots and convert them
    to base64 encoded format for easy storage or transmission.
    """
    @staticmethod
    def plot_image(plot_func):
        """
        Generate a plot and return it as a base64 encoded string.
```

```
    Parameters:
        plot_func (callable): A function that creates the plot.

    Returns:
        str: A base64 encoded string representing the generated plot.
    """
    # Call the function to get the figure
    fig = plot_func()

    if isinstance(fig, plt.Figure):
        # Handle Matplotlib figures
        buf = io.BytesIO()
        fig.savefig(buf, format='png')
        plt.close(fig)
        buf.seek(0)
        img_base64 = base64.b64encode(buf.read()).decode('utf-8')
    else:
        # Handle Plotly figures
        img_bytes = pio.to_image(fig, format='png')
        img_base64 = base64.b64encode(img_bytes).decode('utf-8')

    return img_base64

itemsq1 = [
    "Pie Chart for How many students got more than 70 score, between 60 to 70, between 40 to 60 and below 40 in math.",
    "Pie Chart for How many students got more than 70 score, between 60 to 70, between 40 to 60 and below 40 in reading.",
    "Pie Chart for How many students got more than 70 score, between 60 to 70, between 40 to 60 and below 40 in writing.",
```

```
    "Bar chart for how many student completed test preparation course and how many student not completed test preparation course.",

    "Line chart for math score of 20 to 30 roll nos.",

    "Display the count with proper design that how many students parent having bachelor and master degree.",

    "In dataset, replace data of lunch for free/reduced to premium and display the count of no of students who choose standard lunch and premium lunch."]


df = pd.read_csv('./StudentsPerformance.csv')


def plot_q1():

    i = 'math score'

    above_70 = df[df[i] > 70].shape[0]

    between_60_and_70 = df[(df[i] > 60) & (df[i] <= 70)].shape[0]

    between_40_and_60 = df[(df[i] > 40) & (df[i] <= 60)].shape[0]

    below_40 = df[df[i] <= 40].shape[0]


    values = [above_70,between_60_and_70,between_40_and_60,below_40]

    dist = ["Distinction (Above 70)","First class (Between 70 - 60)","Second class (Between 40 - 60)","Fail (Below 40)"]

    # Plotly interactive pie chart

    fig = px.pie(

        values=values,

        names=dist,

        title=f'{i} Score Distribution',

        hover_name=dist

    )

    fig.update_traces(hovertemplate='%{label}: %{value} students')

    return fig


def plot_q2():
```

```
    i = 'reading score'
    above_70 = df[df[i] > 70].shape[0]
    between_60_and_70 = df[(df[i] > 60) & (df[i] <= 70)].shape[0]
    between_40_and_60 = df[(df[i] > 40) & (df[i] <= 60)].shape[0]
    below_40 = df[df[i] <= 40].shape[0]

    values = [above_70,between_60_and_70,between_40_and_60,below_40]
    dist = ["Distinction (Above 70)","First class (Between 70 - 60)","Second class (Between 40 - 60)","Fail (Below 40)"]
    # Plotly interactive pie chart
    fig = px.pie(
        values=values,
        names=dist,
        title=f'{i} Score Distribution',
        hover_name=dist
    )
    fig.update_traces(hovertemplate='%{label}: %{value} students')
    return fig

def plot_q3():
    i = 'writing score'
    above_70 = df[df[i] > 70].shape[0]
    between_60_and_70 = df[(df[i] > 60) & (df[i] <= 70)].shape[0]
    between_40_and_60 = df[(df[i] > 40) & (df[i] <= 60)].shape[0]
    below_40 = df[df[i] <= 40].shape[0]

    values = [above_70,between_60_and_70,between_40_and_60,below_40]
    dist = ["Distinction (Above 70)","First class (Between 70 - 60)","Second class (Between 40 - 60)","Fail (Below 40)"]
    # Plotly interactive pie chart
```

```
    fig = px.pie(
        values=values,
        names=dist,
        title=f'{i} Score Distribution',
        hover_name=dist
    )
    fig.update_traces(hovertemplate='%{label}: %{value} students')
    return fig



def plot_q4():
    completeTest = df[df['test preparation course'] == "completed"]['Roll No'].count()
    pendingTest = df[df['test preparation course'] == "none"]['Roll No'].count()
    plt.bar(['Complete Test Preparation','Pending Test
Preparation'],[completeTest,pendingTest],color=['purple','indigo'])
    plt.title("Parent having complete and pending test preparation")
    plt.xlabel("Test")
    plt.ylabel("No. Of Test")
    return plt.gcf()


def plot_q5():
    x = df[(df['Roll No'] > 20) & (df['Roll No'] < 30)]
    plt.plot(x['math score'],x['Roll No'],'o')
    plt.plot(x['math score'],x['Roll No'])
    plt.title("Line Chart for math score of 20 - 30 roll no")
    plt.xlabel("Math Score")
    plt.ylabel("Roll No.")
    return plt.gcf()


def plot_q6():
```

```
    bachelor = df[df['parental level of education'] == "bachelor's degree"]['Roll No'].count()
    master   = df[df['parental level of education'] == "master's degree"]['Roll No'].count()
    plt.bar(['Bachelor\'s degree','Master\'s degree'],[bachelor,master],color=['orange','red'])
    plt.title("Parent having bachelor and master")
    plt.xlabel("Degree")
    plt.ylabel("No. Of Degree")
    return plt.gcf()

def plot_q7():
    newDf = df
    newDf['lunch'] = newDf['lunch'].str.replace('free/reduced','premium')
    standard = newDf[newDf['lunch'] == "standard"]['Roll No'].count()
    premium   = newDf[newDf['lunch'] == "premium"]['Roll No'].count()
    plt.bar(['Premium lunch','Standard lunch'],[premium,standard],color=['teal','green'])
    plt.title("Student having premium and standard")
    plt.xlabel("Lunch")
    plt.ylabel("No. Of student")
    return plt.gcf()

@app.route('/')
def index():
    return render_template('welcome.html',itemsq1=itemsq1)

@app.route('/graph/<int:question>')
def graph(question):
    if 1 == question:
        img = ImagePlot.plot_image(plot_q1)
    elif 2 == question:
        img = ImagePlot.plot_image(plot_q2)
    elif 3 == question:
```

```
        img = ImagePlot.plot_image(plot_q3)
    elif 4 == question:
        img = ImagePlot.plot_image(plot_q4)
    elif 5 == question:
        img = ImagePlot.plot_image(plot_q5)
    elif 6 == question:
        img = ImagePlot.plot_image(plot_q6)
    elif 7 == question:
        img = ImagePlot.plot_image(plot_q7)

    return render_template('graph.html',title=itemsq1[question-1],src=img)

if __name__ == "__main__":
    app.run(debug=True)
```

==========

util/footer.html

==========

```
<footer class="bg-rose-900 text-white mt-10">
    <div class="container mx-auto px-4 py-4 text-center">
        <h2 class="text-lg font-bold">Assignment-4</h2>
        <p class="text-sm">© 2024 Ansh Yadav. All rights reserved.</p>
    </div>
</footer>
```

==========

util/header.html

==========

```
<header class="bg-rose-700 text-white fixed w-full">
   <div class="container mx-auto px-4 py-4 flex justify-between items-center">
      <div class="text-lg font-bold">
         <a href="{{ url_for('index') }}" class="hover:text-gray-400">Assignment-4</a>
      </div>
      <nav>
         <ul class="flex space-x-6">
            <li>
               <a href="{{ url_for('index') }}" class="hover:text-rose-400 hover:underline
hover:underline-offset-4">Home</a>
            </li>
            {% for i in range(1,8) %}
            <li>
               <a href="/graph/{{ i }}" class="hover:text-rose-400 hover:underline hover:underline-
offset-4">Question {{i}}</a>
            </li>
            {% endfor %}
         </ul>
      </nav>
   </div>
</header>
```

==========

util/macros.html

==========

```
{% macro image(title, src) %}
<div class="mb-10">
    <div class="text-2xl font-bold">{{ title }}</div>
    <form action="/graph" method="POST">
        <input type="hidden" name="title" value="{{ title }}">
        <input type="hidden" name="src" value="{{ src }}">
        <button type="submit">
            <img class="h-auto w-full max-w-4xl" src="{{ src }}"
alt="{{ title }}">
        </button>
    </form>
</div>
{% endmacro %}
```

==========

graph.html

==========

```
{% extends "index.html" %}
{% block title %}Graphs {% endblock %}
{% block body %}
    <div class=" m-5">
    <div class="text-3xl font-bold">{{ title }}</h1>
        <img src="{{ src }}">
    </div>
    </div>
{% endblock %}
```

==========

index.html

==========

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://cdn.tailwindcss.com"></script>
    <title>{% block title required %}{% endblock %} - Assignment-4</title>
</head>
<body>
    {% include 'util/header.html' %}
    <div class="px-5 pt-20 min-h-screen">
        {% block body required %} {% endblock %}
    </div>
    {% include 'util/footer.html' %}
</body>
</html>
```

==========

Welcome.html

==========

```
{% extends "index.html" %}

{% block title %}Welcome{% endblock %}

{% block body %}

<div>

    <div class="text-3xl mt-5 mb-5">List of Charts</div>

    <ul class="p-5 bg-rose-100 w-full text-rose-900 rounded-lg font-mono border-2 border-rose-300">

        {% for item in itemsq1 %}

        <li class="mb-4">{{ loop.index }}) {{ item }}</li>

        {% endfor %}

    </ul>

</div>

{% endblock %}
```