



data frame named `organism_info`

group	animal	numberLegs
"reptile"	"lizard"	4
"arachnid"	"spider"	8
"annelid"	"worm"	0
"Insect"	"bee"	6

`organism_info[2,1]` `vector` `organism_info[4, 3]`

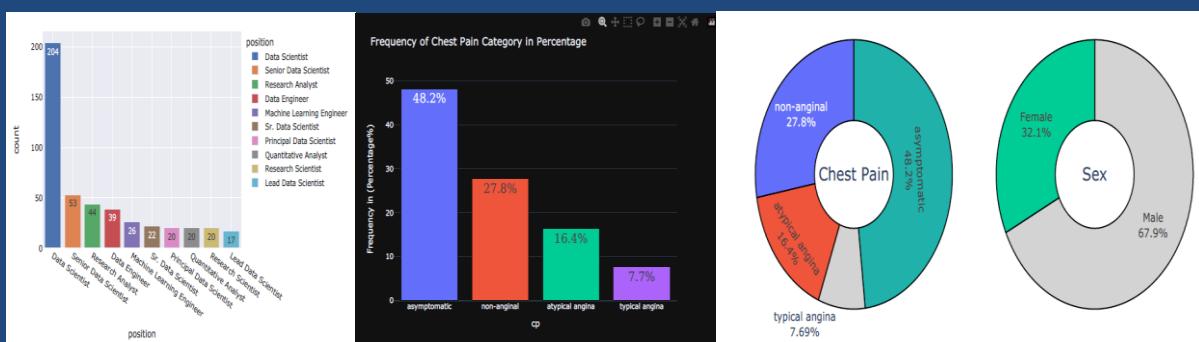
`organism_info$animal[4]`

Single column in a datafram is a Series

Regd. No	Names	Marks%
0	Steve	86.29
1	Mathew	91.63
2	Jose	72.90
3	Patty	69.23
4	vin	88.30

Python Master Data Manipulation & Visualization

Prepared by: Syed Afroz Ali
Data Scientist (Kaggle Grandmaster)



Data Manipulation in Python for Data Analysis

Kaggle Notebook: : Prepared by: Syed Afroz Ali (Kaggle Grandmaster)

<https://www.kaggle.com/code/pythonafroz/python-for-machine-learning-part-01>

Follow for more AI content: <https://www.linkedin.com/in/syed-afroz-70939914/>

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline
```

```
sns.set(rc={"axes.facecolor":"Beige" , "axes.grid" : False})
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
In [2]: # Set Display

pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
pd.set_option('display.precision', 2)
```

```
In [3]: # Check Library Version

import numpy
print('numpy:{0}'.format(numpy.__version__))

numpy:1.26.4
```

```
In [4]: # Load the dataset

df = pd.read_csv("titanic.csv")
display(df.shape)
df.head()
```

```
(891, 12)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... er)	female	38.0	1	0	PC 17599	71.28	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S

```
In [5]: # Set Table Properties

df.head(3).style.set_properties(**{'background-color': 'blue',
                                    'color': 'white',
                                    'border-color': 'darkblack'})
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.250000	nan	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38.000000	1	0	PC 17599	71.283300	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.925000	nan	S

```
In [6]: # Replacing Values/Names in a Column:

df1 = df.copy('Deep')
df1["Survived"].replace({0:"Died" , 1:"Saved"},inplace = True)
df1.head(3)
```



	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	Died	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
1	2	Saved	1	Cumings, Mrs. John Bradley (Florence Briggs Th... ...	female	38.0	1	0	PC 17599	71.28	C85	C
2	3	Saved	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S

```
In [7]: # Drop Columns
df1 = df.copy('Deep')
df1 = df1.drop(['PassengerId', 'Ticket'], axis=1)
df1.head(3)
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.25	NaN	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... ...	female	38.0	1	0	71.28	C85	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.92	NaN	S

```
In [8]: # Drop Rows
df = df.drop(labels=[1,3,5,7], axis=0)
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S

```
In [9]: # Missing Value check
print('Method 1:')
df.isnull().sum()
```

```
Method 1:
PassengerId      0
Survived         0
Pclass            0
Name              0
Sex              0
Age             176
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          685
Embarked         2
dtype: int64
```

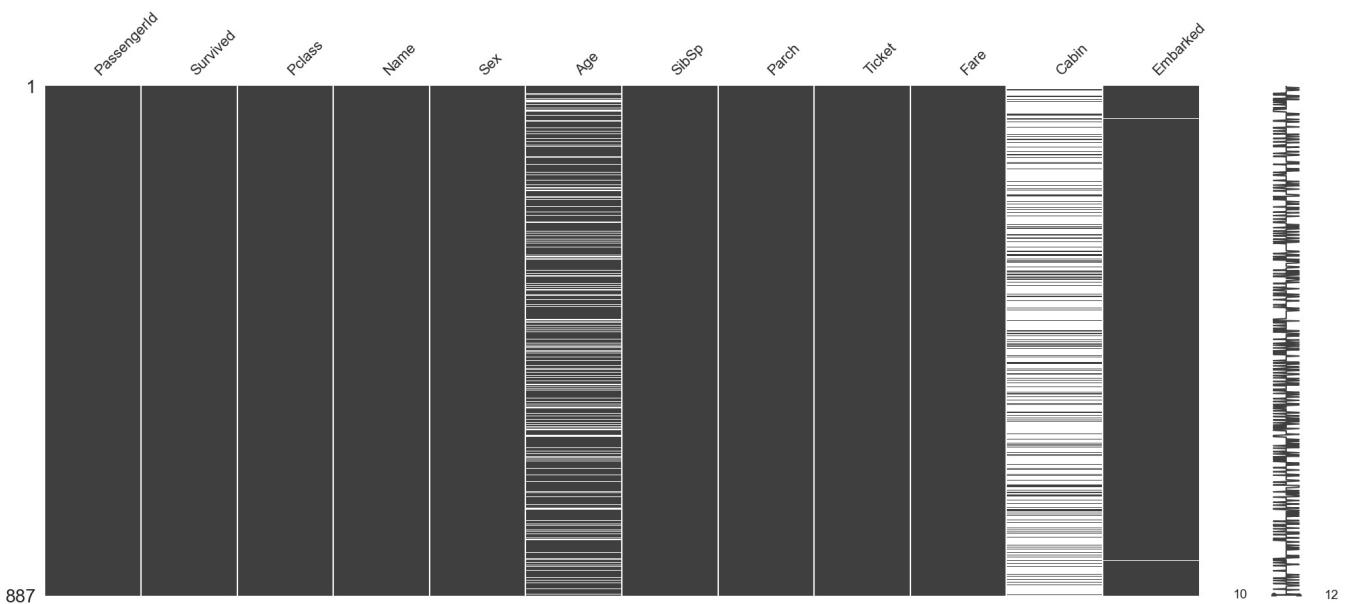
```
In [10]: var1 = [col for col in df.columns if df[col].isnull().sum() != 0]
print(df[var1].isnull().sum())
```

```
Age        176
Cabin     685
Embarked    2
dtype: int64
```

```
In [11]: # Missing Value check
print('Method 12:')
import missingno as msno
msno.matrix(df)
plt.show()
```

Method 12:





```
In [12]: df[df['Embarked'].isnull()]
```

```
Out[12]:   PassengerId  Survived  Pclass          Name     Sex   Age  SibSp  Parch  Ticket  Fare  Cabin  Embarked
      61           62       1      1  Icard, Miss. Amelie  female  38.0      0     0    113572  80.0    B28    NaN
      829          830       1      1  Stone, Mrs. George Nelson (Martha Evelyn)  female  62.0      0     0    113572  80.0    B28    NaN
```

```
In [13]: sample_incomplete_rows = df[df.isnull().any(axis=1)].head()
sample_incomplete_rows
```

```
Out[13]:   PassengerId  Survived  Pclass          Name     Sex   Age  SibSp  Parch  Ticket  Fare  Cabin  Embarked
      0           1       0      3  Braund, Mr. Owen Harris  male  22.0      1     0    A/5 21171  7.25    NaN     S
      2           3       1      3  Heikkinen, Miss. Laina  female  26.0      0     0  STON/O2. 3101282  7.92    NaN     S
      4           5       0      3  Allen, Mr. William Henry  male  35.0      0     0    373450  8.05    NaN     S
      8           9       1      3  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0     2    347742 11.13    NaN     S
      9          10       1      2  Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1     0    237736 30.07    NaN     C
```

```
In [14]: df.describe()
```

```
Out[14]:   PassengerId  Survived  Pclass   Age  SibSp  Parch  Fare
count      887.00    887.00  887.00  711.00  887.00  887.00  887.00
mean      447.99     0.38    2.31   29.72    0.52    0.38   32.18
std       256.22     0.49    0.83   14.52    1.10    0.81   49.78
min        1.00     0.00    1.00   0.42    0.00    0.00    0.00
25%      226.50     0.00    2.00   20.25    0.00    0.00    7.90
50%      448.00     0.00    3.00   28.00    0.00    0.00   14.45
75%      669.50     1.00    3.00   38.00    1.00    0.00   31.00
max      891.00     1.00    3.00   80.00    8.00    6.00  512.33
```

```
In [15]: # Describe
df[df['Survived']==0].describe().T.style.background_gradient(subset=['mean', 'std', '50%', 'count'], cmap='RdPu')
```



Out[15]:

	count	mean	std	min	25%	50%	75%	max
PassengerId	547.000000	448.625229	259.750818	1.000000	213.500000	457.000000	675.500000	891.000000
Survived	547.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Pclass	547.000000	2.530165	0.736605	1.000000	2.000000	3.000000	3.000000	3.000000
Age	423.000000	30.693853	14.120135	1.000000	21.000000	28.000000	39.000000	74.000000
SibSp	547.000000	0.550274	1.286281	0.000000	0.000000	0.000000	1.000000	8.000000
Parch	547.000000	0.329068	0.824052	0.000000	0.000000	0.000000	0.000000	6.000000
Fare	547.000000	22.144765	31.440164	0.000000	7.854200	10.500000	26.000000	263.000000

In [16]: df.describe(percentiles=[0.05, 0.25, 0.35, 0.5, 0.75, 0.85, 0.95, 0.995, 0.999])

Out[16]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	887.00	887.00	887.00	711.00	887.00	887.00	887.00
mean	447.99	0.38	2.31	29.72	0.52	0.38	32.18
std	256.22	0.49	0.83	14.52	1.10	0.81	49.78
min	1.00	0.00	1.00	0.42	0.00	0.00	0.00
5%	49.30	0.00	1.00	4.00	0.00	0.00	7.22
25%	226.50	0.00	2.00	20.25	0.00	0.00	7.90
35%	315.10	0.00	2.00	24.00	0.00	0.00	9.00
50%	448.00	0.00	3.00	28.00	0.00	0.00	14.45
75%	669.50	1.00	3.00	38.00	1.00	0.00	31.00
85%	758.10	1.00	3.00	45.00	1.00	1.00	56.50
95%	846.70	1.00	3.00	56.00	2.70	2.00	112.56
99.5%	886.57	1.00	3.00	70.73	8.00	5.00	263.00
99.9%	890.11	1.00	3.00	75.74	8.00	5.11	512.33
max	891.00	1.00	3.00	80.00	8.00	6.00	512.33

In [17]:

Agg
df[['Age', 'Fare', 'Pclass']].agg(['sum', 'max', 'mean', 'std', 'skew', 'kurt'])

Out[17]:

	Age	Fare	Pclass
sum	21130.17	28540.03	2049.00
max	80.00	512.33	3.00
mean	29.72	32.18	2.31
std	14.52	49.78	0.83
skew	0.40	4.79	-0.63
kurt	0.18	33.33	-1.27

In [18]:

value_counts
df['Embarked'].value_counts().to_frame()

Out[18]:

	count
Embarked	
S	642
C	167
Q	76

In [19]:

df['Embarked'].value_counts().tolist()

Out[19]:

[642, 167, 76]

In [20]:

value_counts for Multiple Columns
for col in df[['Survived', 'Sex', 'Embarked']]:
 print(df[col].value_counts().to_frame())
 print("****" * 7)

```

        count
Survived
0      547
1      340
*****
        count
Sex
male    575
female   312
*****
        count
Embarked
S       642
C       167
Q       76
*****

```

```
In [21]: #Count
df[['Age', 'Embarked', 'Sex']].count()
```

```
Out[21]: Age      711
Embarked  885
Sex       887
dtype: int64
```

```
In [22]: df['Embarked'][df['Sex']=='female'].value_counts(normalize=True)*100
```

```
Out[22]: Embarked
S    65.16
C    23.23
Q    11.61
Name: proportion, dtype: float64
```

```
In [23]: df['Embarked'].value_counts()/len(df['Embarked'])
```

```
Out[23]: Embarked
S    0.72
C    0.19
Q    0.09
Name: count, dtype: float64
```

```
In [24]: #Shuffling the data
df2 = df.sample(frac=1, random_state=3)
df2.head()
```

```
Out[24]:   PassengerId  Survived  Pclass          Name     Sex   Age  SibSp  Parch  Ticket  Fare Cabin Embarked
  733         734       0      2  Berriman, Mr. William John   male  23.0      0      0   28425  13.00   NaN      S
   95         96       0      3  Shorney, Mr. Charles Joseph   male   NaN      0      0   374910  8.05   NaN      S
  161        162       1      2  Watt, Mrs. James (Elizabeth "Bessie" Inglis Mi... female  40.0      0      0      C.A.  33595  15.75   NaN      S
  392        393       0      3  Gustafsson, Mr. Johan Birger   male  28.0      2      0   3101277  7.92   NaN      S
  614        615       0      3  Brocklebank, Mr. William Alfred   male  35.0      0      0   364512  8.05   NaN      S
```

```
In [25]: df1 = df.copy()

columns = ['Age']

for col in columns:
    df1[col].replace(0, np.NaN, inplace=True)

df1.describe()
```

```
Out[25]:   PassengerId  Survived  Pclass   Age  SibSp  Parch  Fare
  count      887.00   887.00  887.00  711.00  887.00  887.00  887.00
  mean      447.99    0.38    2.31  29.72    0.52    0.38   32.18
  std      256.22    0.49    0.83  14.52    1.10    0.81   49.78
  min       1.00    0.00    1.00   0.42    0.00    0.00    0.00
  25%     226.50    0.00    2.00  20.25    0.00    0.00    7.90
  50%     448.00    0.00    3.00  28.00    0.00    0.00  14.45
  75%     669.50    1.00    3.00  38.00    1.00    0.00  31.00
  max      891.00    1.00    3.00  80.00    8.00    6.00  512.33
```

```
In [26]: corr = df.select_dtypes('number').corr()
display(corr)

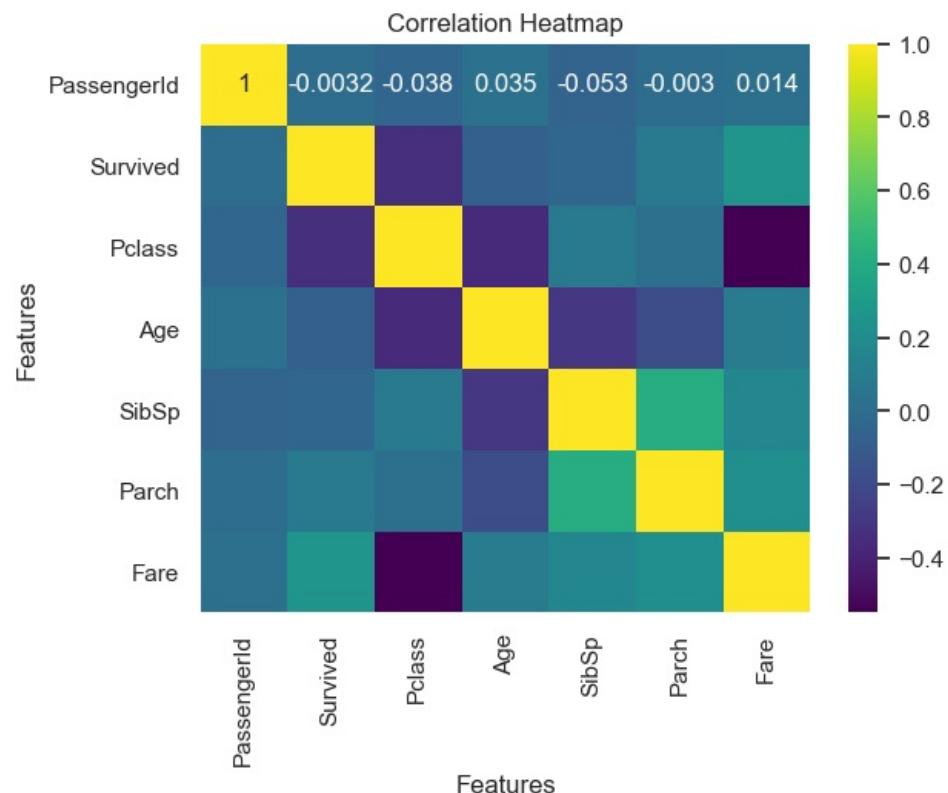
sns.heatmap(corr, annot=True, cmap='viridis')

plt.xlabel('Features')
plt.ylabel('Features')
```



```
plt.title('Correlation Heatmap')
plt.show()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.00e+00	-3.15e-03	-0.04	0.03	-0.05	-2.95e-03	0.01
Survived	-3.15e-03	1.00e+00	-0.33	-0.08	-0.04	8.35e-02	0.26
Pclass	-3.84e-02	-3.35e-01	1.00	-0.37	0.08	1.66e-02	-0.55
Age	3.49e-02	-8.15e-02	-0.37	1.00	-0.30	-1.87e-01	0.09
SibSp	-5.30e-02	-3.52e-02	0.08	-0.30	1.00	4.15e-01	0.16
Parch	-2.95e-03	8.35e-02	0.02	-0.19	0.41	1.00e+00	0.22
Fare	1.38e-02	2.56e-01	-0.55	0.09	0.16	2.17e-01	1.00



```
In [27]: corr = df.groupby(["Embarked"])[["Fare", "Age"]].corr()
display(corr)

n_groups = len(corr.index.levels[0])

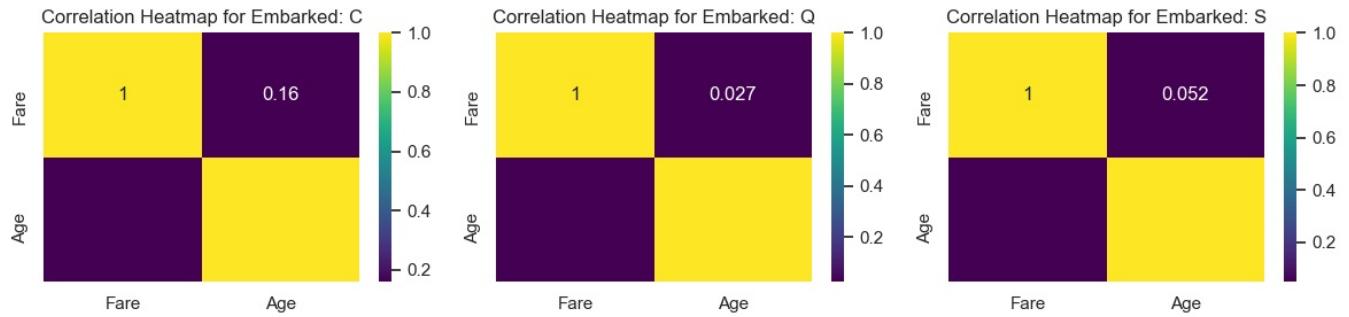
fig, axes = plt.subplots(nrows=1, ncols=min(3, n_groups), figsize=(12, 3))

group_count = 0
for embarked_group in corr.index.levels[0]:
    ax = axes.flat[group_count]
    sns.heatmap(corr.xs(embarked_group), annot=True, cmap='viridis', ax=ax)
    ax.set_title(f"Correlation Heatmap for Embarked: {embarked_group}")
    group_count += 1

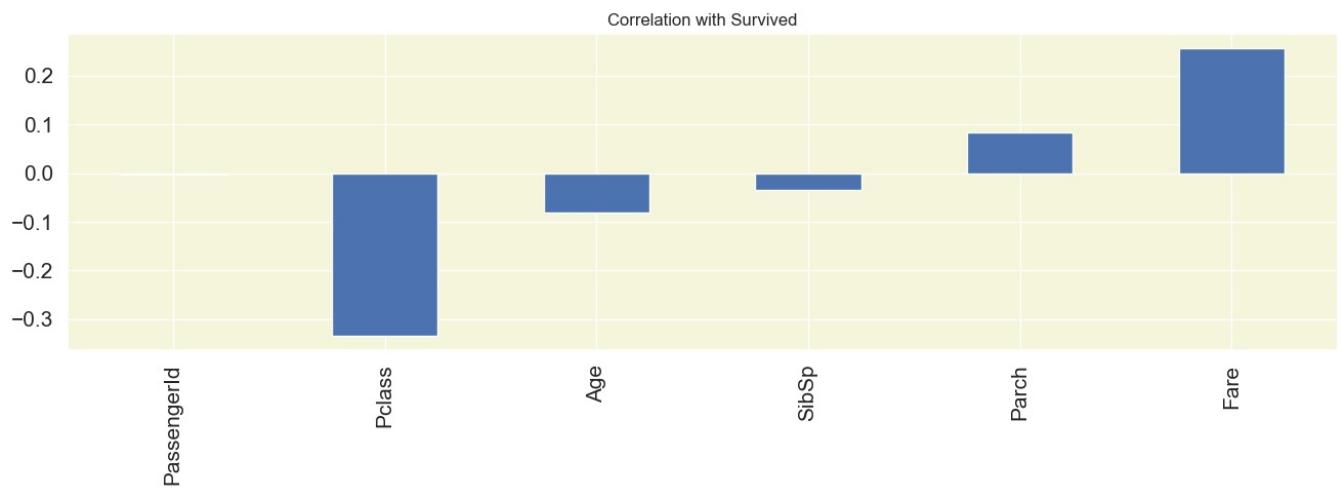
plt.tight_layout()
plt.show()
```

Embarked		Fare	Age
C	Fare	1.00	0.16
	Age	0.16	1.00
Q	Fare	1.00	0.03
	Age	0.03	1.00
S	Fare	1.00	0.05
	Age	0.05	1.00



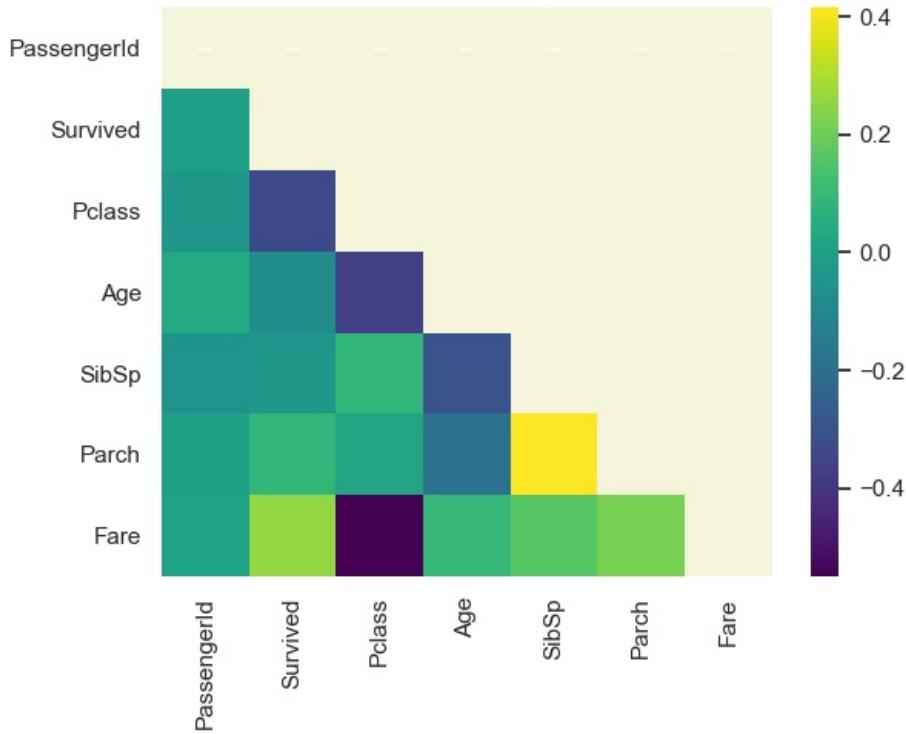


```
In [28]: X = df.select_dtypes('number').drop(['Survived'], axis=1)
y = df.select_dtypes('number')['Survived']
X.corrwith(y).plot.bar(
    figsize=(16, 4), title="Correlation with Survived", fontsize=15,
    rot=90, grid=True)
plt.show()
```



```
In [29]: corr = df.select_dtypes('number').corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
plt.figure(dpi=100)
plt.title('Correlation Analysis')
sns.heatmap(corr, mask=mask, annot=True, lw=0, linecolor='white', cmap='viridis', fmt = "0.2f")
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.show()
```

Correlation Analysis



```
In [30]: abs(df.select_dtypes('number').corr()).style.highlight_min(axis=0)
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	0.003153	0.038432	0.034866	0.052966	0.002953	0.013752
Survived	0.003153	1.000000	0.334575	0.081455	0.035186	0.083506	0.255761
Pclass	0.038432	0.334575	1.000000	0.367249	0.083521	0.016574	0.548991
Age	0.034866	0.081455	0.367249	1.000000	0.304297	0.187338	0.094965
SibSp	0.052966	0.035186	0.083521	0.304297	1.000000	0.414724	0.159978
Parch	0.002953	0.083506	0.016574	0.187338	0.414724	1.000000	0.217091
Fare	0.013752	0.255761	0.548991	0.094965	0.159978	0.217091	1.000000

```
In [31]: df1 = df1[df1['Cabin'].notna()]
df1.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.70	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	C103	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.00	D56	S
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.50	A6	S

```
In [32]: print('Passenger Survived in Titanic: {}'.format(df1['Survived'].value_counts()[0]))
print('Passenger Died in Titanic: {}'.format(df1['Survived'].value_counts()[1]))
```

Passenger Survived in Titanic: 68
Passenger Died in Titanic: 134

```
In [33]: print('Survived sample ratio: {:.3f} %'.format(df1['Survived'].value_counts()[0]/len(df1)*100))
print('Died sample ratio: {:.3f} %'.format(df1['Survived'].value_counts()[1]/len(df1)*100))
```

Survived sample ratio: 33.663 %
Died sample ratio: 66.337 %

```
In [34]: # Fillna Method
```

```
df1 = df.copy()
df1 = df1.dropna()
```

```
In [35]: # Fillna Method
```

```
df1 = df.copy()
df1.fillna(method="ffill", inplace=True)
```

```
In [36]: # Fill Null Values by Mean Value
```

```
df1 = df.copy()
```



```
df1["Age"] = df1["Age"].fillna(df1["Age"].mean())
```

In [37]: # Fill Null Values by Desiresd Value

```
df1 = df.copy()
df1['Embarked'] = df1['Embarked'].fillna(df1['Embarked'] == 'Q')
df1.head(3)
```

Out[37]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S

In [38]: #Fill Method :

```
df1 = df.copy()
df1['Age'] = df1['Age'].fillna(0)
df1['Age'] = df1['Age'].fillna('None')
df1["Age"].fillna(method="backfill",inplace=True)
df1["Embarked"].fillna(value="A",inplace=True)
df1["Pclass"].fillna(value= 0,inplace=True)
df1.head()
```

Out[38]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S

In [39]: # Find Method/Select Method

```
# Find All Null Values in the dataframe

df1 = df.copy()
df1 = df1.drop(['Cabin'],axis =1)

sample_incomplete_rows = df1[df1.isnull().any(axis=1)]
display(sample_incomplete_rows.shape)
sample_incomplete_rows.head()
```

(178, 11)

Out[39]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.00	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.22	C
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.22	C
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.88	Q
29	30	0	3	Todoroff, Mr. Lalio	male	NaN	0	0	349216	7.90	S

In [40]: titanic_Fare500 = df1[df1['Fare'] > 500][['Name', 'Embarked']]
display(titanic_Fare500.shape)
titanic_Fare500

(3, 2)

Out[40]:

	Name	Embarked
258	Ward, Miss. Anna	C
679	Cardeza, Mr. Thomas Drake Martinez	C
737	Lesurer, Mr. Gustave J	C

In [41]: titanic_age_70 = df1.loc[df1['Age'] > 70, ["Name", "Embarked", "Sex"]]
display(titanic_age_70.shape)
titanic_age_70

(5, 3)



Out[41]:

	Name	Embarked	Sex
96	Goldschmidt, Mr. George B	C	male
116	Connors, Mr. Patrick	Q	male
493	Artagaveytia, Mr. Ramon	C	male
630	Barkworth, Mr. Algernon Henry Wilson	S	male
851	Svensson, Mr. Johan	S	male

In [42]:

```
titanic_age_selection = df[(df["Sex"] == "male") & (df["Age"] > 50.00)]
display(titanic_age_selection.shape)
titanic_age_selection.head()
```

(47, 12)

Out[42]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
33	34	0	2	Whealon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.50	NaN	S
54	55	0	1	Ostby, Mr. Engelhart Cornelius	male	65.0	0	1	113509	61.98	B30	C
94	95	0	3	Coxon, Mr. Daniel	male	59.0	0	0	364500	7.25	NaN	S
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.65	A5	C

In [43]:

```
women = df1.loc[df1['Sex'] == 'female']["Survived"]
rate_women = (women.sum()/len(women)).round(3)*100
print("Percentage of women who survived:", rate_women, "%")
```

Percentage of women who survived: 74.0 %

In [44]:

```
men = df1.loc[df1['Sex'] == 'male']["Survived"]
rate_men = (men.sum()/len(men)).round(3)*100
print("Percentage of Men who survived:", rate_men, "%")
```

Percentage of Men who survived: 19.0 %

In [45]:

```
titanic_Pclass = df1[df1["Pclass"].isin([1, 2])]
display(titanic_Pclass.shape)
titanic_Pclass.head()
```

(398, 11)

Out[45]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.07	C
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.00	S
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.00	S

In [46]:

```
df1 = df.copy()
cabin_no_na = df1[df1["Cabin"].notna()]
display(cabin_no_na.shape)
cabin_no_na.head()
```

(202, 12)

Out[46]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.70	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	C103	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.00	D56	S
23	24	1	1	Slusher, Mr. William Thompson	male	28.0	0	0	113788	35.50	A6	S

In [47]:

```
titanic_Pclass = df1[(df1["Pclass"] == 1) & (df1["Sex"] == 'female') & (df1["Age"] > 50) ]
display(titanic_Pclass.shape)
titanic_Pclass.head()
```

(13, 12)



Out[47]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
	11	12	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	C103	S
	195	196	1	Lurette, Miss. Elise	female	58.0	0	0	PC 17569	146.52	B80	C
	268	269	1	Graham, Mrs. William Thompson (Edith Junkins)	female	58.0	0	1	PC 17582	153.46	C125	S
	275	276	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.96	D7	S
	366	367	1	Warren, Mrs. Frank Manley (Anna Sophia Atkinson)	female	60.0	1	0	110813	75.25	D37	C

In [48]: # np.where

```
df1 = df.copy()
df1['Cabin_null'] = np.where(df1['Cabin'].isnull(), 0, 1)
df1.head()
```

Out[48]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Cabin_null
	0	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	0
	2	3	1	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0
	4	5	0	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	0
	6	7	0	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1
	8	9	1	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0

In [49]: df1 = df.copy()

```
df1["Bucket"] = np.where(df1["Fare"] < 250, "Low", "High")
df1.head()
```

Out[49]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Bucket
	0	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	Low
	2	3	1	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	Low
	4	5	0	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	Low
	6	7	0	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	Low
	8	9	1	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	Low

In [50]: df1 = df.copy()

```
titanic_age_missing_first = df1.sort_values(by='Age', ascending=False, na_position='first')
titanic_age_missing_first.head()
```

Out[50]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
	17	18	1	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.00	NaN	S
	19	20	1	Masselmann, Mrs. Fatima	female	NaN	0	0	2649	7.22	NaN	C
	26	27	0	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.22	NaN	C
	28	29	1	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.88	NaN	Q
	29	30	0	Todoroff, Mr. Lalio	male	NaN	0	0	349216	7.90	NaN	S

In [51]: df1 = df.copy()
df1.sort_values(by = 'Age' , ascending = False)[['Name','Ticket','Survived','Pclass', 'Age']].head()

Out[51]:	Name	Ticket	Survived	Pclass	Age
	630 Barkworth, Mr. Algernon Henry Wilson	27042	1	1	80.0
	851 Svensson, Mr. Johan	347060	0	3	74.0
	96 Goldschmidt, Mr. George B	PC 17754	0	1	71.0
	493 Artagaveitia, Mr. Ramon	PC 17609	0	1	71.0
	116 Connors, Mr. Patrick	370369	0	3	70.5

In [52]: Numerical_data = df1.select_dtypes(include=['number'])
Numerical_data.head()



```
Out[52]:
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
0	1	0	3	22.0	1	0	7.25
2	3	1	3	26.0	0	0	7.92
4	5	0	3	35.0	0	0	8.05
6	7	0	1	54.0	0	0	51.86
8	9	1	3	27.0	0	2	11.13

```
In [53]: Categorical_data = df1.select_dtypes(include=['object'])
Categorical_data.head()
```

```
Out[53]:
```

	Name	Sex	Ticket	Cabin	Embarked
0	Braund, Mr. Owen Harris	male	A/5 21171	Nan	S
2	Heikkinen, Miss. Laina	female	STON/O2. 3101282	Nan	S
4	Allen, Mr. William Henry	male	373450	Nan	S
6	McCarthy, Mr. Timothy J	male	17463	E46	S
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	347742	Nan	S

```
In [54]: cabin_notna = df1[df1['Cabin'].notna()]
cabin_notna.head()
```

```
Out[54]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.70	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	38.0	0	0	113783	26.55	C103	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.00	D56	S
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.50	A6	S

```
In [55]: #groupby
```

```
df1 = df.copy()
titanic_room = df1.groupby(['Embarked'])['Age'].mean().reset_index()
titanic_room.head()
```

```
Out[55]:
```

Embarked	Age
C	30.76
Q	28.09
S	29.49

```
In [56]: df1.groupby("Embarked").agg({"Fare": np.mean, "Sex": np.size})
```

```
Out[56]:
```

Embarked	Fare	Sex
C	59.89	167
Q	13.34	76
S	27.05	642

```
In [57]: temp = df1.groupby("Sex")['Age'].min().to_frame().reset_index()
temp
```

```
Out[57]:
```

Sex	Age
0	female 0.75
1	male 0.42

```
In [58]: df1.groupby(["Embarked", "Pclass"]).agg({"Fare": [np.size, np.mean]})
```

```
Out[58]:
```

		Fare	
	size	mean	
Embarked	Pclass		
C	1	84	105.12
	2	17	25.36
	3	66	11.21
Q	1	2	90.00
	2	3	12.35
	3	71	11.22
S	1	126	70.50
	2	164	20.33
	3	352	14.63

```
In [59]: df1.groupby(['Survived', "Sex"])['Fare'].first().to_frame()
```

```
Out[59]:
```

		Fare
Survived	Sex	
0	female	7.85
	male	7.25
1	female	7.92
	male	13.00

```
In [60]: Tit_groupby = df1.groupby("Pclass")["Pclass"].count().to_frame()
Tit_groupby
```

```
Out[60]:
```

Pclass	
1	214
2	184
3	489

```
In [61]: df1.groupby('Survived')[ 'Sex'].value_counts().to_frame()
```

```
Out[61]:
```

		count
Survived	Sex	
0	male	466
	female	81
1	female	231
	male	109

```
In [62]: df1.groupby(['Survived', "Sex"])['Pclass'].count()/df1.groupby(["Sex"])[ 'Pclass'].count()*100
```

```
Out[62]:
```

Survived	Sex	
0	female	25.96
	male	81.04
1	female	74.04
	male	18.96

Name: Pclass, dtype: float64

```
In [63]: (df1.groupby(['Embarked', 'Pclass']).count()['Fare']/df1.groupby(['Embarked']).count()['Fare'])*100
```

```
Out[63]:
```

Embarked	Pclass	
C	1	50.30
	2	10.18
	3	39.52
Q	1	2.63
	2	3.95
	3	93.42
S	1	19.63
	2	25.55
	3	54.83

Name: Fare, dtype: float64

```
In [64]: df1.groupby("Sex")[["Age", "Pclass"]].mean()
```



```
Out[64]:    Age  Pclass
```

Sex

female	27.85	2.17
male	30.79	2.39

```
In [65]: df1.groupby(["Sex", "Pclass"])["Fare"].mean()
```

```
Out[65]: Sex      Pclass
```

female	1	107.08
	2	21.97
	3	16.12
male	1	67.23
	2	19.74
	3	12.65

Name: Fare, dtype: float64

```
In [66]: titanic_summed = df1.groupby(["Sex", "Embarked"])[["Fare", "Age"]].sum()
```

```
Out[66]:          Fare      Age
```

Sex Embarked

female	C	5416.11	1691.00
	Q	454.86	291.50
	S	7811.31	5130.50
male	C	4584.90	2276.92
	Q	558.94	495.00
	S	9553.92	11145.25

```
In [67]: df1.groupby(["Embarked", "Pclass"]).agg({"Fare": [np.size, np.mean]})
```

```
Out[67]:          Fare
```

size mean

Embarked Pclass

C	1	84	105.12
	2	17	25.36
	3	66	11.21
Q	1	2	90.00
	2	3	12.35
	3	71	11.22
S	1	126	70.50
	2	164	20.33
	3	352	14.63

```
In [68]: temp = df1.groupby("Sex")['Age'].min().reset_index()
```

temp

```
Out[68]:   Sex  Age
```

0	female	0.75
1	male	0.42

```
In [69]: titanic_room= df1.groupby(['Embarked', 'Sex'])[['Age', 'Fare']].mean().reset_index()
```

titanic_room

```
Out[69]:   Embarked  Sex  Age  Fare
```

0	C	female	28.18	75.22
1	C	male	33.00	48.26
2	Q	female	24.29	12.63
3	Q	male	30.94	13.97
4	S	female	27.73	38.67
5	S	male	30.37	21.71

```
In [70]: df1.groupby(['Survived', "Sex", "Embarked"])['Pclass'].count().to_frame()
```



```
Out[70]:
```

Pclass			
Survived	Sex	Embarked	
0	female	C	9
		Q	9
		S	63
	male	C	66
		Q	37
		S	363
1	female	C	63
		Q	27
		S	139
	male	C	29
		Q	3
		S	77

```
In [71]: df1.groupby("Embarked").agg({"Fare": np.mean, "Sex": np.size})
```

```
Out[71]: Fare Sex
```

Embarked	Fare	Sex
C	59.89	167
Q	13.34	76
S	27.05	642

```
In [72]: (df1.groupby(['Survived', "Sex"])['Fare'].count()/df1.groupby(['Survived'])['Fare'].count()).to_frame()*100
```

```
Out[72]: Fare
```

Survived	Sex	Fare
0	female	14.81
	male	85.19
1	female	67.94
	male	32.06

```
In [73]: df1.groupby('Sex')['Embarked'].count().nlargest(2).reset_index()
```

```
Out[73]: Sex Embarked
```

Sex	Embarked	
0	male	575
1	female	310

```
In [74]: df1[['Pclass', 'Fare']].groupby(['Pclass'], as_index=True).mean()
```

```
Out[74]: Fare
```

Pclass	Fare
1	84.36
2	20.66
3	13.67

```
In [75]: #pivot_table
```

```
quality_pivot = df1.pivot_table(index='Pclass', values='Age', aggfunc=np.mean)
```

```
quality_pivot
```

```
Out[75]: Age
```

Pclass	Age
1	38.25
2	29.88
3	25.21

```
In [76]: x=pd.DataFrame(pd.pivot_table(df1,index=['Sex', 'Embarked'],aggfunc='count')['Fare'])
```

```
x
```



```
Out[76]: Fare
```

	Sex	Embarked	Fare
female	C	72	
	Q	36	
	S	202	
male	C	95	
	Q	40	
	S	440	

```
In [77]: quality_pivot = df1.pivot_table(index='Pclass', values='Age', aggfunc=np.median)  
quality_pivot
```

```
Out[77]: Age
```

	Pclass	Age
1	37.0	
2	29.0	
3	24.0	

```
In [78]: pd.crosstab(df1['Pclass'], df1['Survived'])
```

```
Out[78]: Survived    0    1
```

	Pclass	0	1
1	80	134	
2	97	87	
3	370	119	

```
In [79]: #Cross Tab  
pd.crosstab(df1['Sex'], df1['Embarked'])
```

```
Out[79]: Embarked  C   Q   S
```

	Sex	C	Q	S
female	72	36	202	
male	95	40	440	

```
In [80]: #Cross Tab  
plot_criteria= ['Sex', 'Pclass']  
cm = sns.light_palette("red", as_cmap=True)  
(round(pd.crosstab(df1[plot_criteria[0]], df1[plot_criteria[1]]), normalize='columns') * 100,2)).style.background
```

```
Out[80]: Pclass      1      2      3
```

	Sex	1	2	3
female	Sex	42.990000	41.300000	29.450000
male	Sex	57.010000	58.700000	70.550000

```
In [81]: pd.crosstab(df1['Sex'], df1['Embarked'], normalize = "index").style.background_gradient(cmap='crest')
```

```
Out[81]: Embarked      C      Q      S
```

	Sex	C	Q	S
female	Sex	0.232258	0.116129	0.651613
male	Sex	0.165217	0.069565	0.765217

```
In [82]: plot_criteria= ['Embarked', 'Pclass']  
cm = sns.light_palette("red", as_cmap=True)  
(round(pd.crosstab(df1[plot_criteria[0]], df1[plot_criteria[1]]), normalize='columns') * 100,2)).style.background
```

```
Out[82]: Pclass      1      2      3
```

	Embarked	1	2	3
C	Embarked	39.620000	9.240000	13.500000
Q	Embarked	0.940000	1.630000	14.520000
S	Embarked	59.430000	89.130000	71.980000

```
In [83]: pd.crosstab(df1['Pclass'], df1['Survived'], margins=True)
```



```
Out[83]: Survived 0 1 All
```

Pclass	1	2	3	All
1	80	134	214	547
2	97	87	184	340
3	370	119	489	887

```
In [84]: #Create New Column
```

```
df1['sex Titanic map']=df1['Sex'].map({'male':1,'female':0})  
df1.head()
```

```
Out[84]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	1
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0

```
In [85]: df1["Fare Range"] = np.where(df1["Fare"] < 200, "low", "high")  
df1.head()
```

```
Out[85]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	1	low
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1	low
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low

```
In [86]: df1["age_bins"] = pd.cut(df1["Age"], bins=[1,18,29, 40, 50, 60, 80], labels=["child","teen","adult", "forty plus"])  
df1.head()
```

```
Out[86]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1	low	old
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen

```
In [87]: df1['is_train'] = np.random.uniform(0,1,len(df1)) <=.75  
df1.head()
```



Out[87]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is_tr
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen	F
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen	T
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult	T
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1	low	old	F
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen	T

In [88]:

#iloc & loc
df1.iloc[9:12, 2:5] # [Row , Column]

Out[88]:

	Pclass	Name	Sex
13	3	Andersson, Mr. Anders Johan	male
14	3	Vestrom, Miss. Hulda Amanda Adolfina	female
15	2	Hewlett, Mrs. (Mary D Kingcome)	female

In [89]:

df1.iloc[2:4, 3:6]

Out[89]:

	Name	Sex	Age
4	Allen, Mr. William Henry	male	35.0
6	McCarthy, Mr. Timothy J	male	54.0

In [90]:

df1.iloc[0:4, 3] = "Anonymous"
df1.head()

Out[90]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is_
0	1	0	3	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen	F
2	3	1	3	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen	T
4	5	0	3	Anonymous	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult	T
6	7	0	1	Anonymous	male	54.0	0	0	17463	51.86	E46	S	1	low	old	F
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen	T

In [91]:

df1.iloc[[3,6,9],[2,3]]

Out[91]:

	Pclass	Name
6	1	Anonymous
10	3	Sandstrom, Miss. Marguerite Rut
13	3	Andersson, Mr. Anders Johan

In [92]:

#Replace
df1["Survived"].replace({0:"Died" , 1:"Saved"} , inplace=True)
df1.head()

Out[92]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is_
0	1	Died	3	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen	F
2	3	Saved	3	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen	
4	5	Died	3	Anonymous	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult	
6	7	Died	1	Anonymous	male	54.0	0	0	17463	51.86	E46	S	1	low	old	F
8	9	Saved	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen	

In [93]:

```
#Rename
df1.rename(columns={"Name" : 'Person Name'},inplace=True)
df1.head()
```

Out[93]:

	PassengerId	Survived	Pclass	Person Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is_
0	1	Died	3	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen	F
2	3	Saved	3	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen	
4	5	Died	3	Anonymous	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult	
6	7	Died	1	Anonymous	male	54.0	0	0	17463	51.86	E46	S	1	low	old	F
8	9	Saved	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen	

In [94]:

```
# Replace the codes with their full names
df1['Embarked'] = df1['Embarked'].replace({'S': 'Southampton', 'C': 'Cherbourg', 'Q': 'Queenstown'})
df1.sample(5)
```

Out[94]:

	PassengerId	Survived	Pclass	Person Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is_
290	291	Saved	1	Barber, Miss. Ellen "Nellie"	female	26.0	0	0	19877	78.85	NaN	Southampton	0	low	teen	
887	888	Saved	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	Southampton	0	low	teen	
135	136	Died	2	Richard, Mr. Emile	male	23.0	0	0	SC/PARIS 2133	15.05	NaN	Cherbourg	1	low	teen	
41	42	Died	2	Turpin, Mrs. William John Robert (Dorothy Ann ...	female	27.0	1	0	11668	21.00	NaN	Southampton	0	low	teen	
500	501	Died	3	Calic, Mr. Petar	male	17.0	0	0	315086	8.66	NaN	Southampton	1	low	child	

In [95]:

```
df1['Pclass'][df1['Pclass'] == 1] = 'Rich'
df1['Pclass'][df1['Pclass'] == 2] = 'Middle Class'
df1['Pclass'][df1['Pclass'] == 3] = 'Poor'
df1.head()
```



Out[95]:

	PassengerId	Survived	Pclass	Person Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex	Titanic map	Fare Range	age_bins	i
0	1	Died	Poor	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	Southampton	1	low	teen		
2	3	Saved	Poor	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	Southampton	0	low	teen		
4	5	Died	Poor	Anonymous	male	35.0	0	0	373450	8.05	NaN	Southampton	1	low	adult		
6	7	Died	Rich	Anonymous	male	54.0	0	0	17463	51.86	E46	Southampton	1	low	old		
8	9	Saved	Poor	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	Southampton	0	low	teen		

In [96]:

```
#Converting Zero Value to NaN Value
df1.loc[df1['Fare'] == 0, 'Fare'] = np.nan
df1.head()
```

Out[96]:

	PassengerId	Survived	Pclass	Person Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex	Titanic map	Fare Range	age_bins	i
0	1	Died	Poor	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	Southampton	1	low	teen		
2	3	Saved	Poor	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	Southampton	0	low	teen		
4	5	Died	Poor	Anonymous	male	35.0	0	0	373450	8.05	NaN	Southampton	1	low	adult		
6	7	Died	Rich	Anonymous	male	54.0	0	0	17463	51.86	E46	Southampton	1	low	old		
8	9	Saved	Poor	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	Southampton	0	low	teen		

In [97]:

```
# Replace First Three name with any symbol or Value
df1 = df.copy()
df1.loc[0:2, 'Name'] = '?'
df1.head()
```

Out[97]:

	PassengerId	Survived	Pclass	Name				Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	?	male	22.0	1	0	A/5 21171	7.25	NaN	S			
2	3	1	3	?	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S			
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S			
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S			
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S			

In [98]:

```
#Replace '?' values in workclass variable with NaN
df1['Name'].replace('?', np.NaN, inplace=True)
df1.head()
```

Out[98]:

	PassengerId	Survived	Pclass	Name				Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	NaN	male	22.0	1	0	A/5 21171	7.25	NaN	S			
2	3	1	3	NaN	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S			
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S			
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S			
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S			

In [99]:

```
#Saving to Excel Format
df1.to_excel('titanic.xlsx', sheet_name="Passenger", index=False)
```

In [100]:

```
df1 = df.copy()
df1[df1['Age'].isnull()].index
```

Out[100]:

```
Index([ 17,  19,  26,  28,  29,  31,  32,  36,  42,  45,
       ...,
     832, 837, 839, 846, 849, 859, 863, 868, 878, 888],
      dtype='int64', length=176)
```



```
#JOIN
Join = df1.join(df1, lsuffix = '_1') # lsuffix = Left Suffix
Join.head(2)
```

Out[101]:

	PassengerId_1	Survived_1	Pclass_1	Name_1	Sex_1	Age_1	SibSp_1	Parch_1	Ticket_1	Fare_1	Cabin_1	Embarked_1	PassengerId
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	

```
#Melt
Melt = pd.melt(df1,id_vars = ['Embarked'], value_vars = ['Survived'])
Melt.head()
```

Out[102]:

	Embarked	variable	value
0	S	Survived	0
1	S	Survived	1
2	S	Survived	0
3	S	Survived	0
4	S	Survived	1

```
df1 = df.copy()
df1.dtypes
```

Out[103]:

PassengerId	int64
Survived	int64
Pclass	int64
Name	object
Sex	object
Age	float64
SibSp	int64
Parch	int64
Ticket	object
Fare	float64
Cabin	object
Embarked	object
dtype:	object

```
df1["Sex"] = df1.Sex.apply(lambda x:'male' if x==1 else 'female')
```

```
df1['Pclass_New'] = df1['Pclass'].apply(lambda x: 'UpperClass' if x == 1 else 0)
display(df1.head())
df1['Pclass_New'].value_counts().to_frame()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Pclass_New
0	1	0	Braund, Mr. Owen Harris	female	22.0	1	0	A/5 21171	7.25	NaN	S	0
2	3	1	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0
4	5	0	Allen, Mr. William Henry	female	35.0	0	0	373450	8.05	NaN	S	0
6	7	0	McCarthy, Mr. Timothy J	female	54.0	0	0	17463	51.86	E46	S	UpperClass
8	9	1	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0

Out[105]:

	count
Pclass_New	
0	673
UpperClass	214

```
df1['Pclass_New'] = df1['Pclass'].apply(lambda x: 5 if x > 2 else 0)
display(df1.head())
df1['Pclass_New'].value_counts().to_frame()
```



	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Pclass_New
0	1	0	3	Braund, Mr. Owen Harris	female	22.0	1	0	A/5 21171	7.25	NaN	S	5
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	5
4	5	0	3	Allen, Mr. William Henry	female	35.0	0	0	373450	8.05	NaN	S	5
6	7	0	1	McCarthy, Mr. Timothy J	female	54.0	0	0	17463	51.86	E46	S	0
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	5

Out[106]: count

Pclass_New	count
5	489
0	398

In [107]: df1.columns.tolist()

```
Out[107]: ['PassengerId',
 'Survived',
 'Pclass',
 'Name',
 'Sex',
 'Age',
 'SibSp',
 'Parch',
 'Ticket',
 'Fare',
 'Cabin',
 'Embarked',
 'Pclass_New']
```

In [108]: df1.unique()

```
Out[108]: PassengerId      887
Survived        2
Pclass          3
Name           887
Sex            1
Age           88
SibSp          7
Parch          7
Ticket         679
Fare          246
Cabin         146
Embarked       3
Pclass_New     2
dtype: int64
```

In [109]: df1 = df.copy()

```
women = df1.loc[df1['Sex'] == 'female']["Survived"]
rate_women = (women.sum()/len(women)).round(3)*100
print("Percentage of women who survived:", rate_women, "%")

men = df1.loc[df1['Sex'] == 'male']["Survived"]
rate_men = (men.sum()/len(men)).round(3)*100
print("Percentage of men who survived : ", rate_men, "%")
```

Percentage of women who survived: 74.0 %

Percentage of men who survived : 19.0 %

In [110]: df1 = df.copy()
df1['Age_Range'] = pd.cut(df1['Age'],
bins=[0.,15,30,45,60,65,np.inf],
labels=[1,2,3,4,5,6])
df1.head()

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Age_Range
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	2
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	3
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	4
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	2

In [111]: df1 = df.copy()
df1['AgeBand'] = pd.cut(df1['Age'], 5)



```
df1[['AgeBand', 'Survived']].groupby(['AgeBand'], as_index=False).mean().sort_values(by='AgeBand', ascending=True)
```

Out[111]:

	AgeBand	Survived
0	(0.34, 16.336]	0.56
1	(16.336, 32.252]	0.37
2	(32.252, 48.168]	0.40
3	(48.168, 64.084]	0.43
4	(64.084, 80.0]	0.09

In [112]:

```
Cabin_Not_NA = df1[df1['Cabin'].notna()]
Cabin_Not_NA.head()
```

Out[112]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	AgeBand
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	(48.168, 64.084]
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.70	G6	S	(0.34, 16.336]
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	C103	S	(48.168, 64.084]
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.00	D56	S	(32.252, 48.168]
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.50	A6	S	(16.336, 32.252]

In [113]:

```
Cabin_NaN = df1[df1['Cabin'].isnull()]
Cabin_NaN.head()
```

Out[113]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	AgeBand
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	(16.336, 32.252]
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	(16.336, 32.252]
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	(32.252, 48.168]
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	(16.336, 32.252]
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.07	NaN	C	(0.34, 16.336]

In [114]:

```
df1 = pd.get_dummies(df1, columns = ['Embarked'], drop_first=True)
df1.head()
```

Out[114]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	AgeBand	Embarked_Q	Embarked_S
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	(16.336, 32.252]	False	True
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	(16.336, 32.252]	False	True
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	(32.252, 48.168]	False	True
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	(48.168, 64.084]	False	True
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	(16.336, 32.252]	False	True

In [115]:

```
df1 = df.copy()
```

```
def Grade(Percentage):
    if Percentage >= 500:
        return 'High'
    if Percentage >= 300:
        return 'Medium'
    if Percentage >= 200:
        return 'Average'
    if Percentage >= 100:
        return 'Low'
    if Percentage >= 50:
```



```
    return 'VeryLow'  
    return 'Free'
```

```
df1['Fare_Range']=df1.apply(lambda x: Grade(x['Fare']),axis=1)  
df1.head()
```

Out[115]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Fare_Range
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	Free
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2-3101282	7.92	NaN	S	Free
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	Free
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	VeryLow
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	Free

```
In [116]: wine = pd.read_csv('WineQT.csv')  
wine.head(3)
```

Out[116]:	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.10	25.0	67.0	1.0	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.09	15.0	54.0	1.0	3.26	0.65	9.8	5	2

```
In [117]: # Create correlation matrix  
corr_matrix = wine.corr().abs()  
# Select upper triangle of correlation matrix  
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))  
# Find features with correlation greater than 0.95  
to_drop = [column for column in upper.columns if any(upper[column] > 0.30)]  
to_drop
```

```
Out[117]: ['citric acid',  
'total sulfur dioxide',  
'density',  
'pH',  
'sulphates',  
'alcohol',  
'quality',  
'Id']
```

```
In [118]: wine['good_quality']=[ "yes" if x>=7 else 'no' for x in wine['quality']]  
wine.head()
```

Out[118]:	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id	good_quality
0	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	0	no
1	7.8	0.88	0.00	2.6	0.10	25.0	67.0	1.0	3.20	0.68	9.8	5	1	no
2	7.8	0.76	0.04	2.3	0.09	15.0	54.0	1.0	3.26	0.65	9.8	5	2	no
3	11.2	0.28	0.56	1.9	0.07	17.0	60.0	1.0	3.16	0.58	9.8	6	3	no
4	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	4	no

```
In [119]: #Removing all Negative values
```

```
wine['residual sugar'] = wine['residual sugar'].abs()  
wine.head()
```

Out[119]:	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id	good_quality
0	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	0	no
1	7.8	0.88	0.00	2.6	0.10	25.0	67.0	1.0	3.20	0.68	9.8	5	1	no
2	7.8	0.76	0.04	2.3	0.09	15.0	54.0	1.0	3.26	0.65	9.8	5	2	no
3	11.2	0.28	0.56	1.9	0.07	17.0	60.0	1.0	3.16	0.58	9.8	6	3	no
4	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	4	no

```
In [120]: # Print Row  
row_30 = wine.iloc[75]  
print(row_30)
```



```

fixed acidity      7.8
volatile acidity  0.41
citric acid       0.68
residual sugar    1.7
chlorides         0.47
free sulfur dioxide 18.0
total sulfur dioxide 69.0
density          1.0
pH                3.08
sulphates        1.31
alcohol           9.3
quality          5
Id               106
good_quality     no
Name: '75, dtype: object

```

```
In [121]: any_negative_yield = (wine['chlorides'] < 0).any()

if any_negative_yield:
    print("The 'chlorides' column contains negative values.")
else:
    print("The 'chlorides' column does not contain negative values.")

The 'chlorides' column does not contain negative values.
```

```
In [122]: geo = pd.read_csv('gapminder_full.csv')
geo.head(3)
```

```
Out[122]:   country  year  population  continent  life_exp  gdp_cap
0  Afghanistan  1952     8425333      Asia     28.80    779.45
1  Afghanistan  1957     9240934      Asia     30.33    820.85
2  Afghanistan  1962    10267083      Asia     32.00    853.10
```

```
In [123]: geo[(geo['population']>10000000) & (geo['country']=='Afghanistan')][['gdp_cap','life_exp','continent']]
```

```
Out[123]:   gdp_cap  life_exp  continent
0     853.10    32.00      Asia
1     836.20    34.02      Asia
2     739.98    36.09      Asia
3     786.11    38.44      Asia
4     978.01    39.85      Asia
5     852.40    40.82      Asia
6     649.34    41.67      Asia
7     635.34    41.76      Asia
8     726.73    42.13      Asia
9     974.58    43.83      Asia
```

```
In [124]: reg_medal=geo.groupby(['country','continent']).size().reset_index().head(10)
reg_medal
```

```
Out[124]:   country  continent   0
0  Afghanistan      Asia  12
1    Albania        Europe  12
2    Algeria        Africa  12
3    Angola        Africa  12
4  Argentina      Americas  12
5  Australia      Oceania  12
6    Austria        Europe  12
7   Bahrain        Asia  12
8  Bangladesh        Asia  12
9   Belgium        Europe  12
```

```
In [125]: df2=geo.groupby('country')['continent'].nunique().reset_index()
df2.head()
```



```
Out[125]: country continent
```

0	Afghanistan	1
1	Albania	1
2	Algeria	1
3	Angola	1
4	Argentina	1

```
In [126]: geo.groupby('country')['continent'].count().nlargest(20).reset_index().head(10)
```

```
Out[126]: country continent
```

0	Afghanistan	12
1	Albania	12
2	Algeria	12
3	Angola	12
4	Argentina	12
5	Australia	12
6	Austria	12
7	Bahrain	12
8	Bangladesh	12
9	Belgium	12

```
In [127]: Highest_Population = geo.nlargest(10, 'population', keep='all')  
Highest_Population.head(10)
```

```
Out[127]: country year population continent life_exp gdp_cap
```

299	China	2007	1318683096	Asia	72.96	4959.11
298	China	2002	1280400000	Asia	72.03	3119.28
297	China	1997	1230075000	Asia	70.43	2289.23
296	China	1992	1164970000	Asia	68.69	1655.78
707	India	2007	1110396331	Asia	64.70	2452.21
295	China	1987	1084035000	Asia	67.27	1378.90
706	India	2002	1034172547	Asia	62.88	1746.77
294	China	1982	1000281000	Asia	65.53	962.42
705	India	1997	959000000	Asia	61.77	1458.82
293	China	1977	943455000	Asia	63.97	741.24

```
In [128]: Lowest_Population = geo.nsmallest(10, 'population', keep='all')  
Lowest_Population.head(10)
```

```
Out[128]: country year population continent life_exp gdp_cap
```

1296	Sao Tome and Principe	1952	60011	Africa	46.47	879.58
1297	Sao Tome and Principe	1957	61325	Africa	48.95	860.74
420	Djibouti	1952	63149	Africa	34.81	2669.53
1298	Sao Tome and Principe	1962	65345	Africa	51.89	1071.55
1299	Sao Tome and Principe	1967	70787	Africa	54.42	1384.84
421	Djibouti	1957	71851	Africa	37.33	2864.97
1300	Sao Tome and Principe	1972	76595	Africa	56.48	1532.99
1301	Sao Tome and Principe	1977	86796	Africa	58.55	1737.56
422	Djibouti	1962	89898	Africa	39.69	3020.99
1302	Sao Tome and Principe	1982	98593	Africa	60.35	1890.22

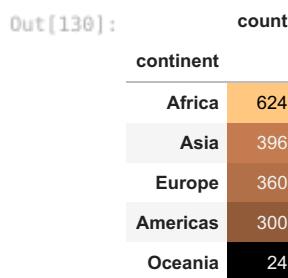
```
In [129]: Population = geo.groupby(by = ['country'], axis = 0)['population'].sum()  
Range = pd.DataFrame(Population).reset_index()  
Range.sort_values(by = ['population'], ascending = False, inplace = True)  
Range.head(10)
```



	country	population
24	China	11497920623
58	India	8413568878
134	United States	2738534790
59	Indonesia	1779874000
14	Brazil	1467745520
66	Japan	1341105696
97	Pakistan	1124200629
8	Bangladesh	1089064744
47	Germany	930564520
94	Nigeria	884496214

```
In [130]: print(f"\033[031m\033[1m")
print("Unique continent Names :", geo['continent'].nunique())
geo['continent'].value_counts().nlargest(10).to_frame().style.background_gradient(cmap='copper')
```

Unique continent Names : 5



```
In [131]: df1 = geo[geo['continent']=='Asia']
df1.groupby('country')['life_exp'].max().sort_values(ascending=False).head(5).reset_index()
```

Out[131]:

	country	life_exp
0	Japan	82.60
1	Hong Kong, China	82.21
2	Israel	80.75
3	Singapore	79.97
4	Korea, Rep.	78.62

```
In [132]: geo.groupby('country')['gdp_cap'].mean().sort_values(ascending=False).index[0:5]
```

Out[132]: Index(['Kuwait', 'Switzerland', 'Norway', 'United States', 'Canada'], dtype='object', name='country')

```
In [133]: geo.groupby('country')['gdp_cap'].mean().sort_values(ascending=False).head(10)
```

Out[133]:

country	gdp_cap
Kuwait	65332.91
Switzerland	27074.33
Norway	26747.31
United States	26261.15
Canada	22410.75
Netherlands	21748.85
Denmark	21671.82
Germany	20556.68
Iceland	20531.42
Austria	20411.92

Name: gdp_cap, dtype: float64

```
In [134]: data = geo[geo['continent']=='Africa']
data[data['gdp_cap'] == data['gdp_cap'].max()]['country']
```

Out[134]: 905 Libya
Name: country, dtype: object

```
In [135]: data = geo[geo['continent']=='Africa']
data[data['gdp_cap'] == data['gdp_cap'].min()]['country']
```

Out[135]: 334 Congo, Dem. Rep.
Name: country, dtype: object

```
In [136]: df1 = geo[(geo['continent'] == 'Europe')]
df1 = df1[df1['gdp_cap'] == df1['gdp_cap'].max()]
df1
```



```

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from lazypredict.Supervised import LazyClassifier
import matplotlib.pyplot as plt

# Assuming df is your DataFrame containing the Titanic dataset
titanic_data = df.copy()
titanic_data = titanic_data.dropna()

# Select relevant features
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']
X = titanic_data[features]
y = titanic_data['Survived']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a LazyClassifier
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metrics=None)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)
print("Models performance:")
models

```

	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
Model					
DecisionTreeClassifier	0.95	0.93	0.93	0.95	0.02
BernoulliNB	0.81	0.84	0.84	0.82	0.03
BaggingClassifier	0.86	0.84	0.84	0.87	0.06
GaussianNB	0.78	0.82	0.82	0.80	0.02
AdaBoostClassifier	0.84	0.82	0.82	0.84	0.14
LinearDiscriminantAnalysis	0.84	0.82	0.82	0.84	0.04
RidgeClassifierCV	0.84	0.82	0.82	0.84	0.02
RidgeClassifier	0.84	0.82	0.82	0.84	0.03
NearestCentroid	0.84	0.82	0.82	0.84	0.03
LogisticRegression	0.84	0.82	0.82	0.84	0.03
LinearSVC	0.84	0.82	0.82	0.84	0.03
PassiveAggressiveClassifier	0.76	0.80	0.80	0.77	0.03
NuSVC	0.81	0.80	0.80	0.82	0.13
CalibratedClassifierCV	0.84	0.78	0.78	0.84	0.07

LGBMClassifier	0.81	0.76	0.76	0.81	0.08
RandomForestClassifier	0.86	0.76	0.76	0.85	0.25
SVC	0.78	0.74	0.74	0.79	0.02
KNeighborsClassifier	0.84	0.74	0.74	0.83	0.03
ExtraTreesClassifier	0.84	0.74	0.74	0.83	0.20
SGDClassifier	0.76	0.73	0.73	0.77	0.02
XGBClassifier	0.81	0.72	0.72	0.81	0.11
ExtraTreeClassifier	0.81	0.69	0.69	0.80	0.02
LabelSpreading	0.76	0.65	0.65	0.75	0.02
LabelPropagation	0.76	0.65	0.65	0.75	0.02
Perceptron	0.65	0.54	0.54	0.65	0.02
QuadraticDiscriminantAnalysis	0.73	0.52	0.52	0.68	0.03
DummyClassifier	0.76	0.50	0.50	0.65	0.02

```
# If you want to evaluate a specific model (e.g., the best performing one)
if not models.empty:
    best_model = models.index[0]
    print(f"\nBest model: {best_model}")

    if best_model in predictions.columns:
        best_model_predictions = predictions[best_model]
        accuracy = accuracy_score(y_test, best_model_predictions)
        print(f"Accuracy of the best model: {accuracy}")
    else:
        print(f"Warning: Predictions for {best_model} not found in the predictions DataFrame.")
        print("Available models in predictions:")
        print(predictions.columns)
else:
    print("No models were successfully trained.")

print("\nShape of predictions DataFrame:", predictions.shape)
print("\nFirst few rows of predictions DataFrame:")
predictions
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
DecisionTreeClassifier	0.95	0.93	0.93	0.95	0.02
BernoulliNB	0.81	0.84	0.84	0.82	0.03
BaggingClassifier	0.86	0.84	0.84	0.87	0.06
GaussianNB	0.78	0.82	0.82	0.80	0.02
AdaBoostClassifier	0.84	0.82	0.82	0.84	0.14
LinearDiscriminantAnalysis	0.84	0.82	0.82	0.84	0.04
RidgeClassifierCV	0.84	0.82	0.82	0.84	0.02
RidgeClassifier	0.84	0.82	0.82	0.84	0.03
NearestCentroid	0.84	0.82	0.82	0.84	0.03
LogisticRegression	0.84	0.82	0.82	0.84	0.03
LinearSVC	0.84	0.82	0.82	0.84	0.03
PassiveAggressiveClassifier	0.76	0.80	0.80	0.77	0.03
NuSVC	0.81	0.80	0.80	0.82	0.13
CalibratedClassifierCV	0.84	0.78	0.78	0.84	0.07
LGBMClassifier	0.81	0.76	0.76	0.81	0.08
RandomForestClassifier	0.86	0.76	0.76	0.85	0.25
SVC	0.78	0.74	0.74	0.79	0.02
KNeighborsClassifier	0.84	0.74	0.74	0.83	0.03
ExtraTreesClassifier	0.84	0.74	0.74	0.83	0.20
SGDClassifier	0.76	0.73	0.73	0.77	0.02

```

plt.figure(figsize=(12, 30))
metrics = ['Accuracy', 'Balanced Accuracy', 'ROC AUC', 'F1 Score']

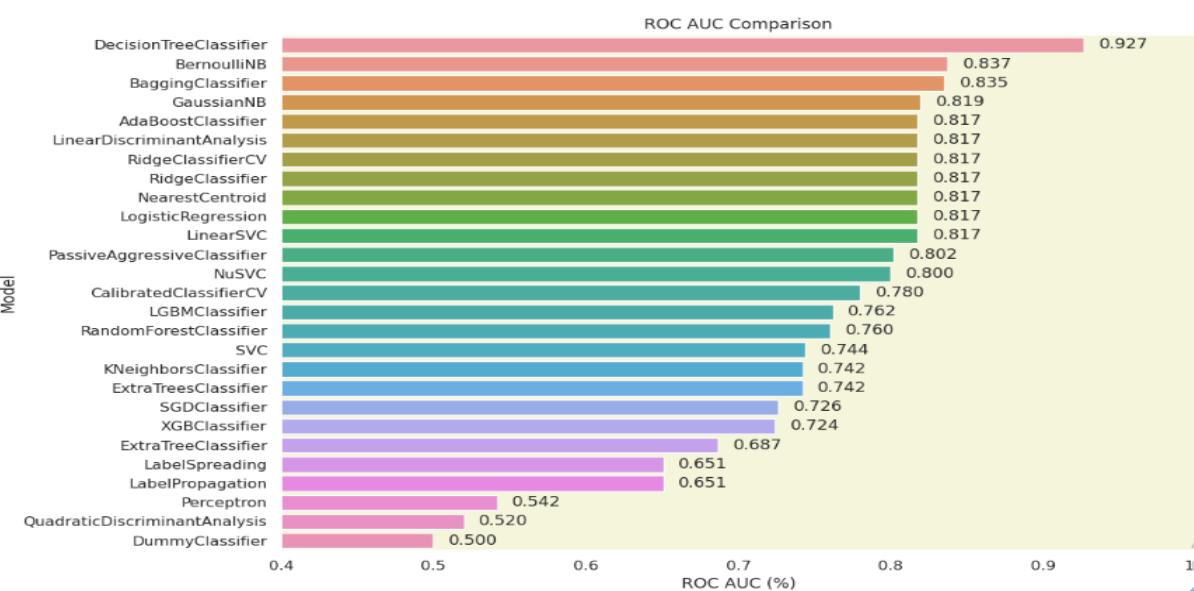
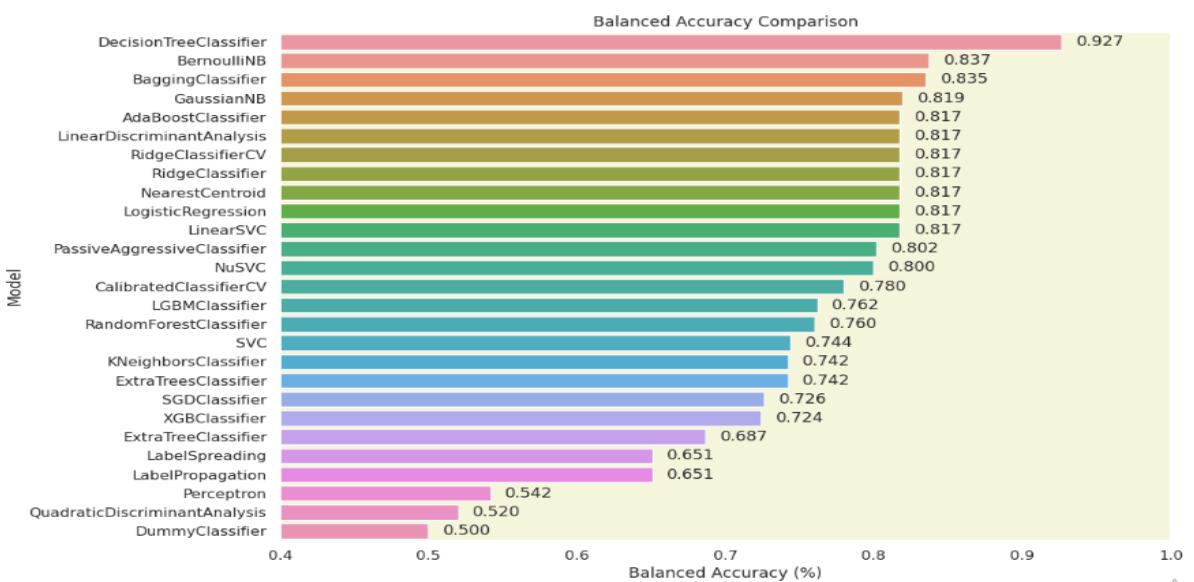
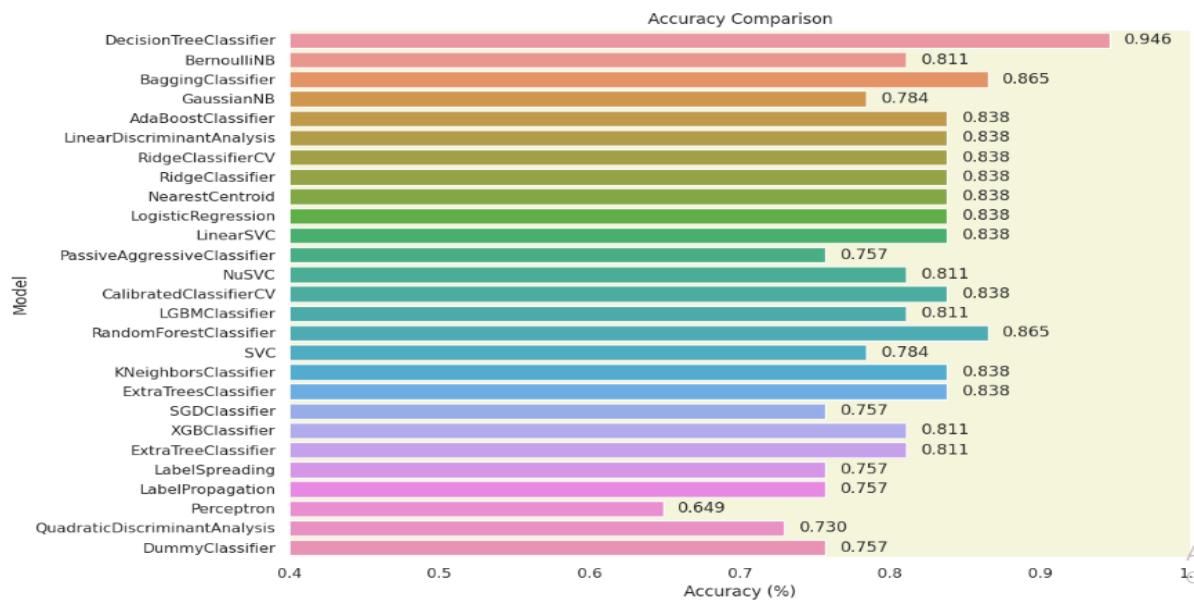
for i, metric in enumerate(metrics):
    plt.subplot(4, 1, i+1)
    ax = sns.barplot(x=models[metric], y=models.index, orient='h')
    plt.title(f'{metric} Comparison')
    plt.xlabel(f'{metric} (%)')
    plt.ylabel('Model')

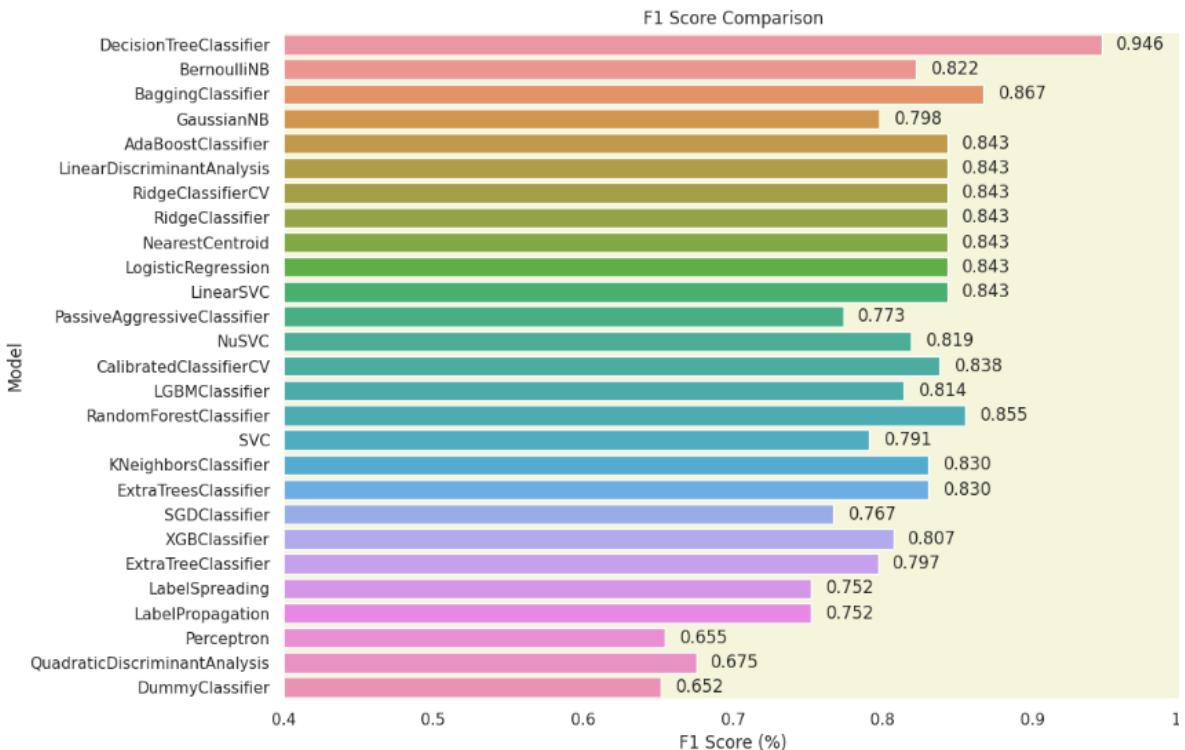
    # Set x-axis to start from 0.40
    plt.xlim(0.40, 1.0)

    # Add value labels on the bars
    for j, v in enumerate(models[metric]):
        ax.text(v + 0.01, j, f'{v:.3f}', va='center')

plt.tight_layout()
plt.show()

```





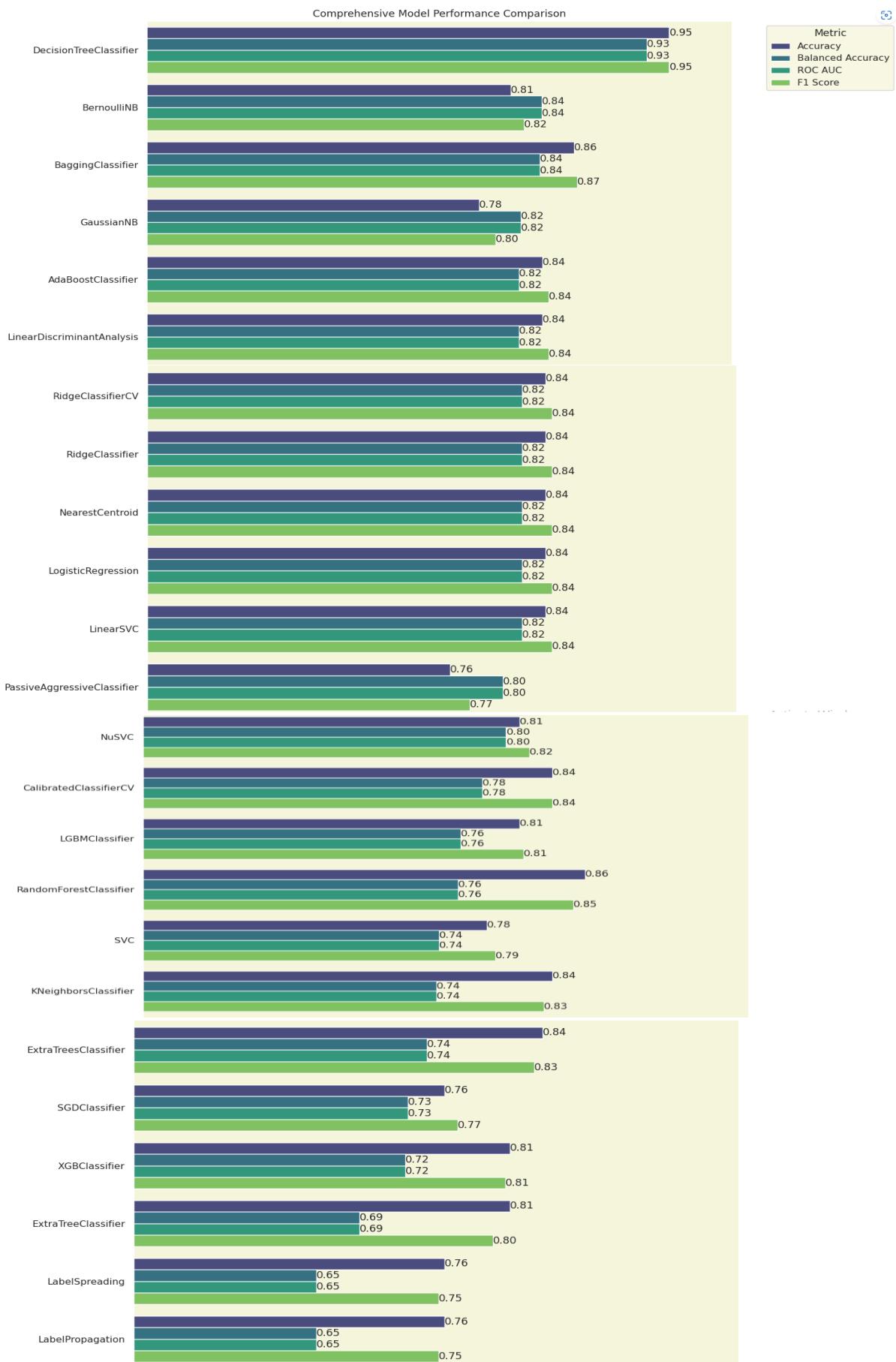
```
# Assuming 'models' and 'metrics' are already defined DataFrames
summary_df = models[metrics].copy()
summary_df['Model'] = models.index
summary_df = summary_df.melt(id_vars=['Model'], var_name='Metric',
                             value_name='Score')

plt.figure(figsize=(15, 30))
barplot = sns.barplot(x='Score', y='Model', hue='Metric', data=summary_df,
                      orient='h', palette='viridis')

# Add values at the edge of the bars
for container in barplot.containers:
    barplot.bar_label(container, fmt='%.2f', label_type='edge')

plt.title('Comprehensive Model Performance Comparison')
plt.xlabel('Score (%)')
plt.ylabel('Model')
plt.legend(title='Metric', bbox_to_anchor=(1.05, 1), loc='upper left')

# Set x-axis to start from 0.50
plt.xlim(0.50, 1.0)
plt.tight_layout()
plt.show()
```



All Types of Data Visualization Kaggle Note Book

Python for Machine Learning Visualization Part 01:

<https://www.kaggle.com/code/pythonafroz/python-for-machine-learning-visualization-part-01>

Python for Machine Learning Visualization Part 02:

<https://www.kaggle.com/code/pythonafroz/python-for-machine-learning-visualization-part-03>

```
In [128...]: import pandas as pd
import numpy as np
import seaborn as sns

import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots

pd.set_option('display.precision', 2)
```

```
In [2]: # Load the dataset
df = pd.read_csv("heart_disease_uci.csv")
df = df.dropna()
df.head(2)
```

```
Out[2]: id age sex dataset cp trestbps chol fbs restecg thalch exang oldpeak slope ca thal num
0 1 63 Male Cleveland typical angina 145.0 233.0 True lv hypertrophy 150.0 False 2.3 down sloping 0.0 fixed defect 0
1 2 67 Male Cleveland asymptomatic 160.0 286.0 False lv hypertrophy 108.0 True 1.5 flat 3.0 normal 2
```

```
In [3]: print(f"Records: {df.shape[0]}")
print(f"Columns: {df.shape[1]}")
```

```
Records: 299
Columns: 16
```

```
In [4]: top_leagues = df['cp'].value_counts().nlargest(4).index
display(top_leagues)

plt.figure(figsize=(15, 6))
sns.scatterplot(x='age', y='chol', data=df[df['cp'].isin(top_leagues)], hue='cp')
plt.title('Age vs. Cholesterol for Top 4 Chest Pain')
plt.xlabel('Age')
plt.ylabel('Cholesterol')
plt.legend(title='Chest Pain Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

```
Index(['asymptomatic', 'non-anginal', 'atypical angina', 'typical angina'], dtype='object', name='cp')
```

Age vs. Cholesterol for Top 4 Chest Pain

Chest Pain Type

- typical angina
- asymptomatic
- non-anginal
- atypical angina

```
In [130...]: import plotly.express as px

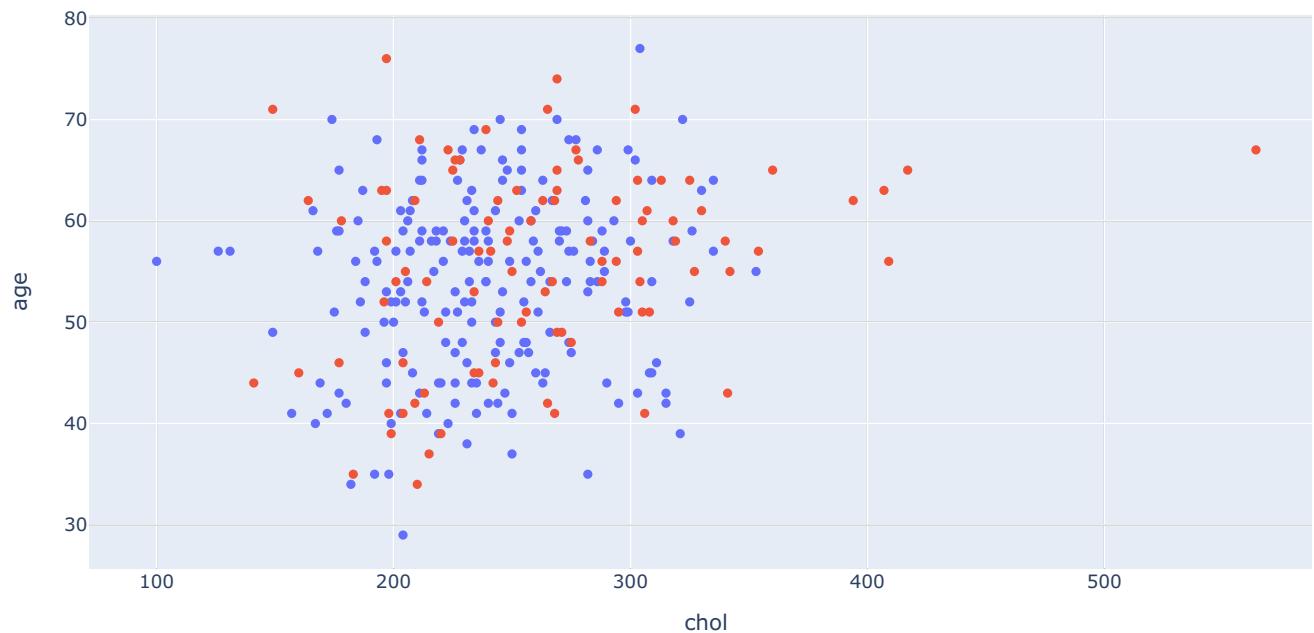
fig = px.scatter(df, x='chol', y='age', color='sex')

fig.update_layout(width=1000, height=500)
```



```
fig.update_layout(title_text='Scatter Plot of Cholesterol vs. Age (colored by Sex)')
fig.show()
```

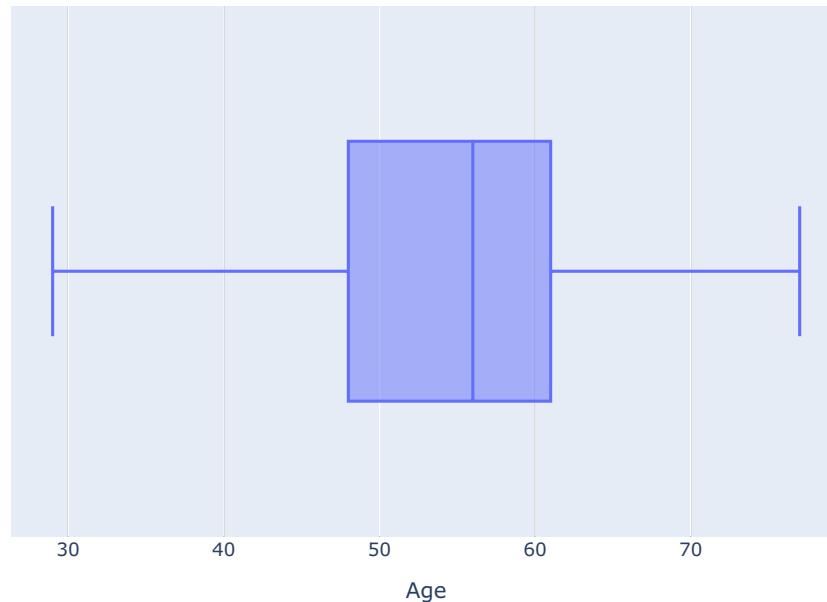
Scatter Plot of Cholesterol vs. Age (colored by Sex)



```
In [6]: from plotly.offline import iplot
```

```
fig = px.box(x = df["age"],
              labels={"x": "Age"},
              title="5-Number-Summary(Box Plot) of Age")
iplot(fig)
```

5-Number-Summary(Box Plot) of Age



```
In [131...]
```

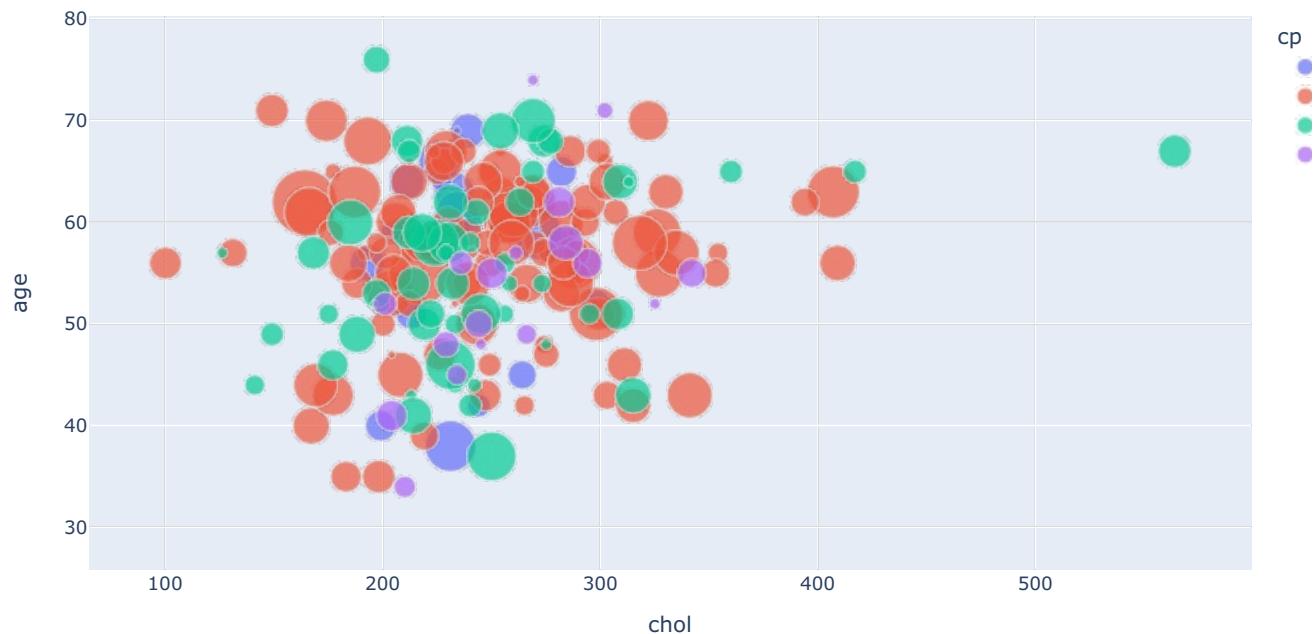
```
import plotly.express as px

fig = px.scatter(df, x='chol', y='age', color='cp', size = 'oldpeak', size_max = 30, hover_name = 'exang')
fig.update_layout(width=1000, height=500)
fig.update_layout(title_text='Scatter Plot of Cholesterol vs. Age (colored by cp)')
```



```
fig.show()
```

Scatter Plot of Cholesterol vs. Age (colored by cp)



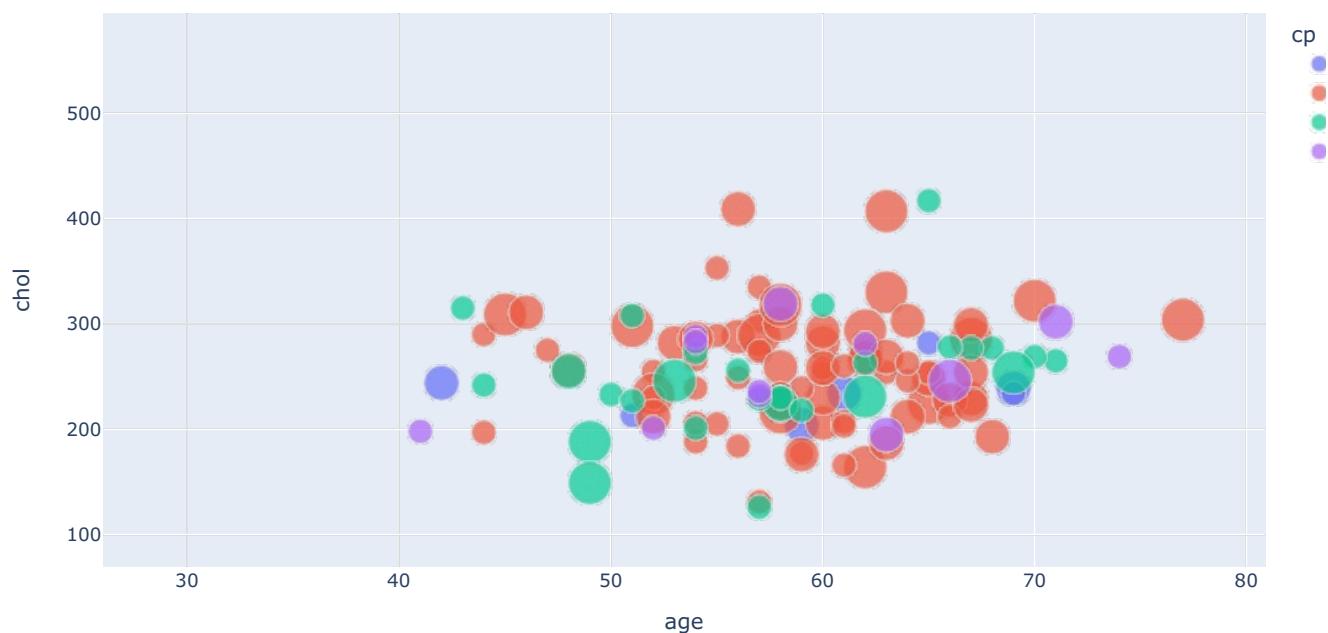
In [132]:

```
fig = px.scatter(data_frame = df,
                  x="age",
                  y="chol",
                  color="cp",
                  size='ca',
                  hover_data=['oldpeak'])

fig.update_layout(title_text="Cholesterol Vs Age",
                  titlefont={'size': 24, 'family':'Serif'},
                  width=1000,
                  height=500,
                  )

fig.show()
```

Cholesterol Vs Age



In [133]:

```
import plotly.express as px
```

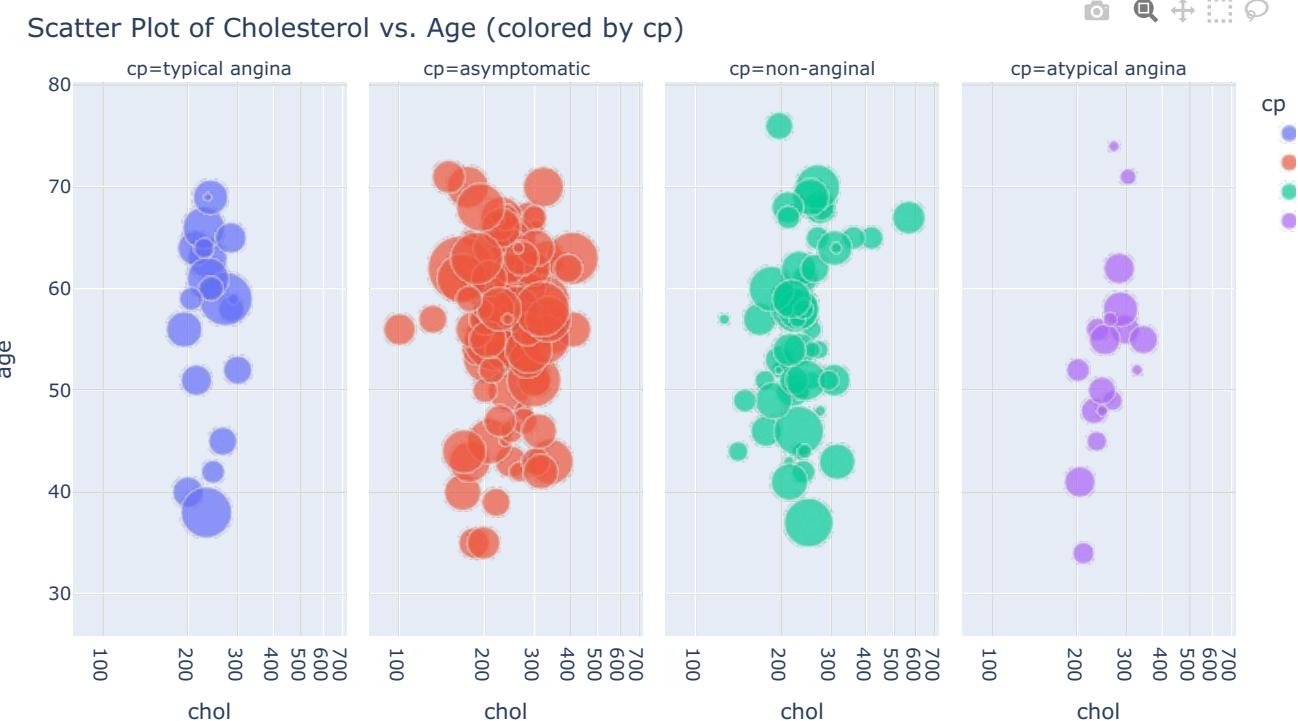
```
fig = px.scatter(df, x='chol', y='age', color='cp', size = 'oldpeak', size_max = 30, hover_name = 'exang', facet
```



```

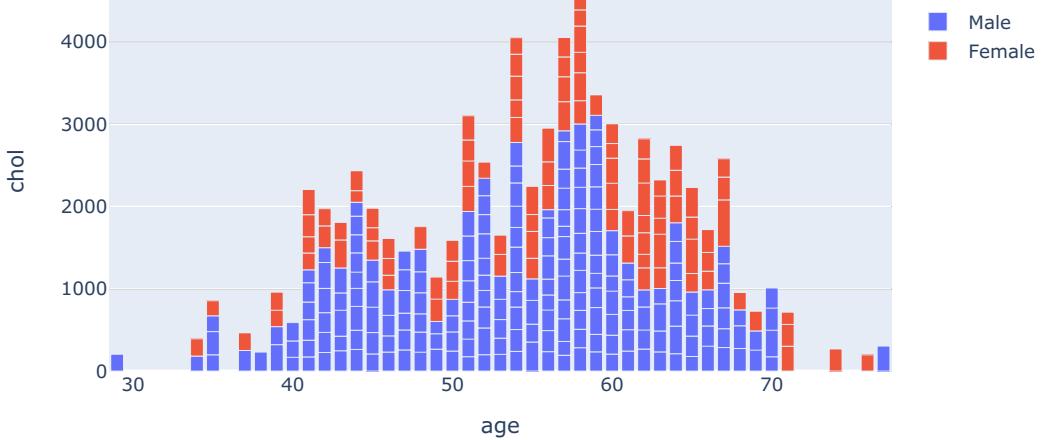
fig.update_layout(width=1000, height=500)
fig.update_layout(title_text='Scatter Plot of Cholesterol vs. Age (colored by cp)')
fig.show()

```



`hover_name='exang'` means that the values in the 'exang' column will be shown as tooltips when you hover over the data points on the scatter plot. This is useful for providing additional information about each data point without cluttering the plot with labels.

```
In [10]: fig=px.bar(df,x='age',y='chol',hover_data=['oldpeak'],color='sex',height=400)
fig.show()
```



```

In [11]: def generate_rating_df(df):
    rating_df = df.groupby(['cp', 'slope']).agg({'id': 'count'}).reset_index()
    rating_df = rating_df[rating_df['id'] != 0]
    rating_df.columns = ['cp', 'slope', 'counts']
    rating_df = rating_df.sort_values('slope')
    return rating_df

rating_df = generate_rating_df(df)
fig = px.bar(rating_df, x='cp', y='counts', color='slope')

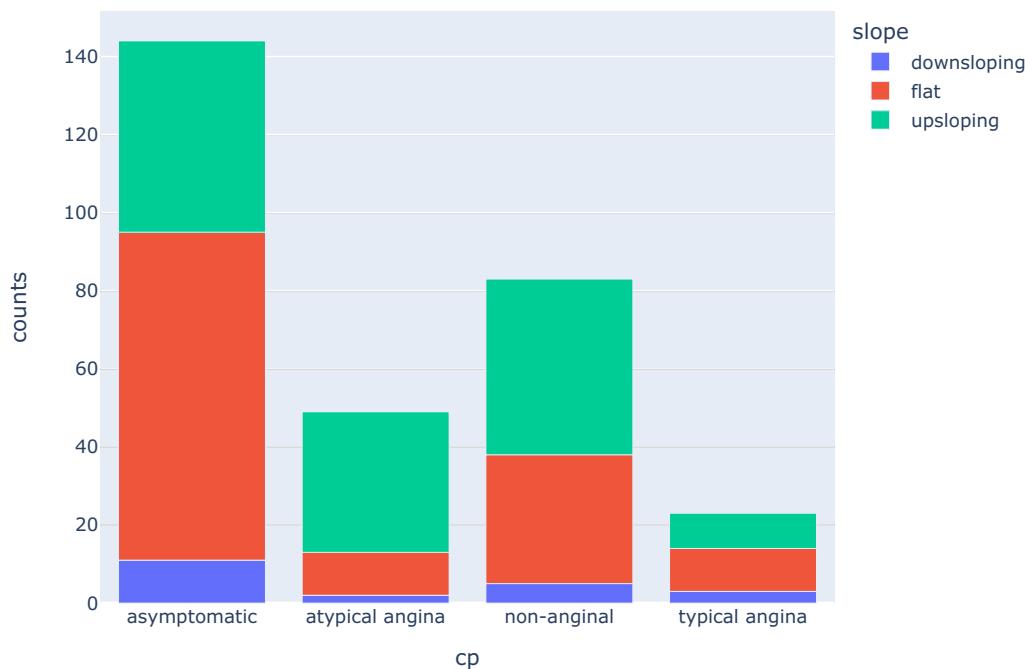
fig.update_traces(textposition='auto',
                  textfont_size=20)

fig.update_layout(barmode='stack')

```



```
fig.show()
```



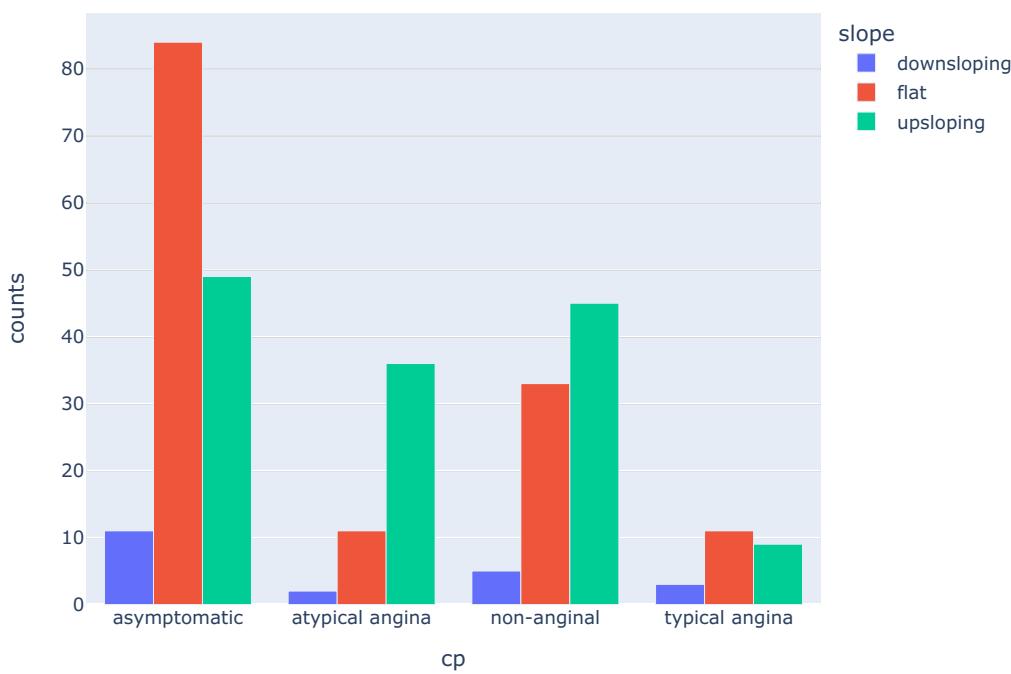
```
In [12]: def generate_rating_df(df):
    rating_df = df.groupby(['cp', 'slope']).agg({'id': 'count'}).reset_index()
    rating_df = rating_df[rating_df['id'] != 0]
    rating_df.columns = ['cp', 'slope', 'counts']
    rating_df = rating_df.sort_values('slope')
    return rating_df

rating_df = generate_rating_df(df)
fig = px.bar(rating_df, x='cp', y='counts', color='slope')

fig.update_traces(textposition='auto',
                   textfont_size=20)

fig.update_layout(barmode='group')

fig.show()
```



```
In [13]: import plotly.express as px
```

```

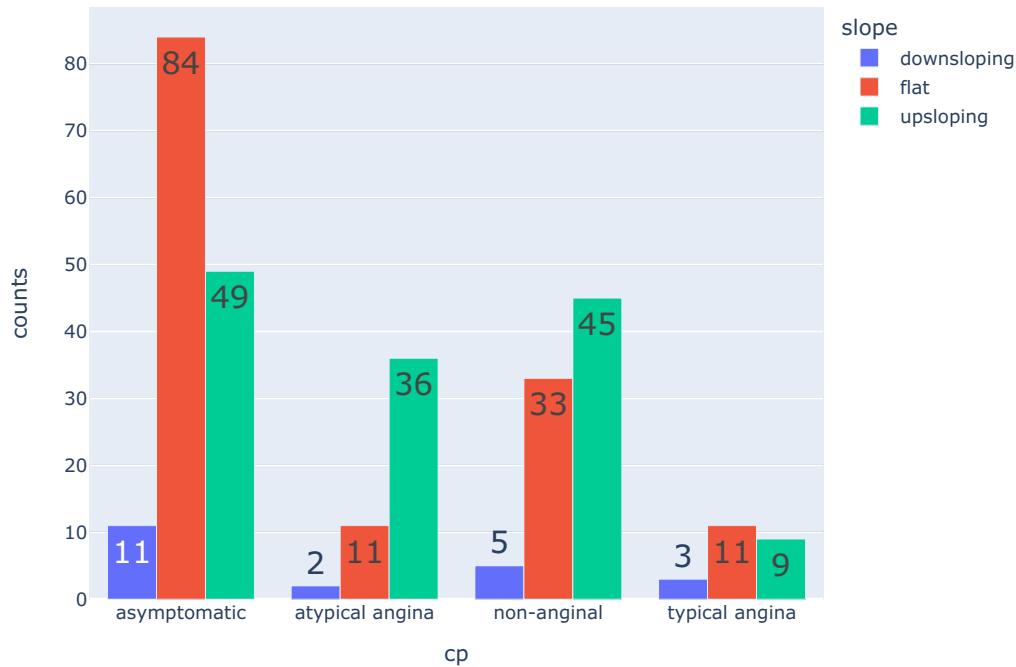
def generate_rating_df(df):
    rating_df = df.groupby(['cp', 'slope']).agg({'id': 'count'}).reset_index()
    rating_df = rating_df[rating_df['id'] != 0]
    rating_df.columns = ['cp', 'slope', 'counts']
    rating_df = rating_df.sort_values('slope')
    return rating_df

rating_df = generate_rating_df(df)

fig = px.bar(rating_df, x='cp', y='counts', color='slope', barmode='group',
             text='counts',
             )
fig.update_traces(textposition='auto',
                   textfont_size=20)

fig.show()

```



```

In [14]: def generate_rating_df(df):
    rating_df = df.groupby(['cp', 'slope']).agg({'id': 'count'}).reset_index()
    rating_df = rating_df[rating_df['id'] != 0]
    rating_df.columns = ['cp', 'slope', 'counts']
    rating_df = rating_df.sort_values('slope')

    # Calculate percentages
    total_counts = rating_df['counts'].sum()
    rating_df['percentage'] = rating_df['counts'] / total_counts * 100

    return rating_df

rating_df = generate_rating_df(df)

fig = px.bar(rating_df, x='cp', y='counts', color='slope', text='percentage')

fig.update_traces(
    texttemplate='%.1f',
    textposition='outside',
    textfont_size=16
)

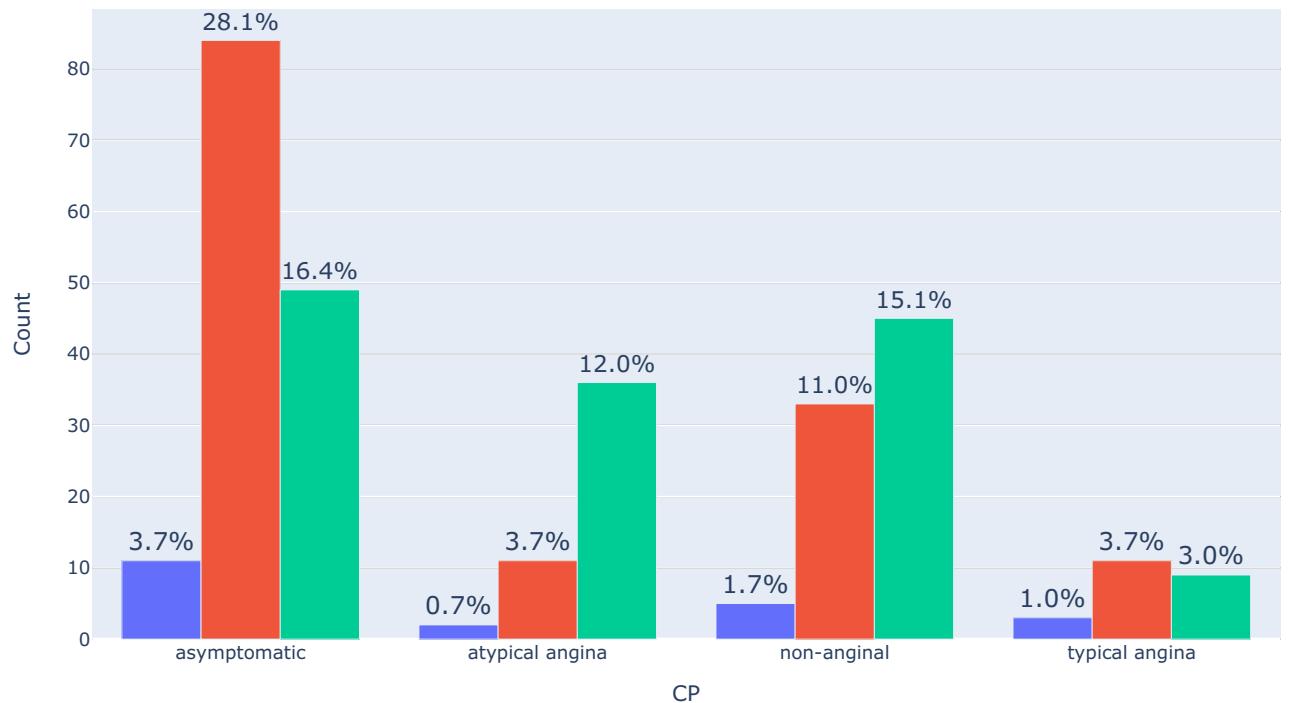
fig.update_layout(
    barmode='group',
    yaxis_title='Count',
    xaxis_title='CP',
    legend_title='Slope'
)

fig.update_layout(
    height=550,
    width=1000,
    title_text="Distribution of Chest Pain Type by Percentage",
    title_font_size=24
)

```

```
fig.show()
```

Distribution of Chest Pain Type by Percentage



In [15]:

```
fig = px.scatter(data_frame = df,
                  x="age",
                  y="chol",
                  color="cp",
                  size='ca',
                  hover_data=['oldpeak'],
                  marginal_x="histogram",
                  marginal_y="box",)

fig.update_layout(title_text="Age vs Cholesterol",
                  titlefont={'size': 24, 'family':'Serif'},
                  width=1000,
                  height=550,
                  )

fig.show()
```

Age vs Cholesterol

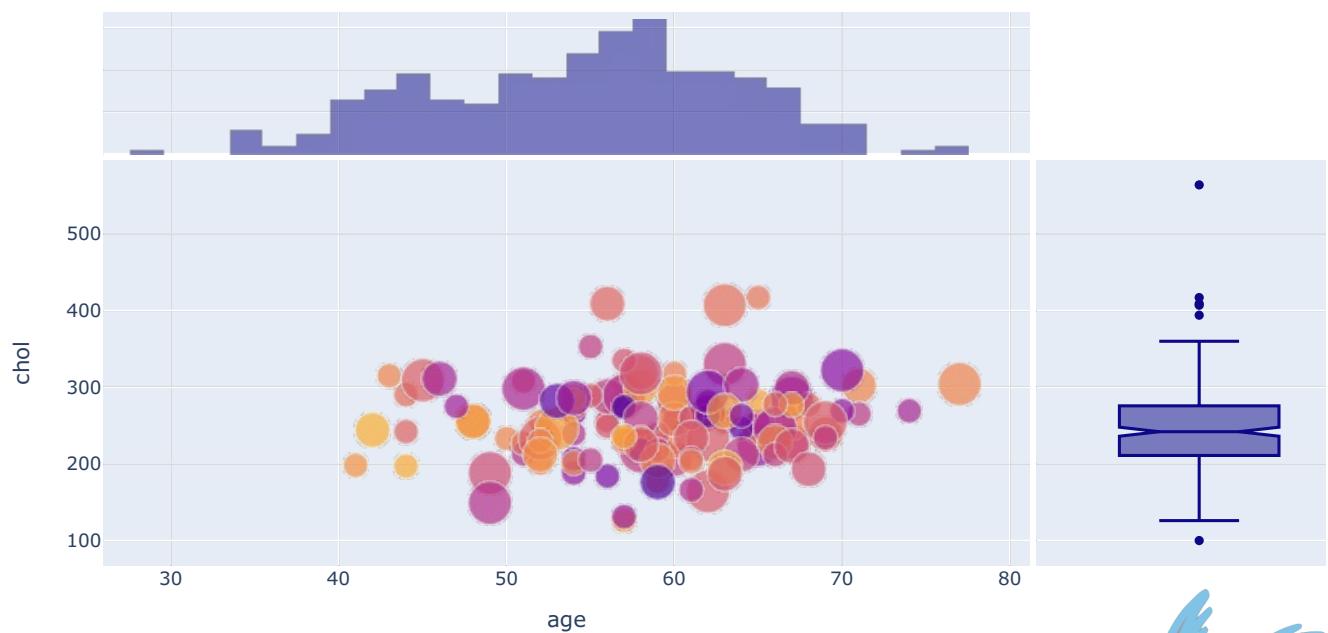


```
In [134]: fig = px.scatter(data_frame = df,
                      x="age",
                      y="chol",
                      color="thalch",
                      size='ca',
                      hover_data=['oldpeak'],
                      marginal_x="histogram",
                      marginal_y="box")

fig.update_layout(title_text="Age vs Cholesterol",
                  titlefont={'size': 24, 'family':'Serif'},
                  width=1000,
                  height=500,
                  )

fig.show()
```

Age vs Cholesterol



```
In [135]: fig = px.scatter(data_frame = df,
                      x="age",
```

```

y="chol",
size ="ca",
size_max=30,
color= "sex",
trendline="ols")
fig.update_layout(title_text=<b> Age vs Cholesterol </b>,
                  titlefont={'size': 24, 'family':'Serif'},
                  width=1000,
                  height=500,
)
fig.show()

```

Age vs Cholesterol



In [18]:

```

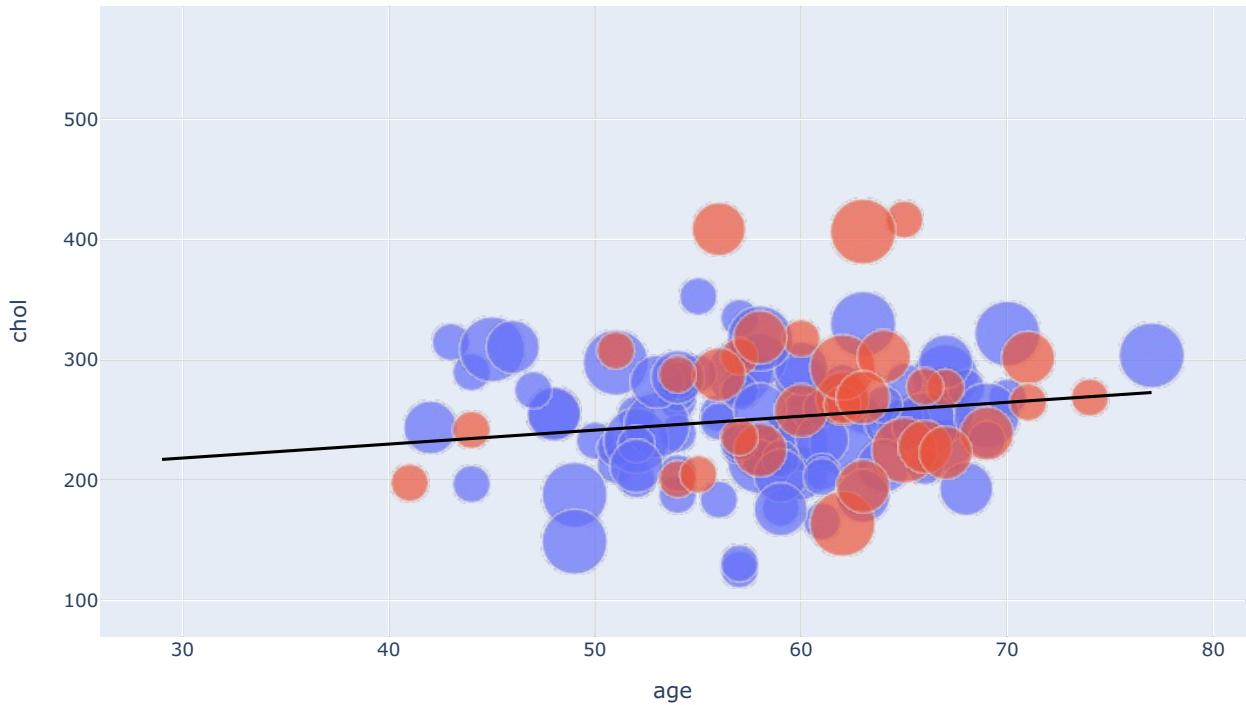
fig = px.scatter(data_frame = df,
                  x="age",
                  y="chol",
                  size ="ca",
                  size_max=30,
                  color= "sex",
                  trendline="ols",
                  trendline_scope="overall",
                  trendline_color_override="black")

fig.update_layout(title_text=<b>Chest Pain vs Gender</b>,
                  titlefont={'size': 24, 'family':'Serif'},
                  width=1000,
                  height=550,
)
fig.show()

```

Chest Pain vs Gender

camera search refresh

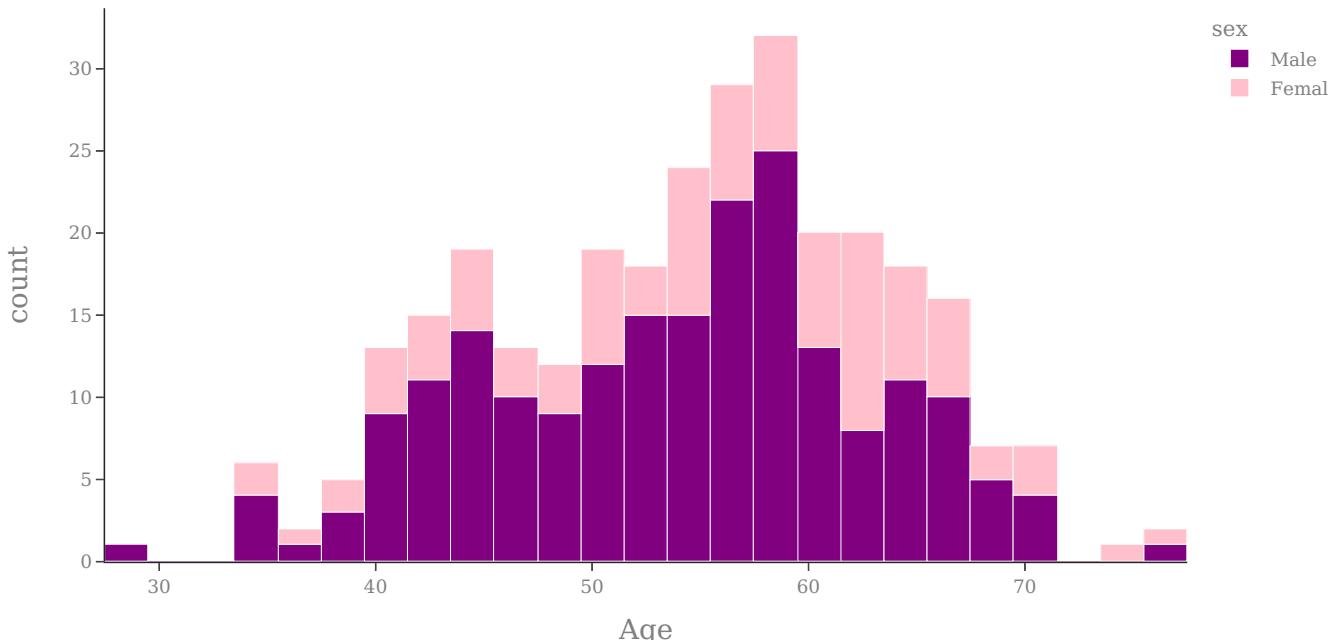


```
In [19]: fig = px.histogram(df, x='age', height=500, width=900, template='simple_white',
                      color='sex', # adding categorical column
                      color_discrete_sequence=['purple','pink'])
```

```
fig.update_layout(title={'text':'Histogram of Persons by Age', 'font':{'size':25}},
                  title_font_family="Times New Roman",
                  title_font_color="darkgrey",
                  title_x=0.2)

fig.update_layout(
    font_family='classic-roman',
    font_color= 'grey',
    yaxis_title={'text': " count", 'font': {'size':18}},
    xaxis_title={'text': " Age", 'font': {'size':18}})
)
fig.show()
```

Histogram of Persons by Age



```
In [20]: import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Assuming df is your DataFrame
asymptomatic = df[df['cp'] == 'asymptomatic']
non_anginal = df[df['cp'] == 'non-anginal']
atypical_angina = df[df['cp'] == 'atypical angina']
typical_angina = df[df['cp'] == 'typical angina']

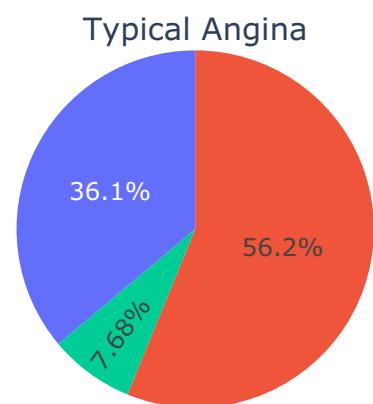
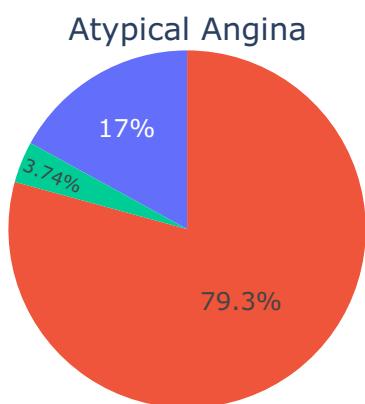
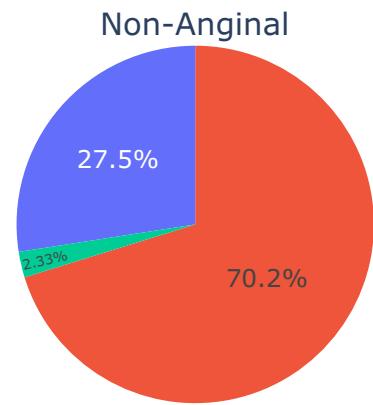
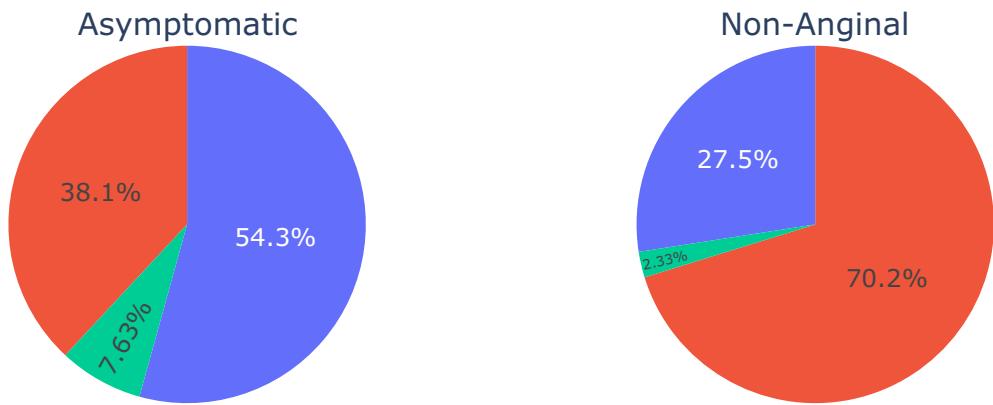
fig = make_subplots(rows=2,
                     cols=2,
                     specs=[[{'type':'domain'}, {'type':'domain'}],
                            [{'type':'domain'}, {"type":'domain'}]],
                     subplot_titles=("Asymptomatic", "Non-Anginal",
                                    "Atypical Angina", "Typical Angina"))

fig.add_trace(go.Pie(labels=asymptomatic["thal"], values=asymptomatic["chol"], name="asymptomatic"), 1, 1)
fig.add_trace(go.Pie(labels=non_anginal["thal"], values=non_anginal["chol"], name="non_anginal"), 1, 2)
fig.add_trace(go.Pie(labels=atypical_angina["thal"], values=atypical_angina["chol"], name="atypical_angina"), 2, 1)
fig.add_trace(go.Pie(labels=typical_angina["thal"], values=typical_angina["chol"], name="typical_angina"), 2, 2)

# Update layout to increase the size of the plot and add main title
fig.update_layout(
    height=800,
    width=1000,
    title_text="Distribution of Cholesterol Levels by Chest Pain Type",
    title_font_size=24
)

# Update traces
fig.update_traces(textposition='inside', textfont_size=16)
fig.update_annotations(font_size=20)
fig.show()
```

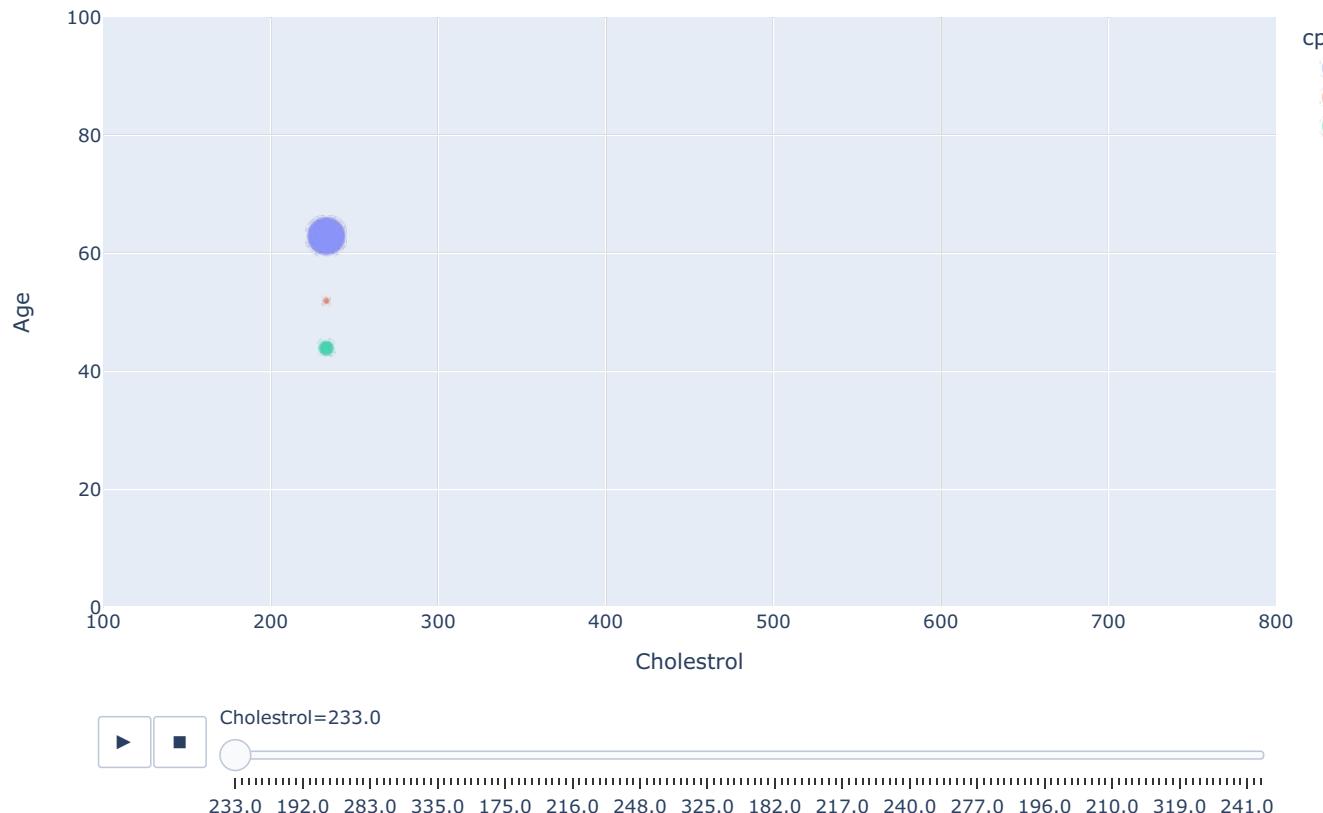
Distribution of Cholesterol Levels by Chest Pain Type



```
In [21]: import plotly.express as px

fig = px.scatter(df, x='chol', y='age', color='cp', size = 'oldpeak', size_max = 30, hover_name = 'exang',
                  labels = dict(oldpeak = 'oldpeak', chol = 'Cholesterol', age = "Age" ), animation_frame = "chol"
                  )
fig.update_layout(width=1000, height=600)
fig.update_layout(title_text='Scatter Plot of Cholesterol vs. Age (colored by cp) with Animation')
fig.show()
```

Scatter Plot of Cholesterol vs. Age (colored by cp) with Animation



```
In [22]: from plotly.offline import iplot

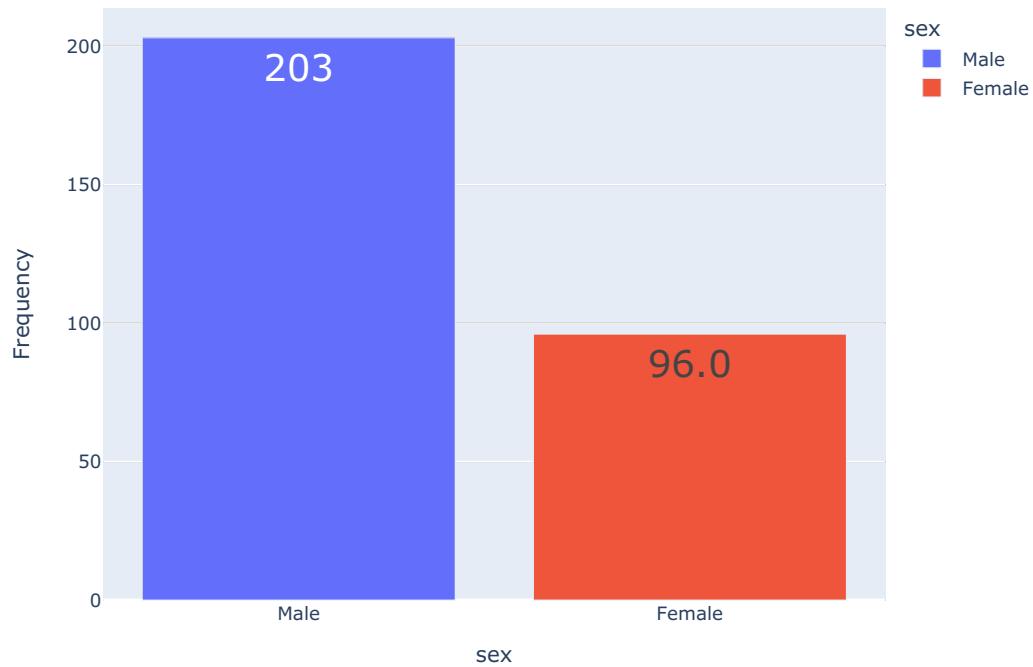
gender = df[["sex"]].value_counts()
display(gender.head().to_frame())

fig = px.bar(data_frame=gender,
              x = gender.index,
              y = gender,
              color=gender.index,
              text_auto="0.3s",
              labels={"y": "Frequency", "index": "Gender"})

)
fig.update_traces(textfont_size=24)

iplot(fig)
```

	count
sex	
Male	203
Female	96

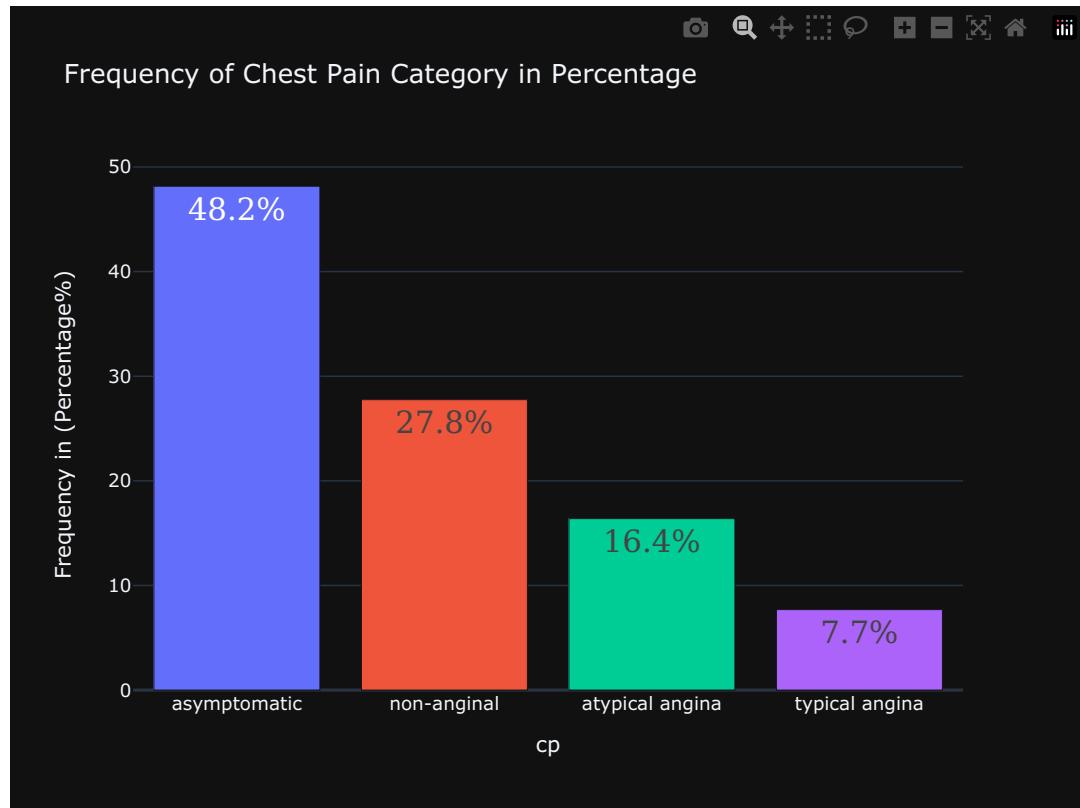


```
In [23]: from plotly.offline import iplot
category = df["cp"].value_counts()

fig = px.bar(category,
              x = category.index,
              y = (category / sum(category)) * 100,
              color=category.index,
              labels={"y" : "Frequency in (Percentage)", "category": "Category"},
              title="Frequency of Chest Pain Category in Percentage",
              text = category.apply(lambda x: f'{(x / sum(category)) * 100:.1f}%'),
              template="plotly_dark"
            )

fig.update_layout(showlegend=False)
fig.update_traces(
    textfont= {
        "family": "consolas",
        "size": 20,
    }
)

iplot(fig)
```

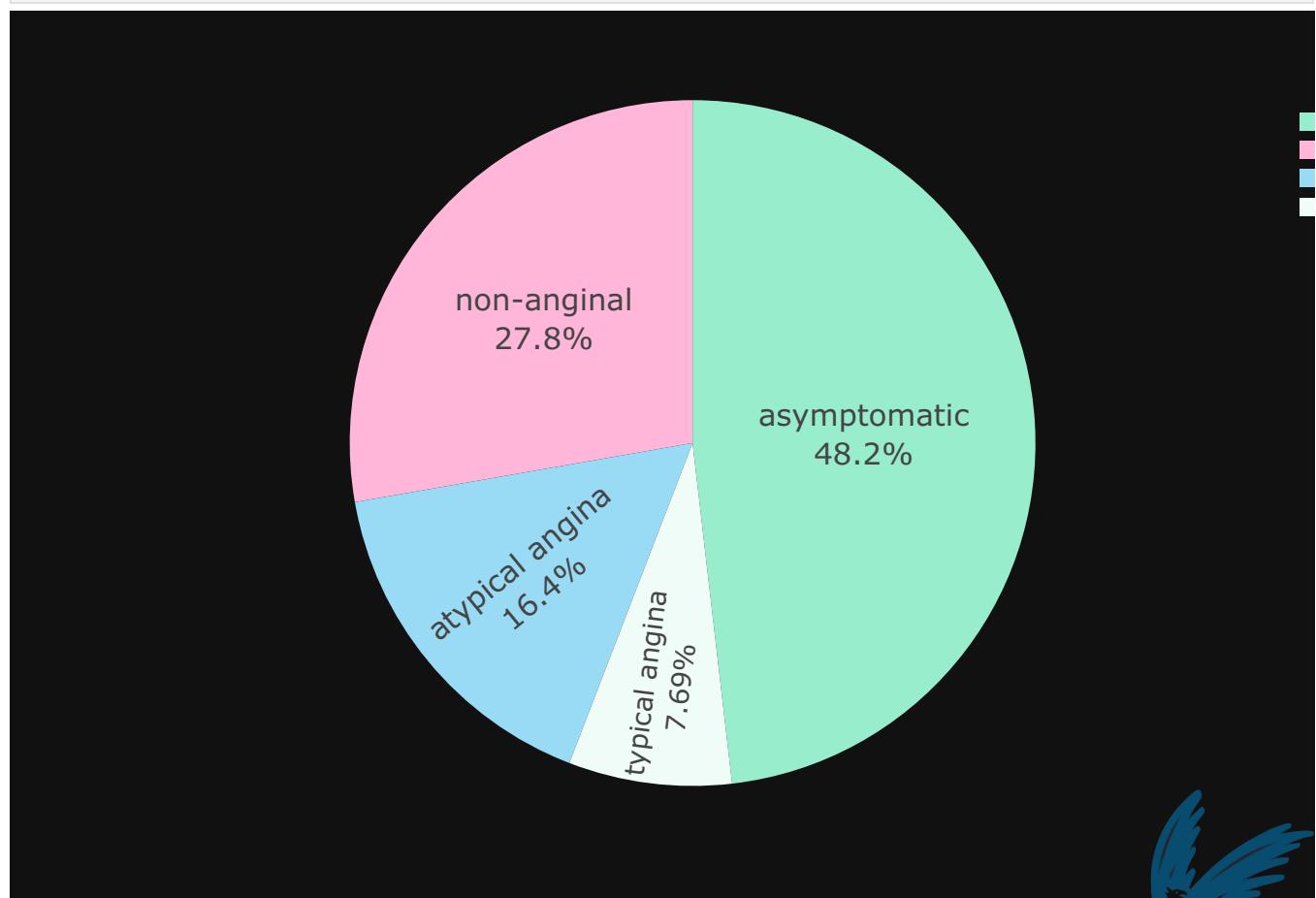


```
In [24]: from plotly.offline import iplot
ChestPain = df["cp"].value_counts()

fig = px.pie(values=ChestPain, names = ChestPain.index,
              color_discrete_sequence= ["#98EECC", "#FFB6D9", "#99DBF5"],
              template="plotly_dark"
            )

fig.update_traces(textposition='inside', textfont_size= 20, textinfo='percent+label')
fig.update_layout(showlegend=True, width=1000, height=600)

iplot(fig)
```



```
In [25]: cp = df["cp"].value_counts()
```



```

fig = px.bar(cp,
    y = cp.index,
    x = (cp / sum(cp)) * 100,
    color=cp.index,
    labels={"x" : "Frequency in Percentage(%)", "cp":"Chest Pain"},
    orientation="h",
    title="Frequency of Chest Pain",
    text = cp.apply(lambda x: f'{(x / sum(cp)) * 100:.1f}%'),
)
fig.update_layout(showlegend=True, width=1000, height=600)

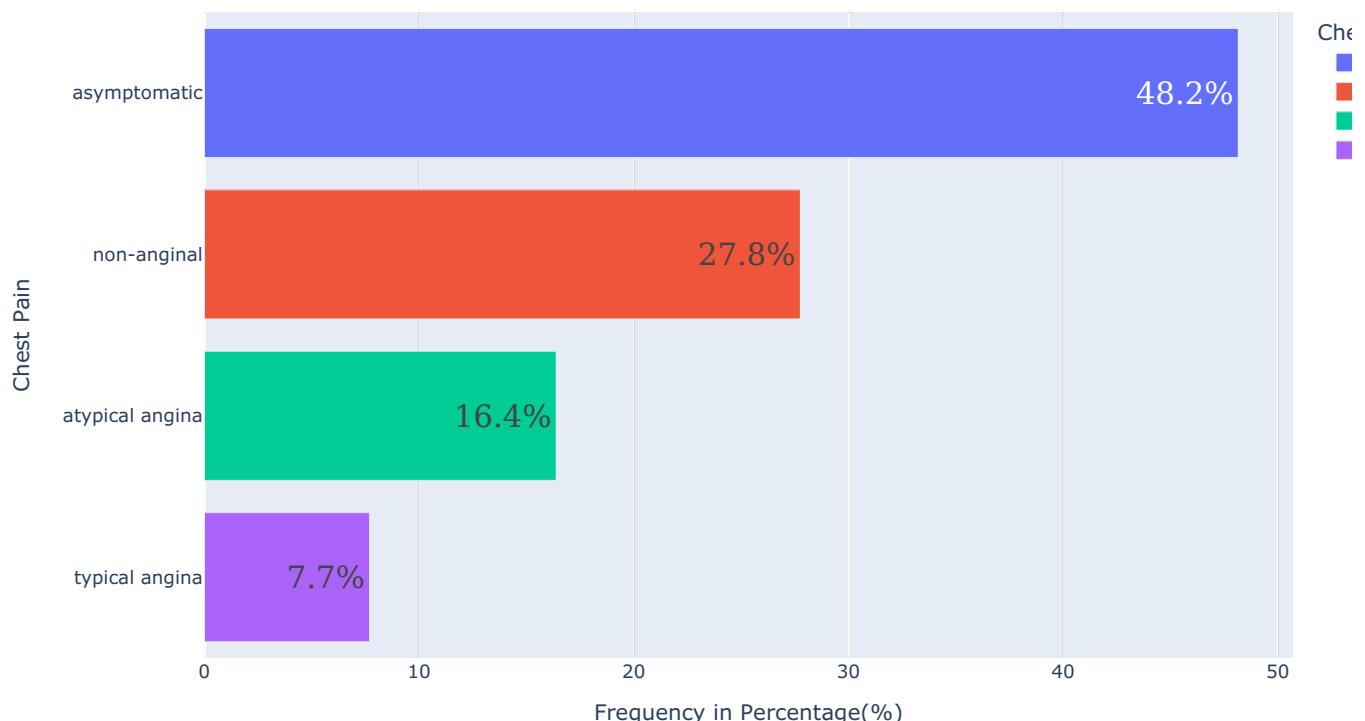
fig.update_traces(
    textfont= {
        "family": "consolas",
        "size": 20
    }
)

iplot(fig)

```



Frequency of Chest Pain



```

In [26]: fig=px.pie(df.groupby('cp',as_index=False)[['sex']].count().sort_values(by='sex',ascending=False).reset_index(drop=True),
                    names='cp',values='sex',color='sex',color_discrete_sequence=px.colors.sequential.Plasma_r,
                    labels={'cp':'Chest Pain','Sex':'Count'},template='seaborn',hole=0.4)

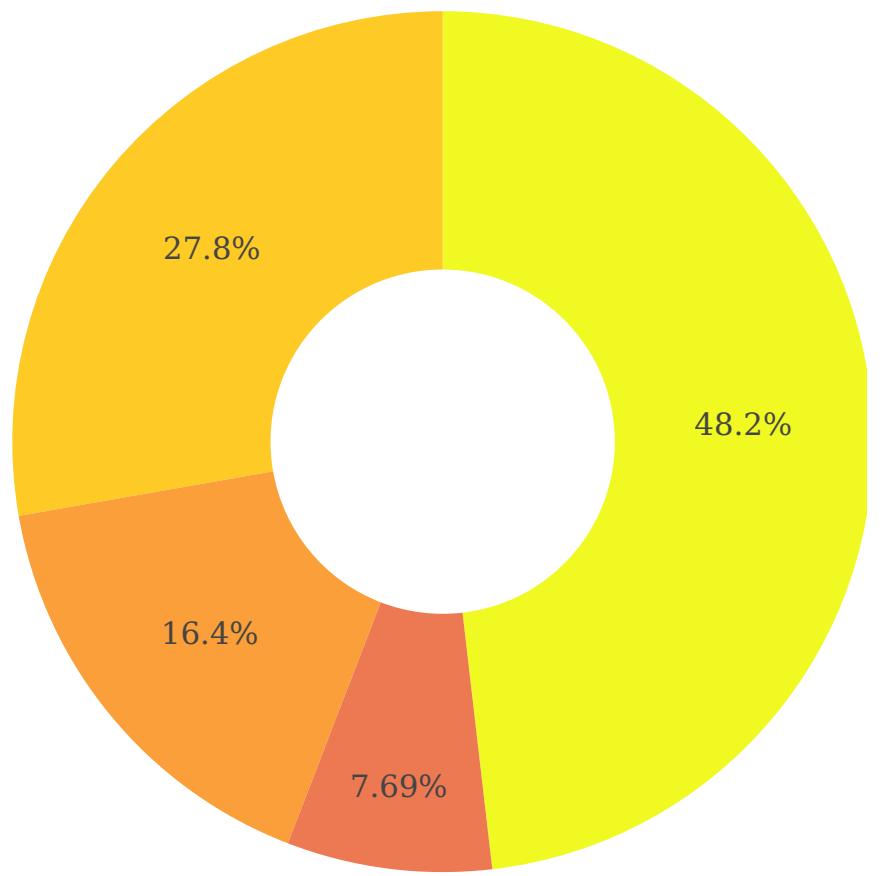
fig.update_layout(autosize=False, width=1200, height=700,legend=dict(orientation='v', yanchor='bottom',y=0.40,x=0.5, showlegend=True))

fig.update_traces(
    textfont= {
        "family": "consolas",
        "size": 20
    }
)

fig.show()

```

Chest Pain



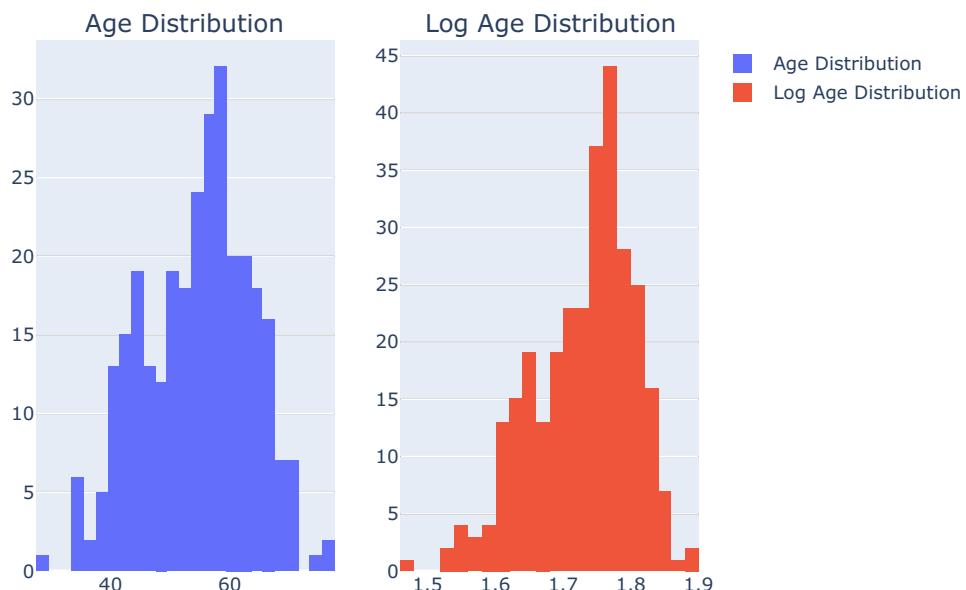
```
In [27]: import plotly.express as px
from plotly.offline import iplot
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(1,2,subplot_titles=('Age Distribution','Log Age Distribution'))

fig.append_trace(go.Histogram(x=df['age'],
                               name='Age Distribution') ,1,1)

fig.append_trace(go.Histogram(x=np.log10(df['age']),
                               name='Log Age Distribution') ,1,2)

iplot(dict(data=fig))
```



```
In [28]: import numpy as np
import plotly.graph_objs as go
from plotly.offline import iplot

# Calculate quartiles and IQR
Q25 = np.quantile(df['chol'], q=0.25)
Q75 = np.quantile(df['chol'], q=0.75)
IQR = Q75 - Q25
cut_off = IQR * 1.5

# Print number of outliers
print('Number of Cholesterol Lower Outliers:', df[df['chol'] <= (Q25 - cut_off)][['chol']].count())
print('Number of Cholesterol Upper Outliers:', df[df['chol'] >= (Q75 + cut_off)][['chol']].count())

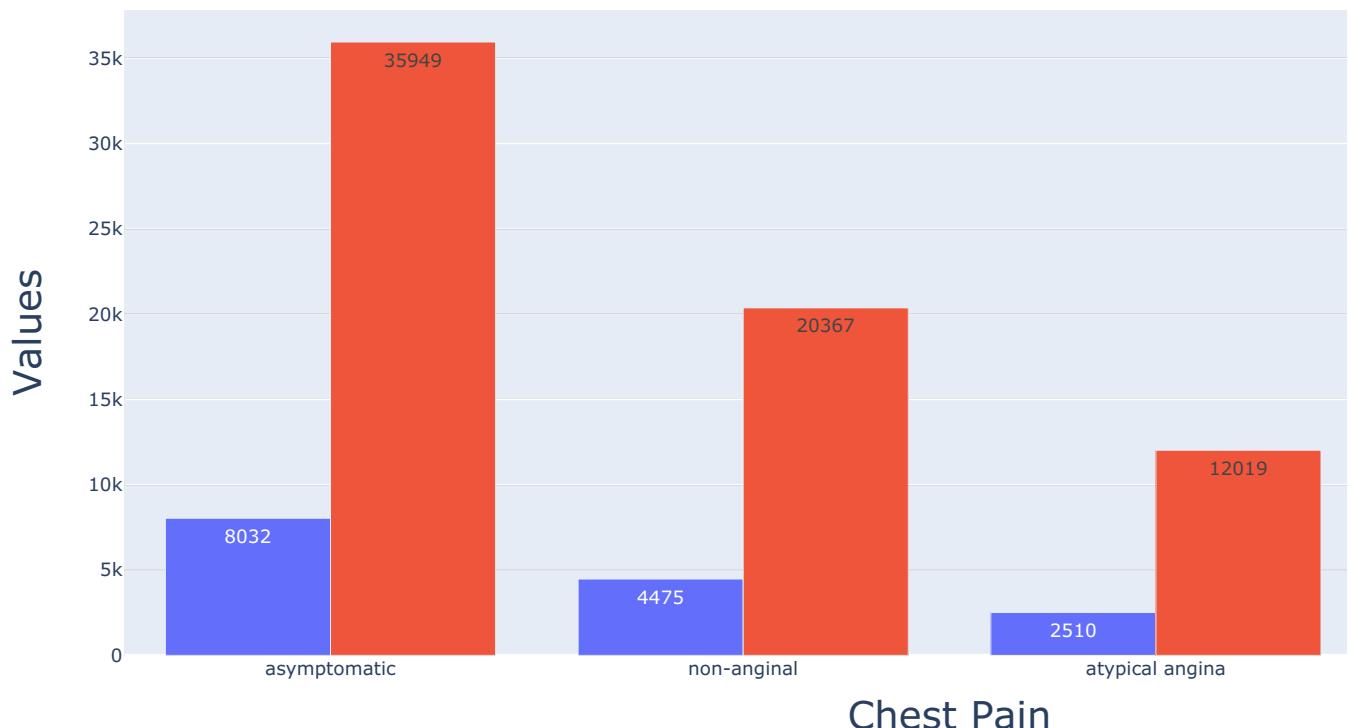
# Group by 'cp' and sort by 'age'
temp = df.groupby('cp').sum().sort_values('age', ascending=False)

# Create bar data
data = [
    go.Bar(x=temp.index, y=temp['age'], name='Age', text=temp['age'], textposition='auto'),
    go.Bar(x=temp.index, y=temp['chol'], name='Cholesterol', text=temp['chol'], textposition='auto')
]

# Define layout
layout = go.Layout(
    xaxis=dict(title='Chest Pain', titlefont=dict(size=25)),
    yaxis=dict(title='Values', titlefont=dict(size=25)),
    showlegend=True,
    width=1300,
    height=600
)

# Create figure and plot
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```

Number of Cholesterol Lower Outliers: 1
Number of Cholesterol Upper Outliers: 5



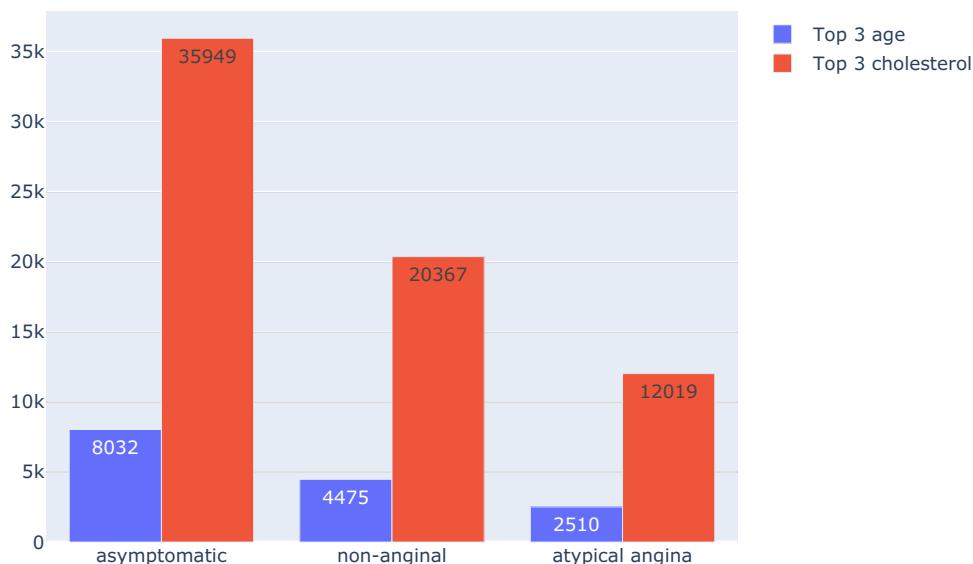
```
In [29]: import plotly.graph_objs as go
from plotly.offline import iplot

# Assuming df is your DataFrame
top_03_cp = df.groupby('cp').sum()['age'].sort_values(ascending=False)[0:3]
top_03_AGE = df.groupby(by='cp').sum().sort_values(by='age', ascending=False)[0:3]['chol']

data = [
    go.Bar(
        x=top_03_cp.index,
        y=top_03_cp,
        name='Top 3 age',
        text=top_03_cp,
        textposition='auto'
    ),
    go.Bar(
        x=top_03_AGE.index,
        y=top_03_AGE,
        name='Top 3 cholesterol',
        text=top_03_AGE,
        textposition='auto'
    )
]

layout = go.Layout(
    title="Grouped Bar Plot For Age and Cholesterol<br>(For The Top Three types of Chest Pain)",
    barmode='group'
)
iplot(dict(data=data, layout=layout))
```

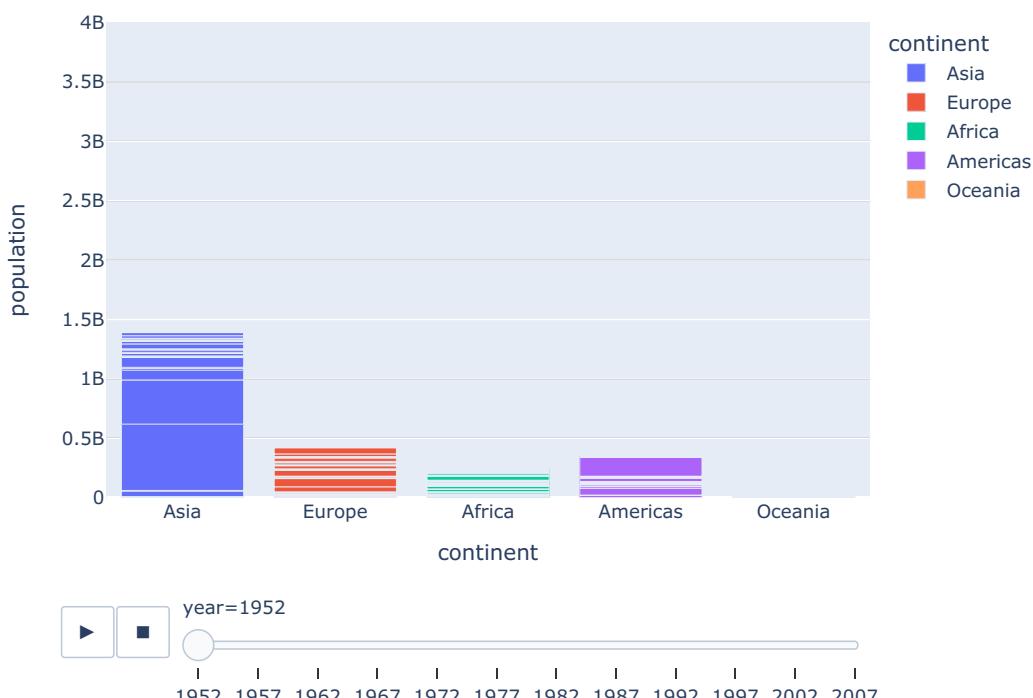
Grouped Bar Plot For Age and Cholesterol (For The Top Three types of Chest Pain)



```
In [37]: gap_df = pd.read_csv("gapminder_full.csv")
display(gap_df.head(2))

fig = px.bar(data_frame=gap_df,
              x="continent",
              y="population",
              color="continent",
              animation_frame="year",
              animation_group="country",
              range_y=[0,4000000000])
fig.show()
```

	country	year	population	continent	life_exp	gdp_cap
0	Afghanistan	1952	8425333	Asia	28.80	779.45
1	Afghanistan	1957	9240934	Asia	30.33	820.85



```
In [40]: fig = px.scatter(gap_df,x='gdp_cap',y='life_exp',color='continent',size='population',size_max=60,hover_name='co
animation_frame="year",animation_group='country',log_x=True,range_x=[100,100000],range_y=[25,90])
```



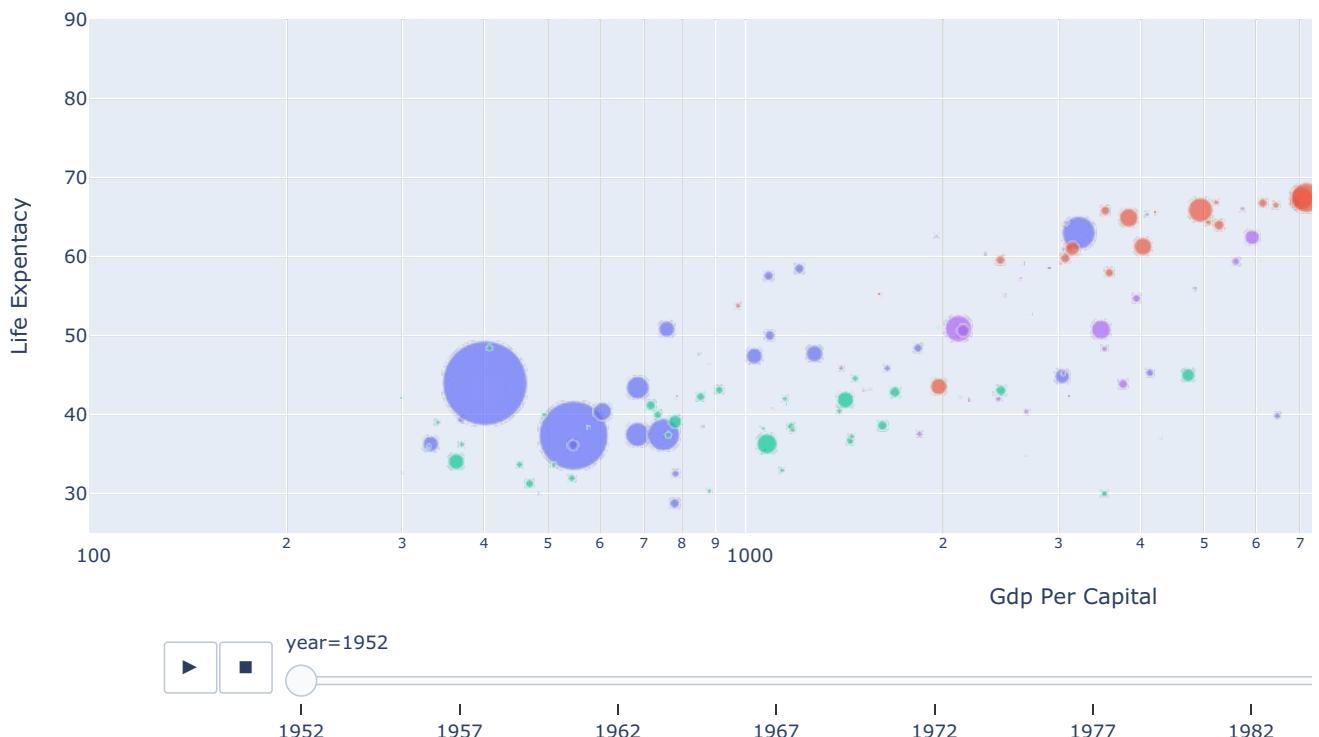
```

        labels=dict(Population ="Populations",gdp_cap="Gdp Per Capital",life_exp="Life Expentacy"))

fig.update_layout(
    height=550,
    width=1500,
    title_text="Distribution of GDP Cap Vs Life Expentacy",
    title_font_size=24
)
fig.show()

```

Distribution of GDP Cap Vs Life Expentacy



```
In [41]: #Grouping the data by state
df1 = df[['cp','age','chol','num']]
df1.groupby('cp').sum().head(10).style.background_gradient(cmap='Blues')
```

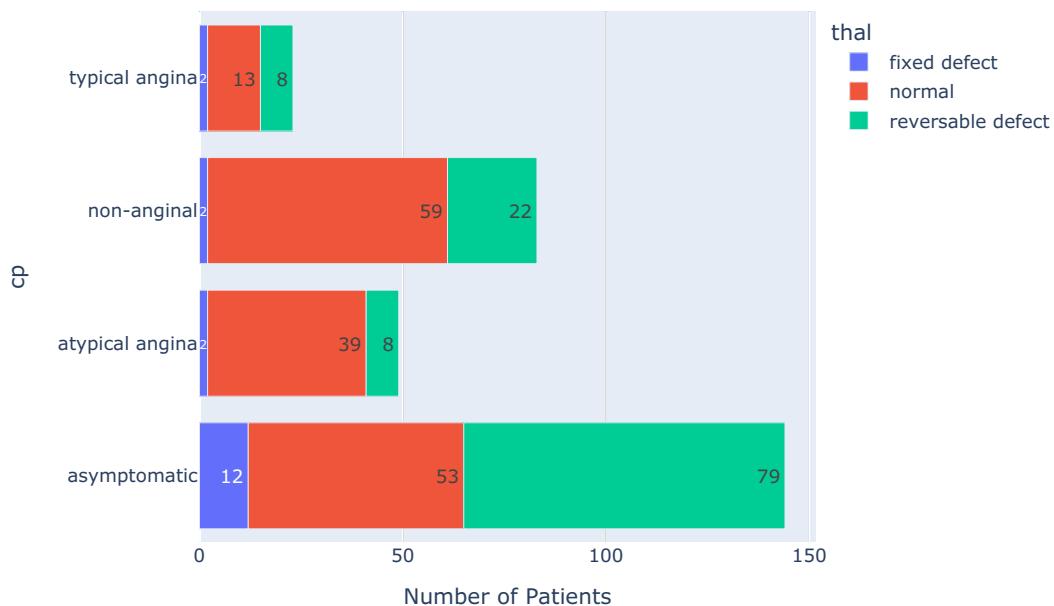
	age	chol	num
cp			
asymptomatic	8032	35949.000000	225
atypical angina	2510	12019.000000	14
non-anginal	4475	20367.000000	33
typical angina	1285	5454.000000	11

```
In [42]: import pandas as pd
import plotly.express as px

grouped_df = df.groupby(['cp', 'thal']).size().reset_index(name='count')

fig = px.bar(grouped_df,
             y="cp",
             x='count',
             color='thal',
             title='Count of Passengers by cp and thal',
             labels={'count': 'Number of Patients'},
             text_auto=True)
fig.show()
```

Count of Passengers by cp and thal



```
In [43]: # color palette for visualizations
import matplotlib.pyplot as plt

colors = ['#2B2E4A', '#E84545', '#903749', '#53354A',]
palette = sns.color_palette(palette = colors)

sns.palplot(palette, size = 2.5)

plt.text(-0.5,
         -0.7,
         'Color Palette',
         {'font':'monospace',
          'size': 24,
          'weight':'normal'}
        )

plt.show()
```

Color Palette



```
In [44]: def format_title(title, subtitle=None, subtitle_font=None, subtitle_font_size=None):
    title = f'{title}
    if not subtitle:
        return title
    subtitle = f'{subtitle}
    return f'{title}<br>{subtitle}'
```

```
import plotly.figure_factory as ff

z = df.groupby(['cp', 'thal']).chol.size().unstack()
z = z.values.tolist()
x = z.columns.tolist()
y = z.index.tolist()

fig = ff.create_annotationed_heatmap(z = z,
                                     x = x,
                                     y = y,
```

```

        xgap = 3,
        ygap = 3,
        colorscale = ['#53354A', '#E84545']
    )

title = format_title('cp',
                      'thal.',
                      'Chol',
                      12
                    )

fig.update_layout(title_text = title,
                  title_x = 0.5,
                  titlefont={'size': 24,
                             'family': 'Proxima Nova',
                             },
                  template='plotly_dark',
                  paper_bgcolor='#2B2E4A',
                  plot_bgcolor='#2B2E4A',

                  xaxis = {'side': 'bottom'},
                  xaxis_showgrid = False,
                  yaxis_showgrid = False,
                  yaxis_autorange = 'reversed',
                  )

fig.show()

```



```

In [45]: # available templates
template = ['ggplot2','plotly_dark', 'seaborn', 'simple_white', 'plotly']

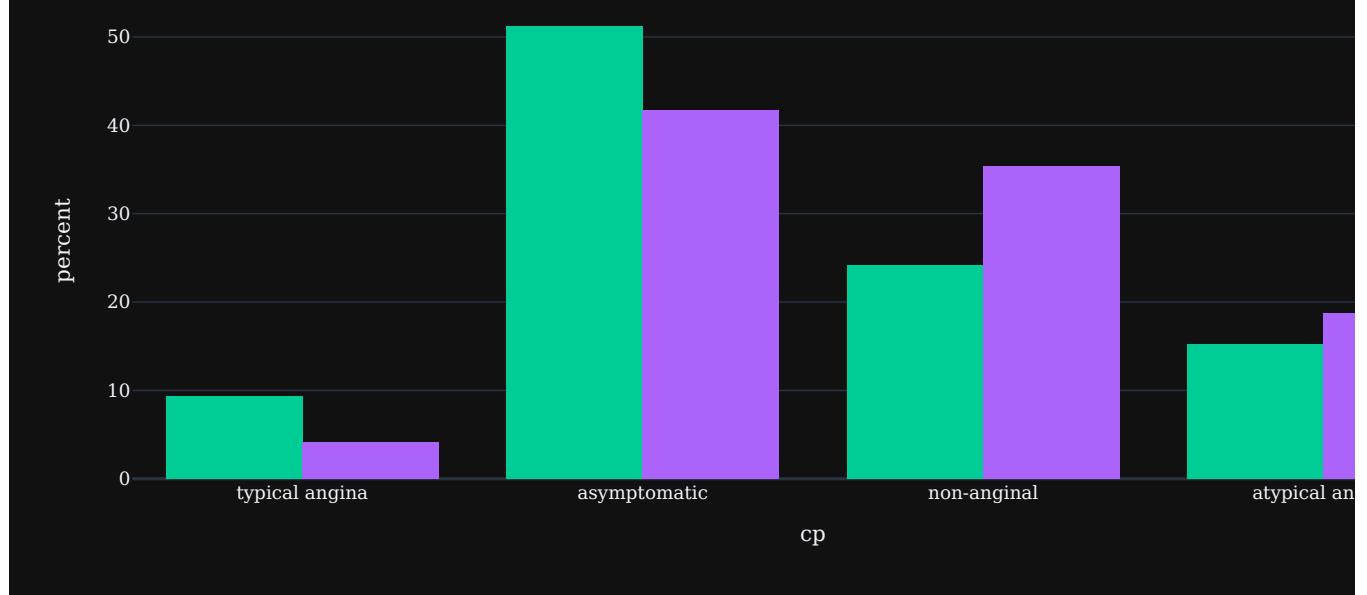
fig = px.histogram(df,
                    x="cp",
                    y=None,
                    color="sex",
                    width=1200,
                    height=450,
                    histnorm='percent',
                    color_discrete_map={
                        "male": "RebeccaPurple", "female": "lightsalmon"
                    },
                    template="plotly_dark"
)

fig.update_layout(title="Gender Chest Pain",
                  font_family="San Serif",
                  bargap=0.2,
                  barmode='group',
                  titlefont={'size': 24},
                  legend=dict(
                      orientation="v", y=1, yanchor="top", x=1.25, xanchor="right"
                  )
)
fig.show()

```



Gender Chest Pain

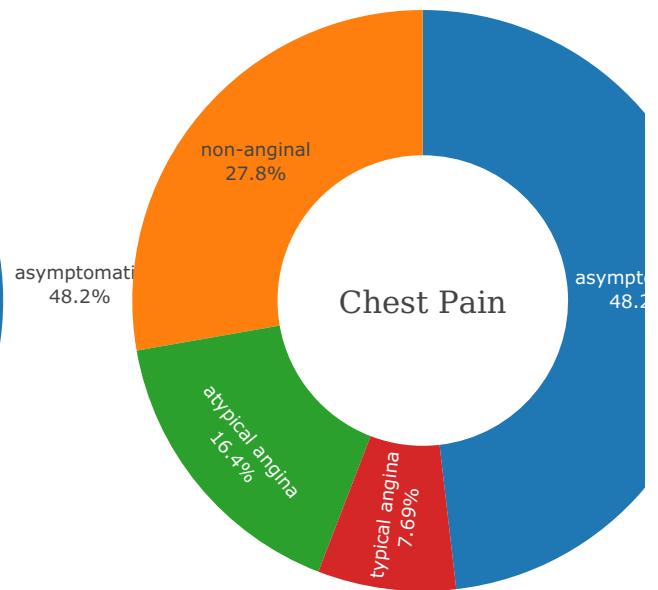
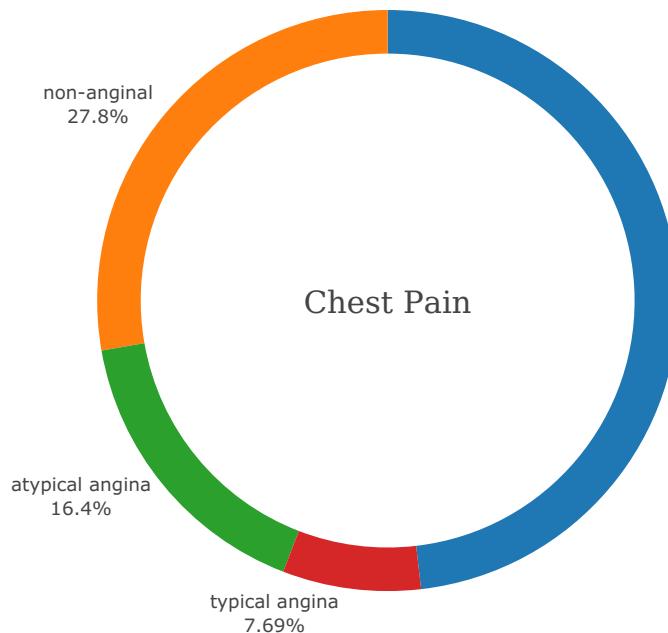


```
In [46]: from plotly.subplots import make_subplots
```

```
# data students performance
fig = make_subplots(rows=1, cols=2,
                     specs=[[{'type':'domain'}, {'type':'domain'}],
                            ])
fig.add_trace(
    go.Pie(
        labels=df['cp'],
        title="Chest Pain",
        titlefont={'size':20, 'family': 'Serif'},
        values=None,
        hole=0.85,
    ), col=1, row=1,
)
fig.update_traces(
    hoverinfo='label+value',
    textinfo='label+percent',
    textfont_size=12,
)

fig.add_trace(
    go.Pie(
        labels=df['cp'],
        title="Chest Pain",
        titlefont={'size':20, 'family': 'Serif'},
        values=None,
        hole=0.5,
    ), col=2, row=1,
)
fig.update_traces(
    hoverinfo='label+value',
    textinfo='label+percent',
    textfont_size=12,
)
fig.layout.update(title=" Heart Disease <b>",
                  titlefont={'size':20, 'family': 'Serif'},
                  showlegend=False,
                  height=600,
                  width=1000,
                  template=None,
)
fig.show()
```

Heart Disease



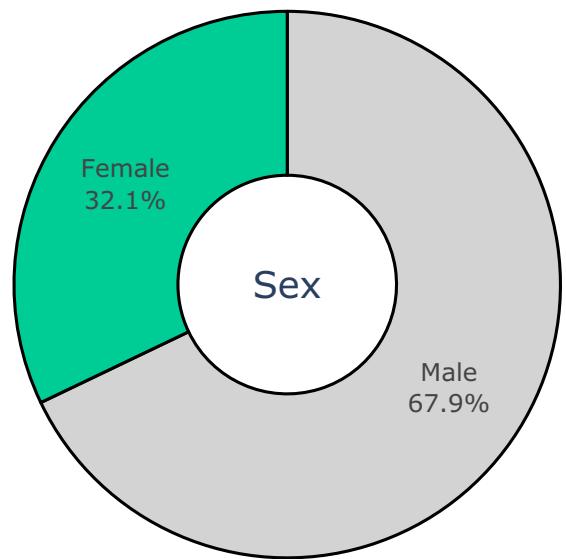
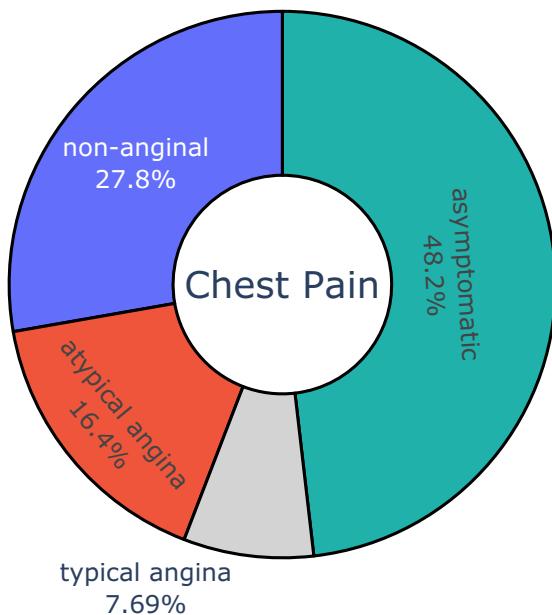
```
In [47]: from plotly.subplots import make_subplots
```

```
# data_titanic
fig = make_subplots(rows=1, cols=2,
                     specs=[[{'type':'domain'}, {'type':'domain'}],
                            ])
fig.add_trace(
    go.Pie(
        labels=df['cp'],
        values=None,
        hole=.4,
        title='Chest Pain',
        titlefont={'color':None, 'size': 24},
        ),
    row=1, col=1
)
fig.update_traces(
    hoverinfo='label+value',
    textinfo='label+percent',
    textfont_size=12,
    marker=dict(
        colors=['lightgray', 'lightseagreen'],
        line=dict(color='#000000',
                  width=2)
    )
)

fig.add_trace(
    go.Pie(
        labels=df['sex'],
        values=None,
        hole=.4,
        title='Sex',
        titlefont={'color':None, 'size': 24},
        ),
    row=1, col=2
)
fig.update_traces(
    hoverinfo='label+value',
    textinfo='label+percent',
    textfont_size=16,
    marker=dict(
        colors=['lightgray', 'lightseagreen'],
        line=dict(color='#000000',
                  width=2)
    )
)
fig.layout.update(title=" Heart Disease ",
```

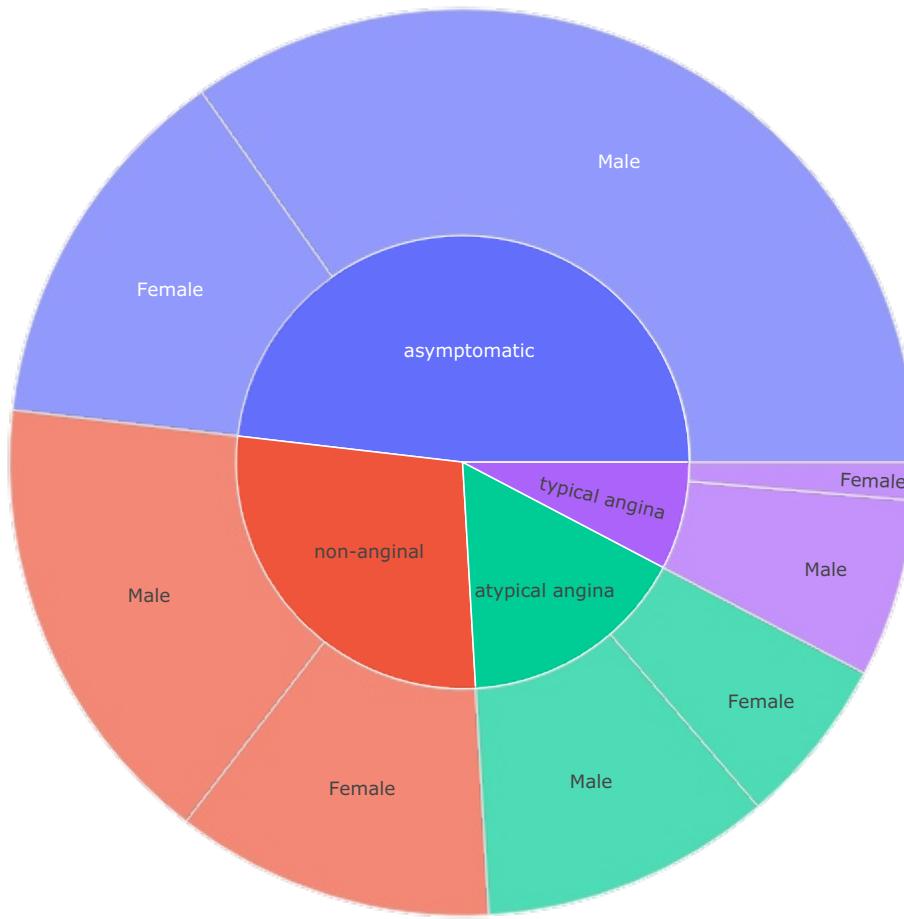
```
titlefont={'color':None, 'size': 24, 'family': 'San-Serif'},
showLegend=False,
height=600,
width=950,
)
fig.show()
```

Heart Desies



```
In [48]: # data students performance
fig = px.sunburst(df,
                  path=['cp', 'sex'])
fig.update_layout(title_text="Chest Pain vs Gender",
                  titlefont={'size': 24, 'family':'Serif'},
                  width=750,
                  height=750,
)
fig.show()
```

Chest Pain vs Gender



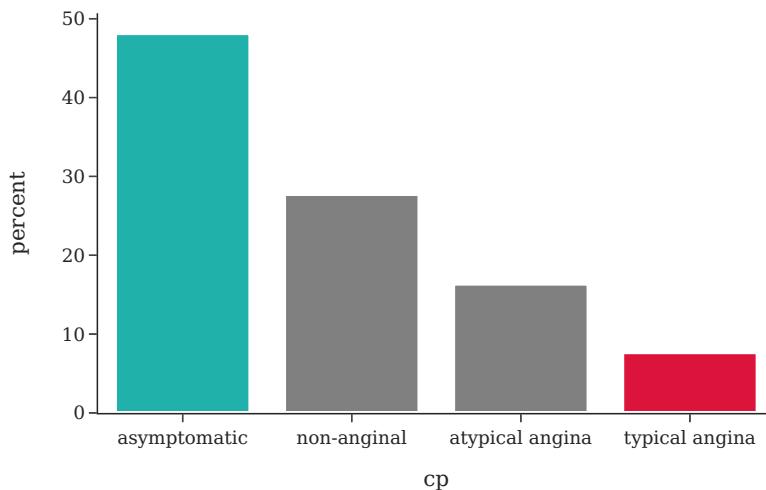
```
In [49]: fig = px.histogram(df, x="cp",
                         width=600,
                         height=400,
                         histnorm='percent',
                         category_orders={
                             "cp": ["asymptomatic", "non-anginal", "atypical angina", "typical angina"],
                             "sex": ["Male", "Female"]
                         },
                         color_discrete_map={
                             "Male": "RebeccaPurple", "Female": "lightsalmon",
                         },
                         template="simple_white"
                     )

fig.update_layout(title="Chest Pain Type",
                  font_family="San Serif",
                  titlefont={'size': 20},
                  legend=dict(
                      orientation="v", y=1, yanchor="top", x=1.0, xanchor="right" )
                  ).update_xaxes(categoryorder='total descending')

# custom color
colors = ['gray',] * 4
colors[3] = 'crimson'
colors[0] = 'lightseagreen'

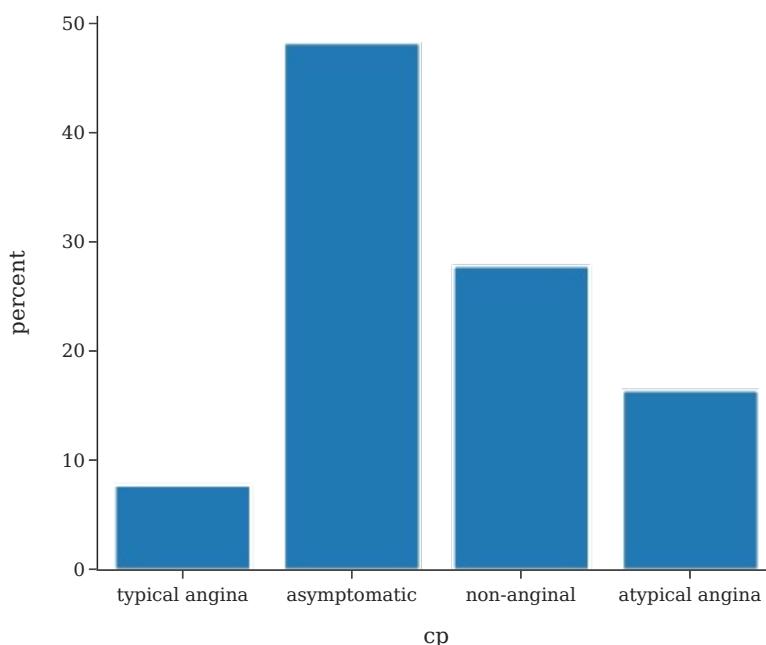
fig.update_traces(marker_color=colors, marker_line_color=None,
                  marker_line_width=2.5, opacity=None)
fig.show()
```

Chest Pain Type



```
In [50]: fig = px.histogram(df, x="cp",
                       width=600,
                       height=500,
                       histnorm='percent',
                       template="simple_white",
                      )
fig.update_layout(title="Types of Chest Pain",
                  font_family="San Serif",
                  titlefont={'size': 20},
                  showlegend=True,
                  legend=dict(
                      orientation="v",
                      y=1.0,
                      yanchor="top",
                      x=1.0,
                      xanchor="right"
                     )
)
fig.update_traces(marker_color=None, marker_line_color='white',
                   marker_line_width=1.5, opacity=0.99)
fig.show()
```

Types of Chest Pain



```
In [51]: colors = ['rgba(38, 24, 74, 0.8)', 'rgba(71, 58, 131, 0.8)',
               'rgba(122, 120, 168, 0.8)', 'rgba(164, 163, 204, 0.85)',
               'rgba(190, 192, 213, 1)']

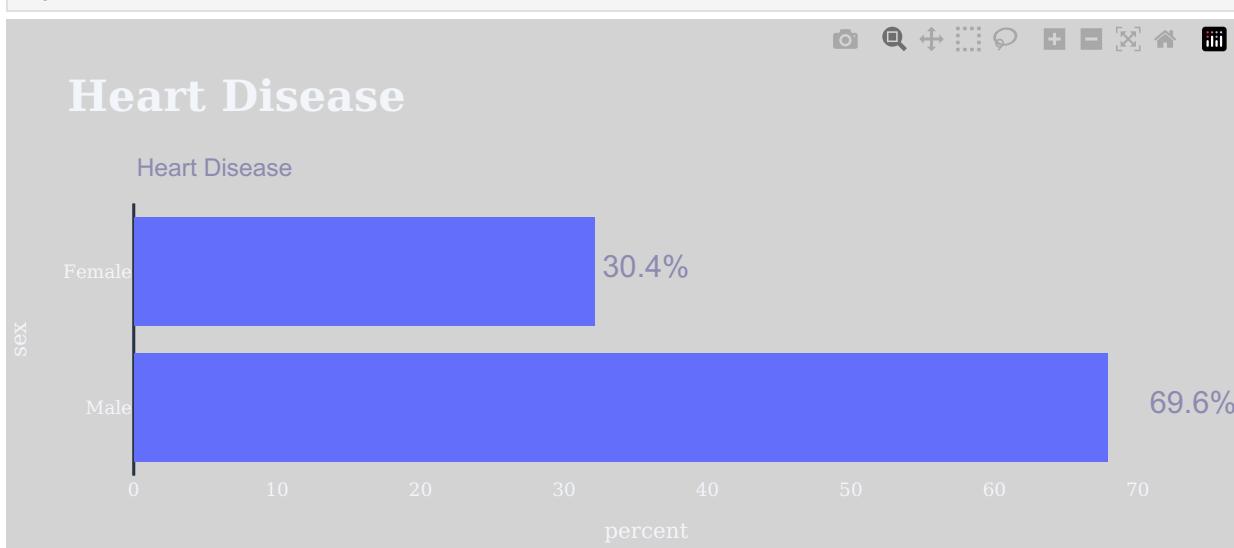
data = df[['sex']]
fig = px.histogram(df,
                    y="sex",
```

```

        orientation='h',
        width=800,
        height=350,
        histnorm='percent',
        template="plotly_dark"
    )
fig.update_layout(title="Heart Disease",
    font_family="San Serif",
    bargap=0.2,
    barmode='group',
    titlefont={'size': 28},
    paper_bgcolor='lightgray',
    plot_bgcolor='lightgray',
    legend=dict(
        orientation="v",
        y=1,
        yanchor="top",
        x=1.250,
        xanchor="right",
    )
)
annotations = []
annotations.append(dict(xref='paper', yref='paper',
    x=0.0, y=1.2,
    text='Heart Disease',
    font=dict(family='Arial', size=16, color=colors[2]),
    showarrow=False))
annotations.append(dict(xref='paper', yref='paper',
    x=0.50, y=0.85,
    text='30.4%',
    font=dict(family='Arial', size=20, color=colors[2]),
    showarrow=False))
annotations.append(dict(xref='paper', yref='paper',
    x=1.08, y=0.19,
    text='69.6%',
    font=dict(family='Arial', size=20, color=colors[2]),
    showarrow=False))

fig.update_layout(
    autosize=False,
    width=800,
    height=350,
    margin=dict(
        l=50,
        r=50,
        b=50,
        t=120,
    ),
)
fig.update_layout(annotations=annotations)
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()

```



In [52]: # Plotting the pie chart
plt.figure(figsize=(20, 5))

```

# Pie chart
plt.subplot(1, 2, 1)
quality_counts = df['cp'].value_counts()
plt.pie(quality_counts, labels=quality_counts.index, colors=sns.color_palette('PuBuGn', len(quality_counts)), autopct='%.1f%%')
plt.title('Chest Pain Distribution')

```

```

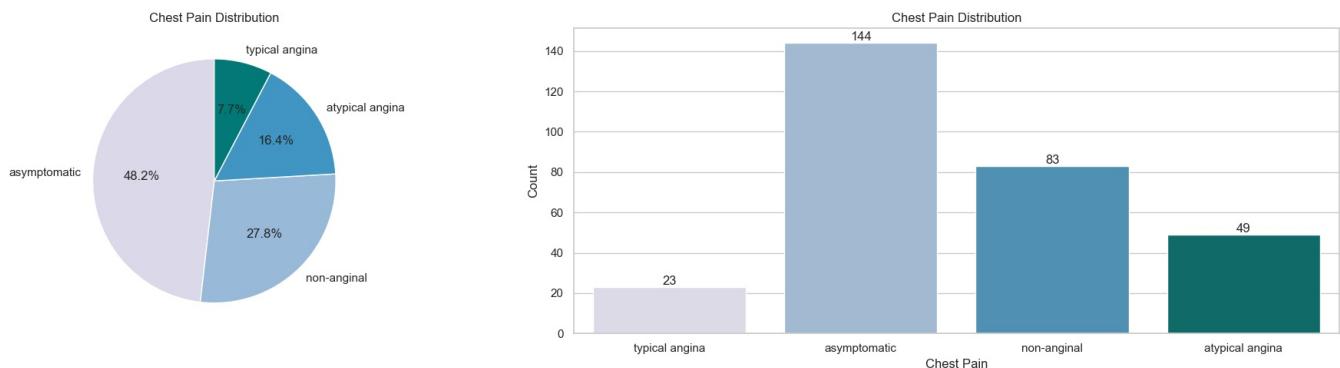
# Count plot
plt.subplot(1, 2, 2)
ax = sns.countplot(data=df, x='cp', palette='PuBuGn')

```



```
# Add count values above each bar
for i in range(len(ax.containers)):
    ax.bar_label(ax.containers[i], label_type='edge')

plt.title('Chest Pain Distribution')
plt.xlabel('Chest Pain')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

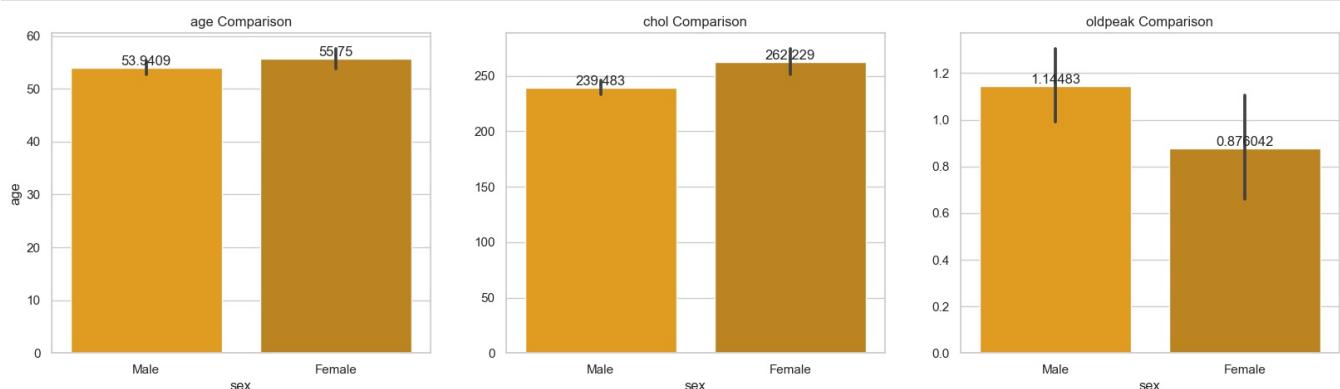


```
In [53]: plt.figure(figsize=(20, 5))

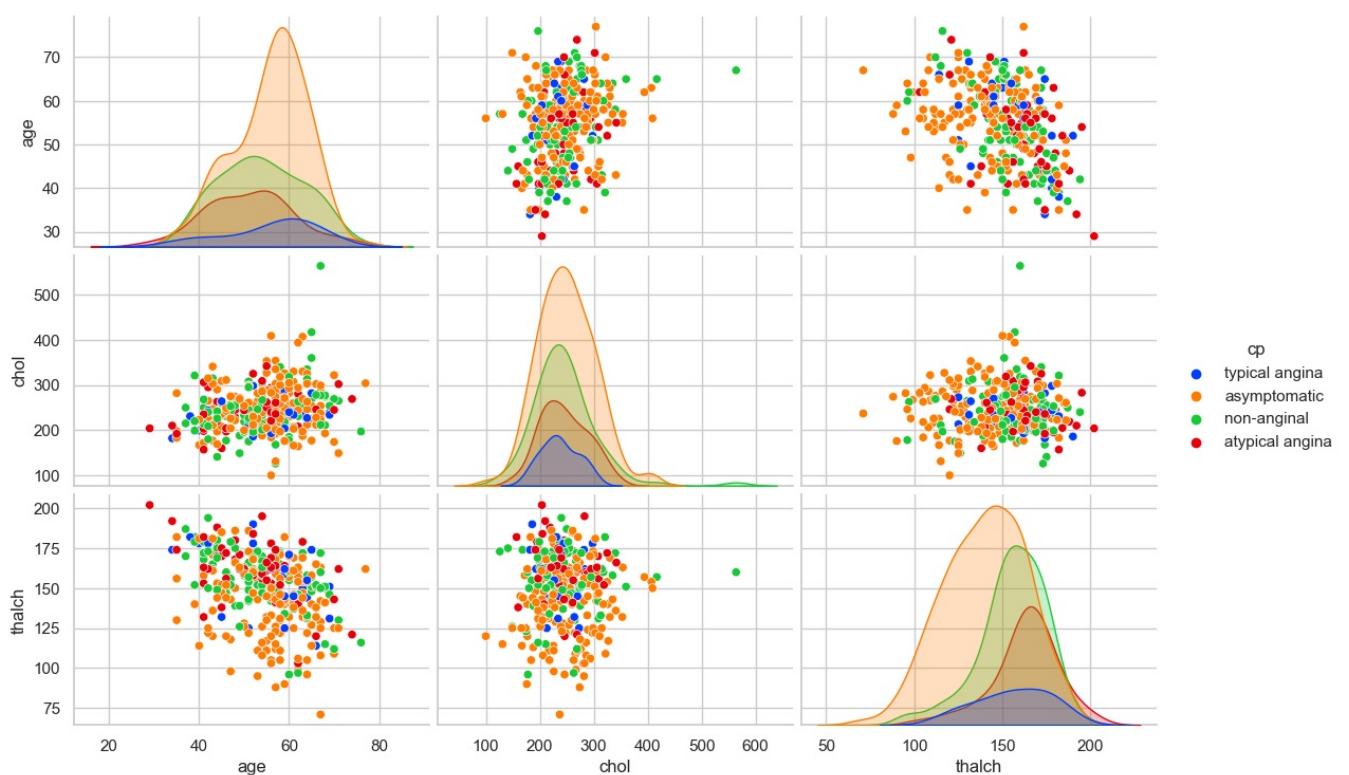
for i, col in enumerate(['age', 'chol', 'oldpeak'], 1):
    plt.subplot(1, 3, i)
    ax = sns.barplot(x='sex', y=col, data=df)
    plt.title(f'{col} Comparison')
    plt.ylabel(col if i == 1 else '')

    # Add count values above each bar
    for i in range(len(ax.containers)):
        ax.bar_label(ax.containers[i], label_type='edge')

plt.show()
```



```
In [54]: sns.pairplot(df[['cp', 'age', 'chol', 'thalch']], hue='cp', aspect=1.5, dropna=True, palette='bright')
plt.show()
```



```
In [55]: # Group by quality and calculate the mean for each quality
grouped_mean = df[['cp','age','trestbps','chol','thalch']].groupby('cp').mean().round(2)

plt.figure(figsize=(20, 6))

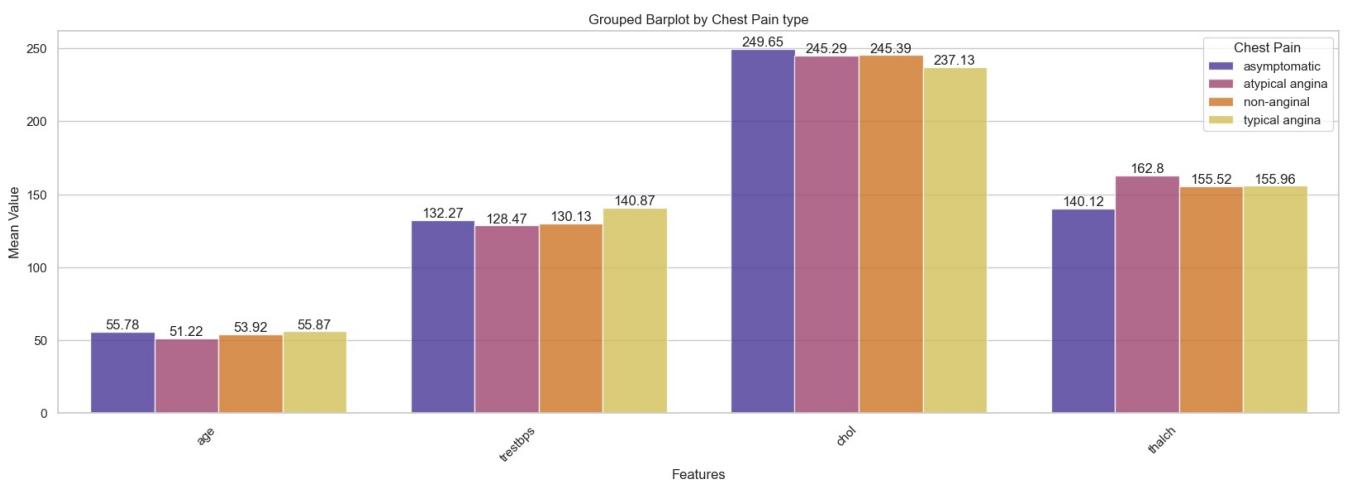
# Plot the grouped bars using Seaborn's barplot
ax = sns.barplot(data=grouped_mean.reset_index().melt(id_vars='cp'),
                  x='variable', y='value', hue='cp', palette='CMRmap', alpha=0.8)

# Add count values above each bar
for i in range(len(ax.containers)):
    ax.bar_label(ax.containers[i], label_type='edge')

plt.xlabel('Features')
plt.ylabel('Mean Value')
plt.title('Grouped Barplot by Chest Pain type')

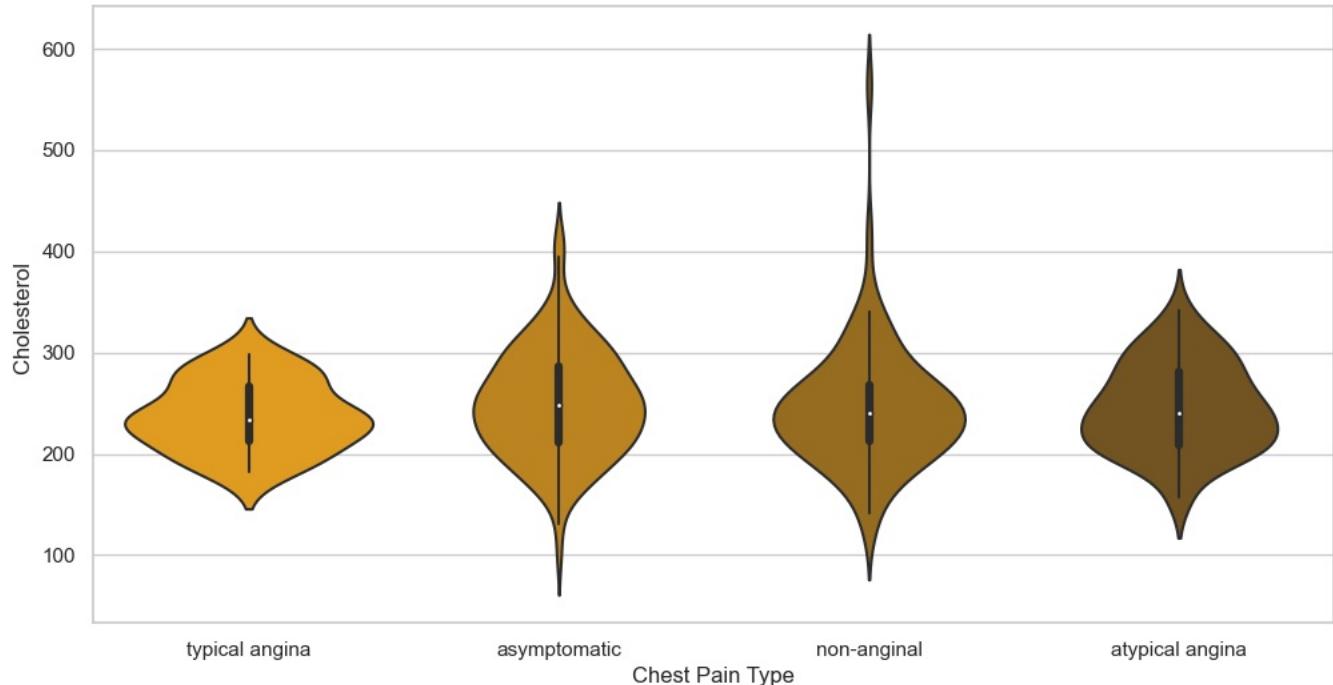
# Rotate x-axis labels
plt.xticks(rotation=45, ha='right')

plt.legend(title='Chest Pain')
plt.show()
```



```
In [56]: # Visualization 8: Violin Plot - Skill Moves Distribution
plt.figure(figsize=(12, 6))
sns.violinplot(x='cp', y='chol', data=df)
plt.title('Distribution of Chest Pain Type with Cholesterol ')
plt.xlabel('Chest Pain Type')
plt.ylabel('Cholesterol ')
plt.show()
```

Distribution of Chest Pain with Cholesterol



```
In [57]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Assuming 'df' is your DataFrame
cp_attributes_comparison = df.loc[df['cp'].isin(['asymptomatic', 'non-anginal', 'atypical angina', 'typical angina'])]
attributes_to_compare = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak', 'ca']

fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(polar=True))

for cp in cp_attributes_comparison['cp'].unique():
    cp_data = cp_attributes_comparison.loc[cp_attributes_comparison['cp'] == cp]

    # Calculate mean values for each attribute
    values = cp_data[attributes_to_compare].mean().values.flatten().tolist()
    values += values[:1] # Close the circle for radar plot

    angles = [n / float(len(attributes_to_compare)) * 2 * np.pi for n in range(len(attributes_to_compare))]
    angles += angles[:1]

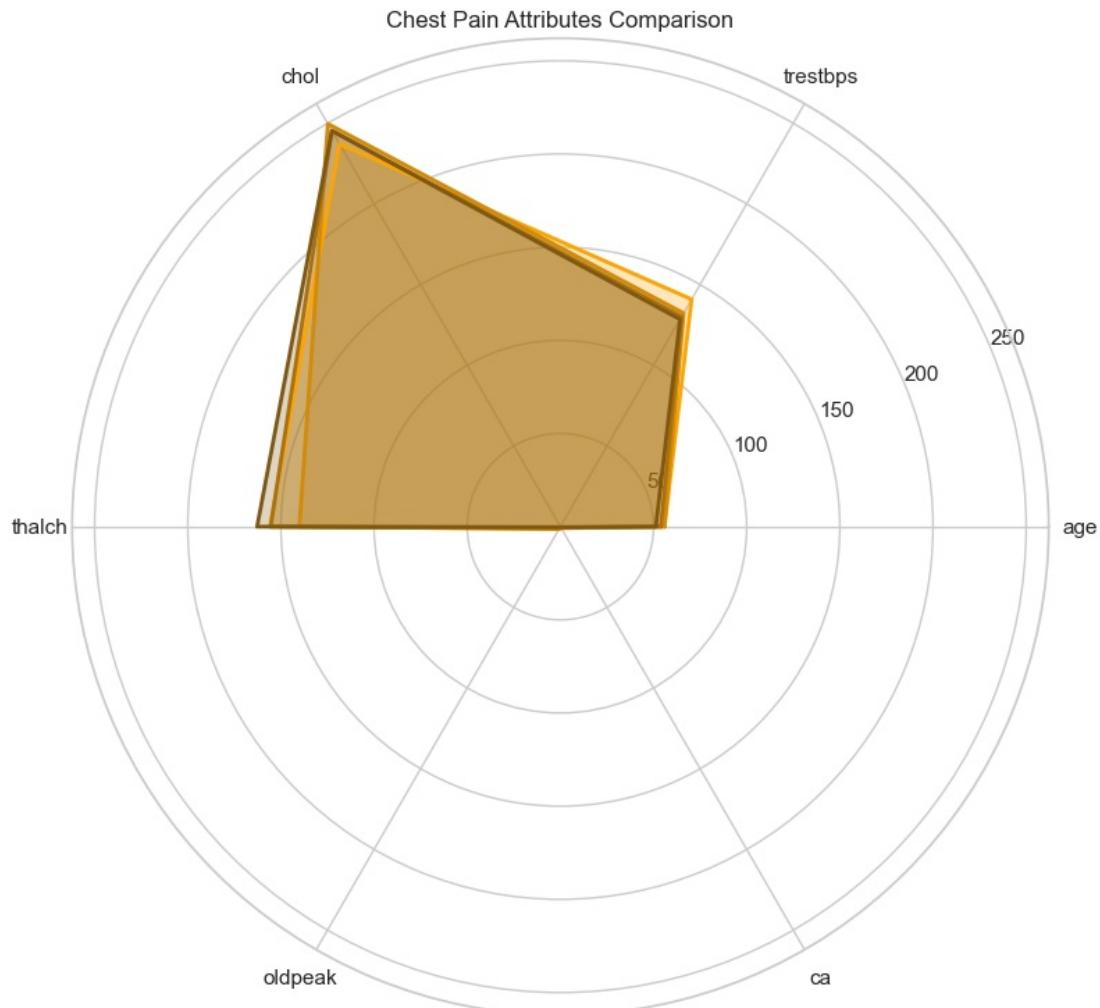
    ax.plot(angles, values, linewidth=2, linestyle='solid', label=cp)
    ax.fill(angles, values, alpha=0.25)

# Set the labels
ax.set_xticks(angles[:-1])
ax.set_xticklabels(attributes_to_compare)

# Add legend
plt.legend(loc='upper right', bbox_to_anchor=(1.3, 1))

# Add title
plt.title('Chest Pain Attributes Comparison')

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



```
In [58]: jobs = pd.read_csv("jobstreet_all_job_dataset.csv")
jobs = jobs.sample(5000)
jobs = jobs.drop(columns=['job_id'], axis=1)
jobs = jobs.reset_index()
jobs = jobs.drop(columns=['index'], axis=1)
display(jobs.shape)
jobs.head(2)
```

(5000, 10)

	job_title	company	descriptions	location	category	subcategory	role	type	salary	listi
0	MARKETING EXECUTIVE	AESD INTERNATIONAL (M) SDN. BHD.	JOB DESCRIPTIONS\nWork closely with the sales ...	Petaling	Marketing & Communications	Marketing Assistants/Coordinators	marketing-executive	Full time	RM 3,000 RM 4,000 per month	21T08
1	E-Commerce Sales Admin	JOBSGURU SDN. BHD.	Job Description\nPerform CS activities by repl...	Petaling	Administration & Office Support	Client & Sales Administration	sales-administration	Full time	RM 2,500 RM 3,500 per month	24T12

```
In [59]: import missingno as msno

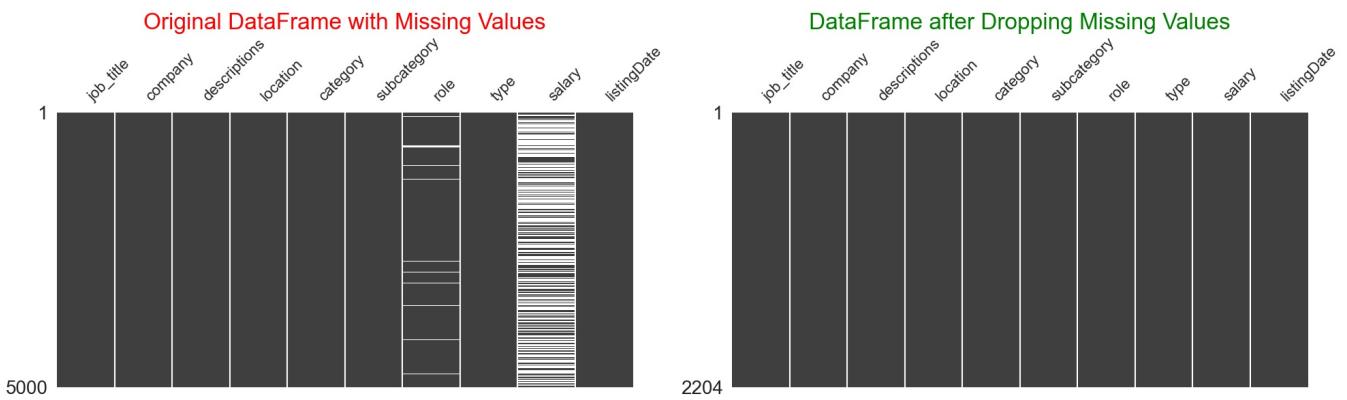
# Create a figure with two subplots arranged in a 1x2 grid
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

# Plot the original DataFrame with missing values
msno.matrix(jobs, ax=axes[0])
axes[0].set_title("Original DataFrame with Missing Values", fontsize=24, color='Red')

# Drop rows with missing values and plot the resulting DataFrame
job = jobs.dropna()

msno.matrix(job, ax=axes[1])
axes[1].set_title("DataFrame after Dropping Missing Values", fontsize=24, color='Green')

plt.tight_layout()
plt.show()
```



```
In [60]: import re

def clean_and_calculate_mean(salary):
    try:
        # Remove currency symbols, words, and extra characters
        salary = salary.replace('RM', '').replace('MYR', '').replace('$', '').replace('per month', '').replace(
            # Handle ranges with different separators
            if '-' in salary:
                salary_range = salary.split('-')
            elif ' - ' in salary:
                salary_range = salary.split(' - ')
            elif ' -' in salary:
                salary_range = salary.split(' -')
            else:
                salary_range = [salary]

        # Convert values to integers, handling potential errors
        salary_values = []
        for value in salary_range:
            try:
                value = int(float(value.replace(',', '').strip()))
                salary_values.append(value)
            except ValueError:
                pass # Ignore non-numeric values

        # Calculate mean if at least two valid values are found
        if len(salary_values) >= 2:
            salary_mean = sum(salary_values) / len(salary_values)
            return salary_mean
        else:
            return None

    except Exception as e:
        print(f'Error processing salary '{salary}': {e}')
        return None

    # Apply the function to the salary column
    job['Salary'] = job['salary'].apply(clean_and_calculate_mean)
    job = job.drop('salary', axis=1)
    job.head(2)
```

```
Out[60]:   job_title      company      descriptions      location      category      subcategory      role      type      listingDate      Sal
0  MARKETING EXECUTIVE  AESD INTERNATIONAL (M) SDN. BHD.  JOB DESCRIPTIONS\nWork closely with the sales ...  Petaling  Marketing & Communications  Marketing Assistants/Coordinators  marketing-executive  Full time  2024-03-21T08:08:18Z  35
1  E-Commerce Sales Admin  JOBSGURU SDN. BHD.  Job Description\nPerform CS activities by repl...  Petaling  Administration & Office Support  Client & Sales Administration  sales-administration  Full time  2024-05-24T12:59:40Z  30
```

```
In [61]: import plotly.express as px
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)

jobType = job['type'].value_counts()

fig = px.pie(values=jobType.values, names=jobType.index.tolist(), color=jobType.index.tolist(), color_discrete_s
fig.update_layout(width=1000, height=800)

fig.update_traces(textfont_size=20)

fig.update_traces(pull=[0.1, 0.1, 0.2], textposition='outside')

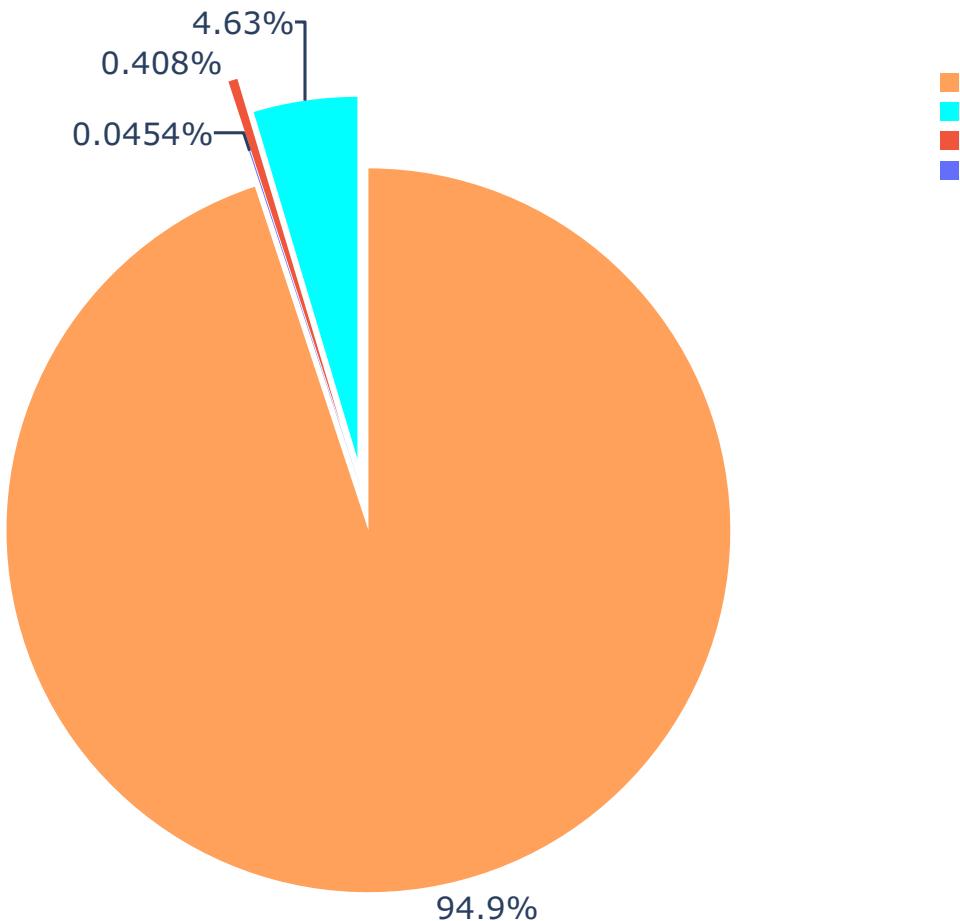
# Set layout properties
fig.update_layout(margin = dict(t=50, l=10, r=10, b=25),
                  title='Employment Types in the Job Market of Malaysia',
                  title_x=0.5,
```



```
title_y=0.98)
```

```
fig.show()
```

Employment Types in the Job Market of Malaysia



```
In [79]: top_n = 50
```

```
filtered_data = job['job_title'].value_counts().head(top_n).reset_index()
filtered_data.columns = ['job_title', 'count']

fig = px.treemap(filtered_data, path=[px.Constant('all'), 'job_title'], values='count')
fig.update_traces(root_color='lightgrey')

fig.update_traces(textfont_size=16)

fig.update_layout(width=1000, height=600)
fig.update_layout(margin=dict(t=50, l=25, r=25, b=25),
                  title='Top Job Openings: Job Roles in Malaysia',
                  title_x=0.5,
                  title_y=0.98)

fig.show()
```

Top Job Openings: Job Roles in Malaysia



```
In [63]: job.replace('Kuala Lumpur Sentral', 'Kuala Lumpur', inplace=True)
job.replace('Bangsar South', 'Bangsar', inplace=True)
job.replace('Klang District', 'Klang/Port Klang', inplace=True)
job.replace('Penang Island', 'Penang', inplace=True)
job['location'] = job['location'].str.replace(' District', '')

top_location = ['Kuala Lumpur', 'Petaling', 'Penang']
filtered_data = job[job['location'].isin(top_location)]

job_title_counts = filtered_data['job_title'].value_counts()

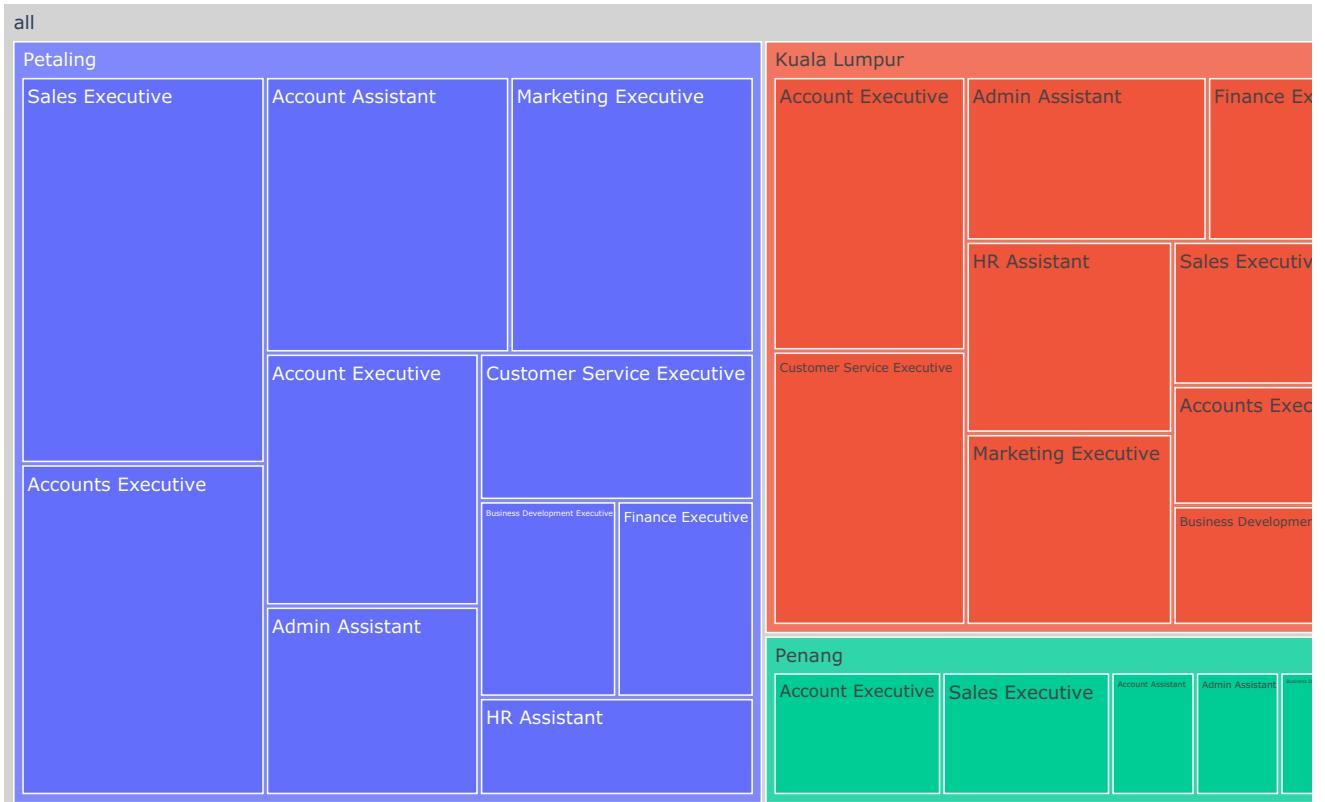
top_n = 10
top_job_titles = job_title_counts.head(top_n).index.tolist()

filtered_data = filtered_data[filtered_data['job_title'].isin(top_job_titles)]

fig = px.treemap(filtered_data, path=[px.Constant('all'), 'location', 'job_title'])
fig.update_traces(root_color='lightgrey')
fig.update_layout(width=1000, height=600)
fig.update_layout(margin=dict(t=50, l=25, r=25, b=25),
                  title='Top Job Opportunities in Kuala Lumpur, Petaling, and Penang',
                  title_x=0.5,
                  title_y=0.98)
fig.show()
```



Top Job Opportunities in Kuala Lumpur, Petaling, and Penang

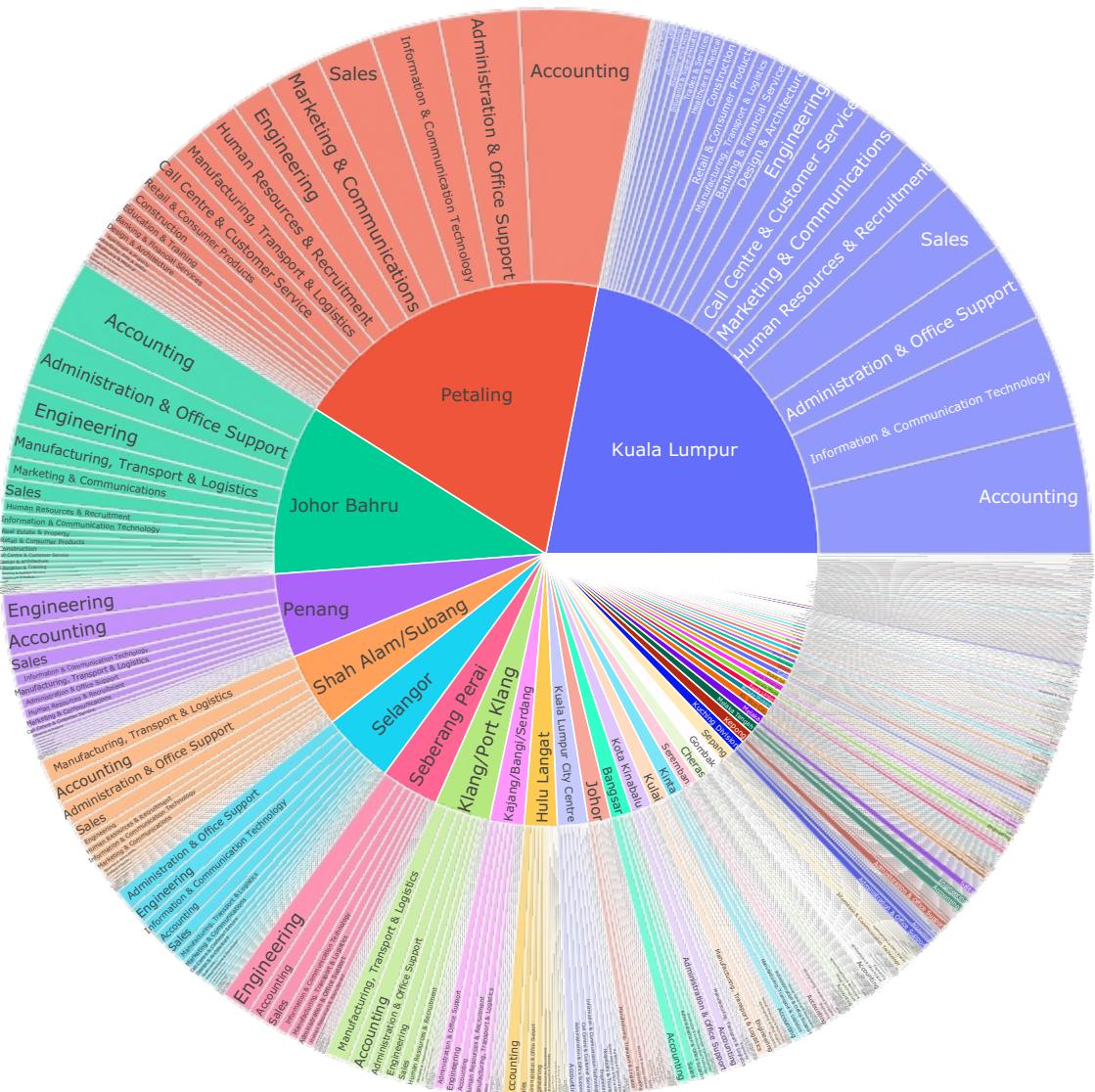


```
In [64]: # plot a sunburst chart
fig = px.sunburst(job, path=['location','category'])

# configurate the plot layout
fig.update_layout(
    margin=dict(t=50, l=25, r=25, b=25),
    width=900, # Set the width of the plot
    height=800, # Set the height of the plot
    title='Job Vacancies Available Across Malaysia by Category',
    title_x=0.48,
    title_y=0.98
)

fig.show()
```

Job Vacancies Available Across Malaysia by Category



```
In [65]: def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.10)
    Q3 = df[column].quantile(0.85)

    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df_outlier_free = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

    return df_outlier_free

jobs = remove_outliers_iqr(job, 'Salary')

#=====#
# Get top 20 job titles from both DataFrames
top_leagues_job = job['job_title'].value_counts().nlargest(20).index
top_leagues_jobs = jobs['job_title'].value_counts().nlargest(20).index

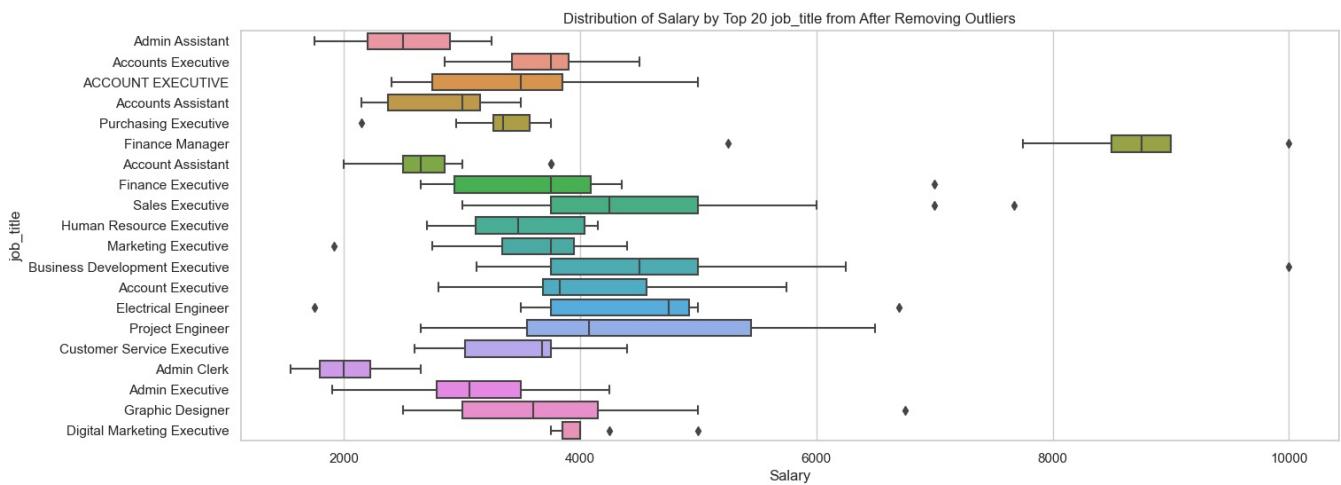
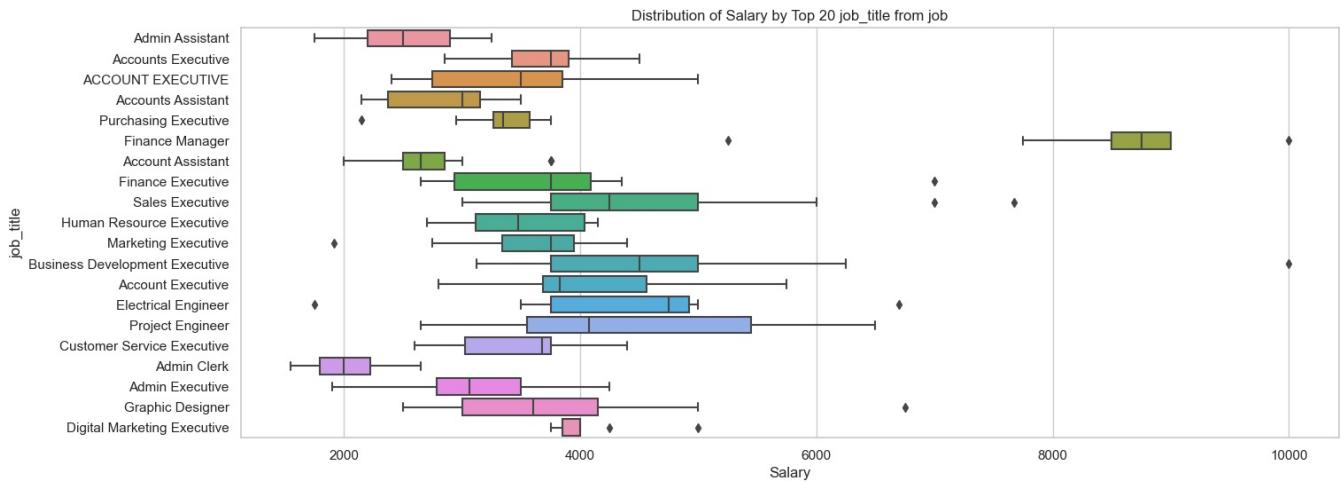
# Combine the top job titles
top_leagues = top_leagues_job.union(top_leagues_jobs)

# Plotting
plt.figure(figsize=(16, 6))
sns.boxplot(x='Salary', y='job_title', data=job[job['job_title'].isin(top_leagues)])
plt.title('Distribution of Salary by Top 20 job_title from job')
plt.xlabel('Salary')
plt.ylabel('job_title')
plt.show()

plt.figure(figsize=(16, 6))
sns.boxplot(x='Salary', y='job_title', data=jobs[jobs['job_title'].isin(top_leagues)])
plt.title('Distribution of Salary by Top 20 job_title from After Removing Outliers')
plt.xlabel('Salary')
```



```
plt.ylabel('job_title')
plt.show()
```



```
In [66]: top_leagues = jobs['job_title'].value_counts().nlargest(15)

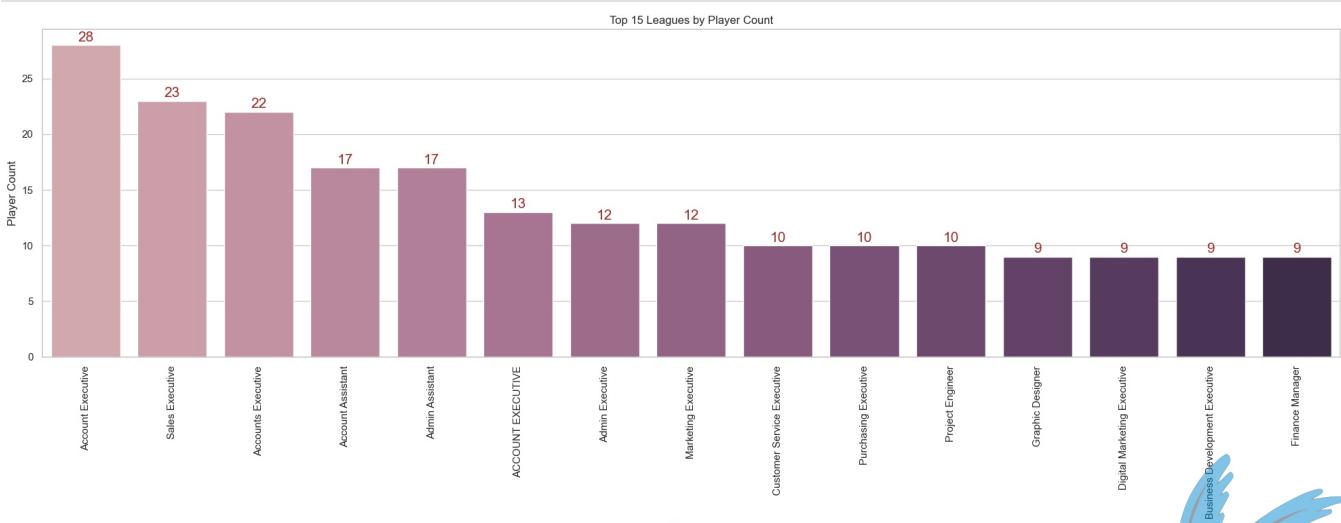
plt.figure(figsize=(20, 8))

colors = sns.cubehelix_palette(len(top_leagues), light=0.7, dark=0.2)
bar_plot = sns.barplot(x=top_leagues.index, y=top_leagues.values, palette=colors)

for index, value in enumerate(top_leagues.values):
    label = f"{value:.1f}"
    plt.text(index, value + 0.1, label, ha='center', va='bottom', fontsize=15, color="#A52A2A")

plt.title('Top 15 Leagues by Player Count')
plt.xlabel('League')
plt.ylabel('Player Count')
plt.xticks(rotation=90)
plt.tight_layout()

plt.show()
```



```
In [67]: tips_df = pd.read_csv("tip.csv")

display(tips_df.head(2))
```

```

fig = px.bar(tips_df,
              x="sex",
              y="total_bill",
              color="smoker",
              barmode="group",
              facet_row="time",
              facet_col="day",
              category_orders={"day": ["Thur", "Fri", "Sat", "Sun"],
                               "time": ["Lunch", "Dinner"]})
fig.show()

```

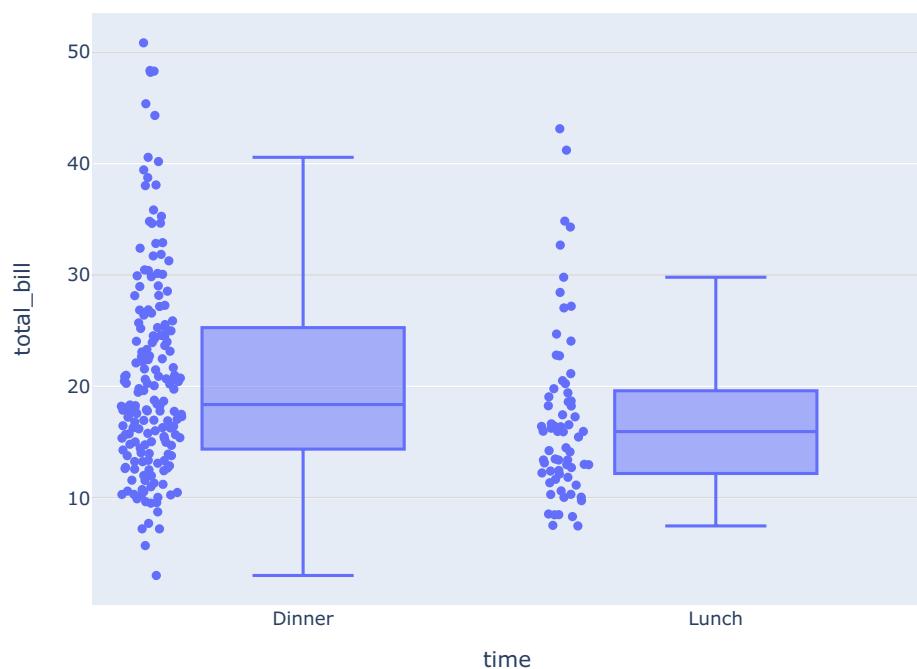
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3



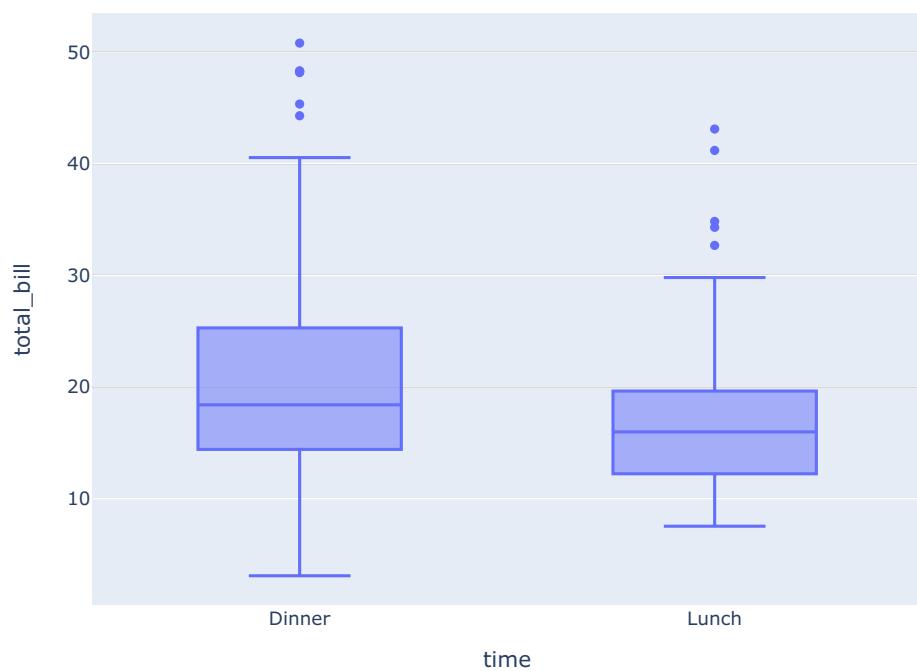
```

In [68]: fig = px.box(tips_df,
                     x="time",
                     y="total_bill",
                     points="all")
fig.show()

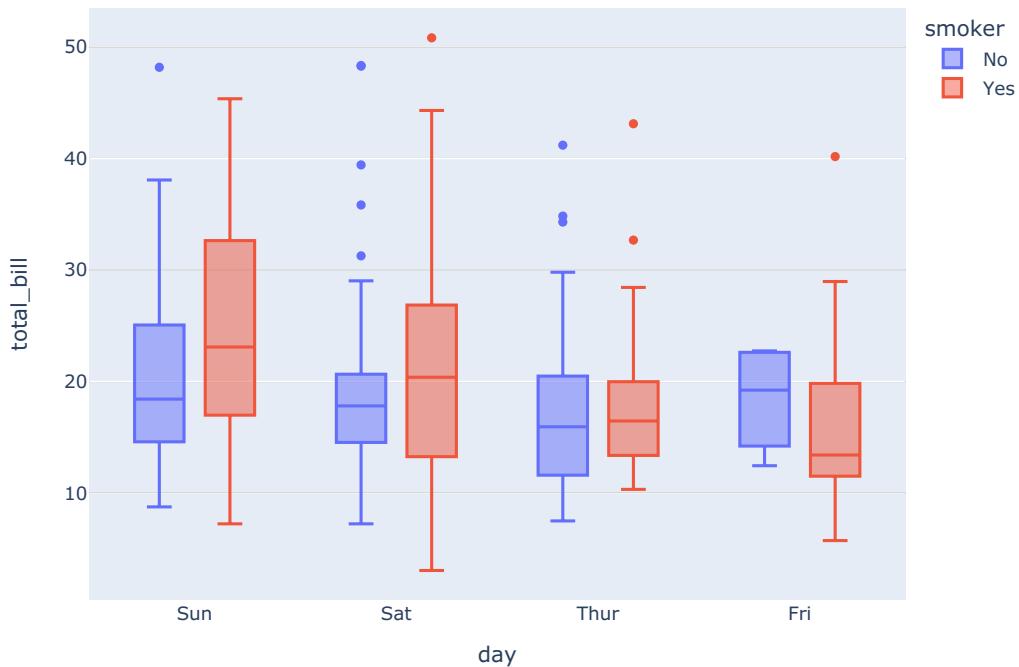
```



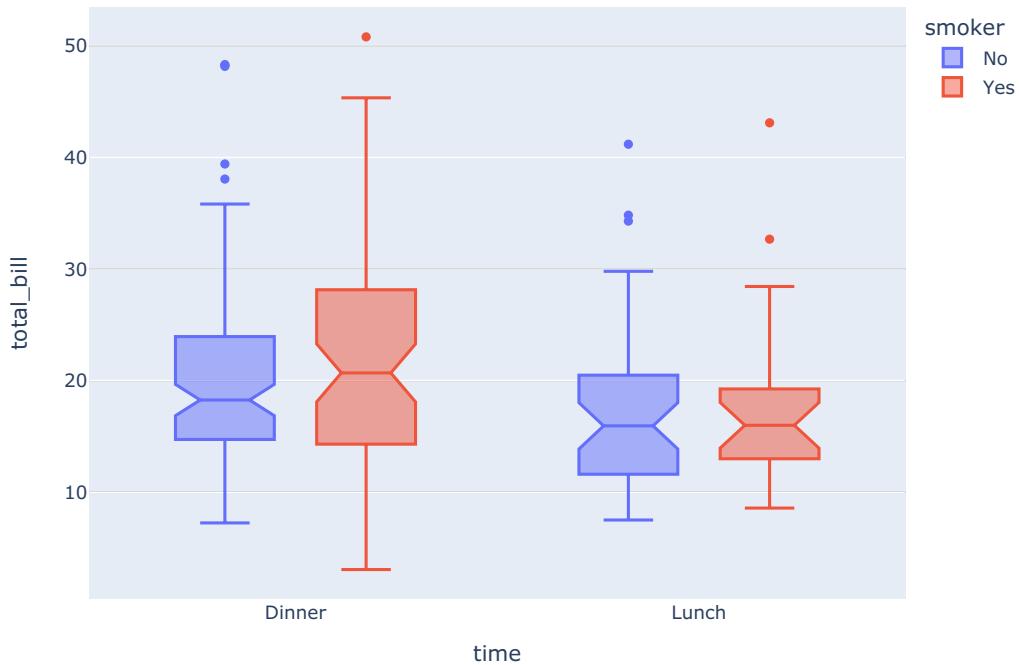
```
In [69]: fig = px.box(tips_df,
                  x="time",
                  y="total_bill",
                  points="outliers")
fig.show()
```



```
In [70]: fig = px.box(tips_df,
                  x="day",
                  y="total_bill",
                  color="smoker" )
fig.update_traces(quartilemethod="linear")
fig.show()
```



```
In [71]: fig = px.box(tips_df,
                  x="time",
                  y="total_bill",
                  color="smoker",
                  notched=True,
                  hover_data=[ "day"] # add day column to hover data
                 )
fig.show()
```



```
In [72]: plt.figure(figsize=(20, 10))

x = jobs['job_title'].head(20)
y = jobs['Salary'].head(20)

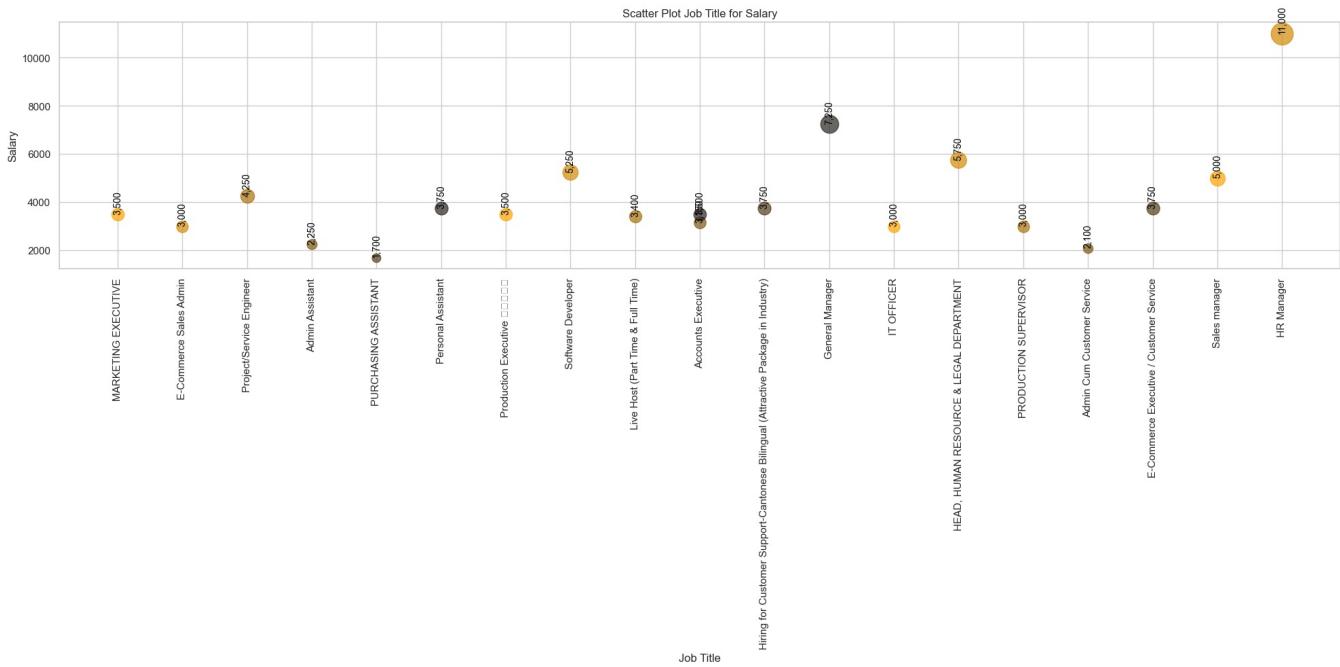
# Plot the scatter plot with country names and numbers on y-axis
marker_sizes = jobs['Salary']
for i, country in enumerate(x):
    plt.scatter(country, y.iloc[i], s=(marker_sizes.iloc[i])/20, label=country, alpha=0.7)
    plt.text(country, y.iloc[i], f'{y.iloc[i]:,.0f}', ha='center', va='bottom', rotation='vertical', fontsize=1)
```



```
# Set y-axis to display numbers in billions
plt.ticklabel_format(style='plain', axis='y', useOffset=False, scilimits=(9, 9))

plt.xlabel('Job Title')
plt.ylabel('Salary')
plt.title('Scatter Plot Job Title for Salary')
plt.xticks(rotation=90)
plt.grid(True)
plt.tight_layout()

plt.show()
```



```
In [73]: jobs = pd.read_csv("jobstreet_all_job_dataset.csv")
jobs = jobs.sample(5000)
jobs = jobs.drop(columns=['job_id'], axis=1)
jobs = jobs.reset_index()
jobs = jobs.drop(columns=['index'], axis=1)
display(jobs.shape)
display(jobs.head(2))
```

```
from wordcloud import WordCloud

text = str(list(jobs['category'])).replace(',', ' ').replace('[', '').replace("'", '').replace(']', '')

wordcloud = WordCloud(background_color = 'white', width = 1600, height = 800, max_words = 121).generate(text)
plt.imshow(wordcloud)

plt.axis('off')
plt.show()
```

(5000, 10)

	job_title	company	descriptions	location	category	subcategory	role	type	salary	listingDate
0	Manager, Government & Authority Liasion	Mass Rapid Transit Corporation Sdn Bhd	JOB PURPOSE \nTo organize & participate in a ...	Kuala Lumpur	Construction	Project Management	manager	Contract/Temp	NaN	2024-05-10T04:06:31Z
1	Junior IT Executive	Saraya Goodmaid Sdn Bhd	Designing solutions, implementation, customiza...	Seremban District	Information & Communication Technology	Developers/Programmers	information- technology- executive	Full time	NaN	2024-04-08T00:14:09Z



```
In [74]: # Drop rows with missing values and plot the resulting DataFrame
job = jobs.dropna()

import re

def clean_and_calculate_mean(salary):
    try:
        # Remove currency symbols, words, and extra characters
        salary = salary.replace('RM', '').replace('MYR', '').replace('$', '').replace('per month', '').replace(
            # Handle ranges with different separators
            if '-' in salary:
                salary_range = salary.split('-')
            elif '-' in salary:
                salary_range = salary.split(' - ')
            elif '-' in salary:
                salary_range = salary.split(' - ')
            else:
                salary_range = [salary]

        # Convert values to integers, handling potential errors
        salary_values = []
        for value in salary_range:
            try:
                value = int(float(value.replace(',', '').strip()))
                salary_values.append(value)
            except ValueError:
                pass # Ignore non-numeric values

        # Calculate mean if at least two valid values are found
        if len(salary_values) >= 2:
            salary_mean = sum(salary_values) / len(salary_values)
            return salary_mean
        else:
            return None

    except Exception as e:
        print(f"Error processing salary '{salary}': {e}")
        return None

# Apply the function to the salary column
job['Salary'] = job['salary'].apply(clean_and_calculate_mean)
job = job.drop('salary', axis=1)
job.head(2)
```

	job_title	company	descriptions	location	category	subcategory	role	type	listingDate	Salary
4	Specialist, Contact Centre	CTOS Data Systems Sdn Bhd	Attend to all inbound and outbound calls/ emails...	Kuala Lumpur	Call Centre & Customer Service	Customer Service - Call Centre	call-centre-role	Full time	2024-04-19T02:57:09Z	2750.0
5	Multilingual Customer Support Specialist	Private Advertiser	Skills and Abilities:\nSkilled communicator.\n...	Kampung Malaysia Raya	Call Centre & Customer Service	Customer Service - Call Centre	customer-support-specialist	Full time	2024-04-05T12:56:47Z	5500.0

```
In [75]: dfp = job['role'].value_counts().head(10).sort_values(ascending = True).reset_index()
dfl = job['location'].value_counts().head(10).sort_values(ascending = True).reset_index()
dfc = job['company'].value_counts().head(10).sort_values(ascending = True).reset_index()

fig = go.Figure()

fig.add_trace(go.Bar(y = dfp['role'],
                      orientation='h',
                      name = 'Position',
                      marker = dict(color = 'LightCoral')))

fig.add_trace(go.Bar(y = dfl['location'],
                      orientation='h',
                      name = 'Location',
                      marker = dict(color = 'CadetBlue')))

fig.add_trace(go.Bar(y = dfc['company'],
                      orientation='h',
                      name = 'Company',
                      marker = dict(color = 'SteelBlue')))

fig.update_layout(
    updatemenus=[dict(
        type = "buttons",
        direction="left",
        pad={"r": 10, "t": 10},
        showactive=True,
        x=0.16,
        xanchor="left",
        y=1.12,
        yanchor="top",
        font = dict(color = 'Indigo',size = 14),
```



```

buttons=list([
    dict(label="All",
        method="update",
        args=[ {"visible": [True, True, True]},
              {'showlegend' : True}
            ]),
    dict(label="Position",
        method="update",
        args=[ {"visible": [True, False, False]},
              {'showlegend' : True}
            ]),
    dict(label='Location',
        method="update",
        args=[ {"visible": [False, True, False]},
              {'showlegend' : True}
            ]),
    dict(label='Company',
        method="update",
        args=[ {"visible": [False, False, True]},
              {'showlegend' : True}
            ]),
]),
])

fig.update_layout(
    annotations=[
        dict(text="Choose:", showarrow=False,
            x=0, y=1.075, yref="paper", align="right",
            font=dict(size=16,color = 'DarkSlateBlue'))])

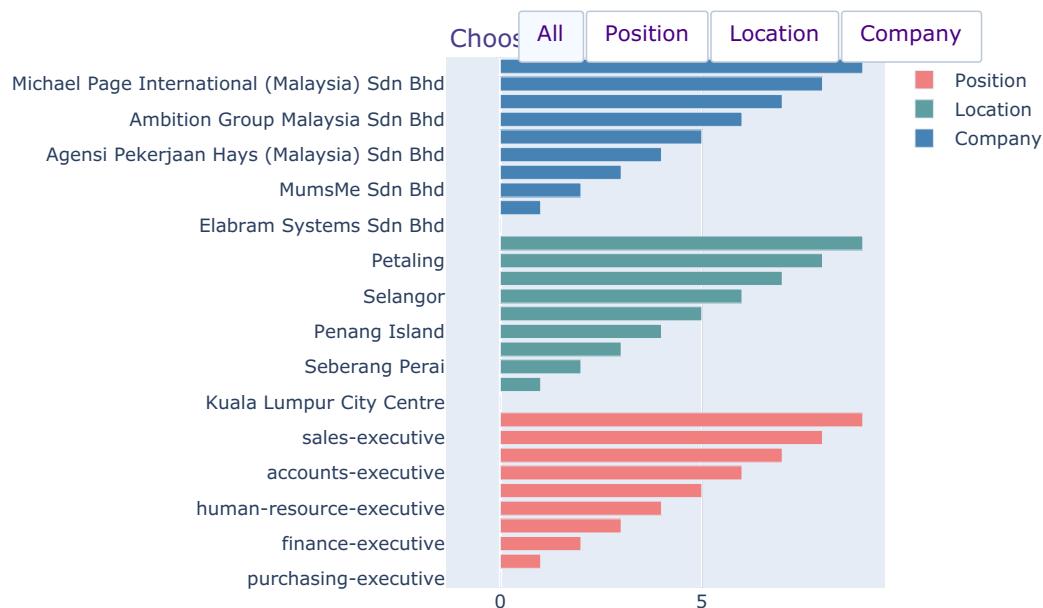
fig.update_layout(title ="Top 10 Positions, Locations and Companies",
                  title_x = 0.5,
                  title_font = dict(size = 20, color = 'MidnightBlue'))

fig.show()

```



Top 10 Positions, Locations and Companies



In [76]: # Read data
df = pd.read_csv('US_Job_Market.csv')
df.head(3)

	position	company	description	reviews	location
0	Development Director	ALS TDI	Development Director\nALS Therapy Development ...	NaN	Atlanta, GA 30301
1	An Ostentatiously-Excitable Principal Research...	The Hexagon Lavish	Job Description\n\n"The road that leads to acc...	NaN	Atlanta, GA
2	Data Scientist	Xpert Staffing	Growing company located in the Atlanta, GA are...	NaN	Atlanta, GA

In [77]: #!pip install dash

In [78]: import pandas as pd
from dash import Dash, dcc, html
from dash.dependencies import Input, Output

```

import plotly.graph_objects as go

dfd1 = df[df['position']=='Data Scientist']
dfd2 = df[df['position']=='Senior Data Scientist']
dfd3 = df[df['position']=='Research Analyst']
dfd4 = df[df['position']=='Data Engineer']

# Add 'position' column to each dataframe
redf1 = dfd1[["location", "position"]].value_counts().nlargest(10).sort_values(ascending = True).reset_index()
redf2 = dfd2[["location", "position"]].value_counts().nlargest(10).sort_values(ascending = True).reset_index()
redf3 = dfd3[["location", "position"]].value_counts().nlargest(10).sort_values(ascending = True).reset_index()
redf4 = dfd4[["location", "position"]].value_counts().nlargest(10).sort_values(ascending = True).reset_index()
# Create Plotly figure
fig = go.Figure()

fig.add_trace(go.Bar(x = redf1["location"],
                     y = redf1["count"],
                     marker = dict(color = 'Tomato'),
                     name = 'Data Scientist'))
fig.add_trace(go.Bar(x = redf2['location'],
                     y = redf2['count'],
                     name = 'Senior Data Scientist',
                     marker = dict(color = 'LightCoral')))
fig.add_trace(go.Bar(x = redf3['location'],
                     y = redf3['count'],
                     name = 'Research Analyst',
                     marker = dict(color = 'SteelBlue')))
fig.add_trace(go.Bar(x = redf4['location'],
                     y = redf4['count'],
                     name = 'Data Engineer',
                     marker = dict(color = 'CadetBlue')))

# Update Layout with dropdown functionality
fig.update_layout(
    updatemenus=[

        dict(
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.13,
            xanchor="left",
            y=1.12,
            yanchor="top",
            font = dict(color = 'Indigo',size = 14),
            buttons=list([
                dict(label="All",
                     method="update",
                     args=[ {"visible": [True, True, True, True]},
                           {'showlegend' : True}
                     ]),
                dict(label="Data Scientist",
                     method="update",
                     args=[ {"visible": [True, False, False, False]},
                           {'showlegend' : True}
                     ]),
                dict(label='Senior Data Scientist',
                     method="update",
                     args=[ {"visible": [False, True, False, False]},
                           {'showlegend' : True}
                     ]),
                dict(label='Research Analyst',
                     method="update",
                     args=[ {"visible": [False, False, True, False]},
                           {'showlegend' : True}
                     ]),
                dict(label='Data Engineer',
                     method="update",
                     args=[ {"visible": [False, False, False, True]},
                           {'showlegend' : True}
                     ]),
            ])
        )
    ])

fig.update_layout(
    annotations=[

        dict(text="Choose:", showarrow=False,
             x=0, y=1.075, yref="paper", align="right",
             font=dict(size=16,color = 'DarkSlateBlue'))]
)

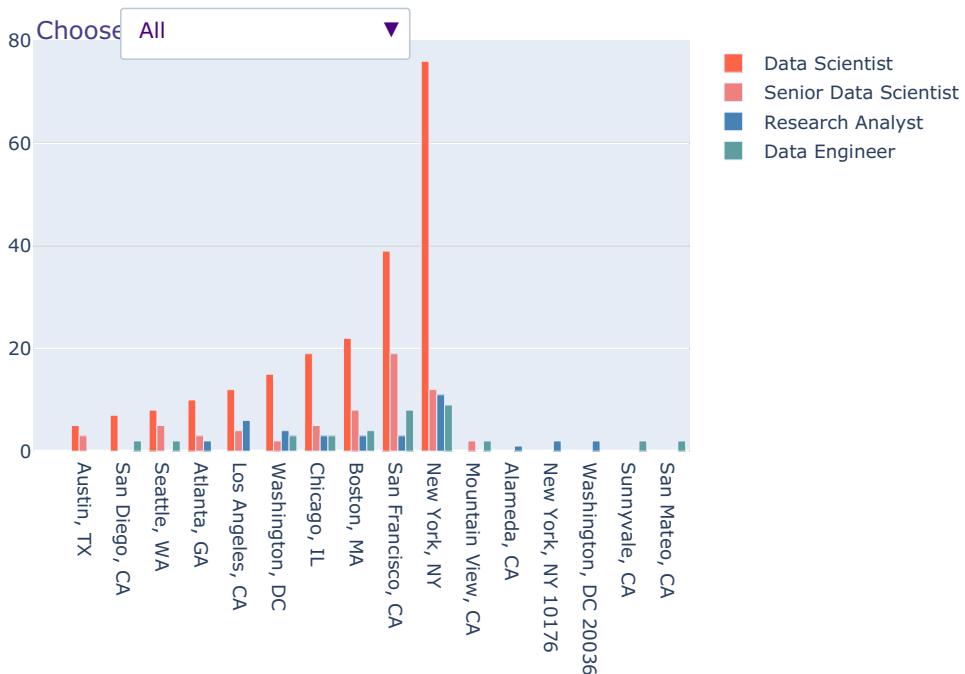
fig.update_layout(title = "The distribution of states by four Positions",
                  title_x = 0.5,
                  title_font = dict(size = 20, color = 'MidnightBlue'))

fig.show()

```



The distribution of states by four Positions

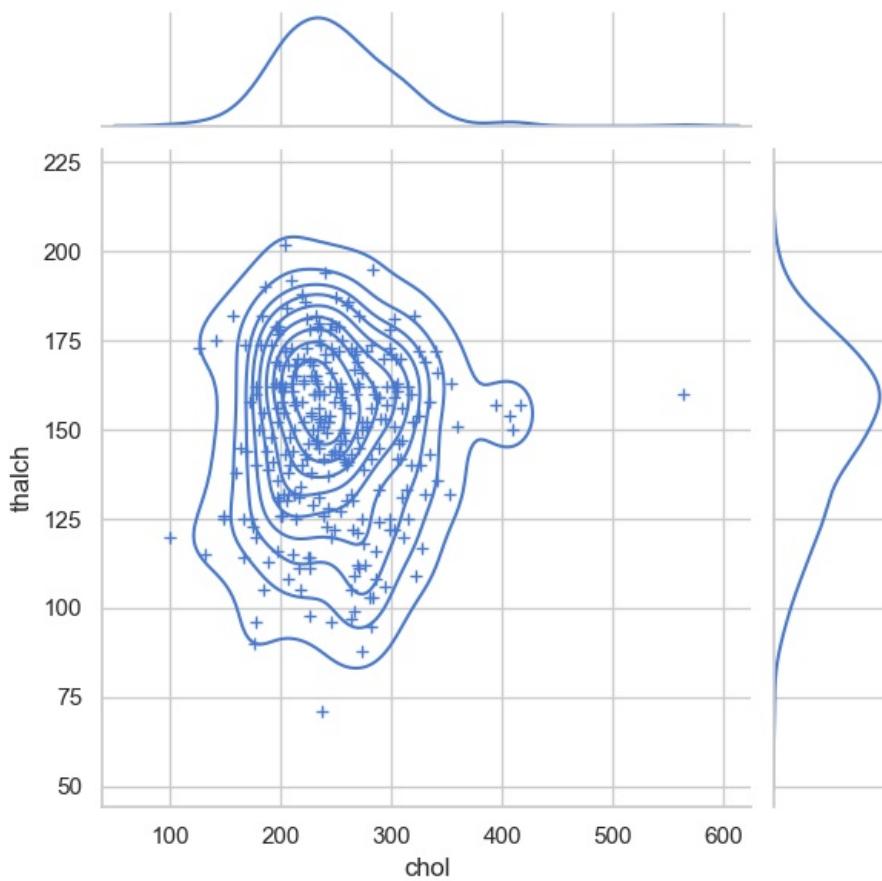


```
In [81]: df = pd.read_csv("heart_disease_uci.csv")
df = df.dropna()
df.head(2)
```

```
Out[81]:
```

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2

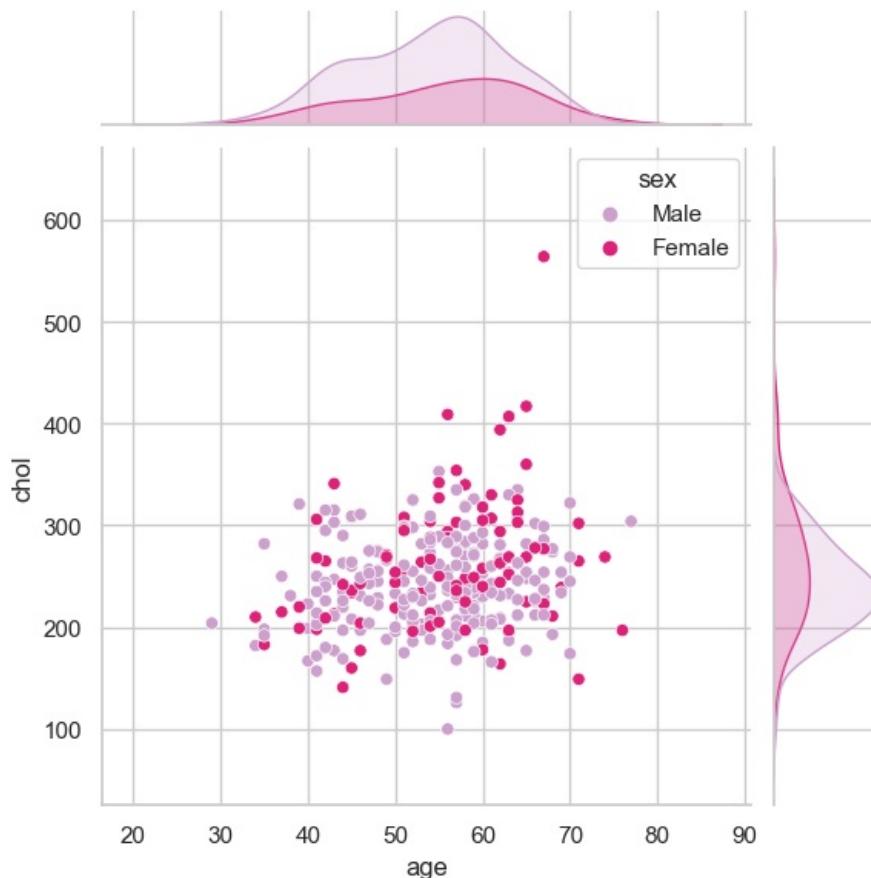
```
In [82]: g = sns.jointplot(x="chol", y="thalch", data=df, kind="kde", color="b")
g.plot_joint(plt.scatter, c="b", s=30, linewidth=1, marker="+")
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("chol", "thalch");
```



```
In [83]: # --- Create Jointplot ---
jointplot = sns.jointplot(x = 'age', y = 'chol', data = df, hue = 'sex', palette = 'PuRd')
```

```
# --- Jointplot Titles & Text ---
jointplot.fig.suptitle('Jointplot between Age and Chol', fontweight = 'heavy', y = 1.05, fontsize = '14',
fontfamily = 'sans-serif', color = 'black');
```

Jointplot between Age and Chol



```
In [84]: import matplotlib.pyplot as plt
```

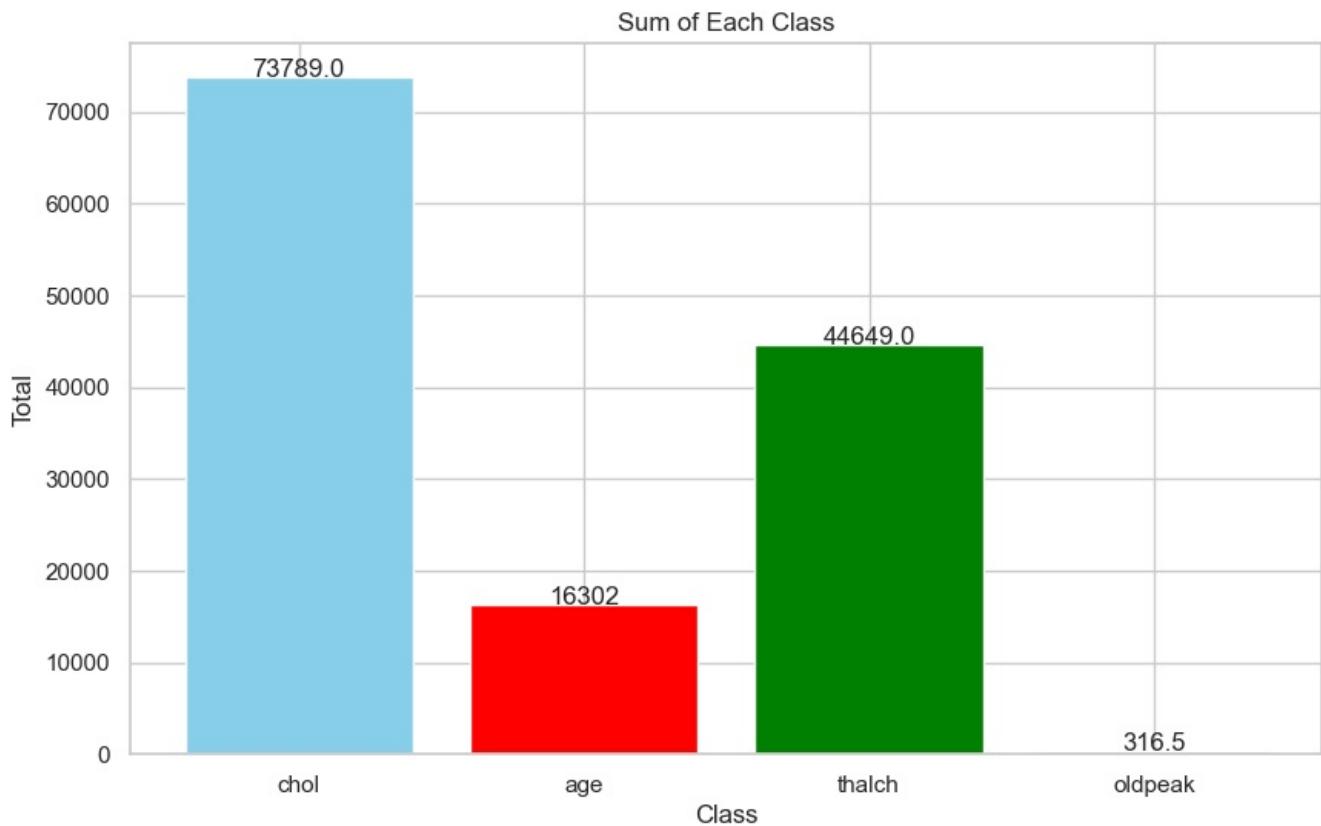
```
# Define the labels
labels = ['chol', 'age', 'thalch','oldpeak']

# Calculate counts
counts = [df[label].sum() for label in labels]

# Create the bar plot
plt.figure(figsize=(10, 6))
bars = plt.bar(labels, counts, color=['skyblue', 'Red', 'Green'])

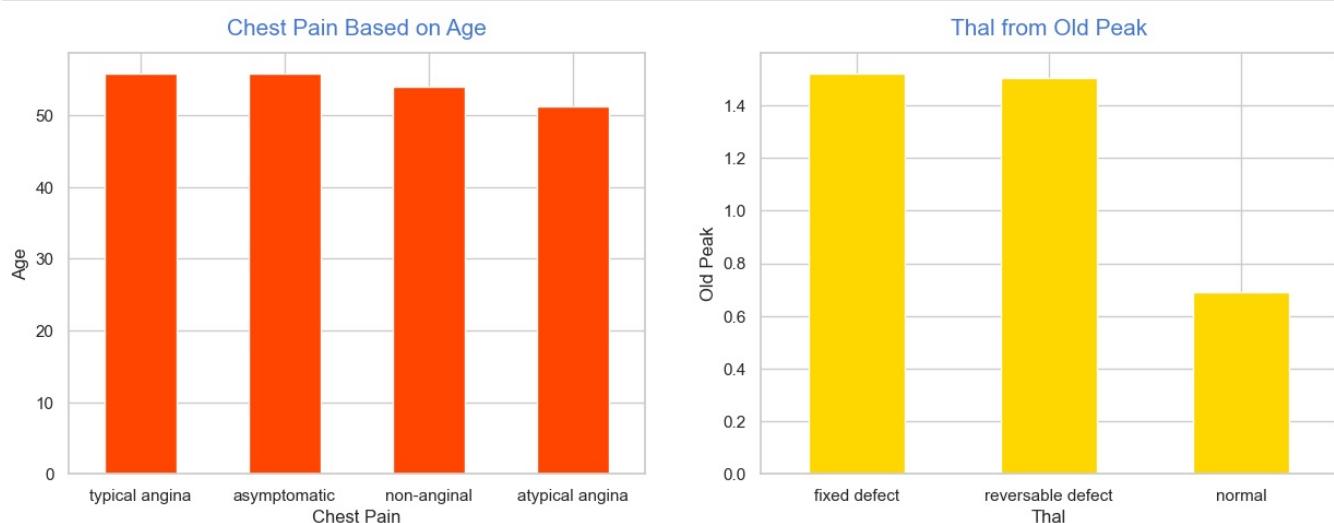
# Add value labels on top of each bar with some vertical offset
for bar, count in zip(bars, counts):
    yval = bar.get_height() + 0.1 # Add a small offset
    plt.text(bar.get_x() + bar.get_width() / 2, yval, str(count), ha='center')

plt.title('Sum of Each Class')
plt.xlabel('Class')
plt.ylabel('Total')
plt.show()
```

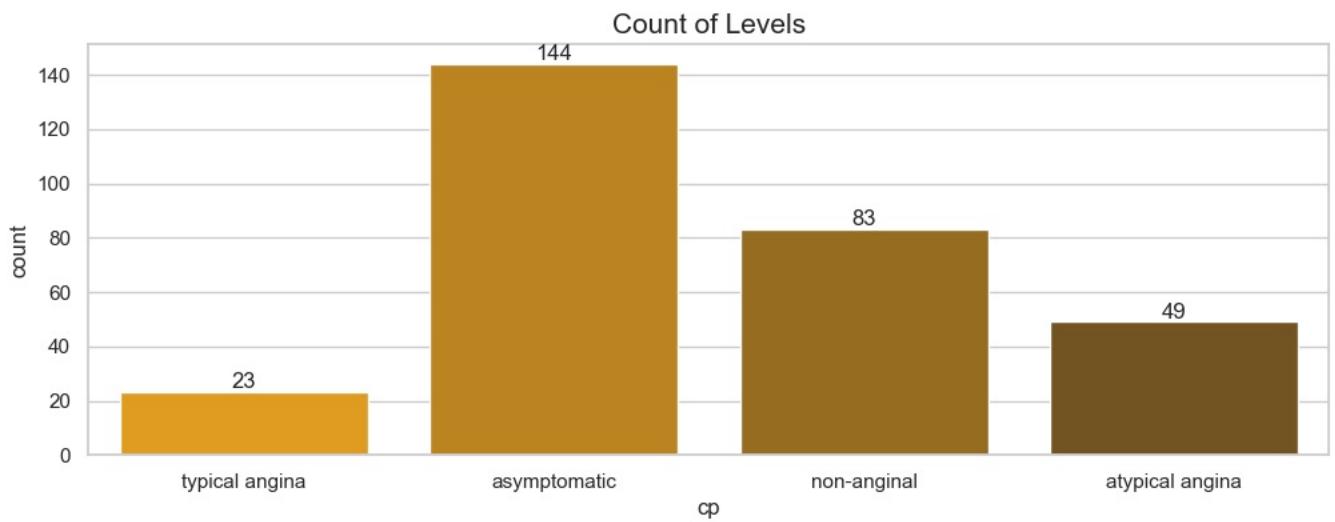


```
In [85]: plt.rcParams['figure.figsize'] = (15,5)
plt.subplot(1, 2, 1)
chart = df.groupby('cp')['age'].mean().sort_values(ascending = False).plot(kind = 'bar', color = 'orangered')
chart.set_xticklabels(chart.get_xticklabels(), rotation = 0)
plt.title('Chest Pain Based on Age', fontsize = 15, color = 'b', pad = 12)
plt.xlabel('Chest Pain')
plt.ylabel('Age')

plt.subplot(1, 2, 2)
chart = df.groupby('thal')['oldpeak'].mean().sort_values(ascending = False).plot(kind = 'bar', color = 'gold')
chart.set_xticklabels(chart.get_xticklabels(), rotation = 0)
plt.title('Thal from Old Peak', fontsize = 15, color = 'b', pad = 12)
plt.xlabel('Thal')
plt.ylabel('Old Peak')
plt.show()
```

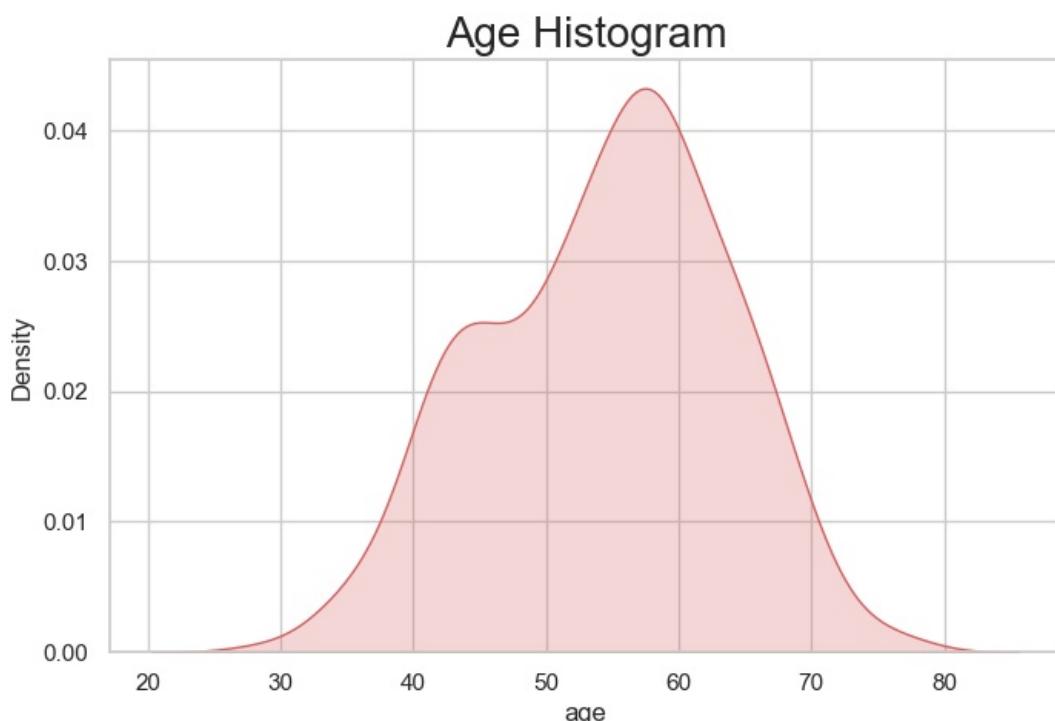


```
In [86]: plt.figure(figsize = (12,4))
ax = sns.countplot(x=df.cp)
for bars in ax.containers:
    ax.bar_label(bars)
plt.title("Count of Levels", fontsize = 15);
```



In [87]:

```
plt.figure(figsize = (8,5))
sns.kdeplot(df.age, shade = True, color = "r")
plt.title("Age Histogram", fontsize = 20)
plt.show()
print("Histogram's skewness is {} and kurtosis is {}".format(df.age.skew(), df.age.kurtosis()))
```



Histogram's skewness is -0.21485314045391055 and kurtosis is -0.5174882052116159

In [88]:

```
import scipy.stats as stats

df_numeric = df.select_dtypes(include='number')

results = []

for col in df_numeric.columns:
    skewness = df_numeric[col].skew()
    kurtosis = df_numeric[col].kurt()
    results.append([col, skewness, kurtosis])

df_stats = pd.DataFrame(results, columns=['Column', 'Skewness', 'Kurtosis'])
df_stats
```

Out[88]:

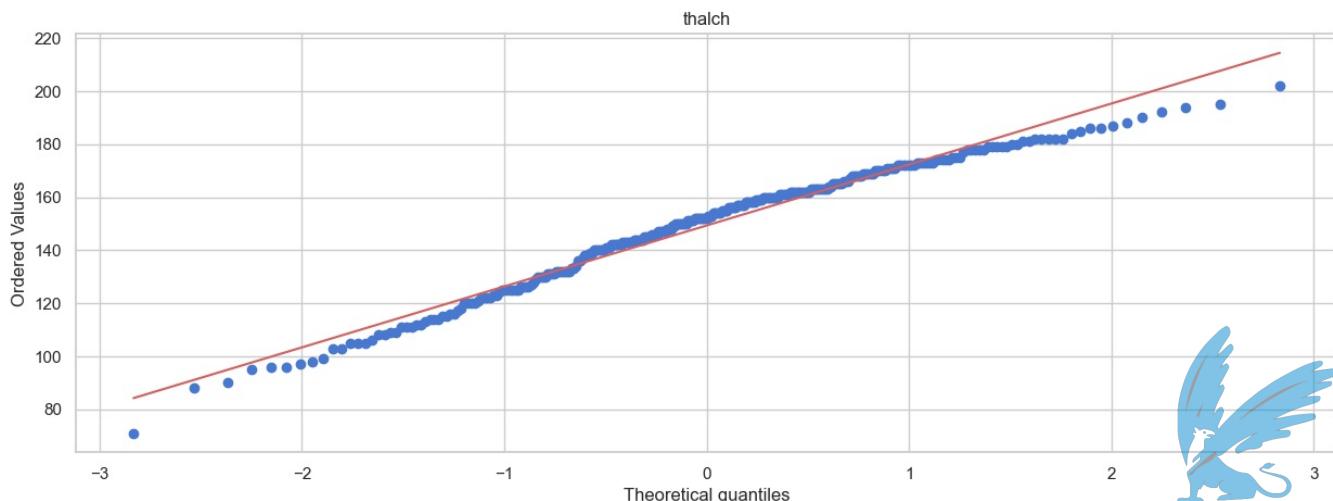
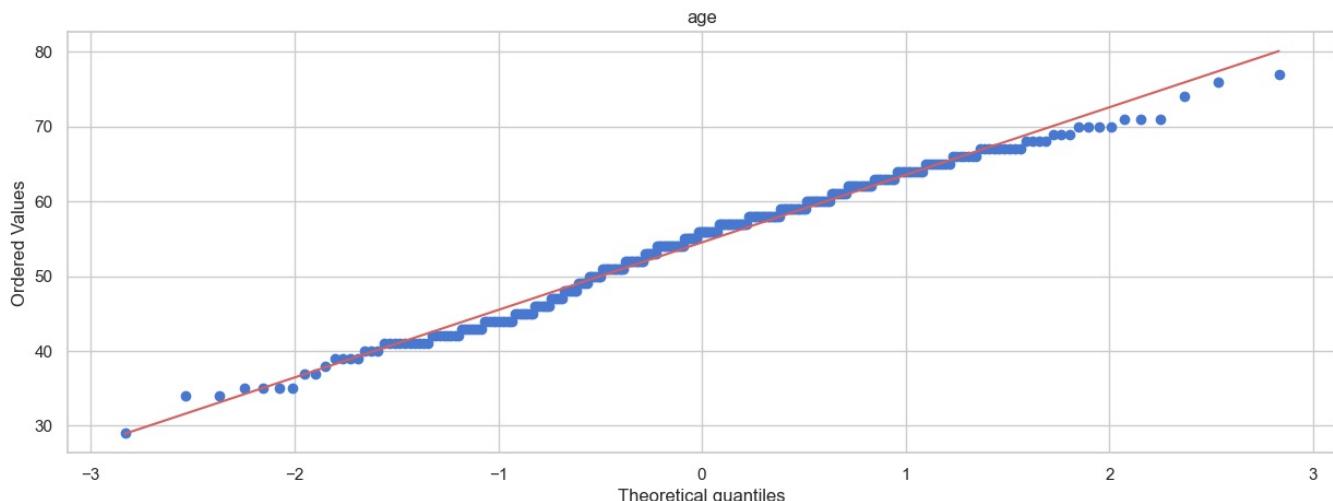
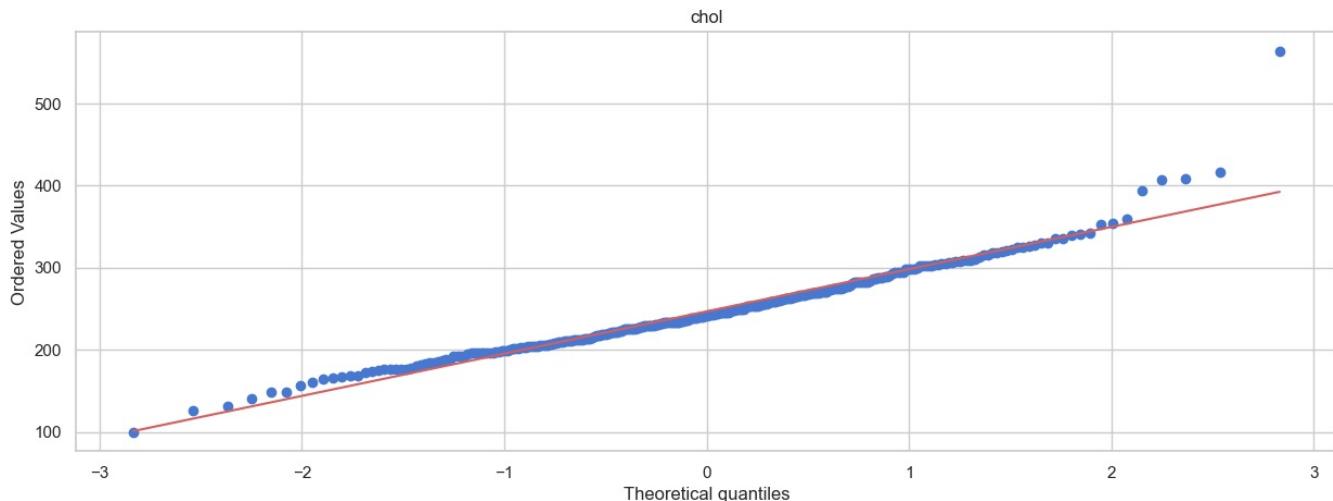
	Column	Skewness	Kurtosis
0	id	0.90	3.95
1	age	-0.21	-0.52
2	trestbps	0.70	0.80
3	chol	1.03	4.35
4	thalch	-0.53	-0.09
5	oldpeak	1.24	1.52
6	ca	1.19	0.26
7	num	1.05	-0.16

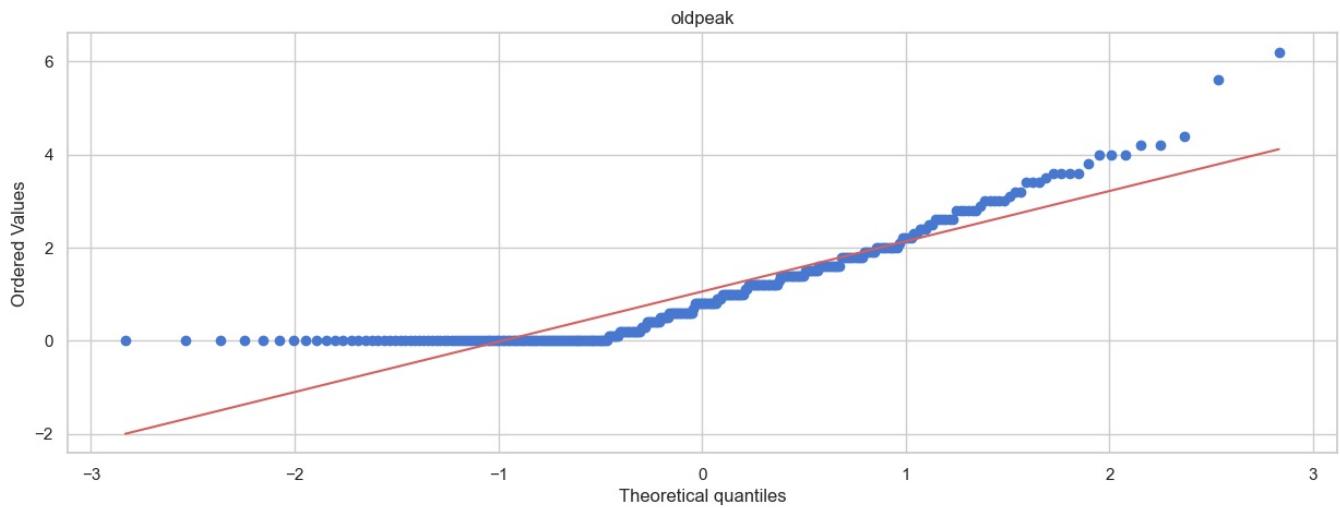
In [89]:

```
from scipy.stats import norm

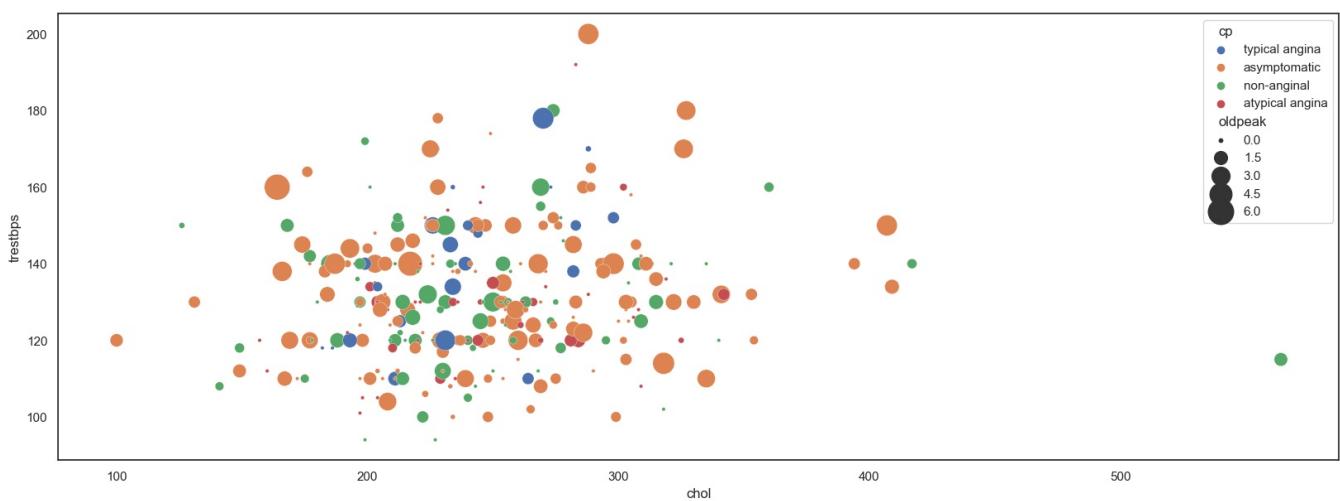
dfx = df[['chol', 'age', 'thalch','oldpeak']]

for col in dfx:
    stats.probplot(dfx[col], plot=plt)
    plt.title(col)
    plt.show();
```

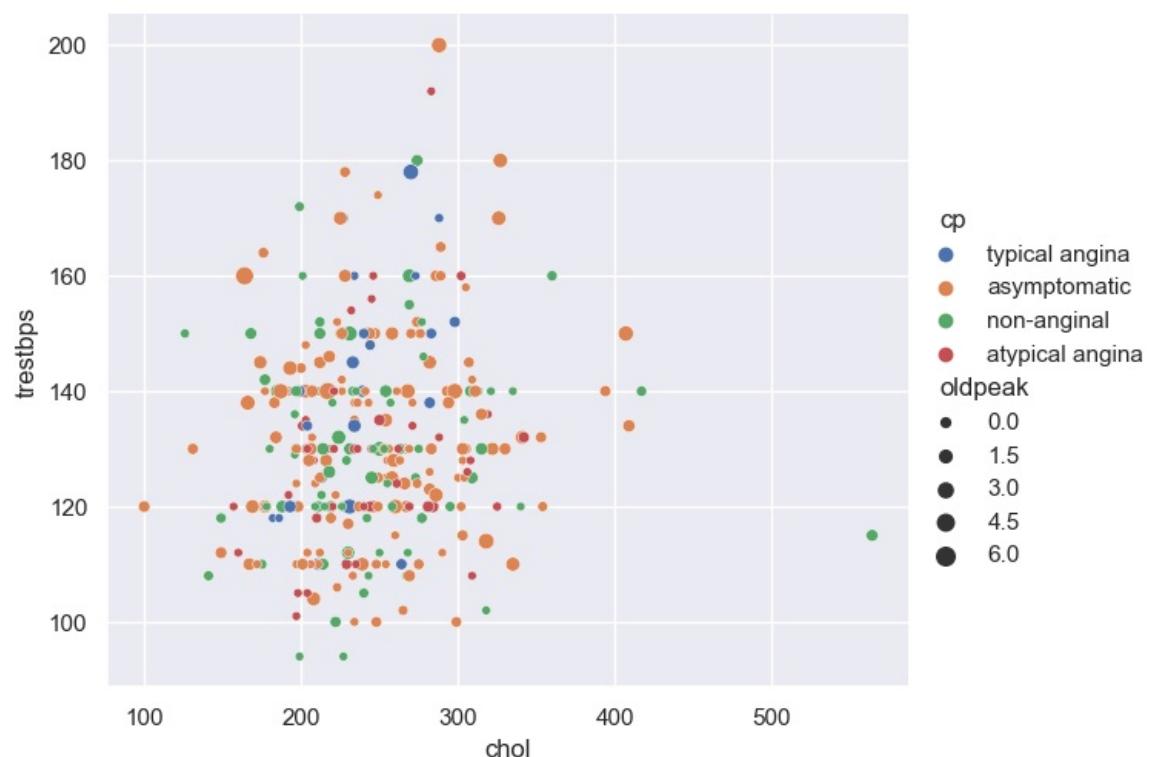




```
In [90]: sns.set(rc={'figure.figsize':(20,7)})
sns.set_style("white")
sns.scatterplot(data=df, x="chol", y="trestbps", size="oldpeak", hue='cp', legend=True, sizes=(10, 500));
```



```
In [91]: sns.set(rc={'figure.figsize':(20,7)})
sns.relplot(y='trestbps',x='chol',data=df,kind='scatter',size='oldpeak',hue='cp',aspect=1.2);
```



```
In [92]: # Filter out rows with 'oldpeak' equal to 0.0
df_filtered = df[df['oldpeak'] != 0.0]

# Create the countplot
plt.figure(figsize=(20, 7))
```



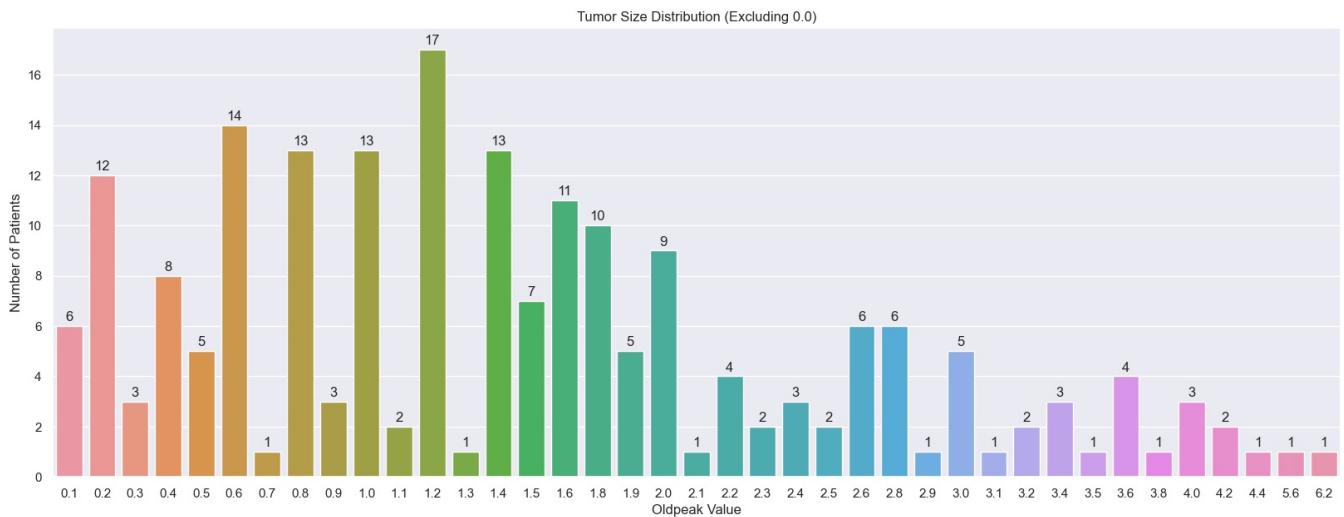
```

sns.countplot(data=df_filtered, x='oldpeak', order=sorted(df_filtered['oldpeak'].unique()))

# Access bars through the current axes
for bar in plt.gca().patches:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.1, int(bar.get_height()), ha='center', va='bottom')

# Add labels and title
plt.title('Tumor Size Distribution (Excluding 0.0)')
plt.ylabel('Number of Patients')
plt.xlabel('Oldpeak Value')
plt.xticks(rotation=0) # Rotate x-axis labels for better readability
plt.show()

```



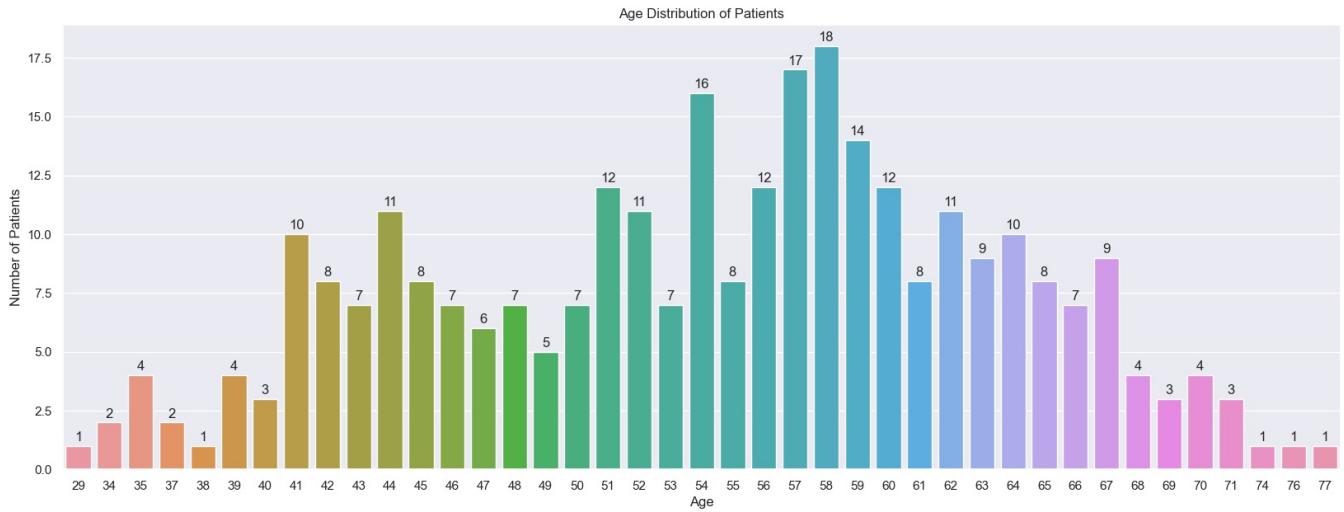
```

In [93]: plt.figure(figsize=(20, 7))
# Create the countplot
sns.countplot(data=df, x='age', order=sorted(df['age'].unique()))

# Access bars through the current axes
for bar in plt.gca().patches:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.1, int(bar.get_height()), ha='center', va='bottom')

# Add labels and title
plt.title('Age Distribution of Patients')
plt.ylabel('Number of Patients')
plt.xlabel('Age')
plt.show()

```



```

In [94]: import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats # Import stats module

def diagnostic_plots(df, variable):

    plt.figure(figsize=(17, 5))
    plt.subplot(1, 3, 1)
    sns.distplot(df[variable])
    plt.title('Histogram')

    plt.subplot(1, 3, 2)
    stats.probplot(df[variable], dist="norm", plot=plt) # Use stats.probplot
    plt.ylabel('RM quantiles')

    plt.subplot(1, 3, 3)
    sns.boxplot(x=df[variable])
    plt.title('Boxplot')

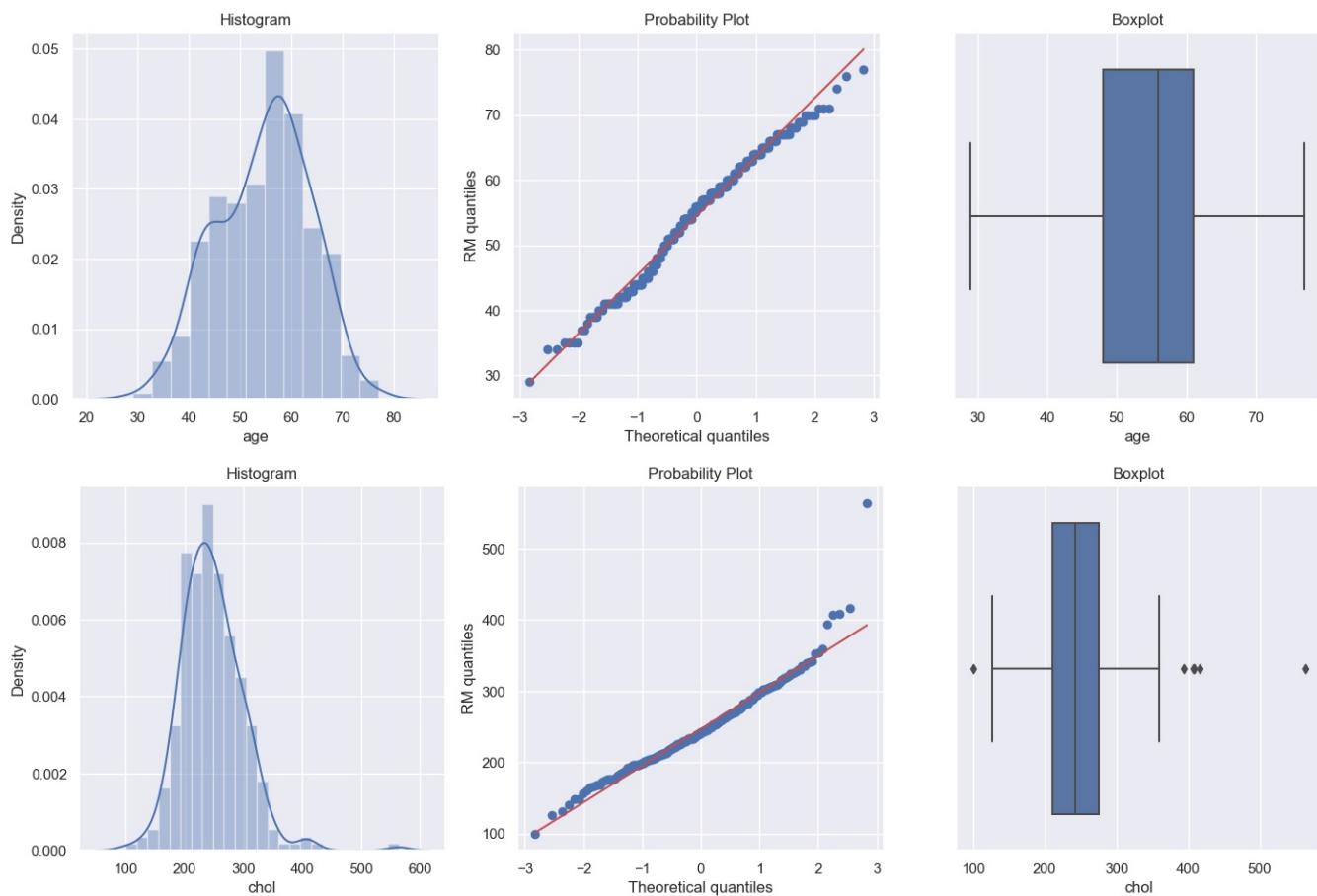
```

```

plt.show()

for col in df[['age','chol']].select_dtypes(exclude="0").columns[:20].to_list():
    diagnostic_plots(df,col)

```

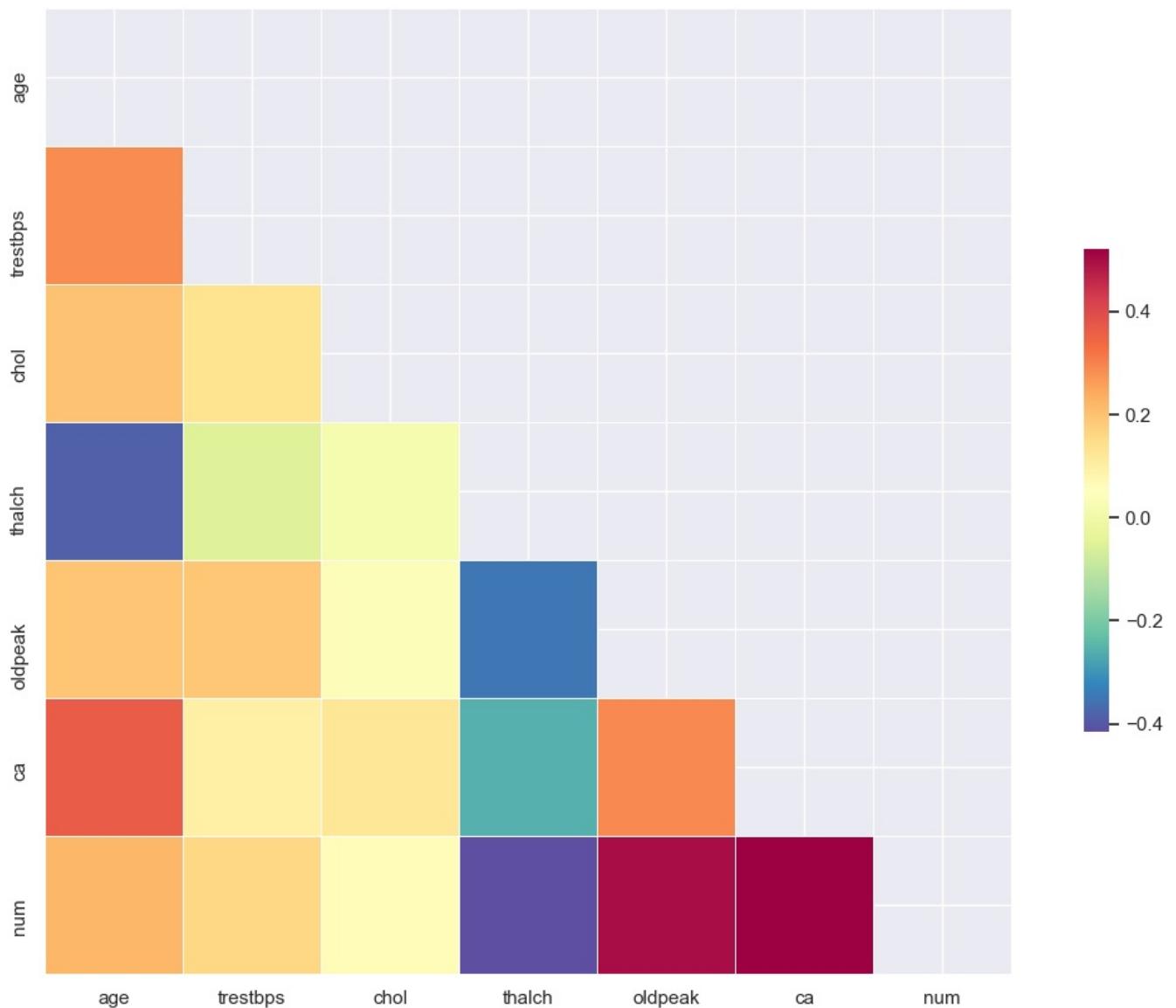


```

In [95]: corr = df.select_dtypes('number').drop('id',axis=1).corr()
# Generate a mask for the upper triangle
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
fig, ax = plt.subplots(figsize=(15,10))
sns.heatmap(corr, cmap='Spectral_r', mask=mask, square=True, annot=True, linewidth=0.5, cbar_kws={"shrink" : 0.5})

```





```
In [96]: df = df.drop('id', axis=1)
df.head(2)
```

	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2

```
In [97]: plt.figure(figsize=(20, 5))
sns.set_context("paper")

kdeplt = sns.kdeplot(
    data=df,
    x="chol",
    hue="sex",
    palette='Dark2',
    alpha=0.7,
    lw=2,
)

kdeplt.set_title("Cholesterol values distribution\nMale VS Female", fontsize=12)
kdeplt.set_xlabel("Cholesterol", fontsize=12)

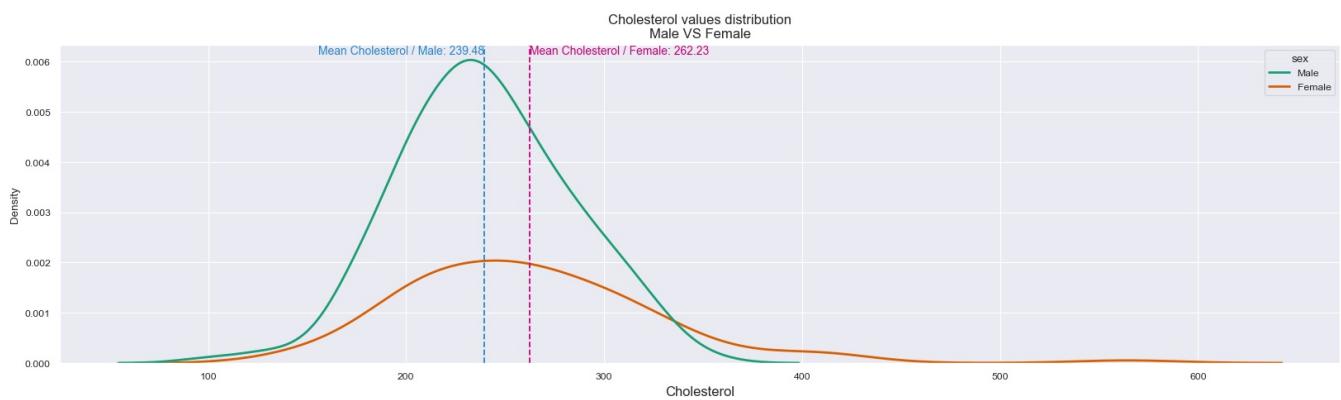
# Calculate mean cholesterol for each sex
mean_male = df[df['sex'] == 'Male']['chol'].mean()
mean_female = df[df['sex'] == 'Female']['chol'].mean()

# Add vertical lines for mean cholesterol
plt.axvline(x=mean_male, color="#2986cc", ls="--", lw=1.3)
plt.axvline(x=mean_female, color="#c90076", ls="--", lw=1.3)

# Add text annotations
plt.text(mean_male, plt.gca().get_ylim()[1], f"Mean Cholesterol / Male: {mean_male:.2f}",
         fontsize=10, color="#2986cc", ha='right', va='top')
plt.text(mean_female, plt.gca().get_ylim()[1], f"Mean Cholesterol / Female: {mean_female:.2f}",
         fontsize=10, color="#c90076", ha='left', va='top')

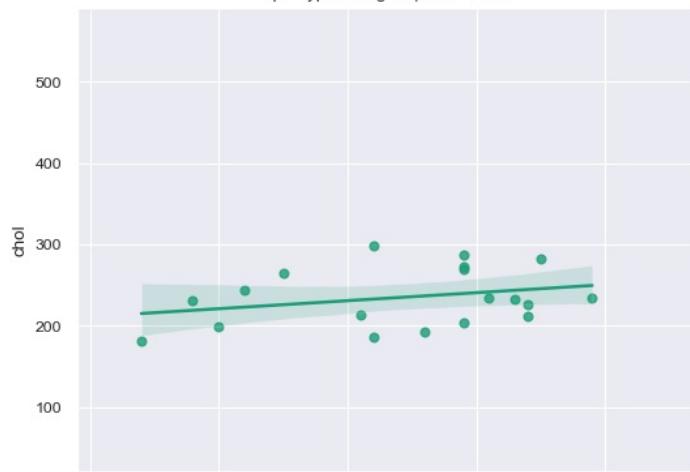
plt.show()
```



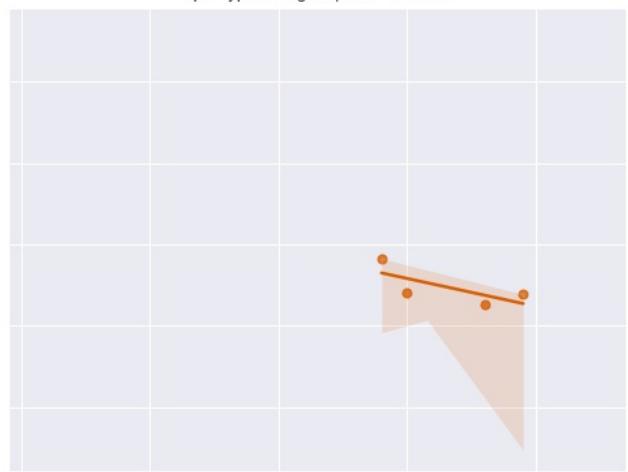


```
In [98]: heart_df_fg = sns.FacetGrid(
    data=df,
    col="sex",
    hue="sex",
    row="cp",
    height=4,
    aspect=1.3,
    palette='Dark2',
    col_order=["Male", "Female"],
)
heart_df_fg.map_dataframe(sns.regplot, "age", "chol")
plt.show()
```

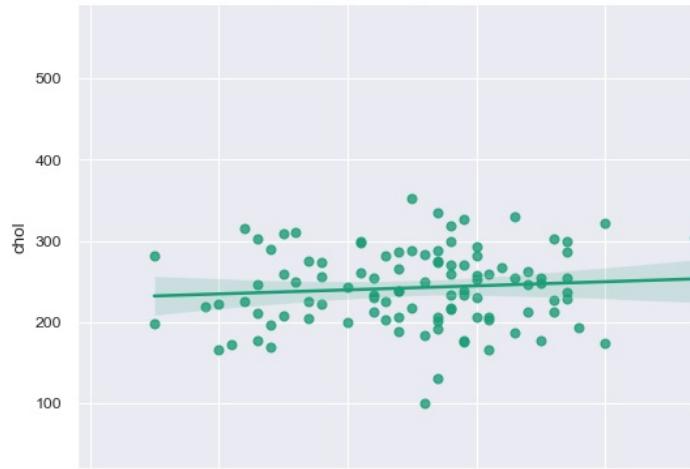
cp = typical angina | sex = Male



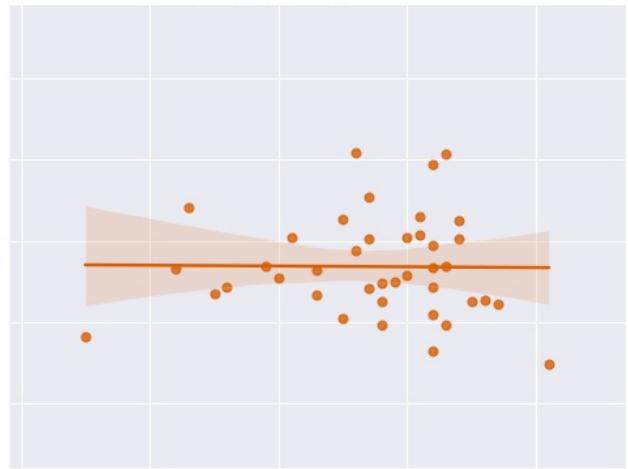
cp = typical angina | sex = Female



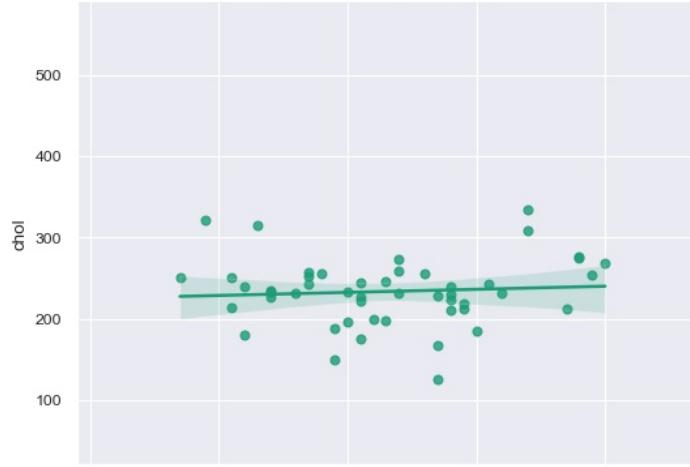
cp = asymptomatic | sex = Male



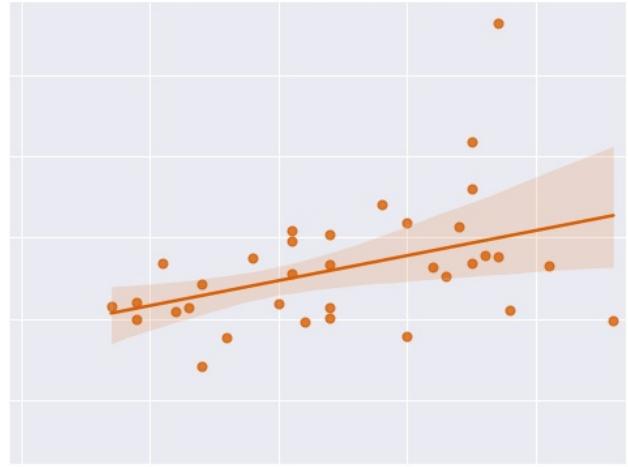
cp = asymptomatic | sex = Female



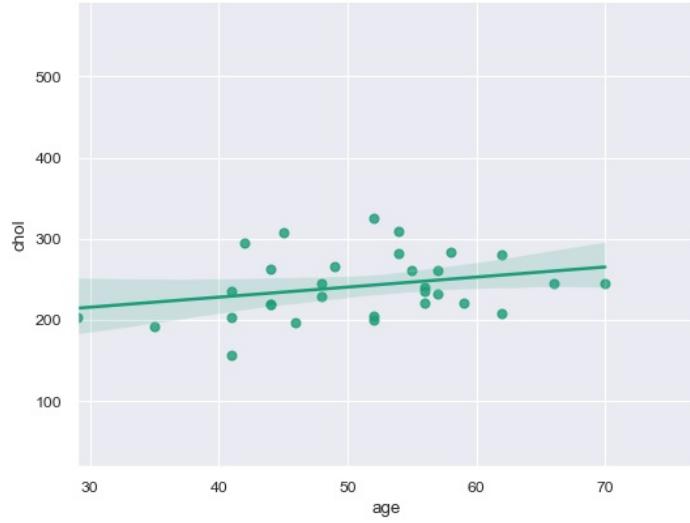
cp = non-anginal | sex = Male



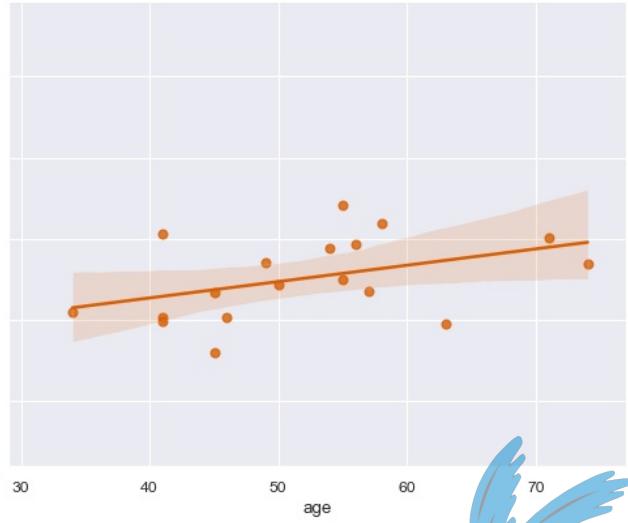
cp = non-anginal | sex = Female



cp = atypical angina | sex = Male



cp = atypical angina | sex = Female



In [99]:
x = df.groupby("cp")["chol"].min().index
y = df.groupby("cp")["chol"].min().values



```

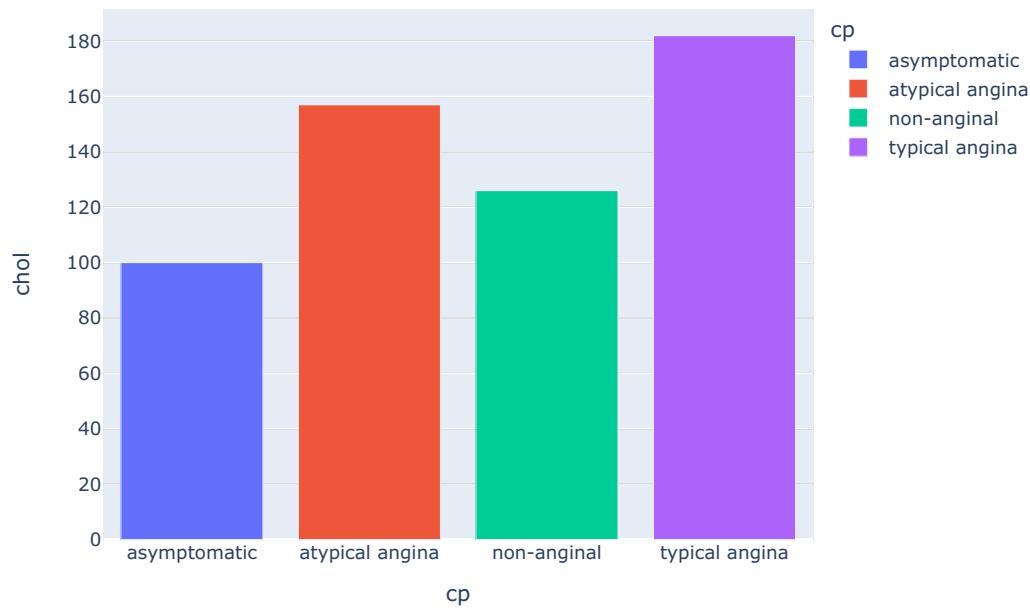
df = pd.DataFrame({'cp':x,
                   'chol':y })

fig = px.bar(df,
              x='cp',
              y='chol',
              color='cp', #color represents brand
              title='Chol Value'
             )
fig.show()

```



Chol Value



```

In [100]: df = pd.read_csv("heart_disease_uci.csv")
df = df.dropna()
df = df.drop('id', axis=1)
df.head(2)

```

```

Out[100]:   age  sex  dataset      cp  trestbps  chol    fbs  restecg  thalch  exang  oldpeak  slope  ca  thal  num
0   63  Male  Cleveland  typical angina  145.0  233.0  True  lv hypertrophy  150.0  False  2.3  downsloping  0.0  fixed defect  0
1   67  Male  Cleveland  asymptomatic  160.0  286.0  False  lv hypertrophy  108.0  True  1.5  flat  3.0  normal  2

```

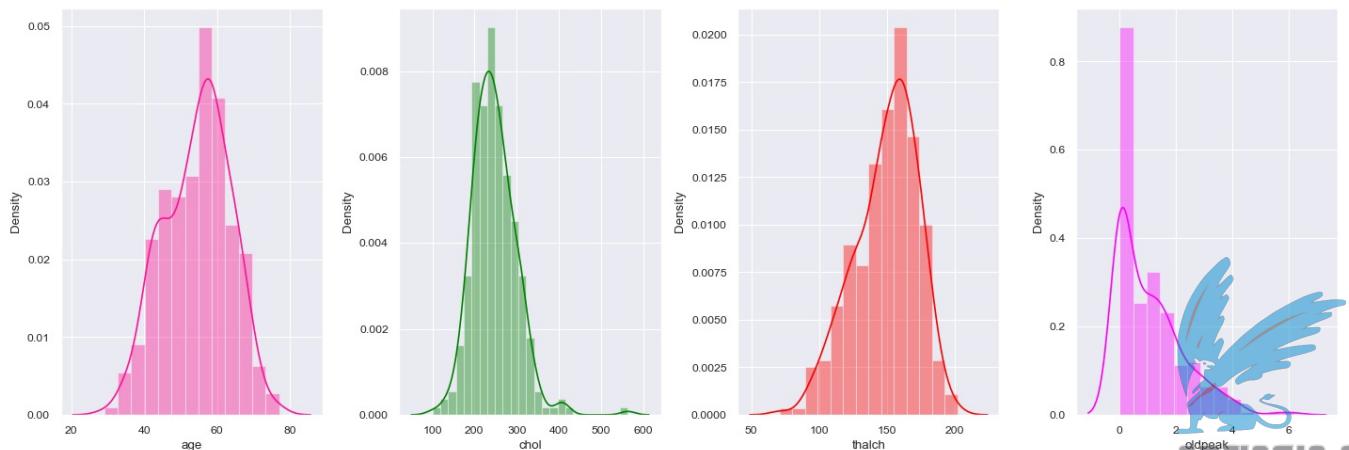
```
In [101]: plt.figure(figsize=(18,5))
```

```

plt.subplot(1,5,1)
sns.distplot(df['age'],color='DeepPink')
plt.subplot(1,5,2)
sns.distplot(df['chol'],color='Green')
plt.subplot(1,5,3)
sns.distplot(df['thalch'],color='Red')
plt.subplot(1,5,4)
sns.distplot(df['oldpeak'],color='Magenta')

plt.tight_layout()
plt.show()

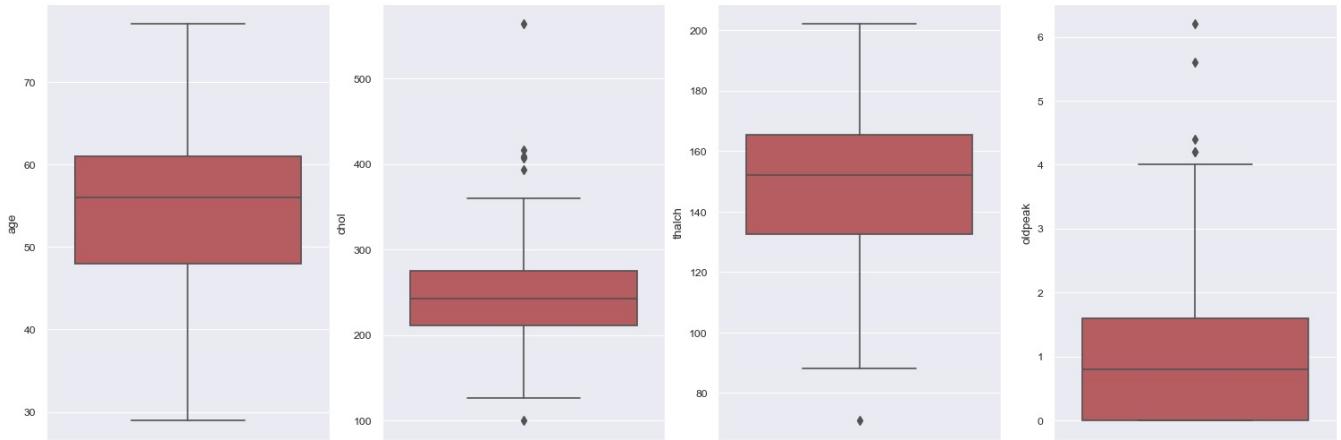
```



```
In [102]: df_cpy = df.copy("Deep")
df_cpy = df_cpy.select_dtypes("number")
df_cpy = df_cpy[['age', 'chol', 'thalch', 'oldpeak']]

fig, axis=plt.subplots(ncols=4, nrows=1, figsize=(15,5))
index=0
axis=axis.flatten()

for col, values in df_cpy.items():
    sns.boxplot(y=col, data=df_cpy, color='r', ax=axis[index])
    index+=1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```

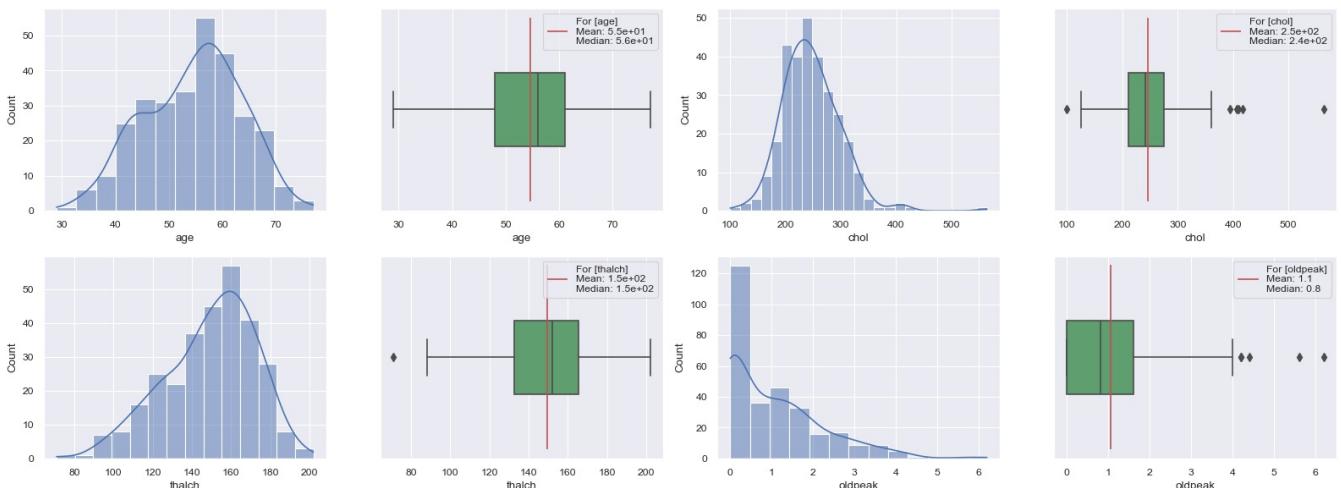


```
In [103]: df_cpy = df.copy("Deep")
df_cpy = df_cpy.select_dtypes("number")
df_cpy = df_cpy[['age', 'chol', 'thalch', 'oldpeak']]

flierprops = dict(markerfacecolor='g', color='g', alpha=0.5)

n_cols = 4
n_rows = int(np.ceil(df_cpy.shape[-1]*2 / n_cols))
fig, axes = plt.subplots(n_rows, n_cols, figsize=(4 * n_cols, 3 * n_rows))
for i, (col) in enumerate(list(df_cpy.columns)):
    mean = df_cpy[col].mean()
    median = df_cpy[col].median()
    sns.histplot(df_cpy[col], ax=axes.flatten()[2*i], kde=True)
    sns.boxplot(x=df_cpy[col], orient='h', ax=axes.flatten()[2*i+1], color='g')
    axes.flatten()[2*i+1].vlines(mean, ymin = -1, ymax = 1, color='r', label=f"For [{col}]\nMean: {mean:.2}\nMedian: {median:.2}")
    axes.flatten()[2*i+1].legend()

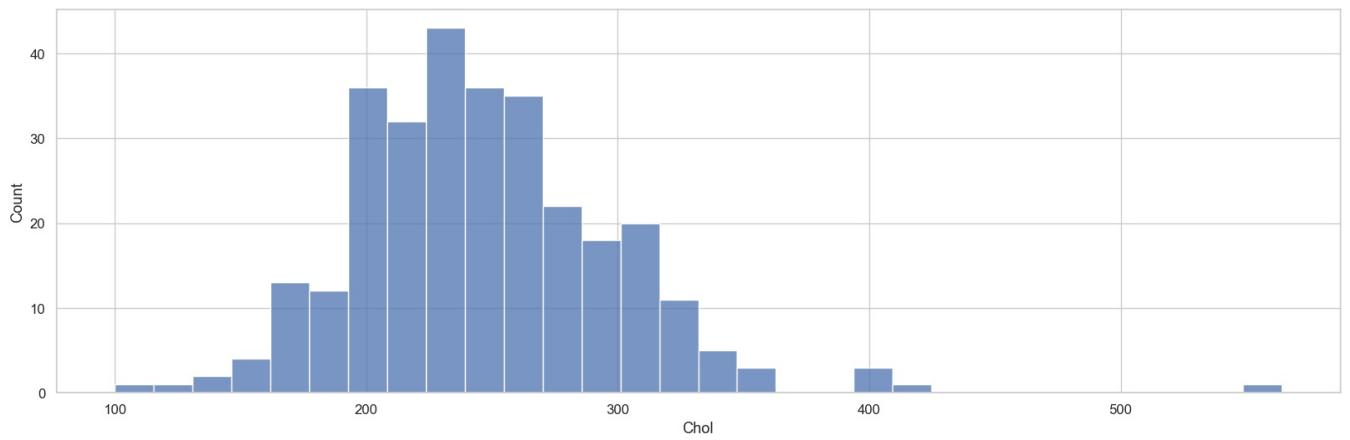
    if i % n_cols == 0:
        ax.set_ylabel('Frequency')
    else:
        ax.set_ylabel('')
plt.tight_layout()
```



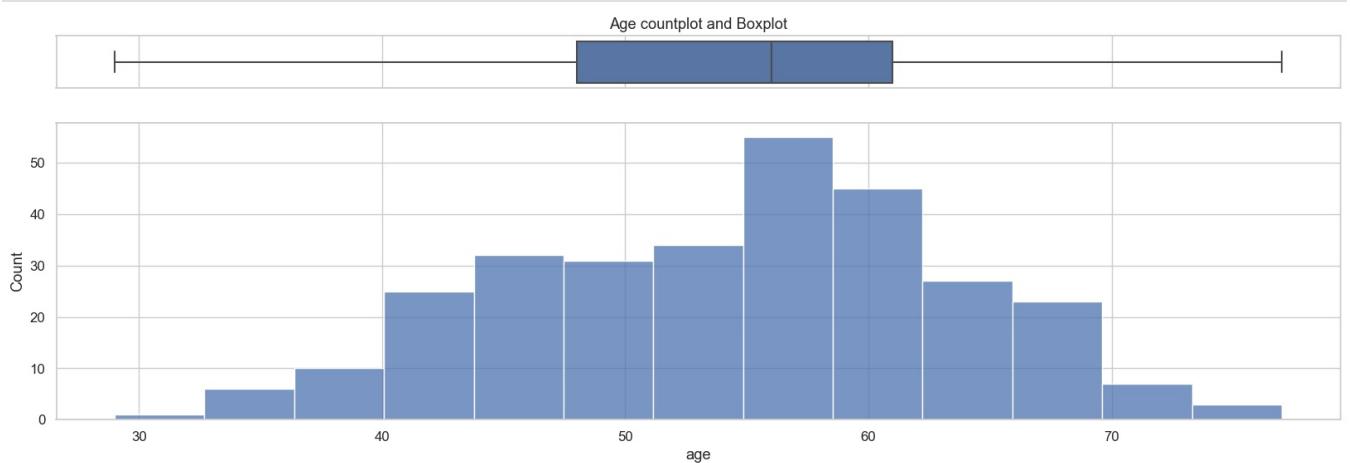
```
In [104]: sns.set(style="whitegrid", palette="deep", font_scale=1.1, rc={"figure.figsize": [20, 6]})

sns.histplot(df['chol'], bins = 30).set(xlabel = "Chol");
```

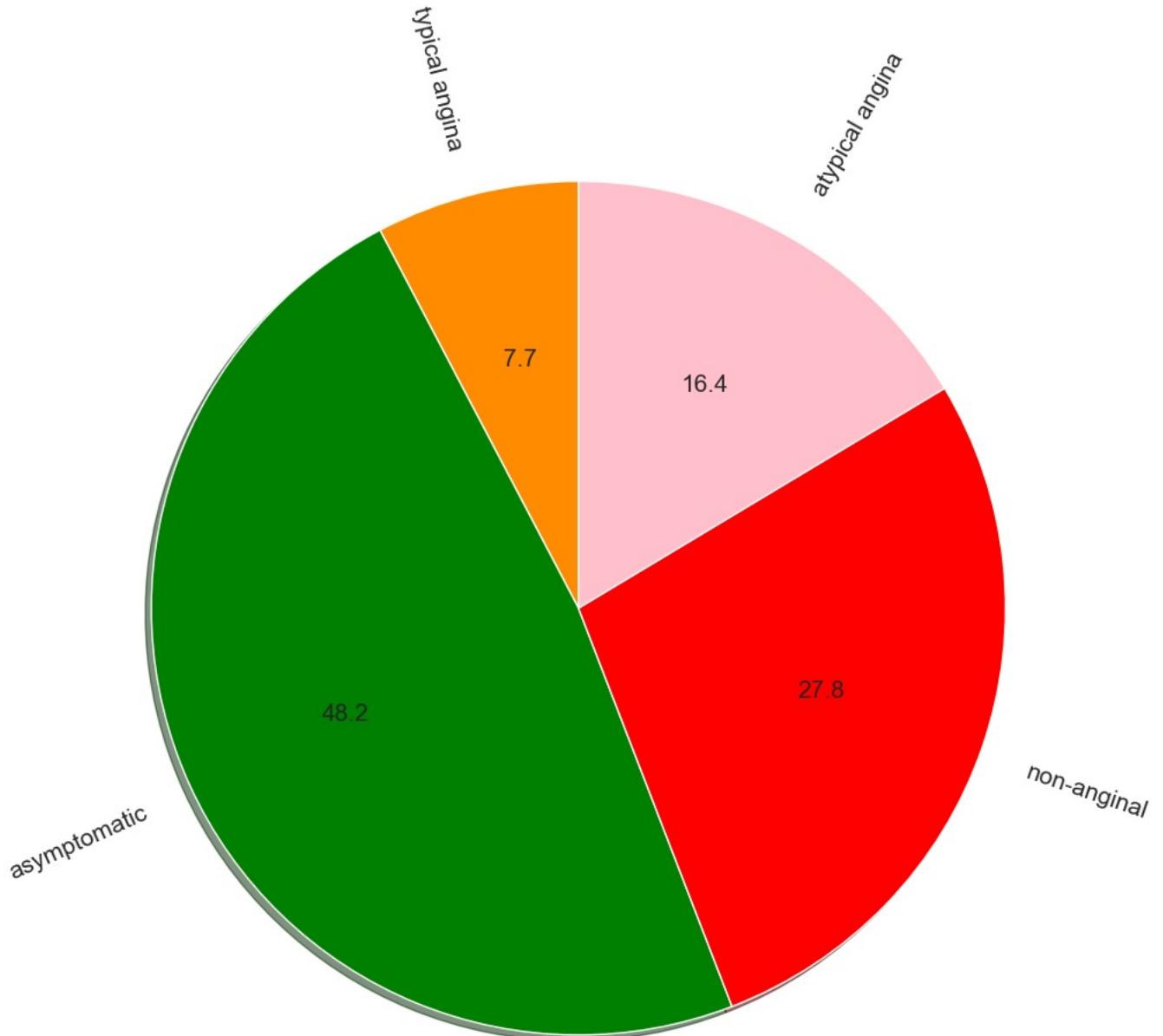




```
In [105]: df2 = df[['age', 'sex', 'chol']]
f, (ax_box, ax_hist) = plt.subplots(2, sharex=True, gridspec_kw={"height_ratios": (.15, .85)})
ax_box.title.set_text('Age countplot and Boxplot')
sns.boxplot(df2["age"], orient="h", ax=ax_box)
sns.histplot(data=df2, x="age", ax=ax_hist)
ax_box.set(xlabel='')
plt.show()
```



```
In [106]: #is online delivery available?
colors = ("darkorange", "green", 'Red', 'Pink')
explodes = [0.5, 0.5, 0.75, .50]
df["cp"].value_counts(sort=False).plot.pie(colors=colors,
                                             textprops={'fontsize': 15},
                                             autopct = '%4.1f',
                                             startangle= 90,
                                             radius =2,
                                             rotatelabels=True,
                                             shadow = True) ;
```

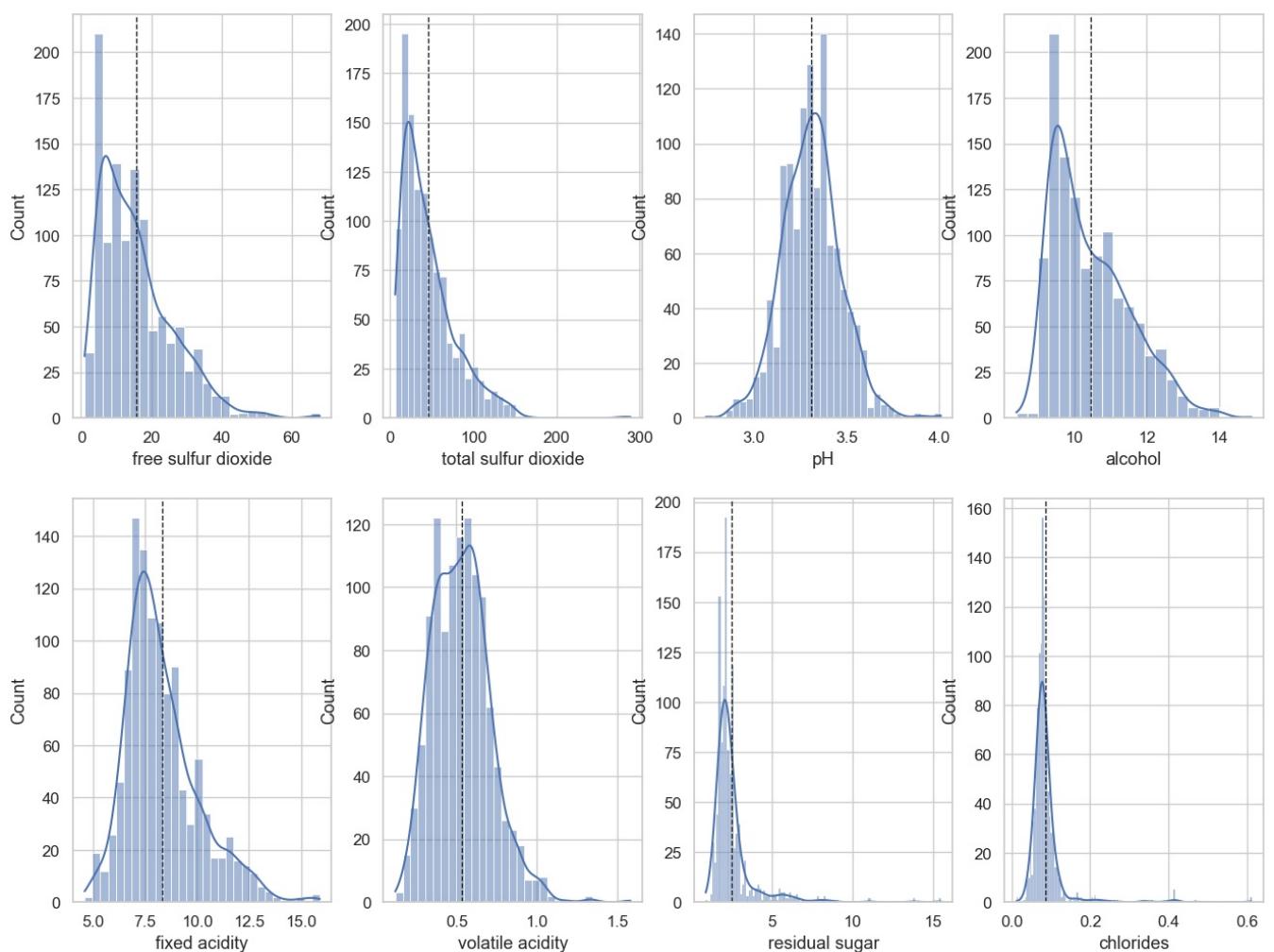


```
In [107]: wine = pd.read_csv("WineQT.csv")
wine.head(2)
```

```
Out[107]:
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.0	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	0
1	7.8	0.88	0.0	2.6	0.10	25.0	67.0	1.0	3.20	0.68	9.8	5	1

```
In [108]: NUMERICAL = wine[['fixed acidity', 'volatile acidity', 'residual sugar',
   'chlorides', 'free sulfur dioxide', 'total sulfur dioxide',
   'pH', 'alcohol']]
fig, axes = plt.subplots(2, 4)
fig.set_figheight(12)
fig.set_figwidth(16)
for i,col in enumerate(NUMERICAL):
    sns.histplot(wine[col],ax=axes[(i // 4) -1 ,(i % 4)], kde = True)
    axes[(i // 4) -1 ,(i % 4)].axvline(wine[col].mean(), color='k', linestyle='dashed', linewidth=1)
```

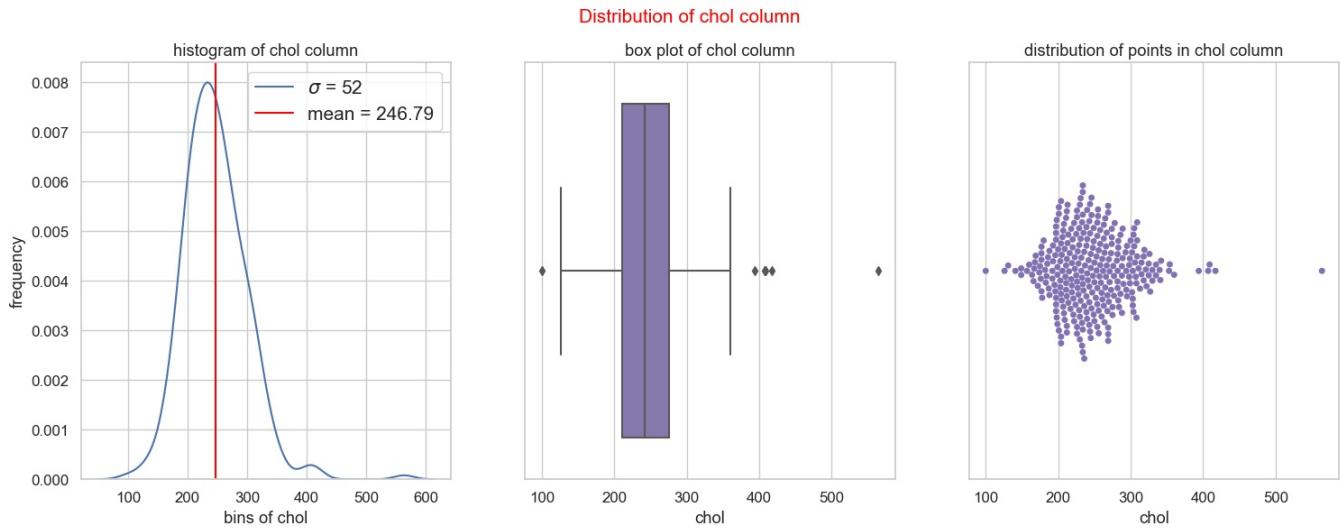


```
In [110]: #set configuration for charts
plt.rcParams["figure.figsize"]=[18 , 6]
plt.rcParams["font.size"]=15
plt.rcParams["legend.fontsize"]="medium"
plt.rcParams["figure.titlesize"]="medium"

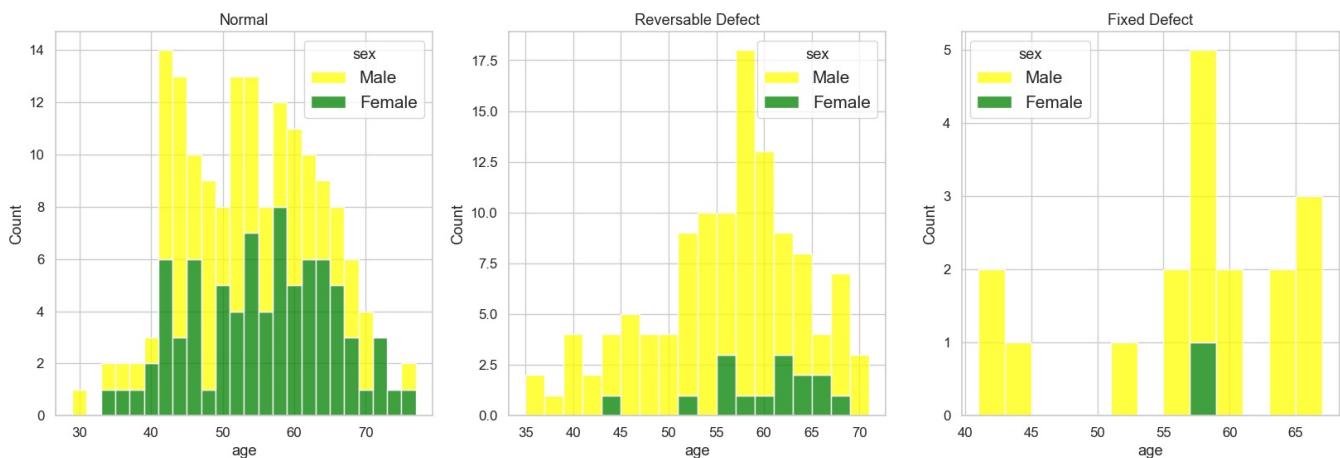
def plot_distribution(data , x ,color,bins ):
    mean = data[x].mean()
    std = data[x].std()
    info=dict(data = data , x = x , color = color)
    plt.subplot(1 , 3 , 1 , title =f"Disttribution of {x} column")
    sns.distplot(a=data[x] , bins = bins)
    plt.xlabel(f"bins of {x}")
    plt.axvline(mean , label ="mean" , color ="red")
    plt.ylabel("frequency")
    plt.legend(["${\sigma}$ = %d"%std , f"mean = {mean:.2f}"])
    plt.title(f"histogram of {x} column")
    plt.subplot(1 , 3 , 2)
    sns.boxplot(**info)
    plt.xlabel(f"{x}")
    plt.title(f"box plot of {x} column")
    plt.subplot(1 , 3 , 3)
    sns.swarmplot(**info)
    plt.xlabel(f"{x}")
    plt.title(f"distribution of points in {x} column")
    plt.suptitle(f"Distribution of {x} column" , fontsize =15 , color="red")
    plt.show()

age_bins = np.arange(29 , 77+5 , 5)
base_color = sns.color_palette()[4]
plot_distribution(data = df , x ="chol" , color = base_color , bins=age_bins)
```



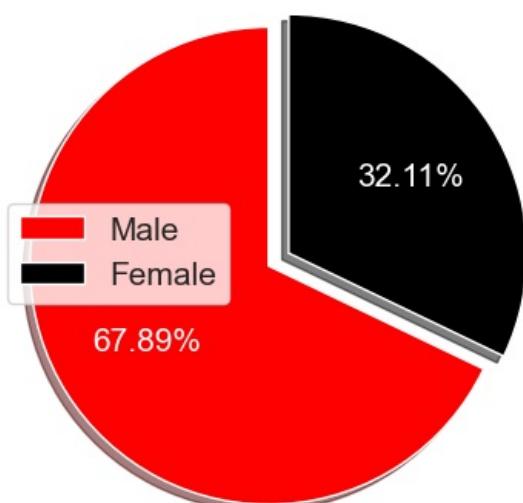


```
In [111]: plot , ax = plt.subplots(1 , 3 , figsize=(20,6))
sns.histplot(data = df.loc[df["thal"] == 'normal'] , x = "age" , hue = "sex",binwidth=2,ax = ax[0],palette = sns
sns.histplot(data = df.loc[df["thal"] == 'reversible defect'] , x = "age" , hue = "sex",binwidth=2,ax = ax[1],pa
sns.histplot(data = df.loc[df["thal"] == 'fixed defect'] , x = "age" , hue = "sex",binwidth=2,ax = ax[2],palette
plt.show()
```



```
In [112]: sex = ["Male", "Female"]
values = df["sex"].value_counts()
color = ["#FF0000", "#000000"]

plt.figure(figsize = (5, 7))
plt.pie(values, labels = sex, colors = color, explode = (0.1, 0), textprops = {"color":"w"}, autopct = "% .2f %"
plt.legend();
```



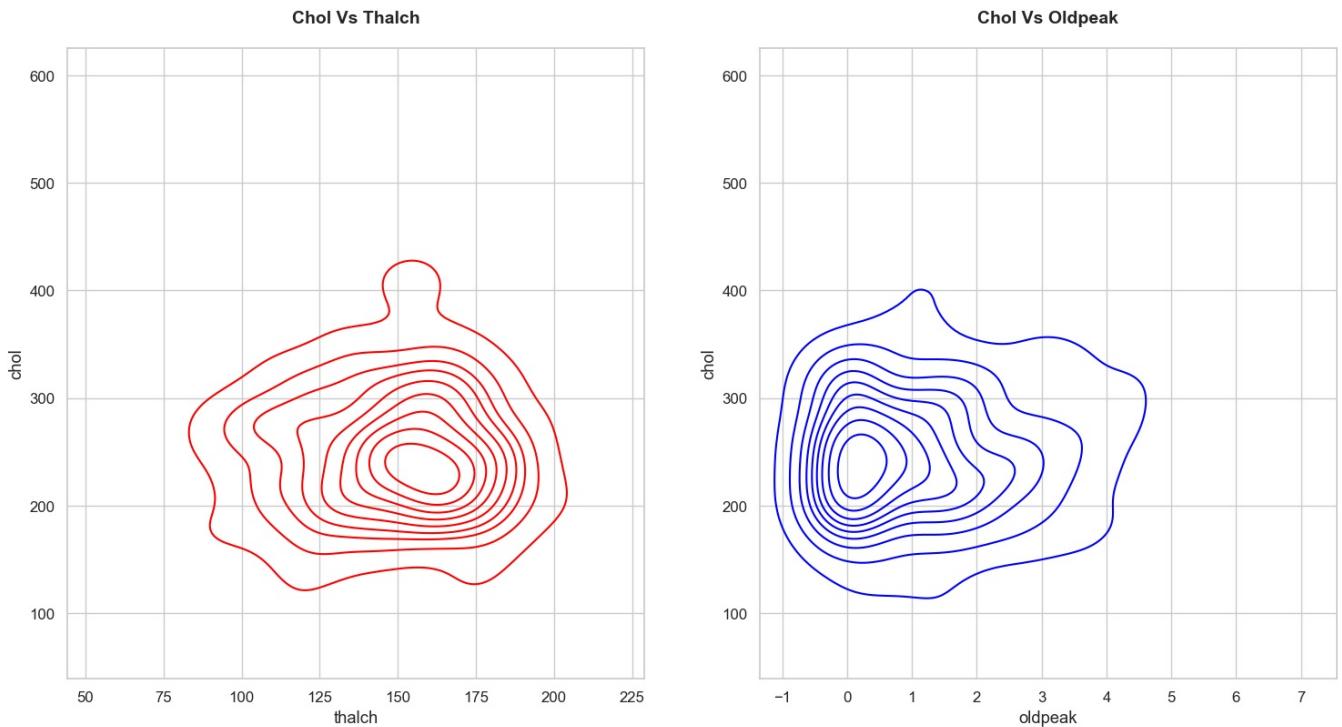
```
In [113]: #plotting
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 9))
fig.suptitle(' Highest and Lowest Correlation ' , size = 20, weight='bold')
axs = [ax1, ax2]
```



```
#kdeplot
sns.kdeplot(data=df, y='chol', x='thalch', ax=ax1, color="red")
ax1.set_title('Chol Vs Thalch', size = 14, weight='bold', pad=20)

#kdeplot
sns.kdeplot(data=df, y='chol', x='oldpeak', ax=ax2, color='Blue')
ax2.set_title('Chol Vs Oldpeak', size = 14, weight='bold', pad=20);
```

Highest and Lowest Correlation



```
In [114]: df1 = pd.read_csv('US_Job_Market.csv')
df1 = df1.dropna().reset_index()
df1 = df1.drop('index', axis=1)
df1.head(2)
```

	position	company	description	reviews	location
0	Data Analyst	Operation HOPE	DEPARTMENT: Program Operations POSITION LOCATION: Atlanta, GA 30303	44.0	Atlanta, GA 30303
1	Assistant Professor -TT - Signal Processing & ...	Emory University	DESCRIPTION\nThe Emory University Department o...	550.0	Atlanta, GA

```
In [115]: plt.figure(figsize=(20, 7))

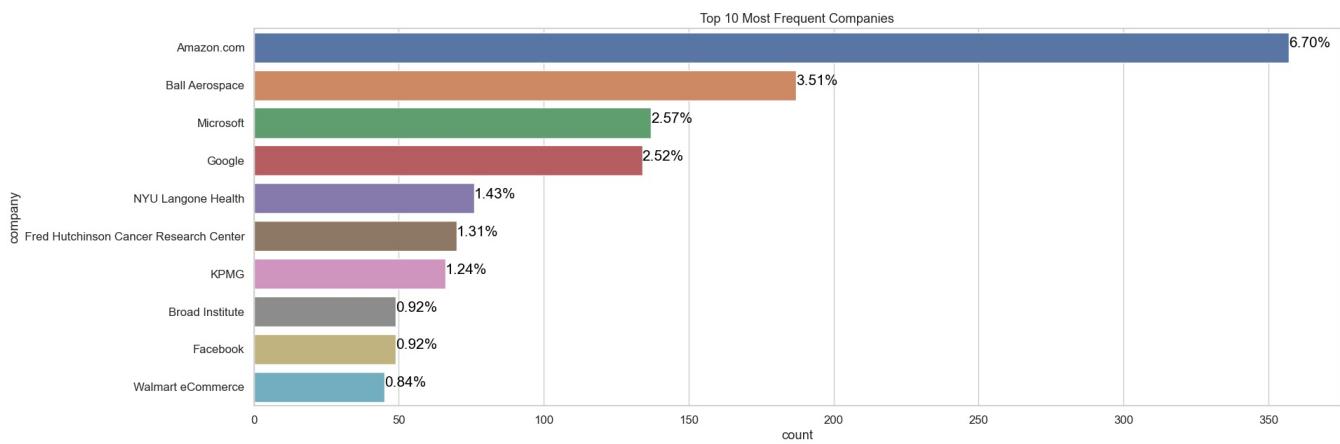
# Filter for the top 10 most frequent companies
df_v = df1['company'].value_counts().head(10).reset_index()
df_v.columns = ['company', 'count']

# Calculate the percentage
total = df1['company'].value_counts().sum()
df_v['percentage'] = (df_v['count'] / total) * 100

# Create the bar plot
plot = sns.barplot(y='company', x='count', data=df_v)

# Annotate the bars with the percentage
for index, row in df_v.iterrows():
    plot.text(row['count'], index, f'{row['percentage']:.2f}%', color='black', ha="left")

plt.xticks(rotation=90)
plt.title('Top 10 Most Frequent Companies')
plt.show()
```

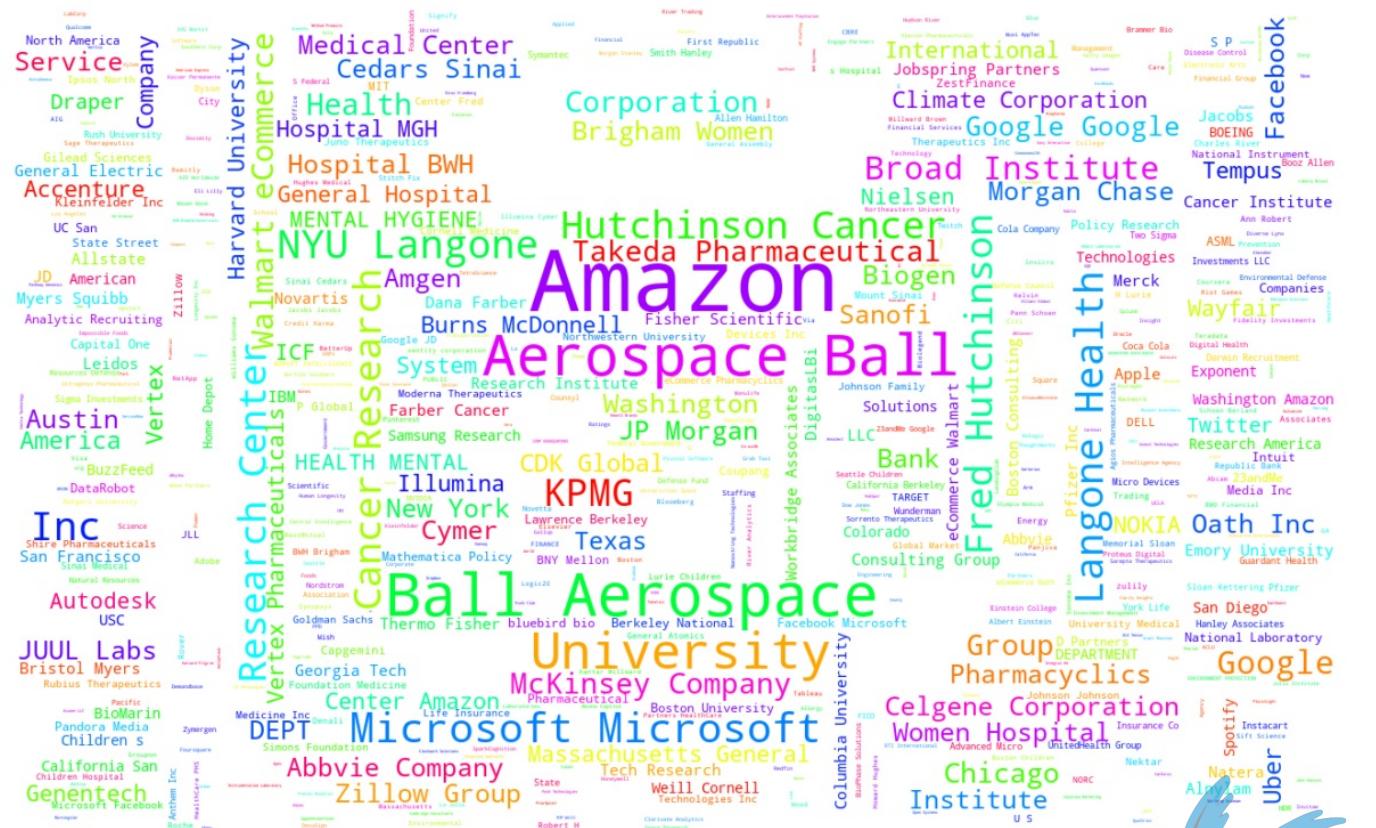


```
In [116]: from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
from wordcloud import WordCloud, STOPWORDS
import pandas as pd
import requests
from io import BytesIO

# Download mask image
mask_url = "https://cdn.pixabay.com/photo/2013/07/12/17/47/test-pattern-152459_1280.png"
response = requests.get(mask_url)
mask_image = Image.open(BytesIO(response.content))
wordcloud_mask = np.array(mask_image)

# Generate word cloud
plt.figure(figsize=(15,15))
all_text = " ".join(df1['company'].values.tolist())
wordcloud = WordCloud(width=800,
                      height=800,
                      stopwords=STOPWORDS,
                      background_color='white',
                      max_words=800,
                      colormap="hsv",
                      mask=wordcloud_mask).generate(all_text)

# Display the word cloud
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



```
In [117]: from wordcloud import WordCloud  
  
# create a word cloud for positive reviews  
positive_reviews = df1[df1['location'] == 'Atlanta, GA']['company'].str.cat(sep=' ')
```



```

positive_cloud = WordCloud(width=1500, height=800, max_words=100, background_color='white').generate(positive_r
plt.figure(figsize=(20, 6), facecolor=None)
plt.imshow(positive_cloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()

```

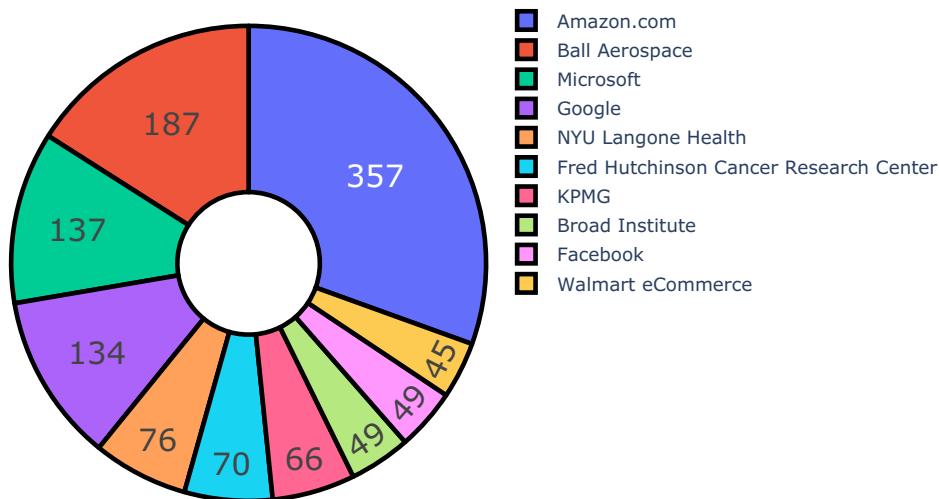


```

In [118]: import plotly.graph_objs as go
values = df1['company'].value_counts()[:10]
labels=values.index
text=values.index
fig = go.Figure(data=[go.Pie(values=values, labels=labels, hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=20,
                   marker=dict(line=dict(color='#000000', width=3)))
fig.update_layout(title="Most popular Jobs in USA",
                  titlefont={'size': 30},
                  )
fig.show()

```

Most popular Jobs in USA



```

In [119]: print("Count of unique Jobs in USA")
locationCount=df1['company'].value_counts().head(10).sort_values(ascending=True)
locationCount

```



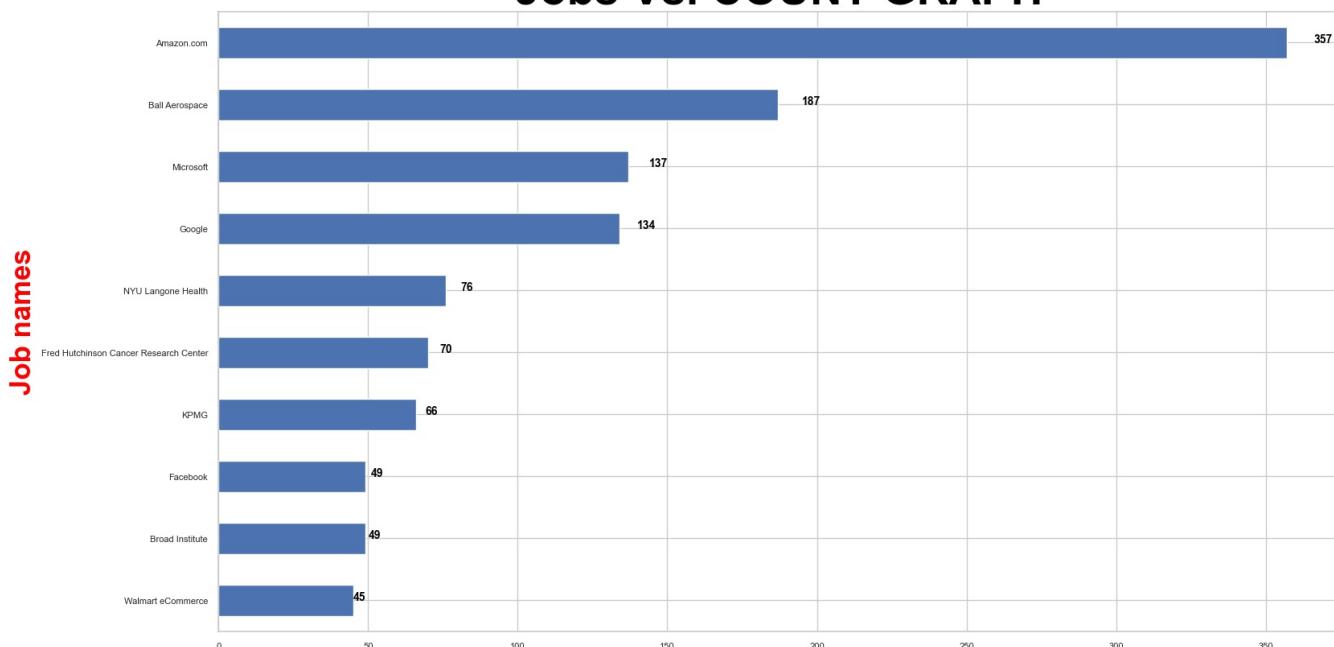
```

fig=plt.figure(figsize=(18,10))
locationCount.plot(kind="barh", fontsize=8)
plt.ylabel("Job names", fontsize=25, color="red", fontweight='bold')
plt.title("Jobs Vs. COUNT GRAPH", fontsize=40, color="BLACK", fontweight='bold')
for v in range(len(locationCount)):
    plt.text(v+locationCount[v],v,locationCount[v],fontsize=10,color="BLACK",fontweight='bold')

```

Count of unique Jobs in USA

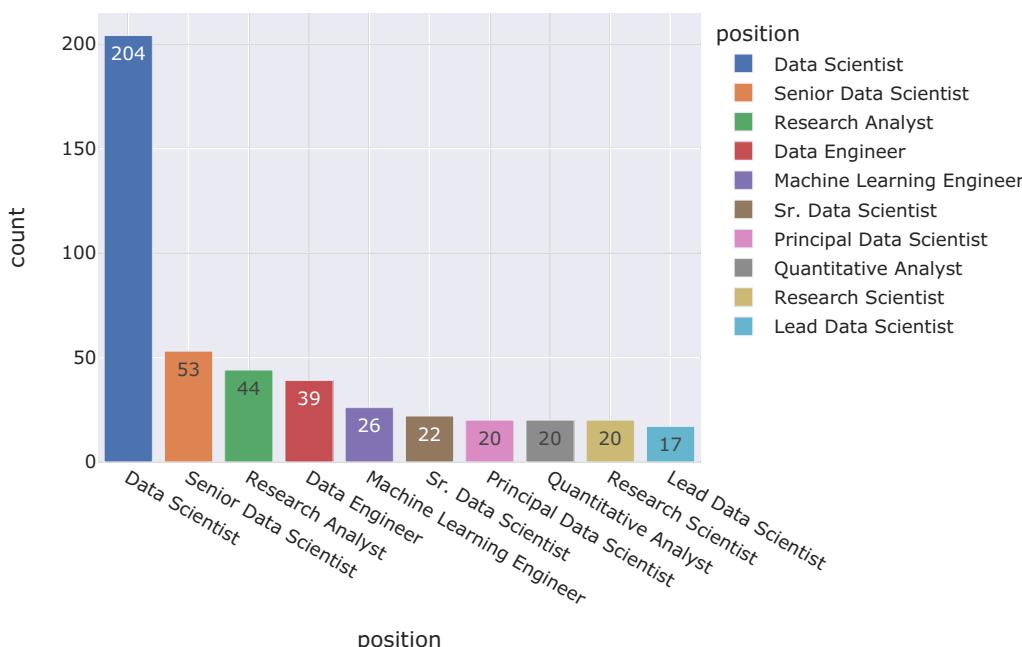
Jobs Vs. COUNT GRAPH



```
In [120]: z=df1['position'].value_counts().head(10)
fig=px.bar(z,x=z.index,y=z.values,color=z.index, text=z.values, labels={'index':'job title', 'y':'count', 'text':'count'})
fig.show()
```



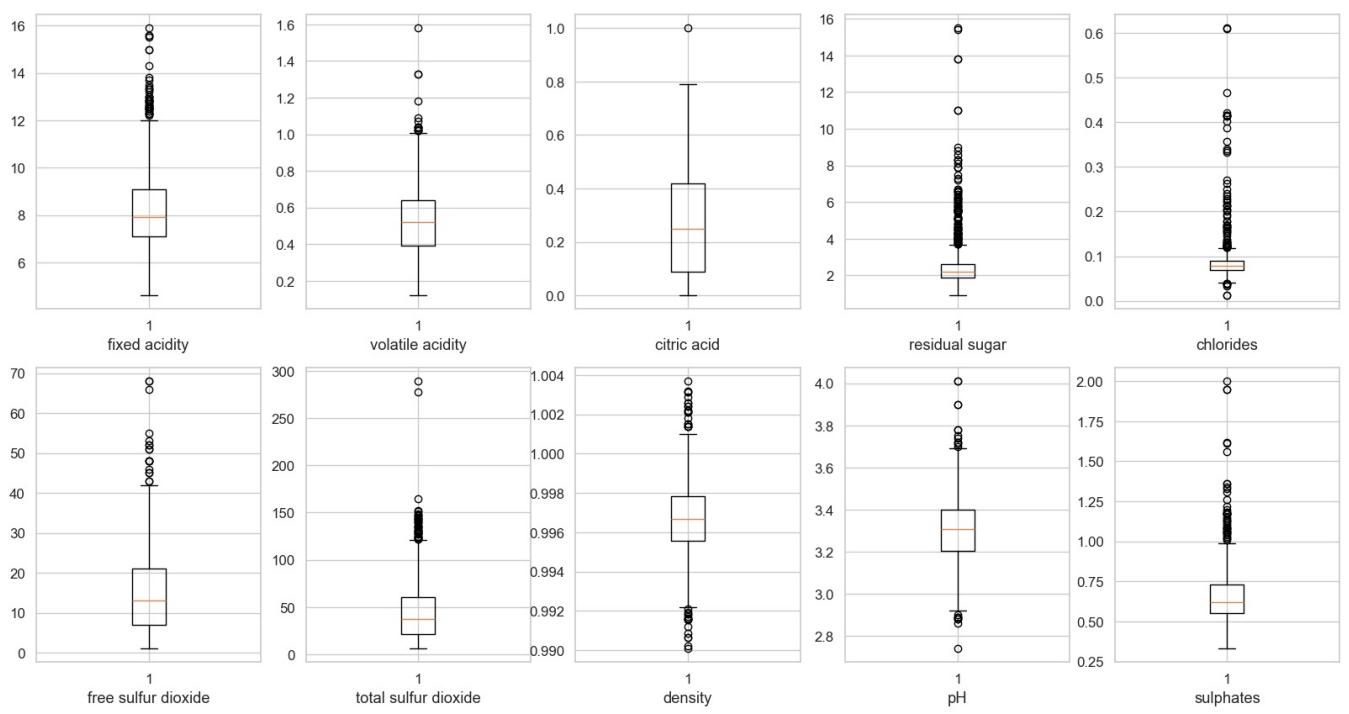
Top 10 Popular Roles in Data Science



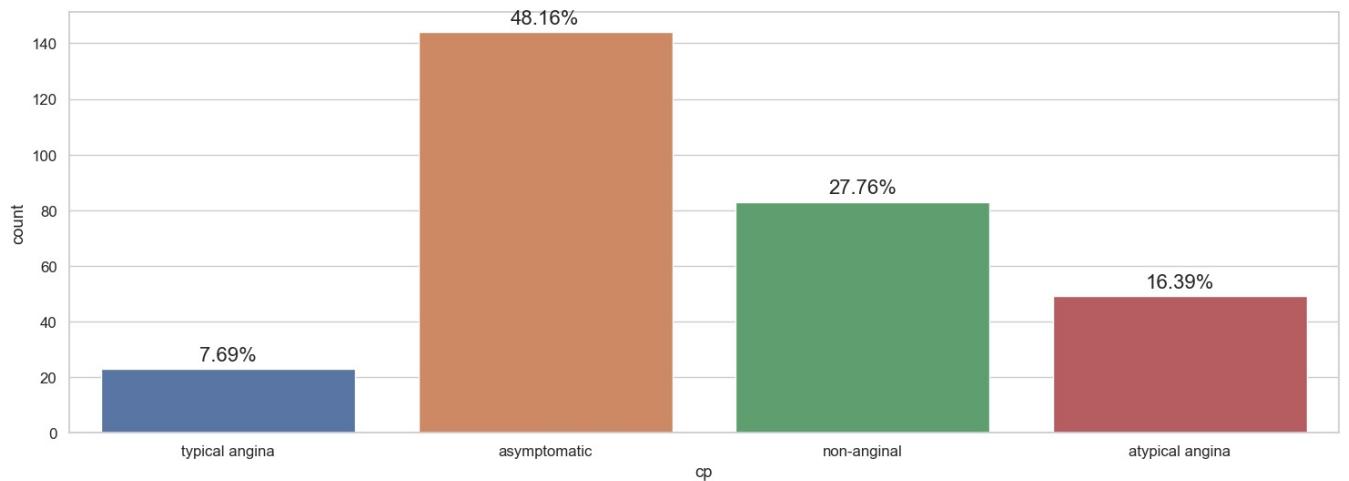
```

In [121]: # Plotting Outliers
col = 1
plt.figure(figsize = (20, 10))
for i in wine.columns:
    if col < 11:
        plt.subplot(2, 5, col)
        plt.boxplot(wine[i])
        plt.xlabel(i)
    col = col + 1

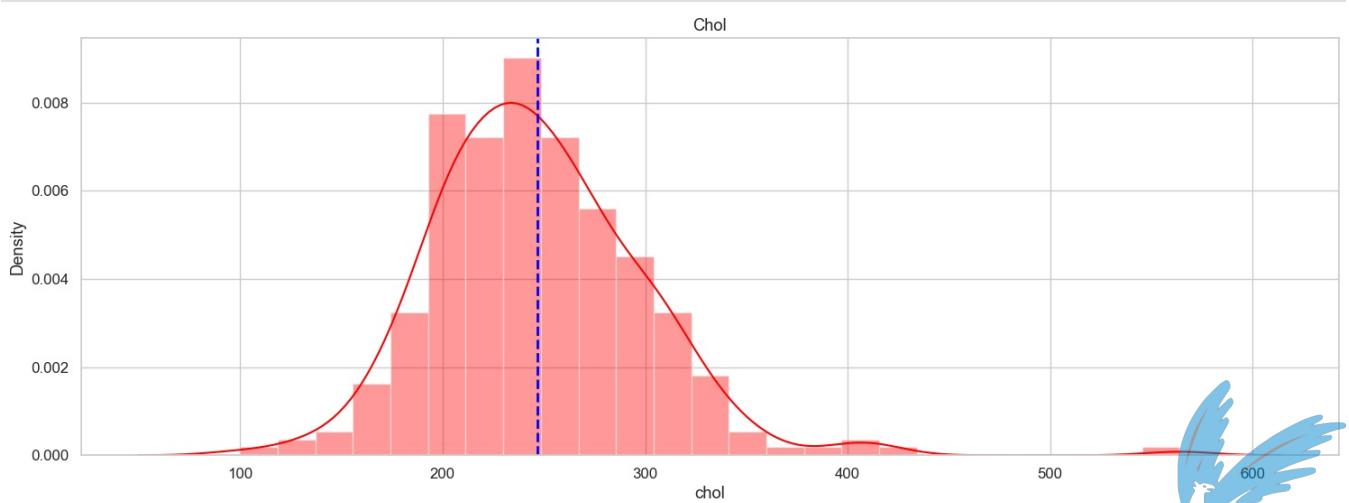
```



```
In [122]: s = sns.countplot(x = 'cp', data = df)
sizes = []
for p in s.patches:
    height = p.get_height()
    sizes.append(height)
    s.text(p.get_x() + p.get_width() / 2.,
           height + 3,
           '{1.2f}%'.format(height / len(df) * 100),
           ha="center", fontsize=16)
```



```
In [123]: #checking the target variables for distribution
sns.distplot(df['chol'], color='Red')
plt.axvline(x=df['chol'].mean(), color='Blue', linestyle='--', linewidth=2)
plt.title('Chol');
```



```
In [124]: wine.iloc[:, :-1].describe().T.sort_values(by='std', ascending = False)\n        .style.background_gradient(cmap='GnBu')\n\n
```

```
.bar(subset=["max"], color="#BB0000")\n.bar(subset=["mean",], color='green')
```

Out[124]:

	count	mean	std	min	25%	50%	75%	max
total sulfur dioxide	1143.000000	45.914698	32.782130	6.000000	21.000000	37.000000	61.000000	289.000000
free sulfur dioxide	1143.000000		10.250486	1.000000	7.000000	13.000000	21.000000	
fixed acidity	1143.000000		1.747595	4.600000	7.100000	7.900000	9.100000	
residual sugar	1143.000000		1.355917	0.900000	1.900000	2.200000	2.600000	
alcohol	1143.000000		1.082196	8.400000	9.500000	10.200000	11.100000	
quality	1143.000000		0.805824	3.000000	5.000000	6.000000	6.000000	
citric acid	1143.000000		0.196686	0.000000	0.090000	0.250000	0.420000	
volatile acidity	1143.000000		0.179633	0.120000	0.392500	0.520000	0.640000	
sulphates	1143.000000		0.170399	0.330000	0.550000	0.620000	0.730000	
pH	1143.000000		0.156664	2.740000	3.205000	3.310000	3.400000	
chlorides	1143.000000		0.047267	0.012000	0.070000	0.079000	0.090000	
density	1143.000000		0.001925	0.990070	0.995570	0.996680	0.997845	

In [125]: df[df["age"] >= 50].describe().style.background_gradient(cmap='RdPu')

Out[125]:

	age	trestbps	chol	thalch	oldpeak	ca	num
count	213.000000	213.000000	213.000000	213.000000	213.000000	213.000000	213.000000
mean	59.159624	134.793427	252.220657	144.708920	1.214085	0.835681	1.103286
std	5.731645	18.575307	54.953695	22.377966	1.197004	0.969460	1.280713
min	50.000000	94.000000	100.000000	71.000000	0.000000	0.000000	0.000000
25%	55.000000	120.000000	214.000000	130.000000	0.100000	0.000000	0.000000
50%	58.000000	132.000000	246.000000	148.000000	1.000000	1.000000	1.000000
75%	63.000000	145.000000	283.000000	161.000000	1.800000	1.000000	2.000000
max	77.000000	200.000000	564.000000	195.000000	6.200000	3.000000	4.000000

In [126]:

```
def highlight_min(s, props=''):
    return np.where(s == np.nanmin(s.values), props, '')
df.describe().style.apply(highlight_min, props='color:yellow;background-color:Grey', axis=0)
```

Out[126]:

	age	trestbps	chol	thalch	oldpeak	ca	num
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000
mean	54.521739	131.715719	246.785953	149.327759	1.058528	0.672241	0.946488
std	9.030264	17.747751	52.532582	23.121062	1.162769	0.937438	1.230409
min	29.000000	94.000000	100.000000	71.000000	0.000000	0.000000	0.000000
25%	48.000000	120.000000	211.000000	132.500000	0.000000	0.000000	0.000000
50%	56.000000	130.000000	242.000000	152.000000	0.800000	0.000000	0.000000
75%	61.000000	140.000000	275.500000	165.500000	1.600000	1.000000	2.000000
max	77.000000	200.000000	564.000000	202.000000	6.200000	3.000000	4.000000

Prepared By: Syed Afroz Ali

In []:

