Prepared By: Syed Afroz Ali (Kaggle Grand Master)

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]:

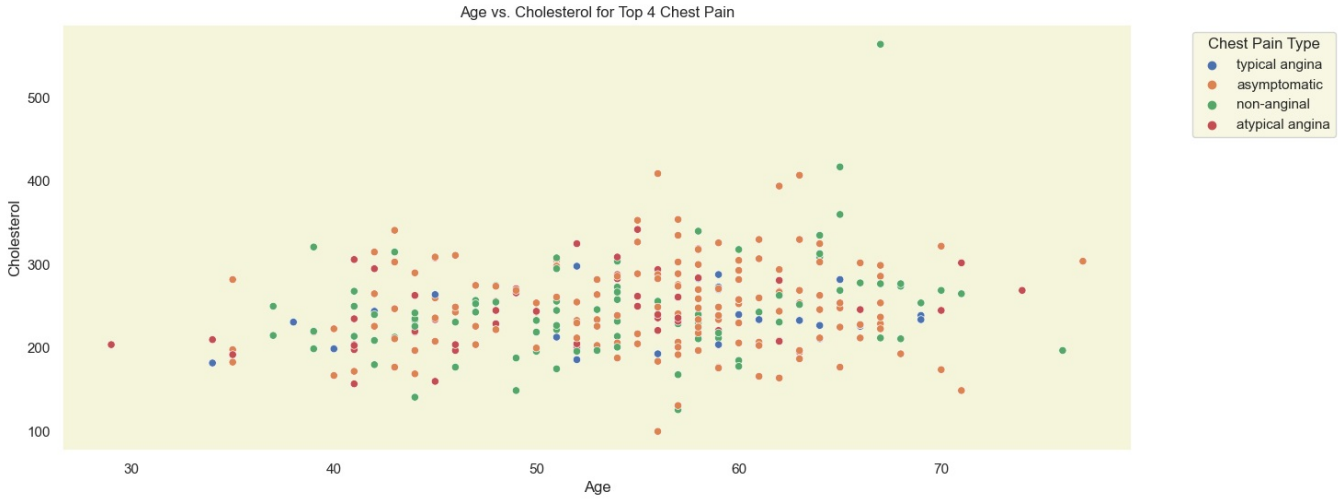
	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal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

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

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

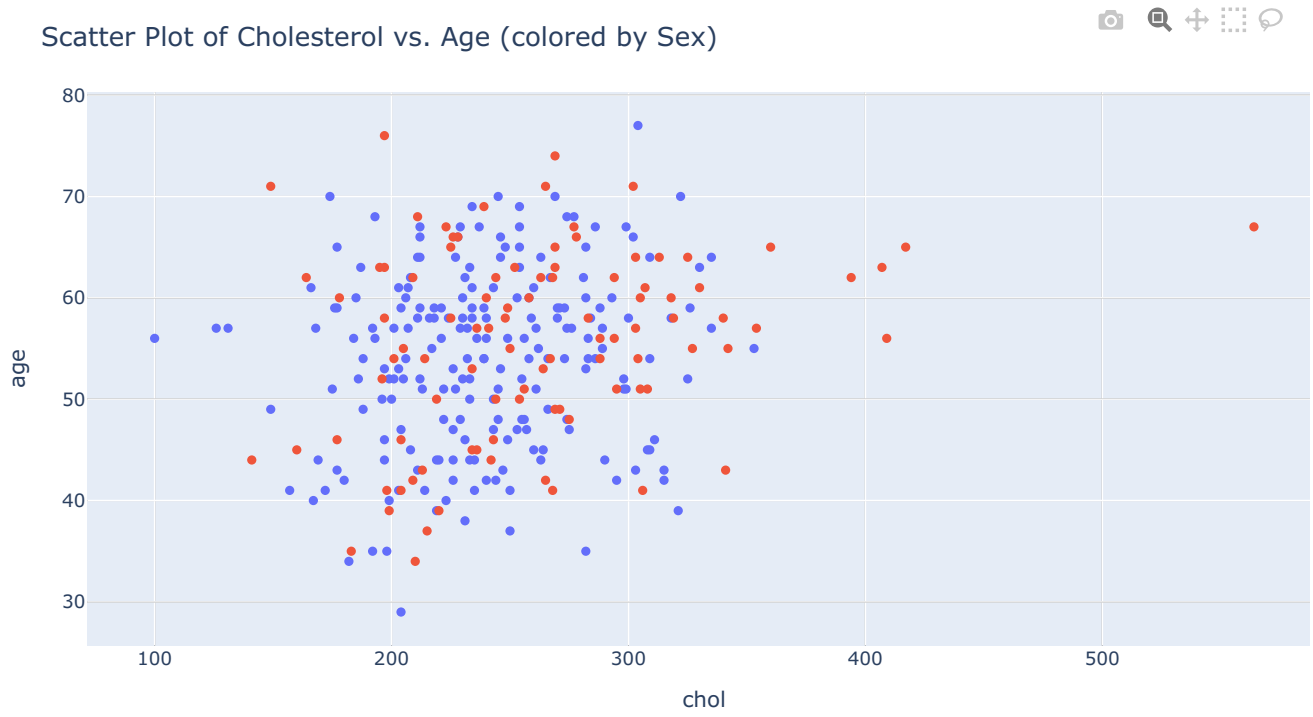
In [4]:
```
top_leagues = df['cp'].value_counts().nlargest(4).index
display(top_leagues)

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')
```

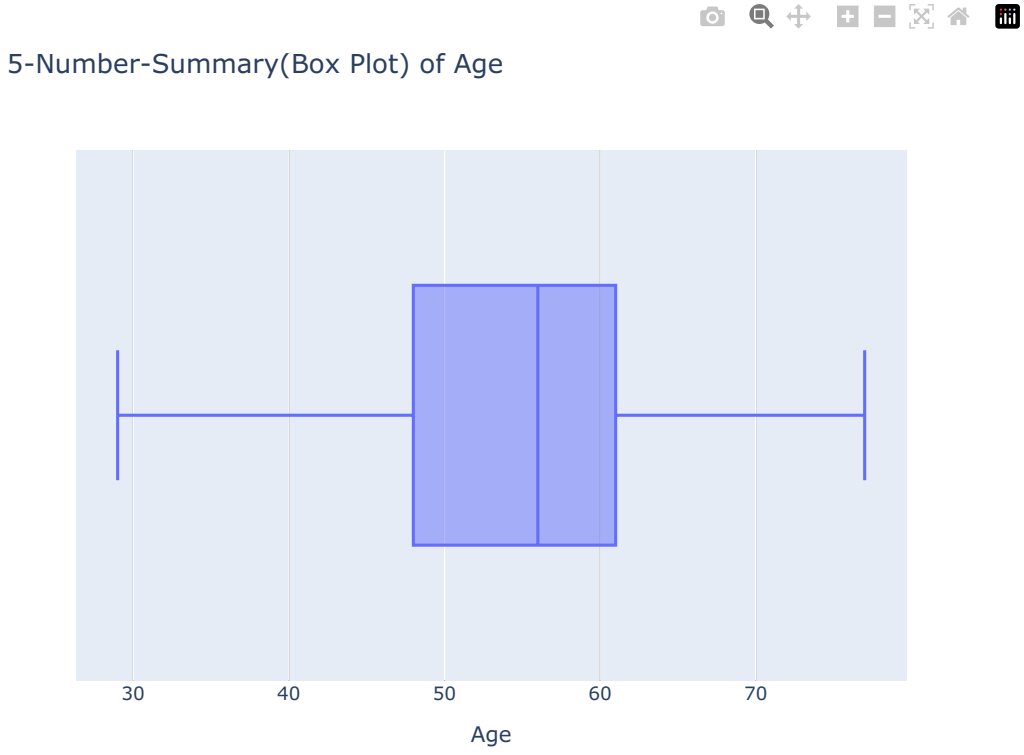


In [130…
```
import plotly.express as px


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()
```

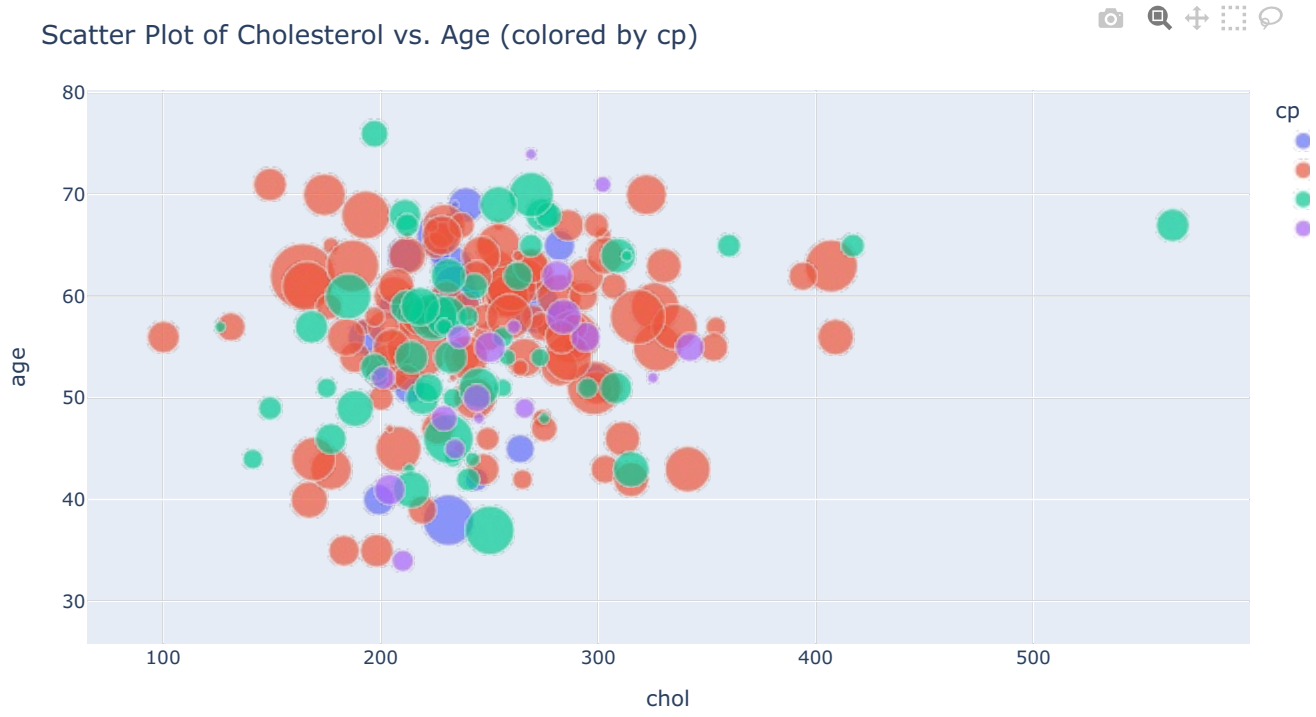


In [6]:
```
from plotly.offline import iplot

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



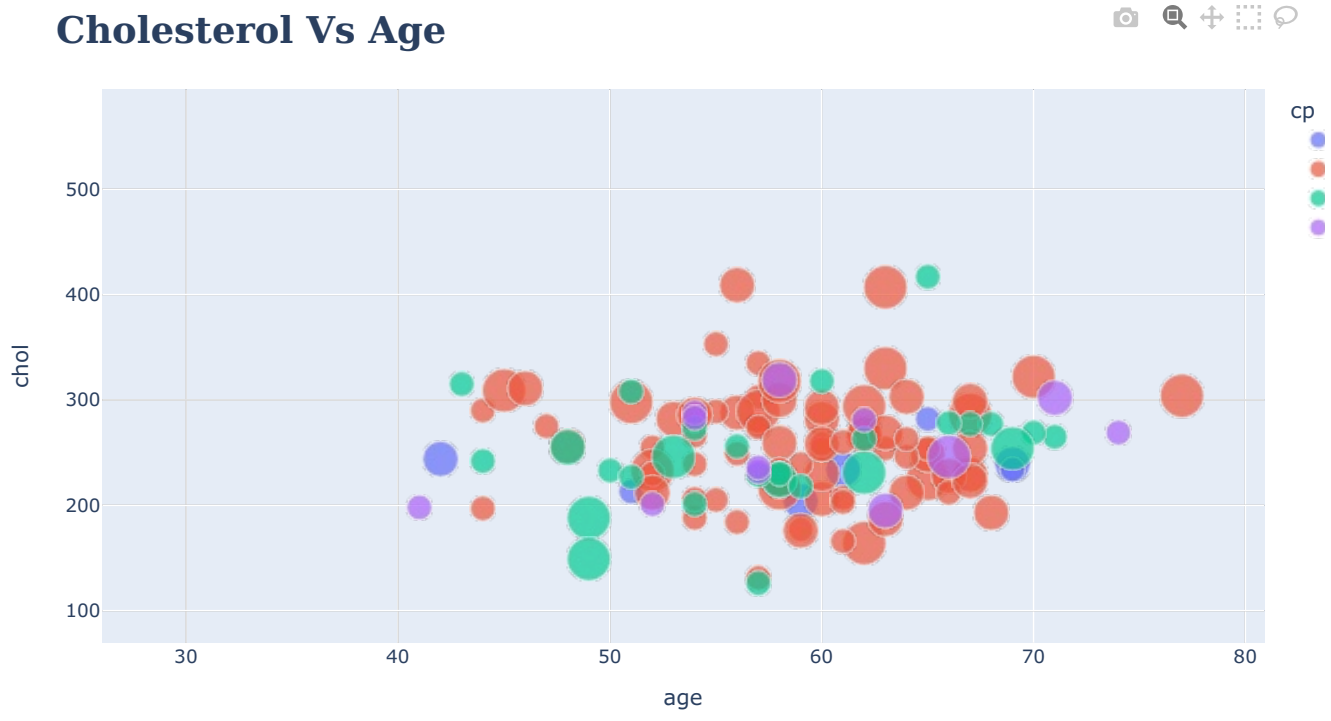
In [131…
```
import plotly.express as px


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()
```



In [132…

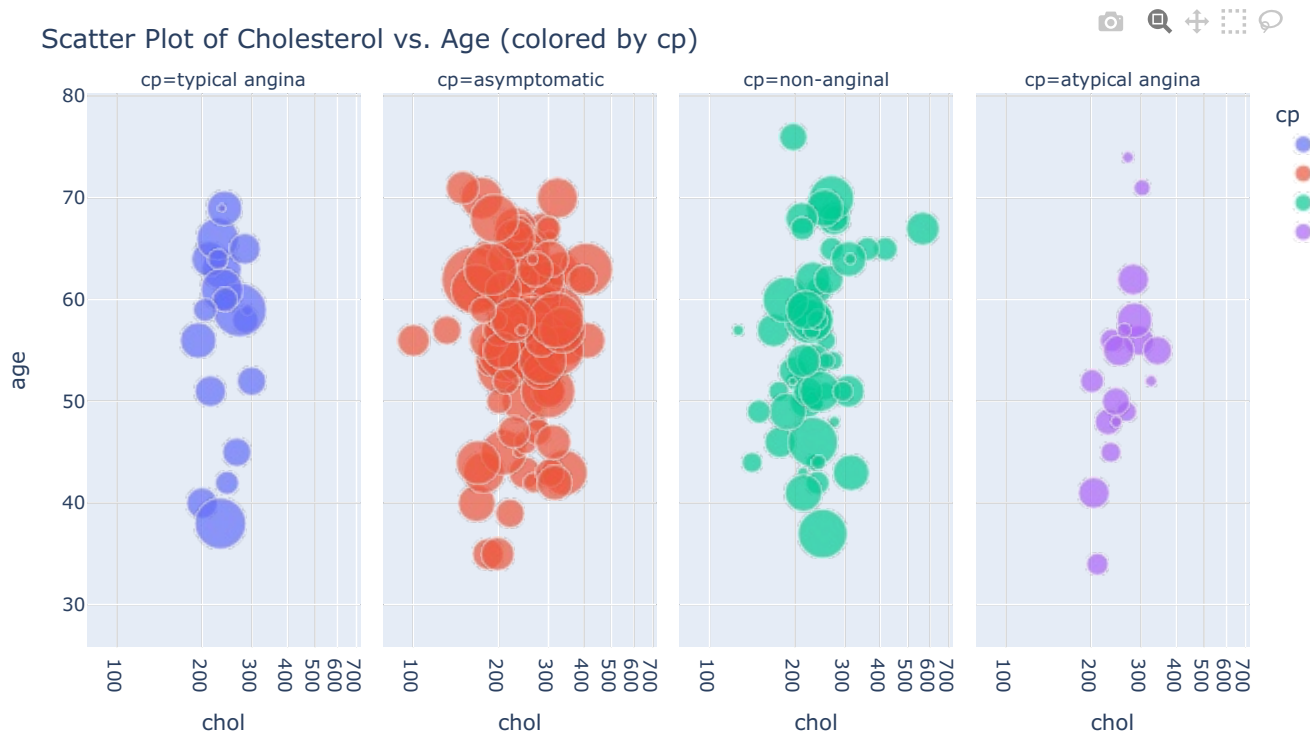
```
fig = px.scatter(data_frame = df,
                 x="age",
                 y="chol",
                 color="cp",
                 size='ca',
                 hover_data=['oldpeak'])

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

fig.show()
```



In [133…

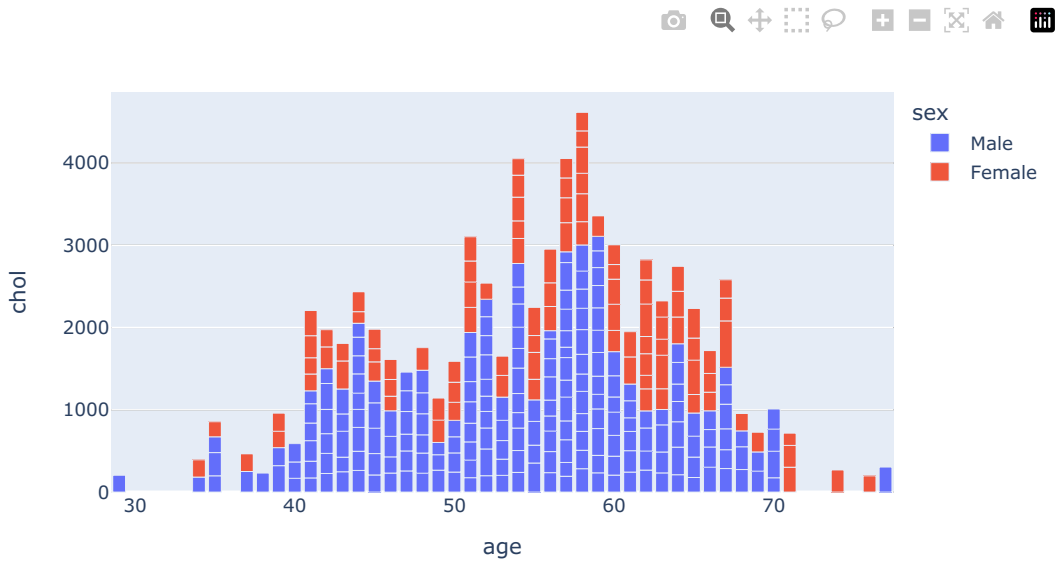
```
import plotly.express as px


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()
```



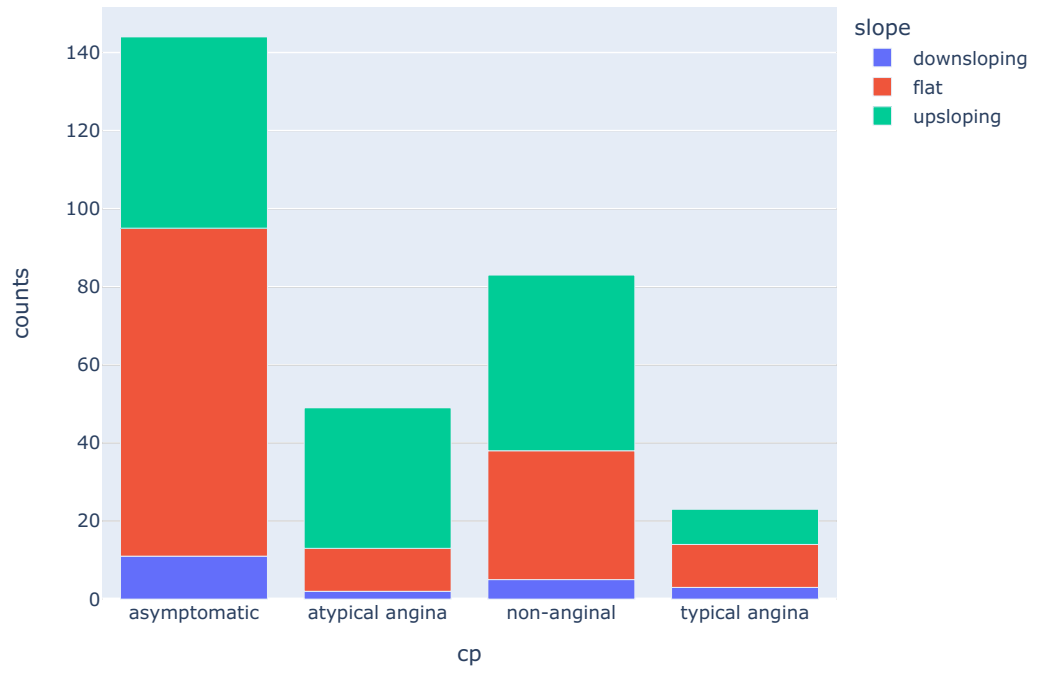
hover_name='exang' means that the values in the 'exang' column will be shown as tooltips when you hover over the data points on the scatter plot. This is useful for providing additional information about each data point without cluttering the plot with labels.

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

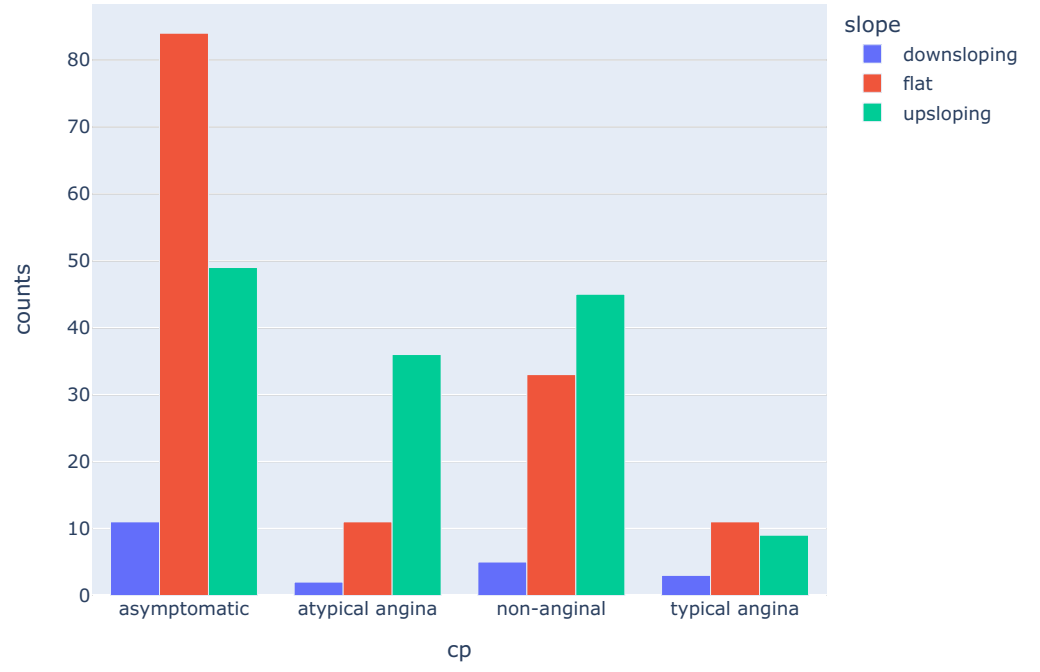


In [11]:
```
def generate_rating_df(df):
    rating_df = df.groupby(['cp', 'slope']).agg({'id': 'count'}).reset_index()
    rating_df = rating_df[rating_df['id'] != 0]
    rating_df.columns = ['cp', 'slope', 'counts']
    rating_df = rating_df.sort_values('slope')
    return rating_df

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()
```

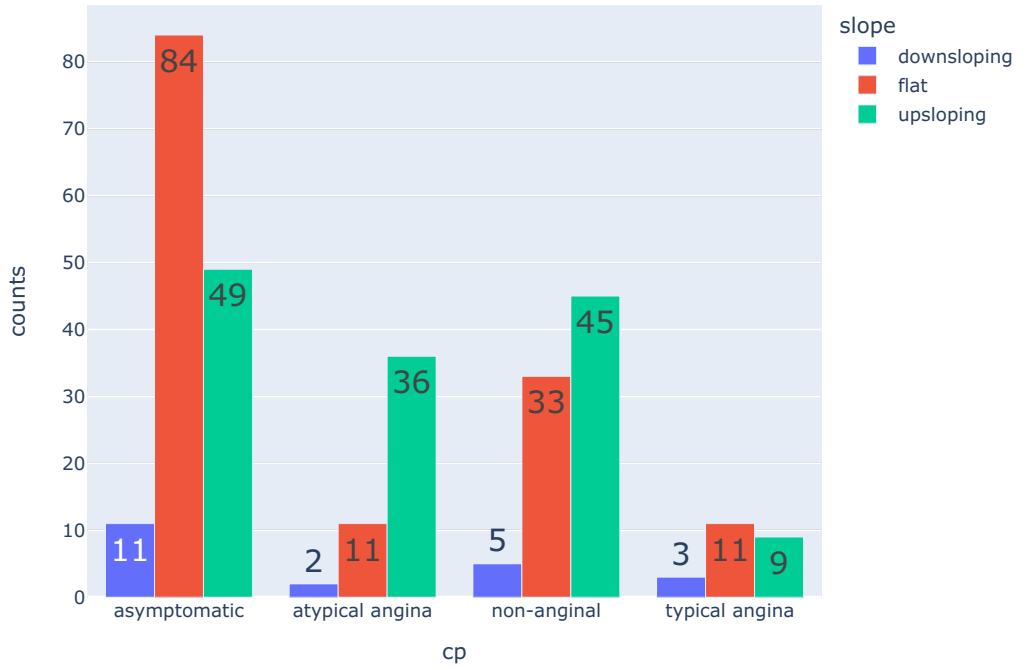


In [12]:
```
def generate_rating_df(df):
    rating_df = df.groupby(['cp', 'slope']).agg({'id': 'count'}).reset_index()
    rating_df = rating_df[rating_df['id'] != 0]
    rating_df.columns = ['cp', 'slope', 'counts']
    rating_df = rating_df.sort_values('slope')
    return rating_df

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
```

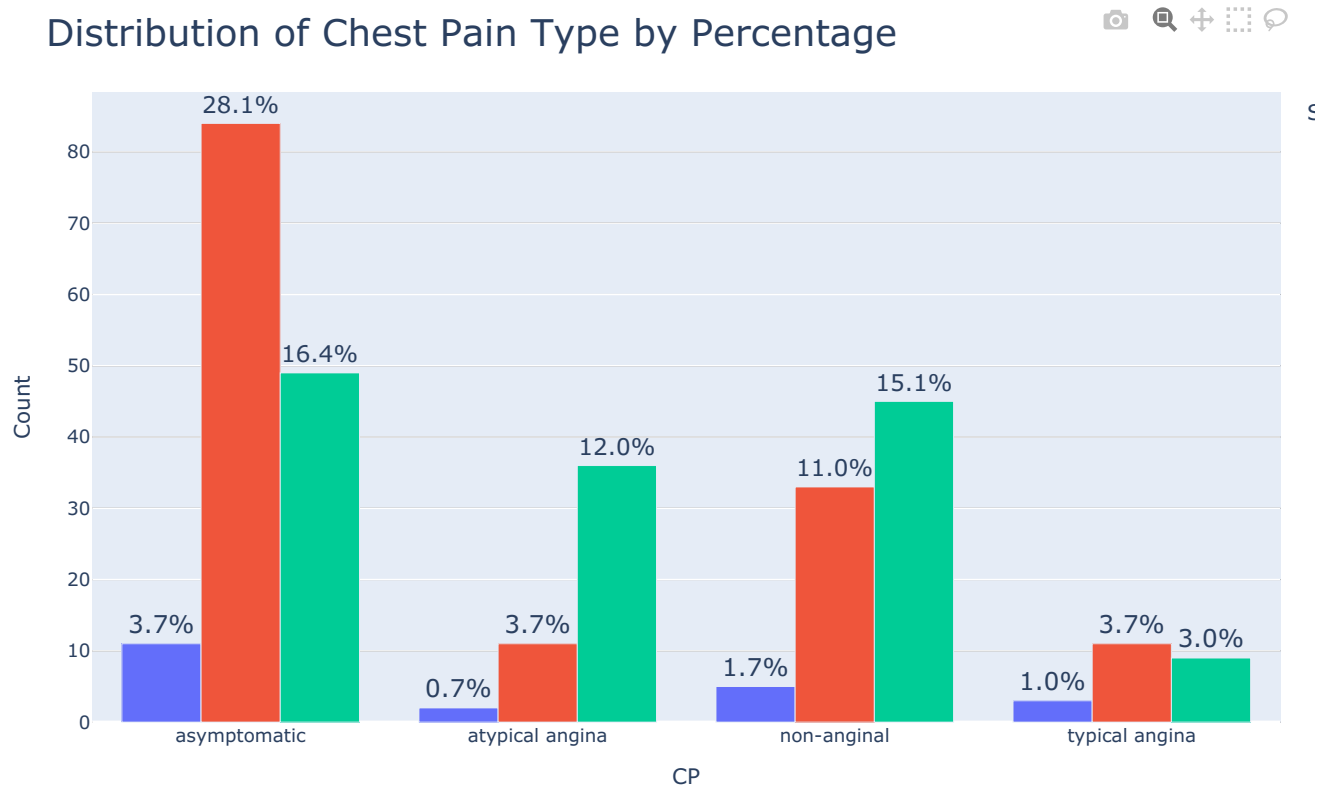
```
def generate_rating_df(df):
    rating_df = df.groupby(['cp', 'slope']).agg({'id': 'count'}).reset_index()
    rating_df = rating_df[rating_df['id'] != 0]
    rating_df.columns = ['cp', 'slope', 'counts']
    rating_df = rating_df.sort_values('slope')
    return rating_df

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]:

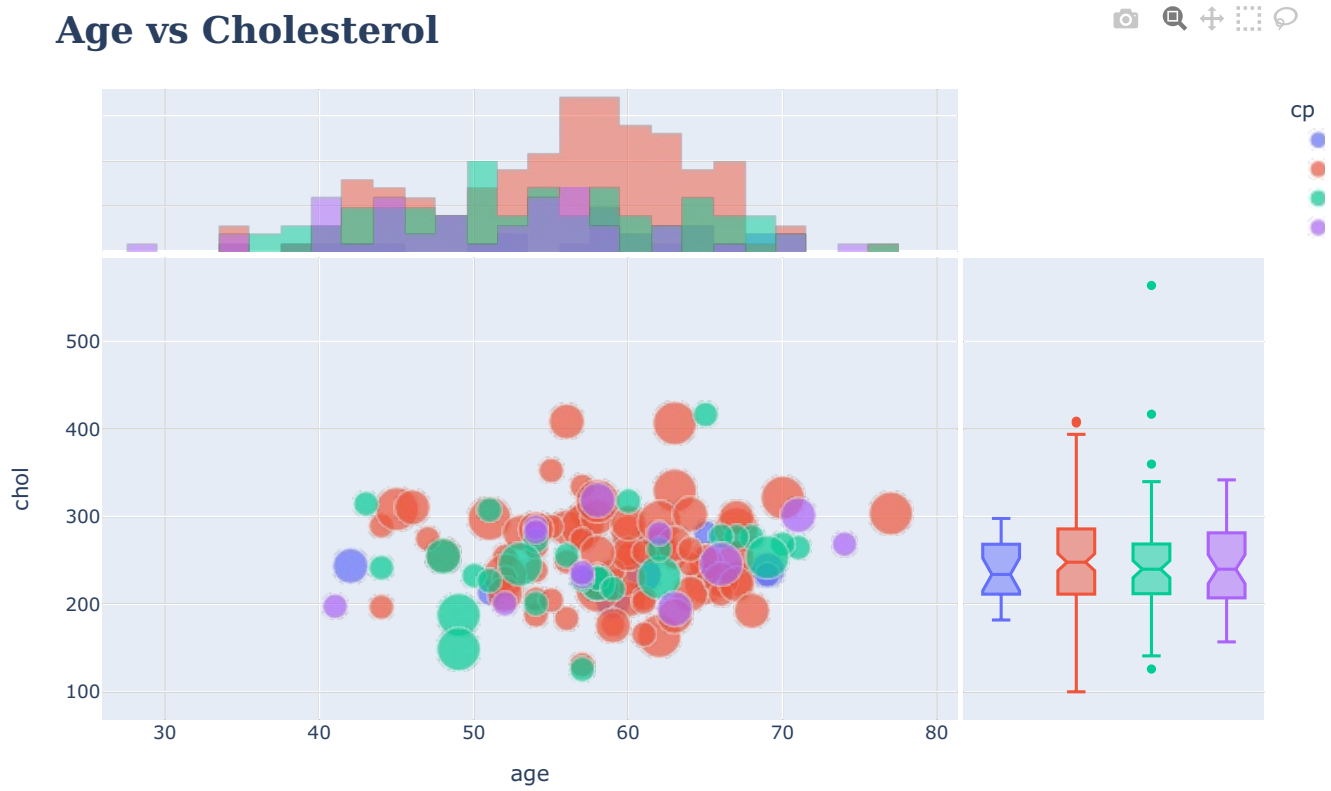
```
def generate_rating_df(df):
    rating_df = df.groupby(['cp', 'slope']).agg({'id': 'count'}).reset_index()
    rating_df = rating_df[rating_df['id'] != 0]
    rating_df.columns = ['cp', 'slope', 'counts']
    rating_df = rating_df.sort_values('slope')

    # 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='%{text:.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()
```



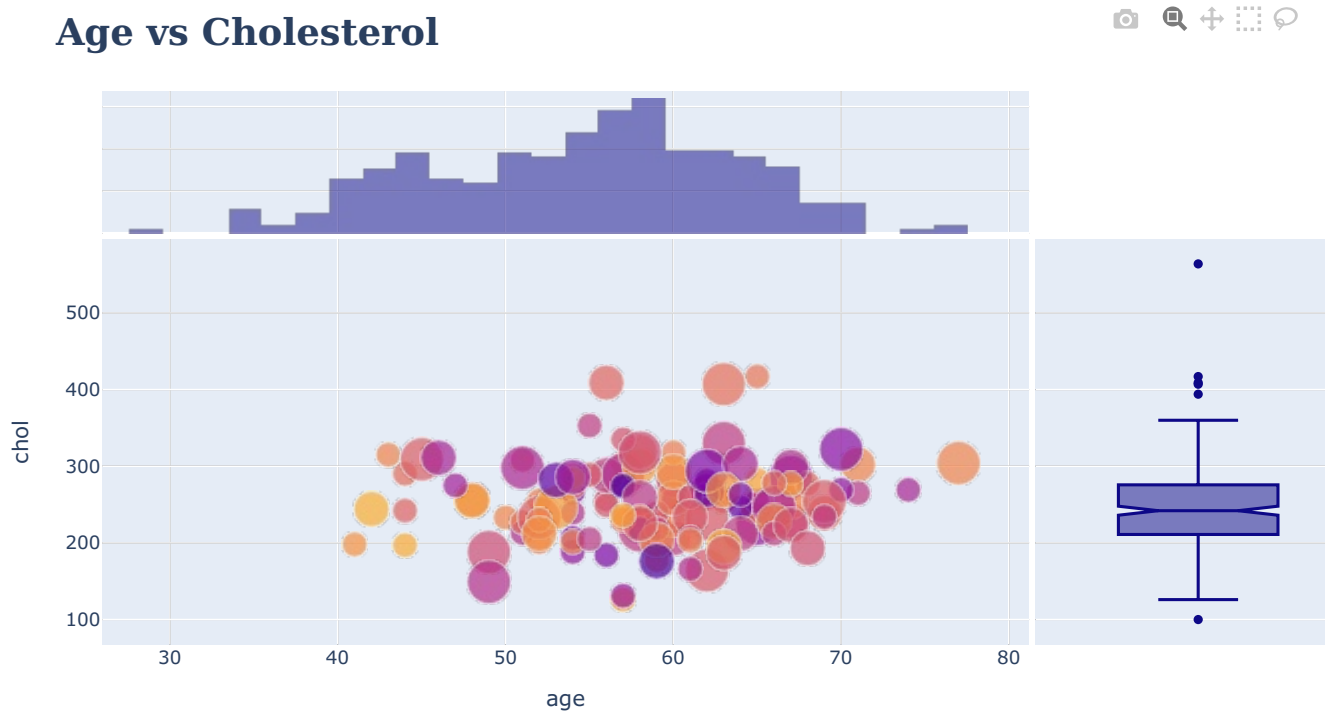
In [15]:

```
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="<b> Age vs Cholesterol <b>",
                  titlefont={'size': 24, 'family':'Serif'},
                  width=1000,
                  height=550,
                 )

fig.show()
```



In [134…

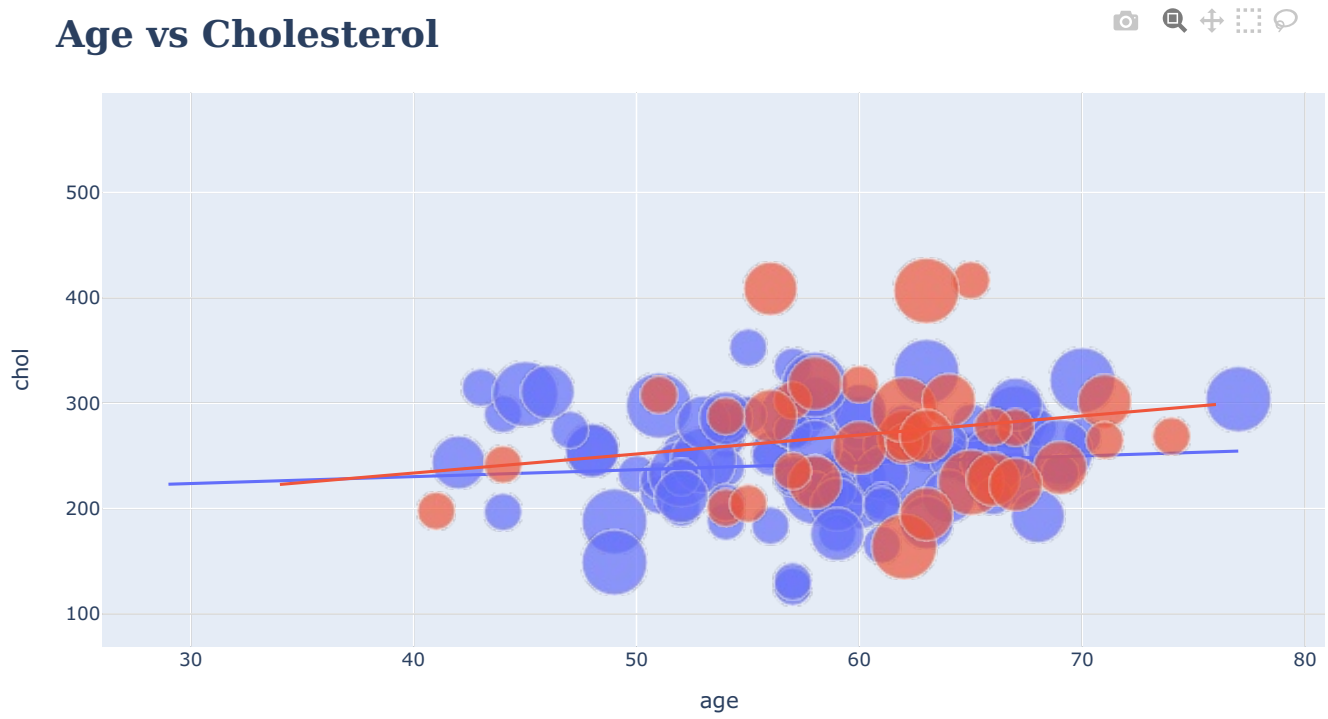
```
fig = px.scatter(data_frame = df,
                 x="age",
                 y="chol",
                 color="thalch",
                 size='ca',
                 hover_data=['oldpeak'],
                 marginal_x="histogram",
                 marginal_y="box")

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

fig.show()
```



In [135…

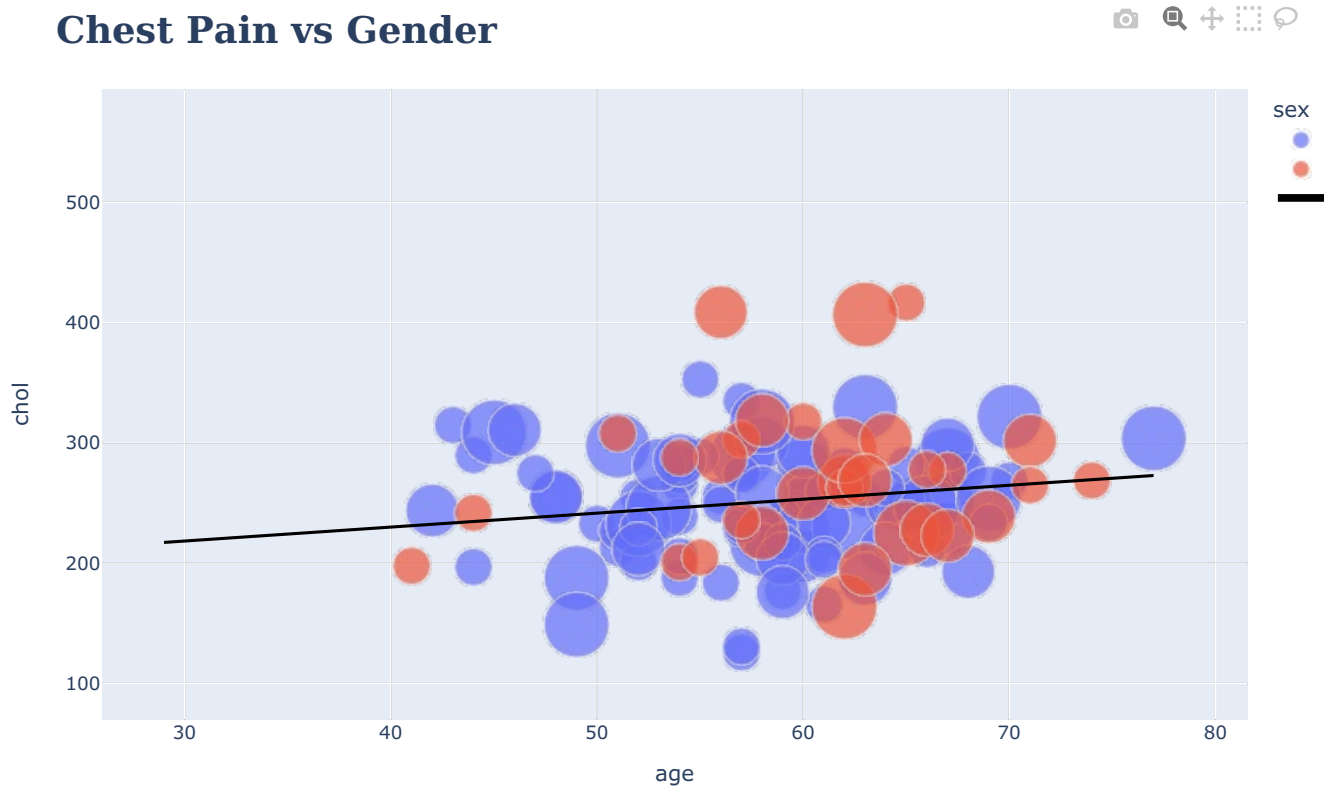
```
fig = px.scatter(data_frame = df,
                 x="age",
```

```
                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()
```

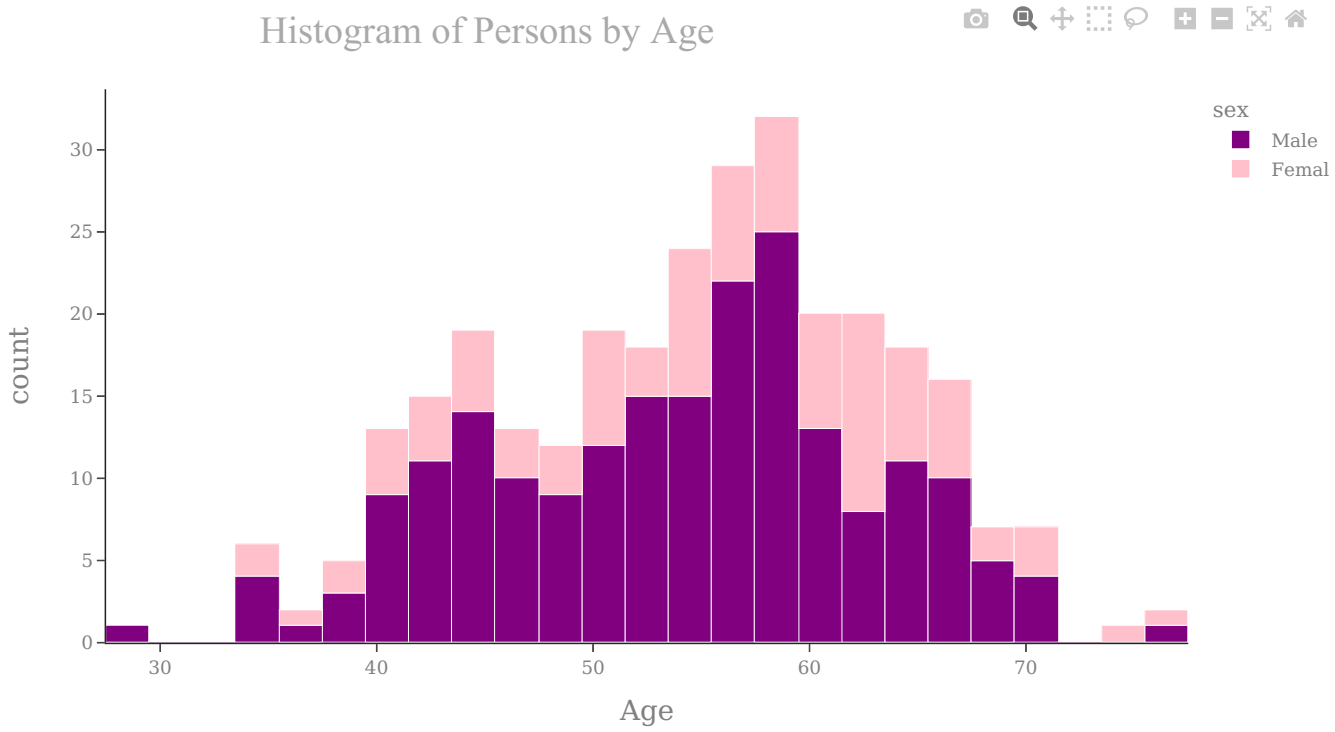


In [18]:

```
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()
```

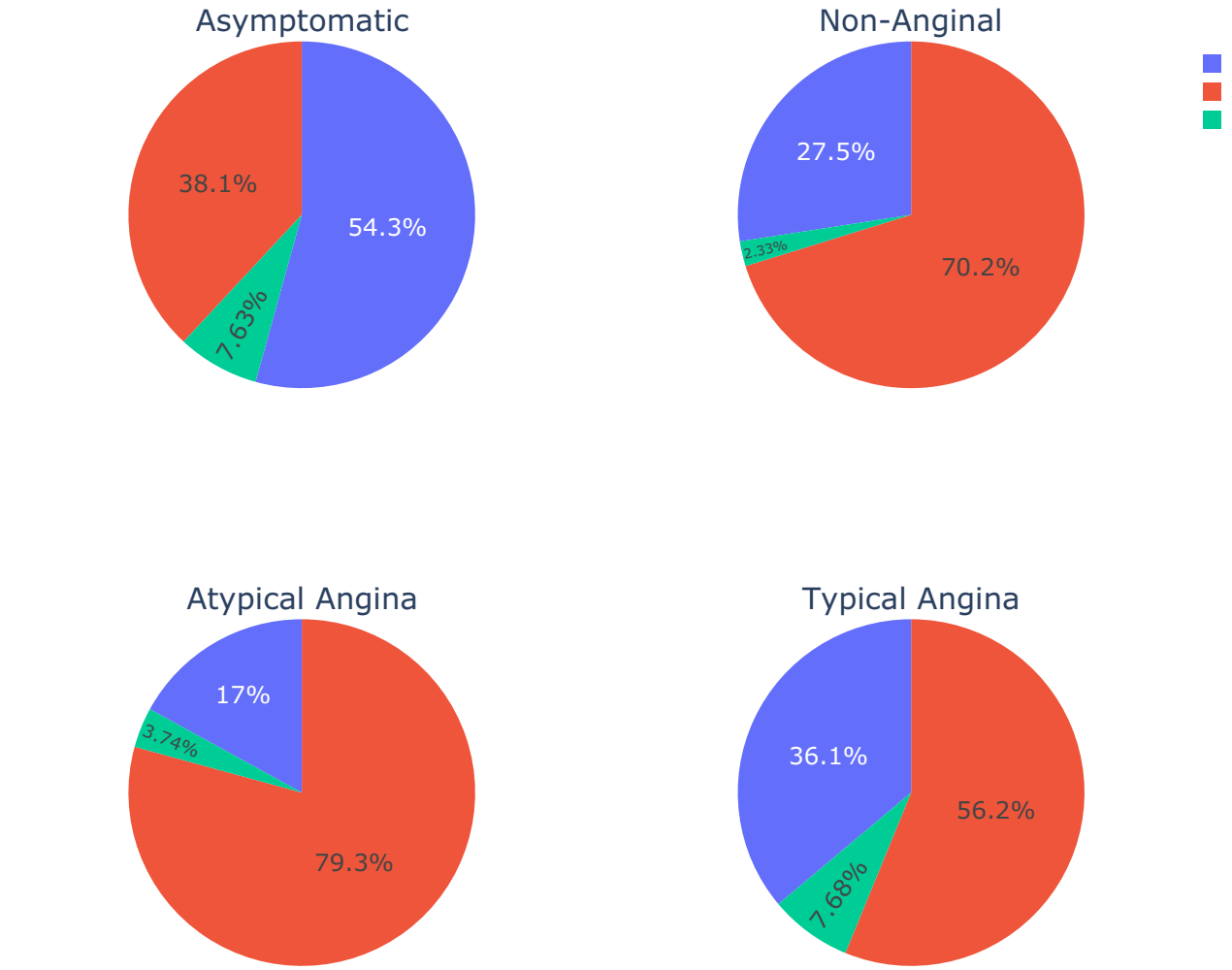


In [19]:
```
fig= px.histogram(df, x='age',height=500,width=900,template='simple_white',
                  color='sex', # adding categorical column
                  color_discrete_sequence=['purple','pink'])



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()
```

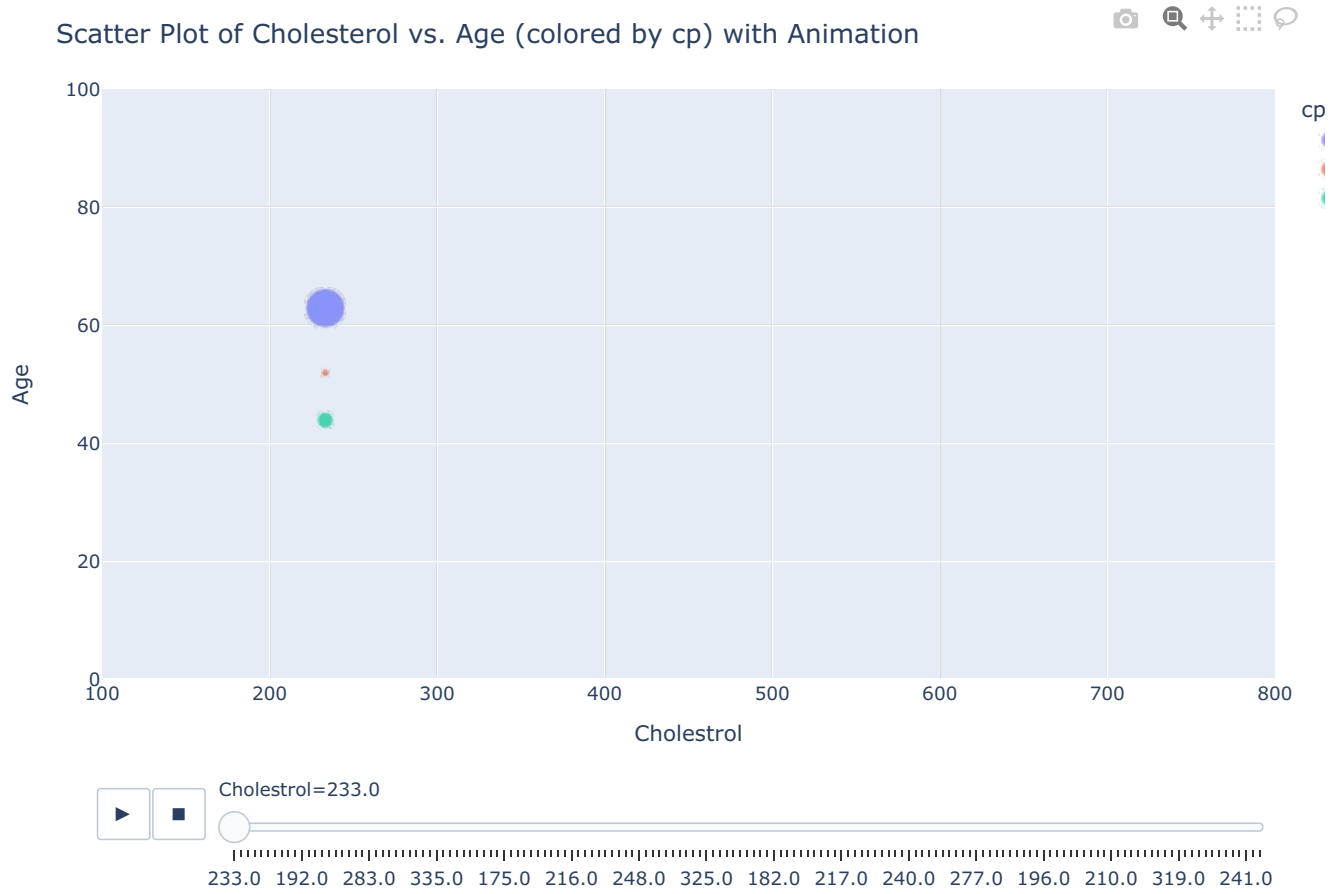


In [20]:

```
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
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()
```

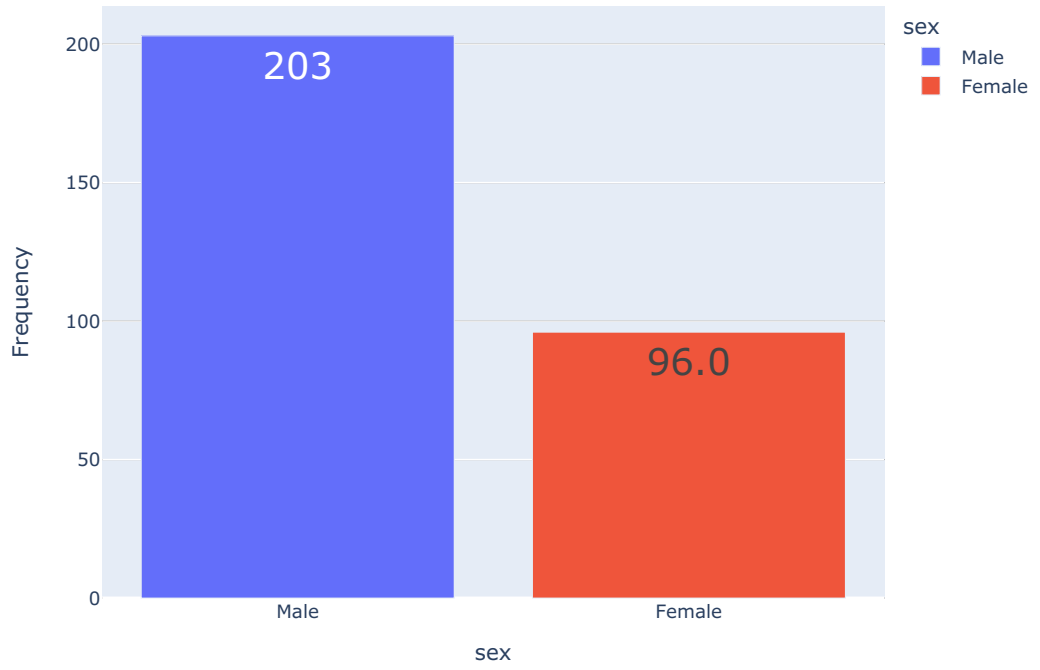


In [21]:
```
import plotly.express as px


fig = px.scatter(df, x='chol', y='age', color='cp', size = 'oldpeak', size_max = 30, hover_name = 'exang', rang
                 labels = dict(oldpeak = 'oldpeak', chol = 'Cholestrol', 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()
```



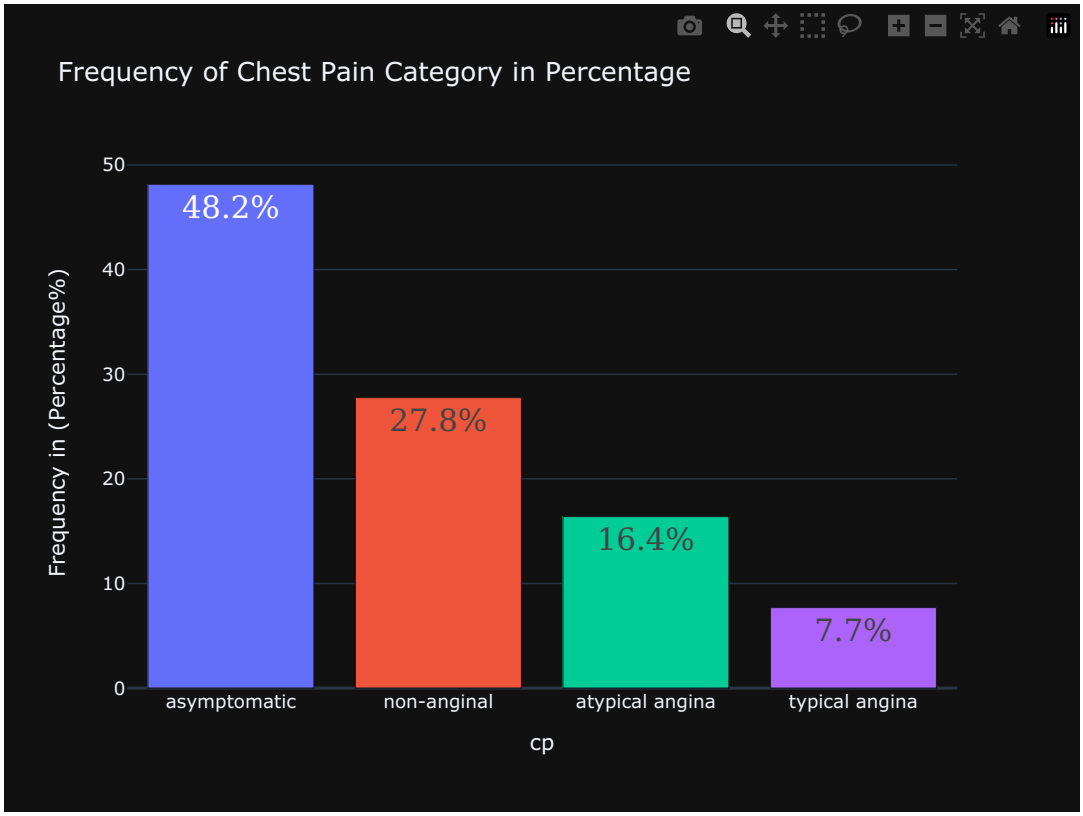
In [22]:
```
from plotly.offline import iplot

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)
```

sex	count
Male	203
Female	96

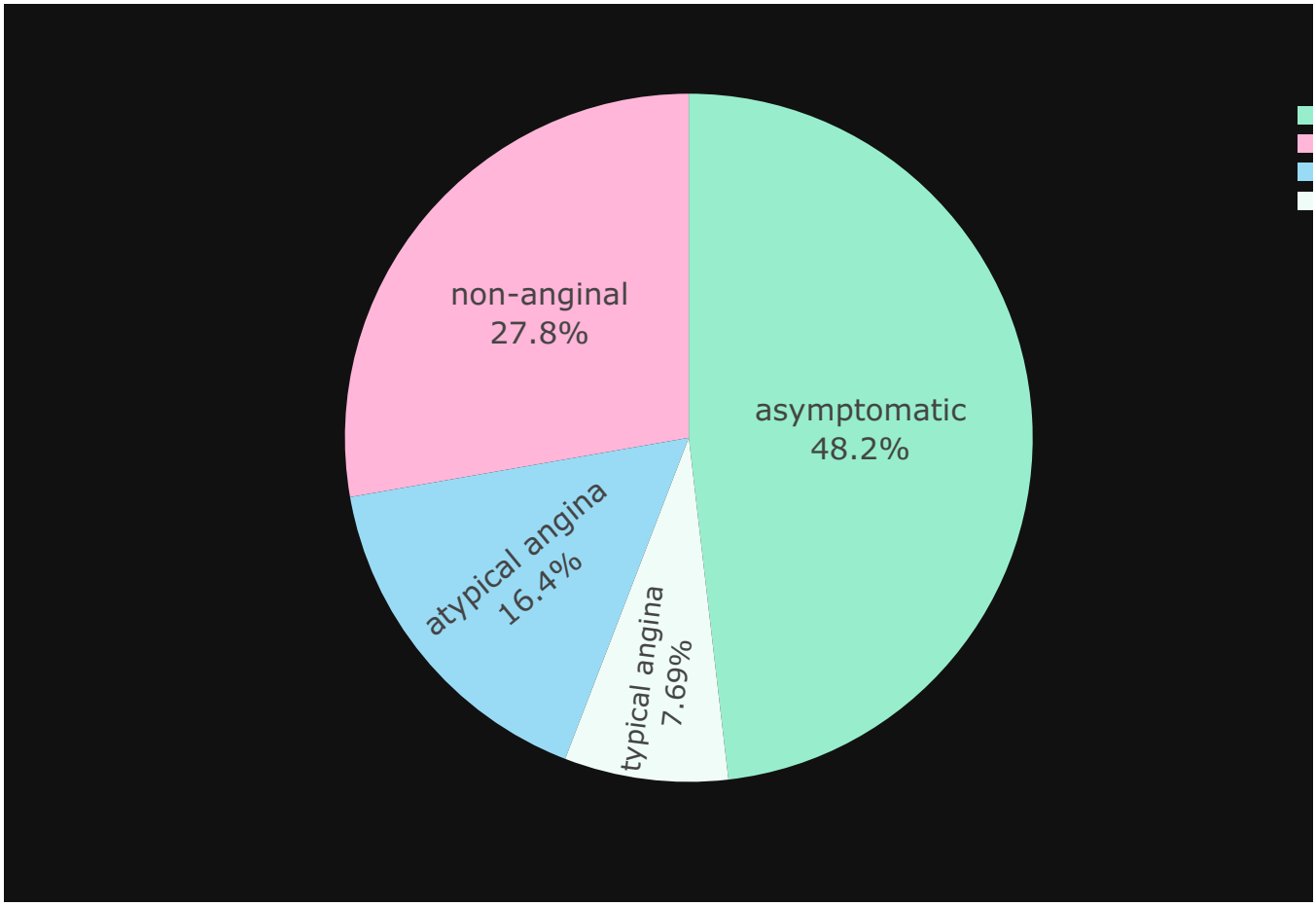


In [23]:

```
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)
```

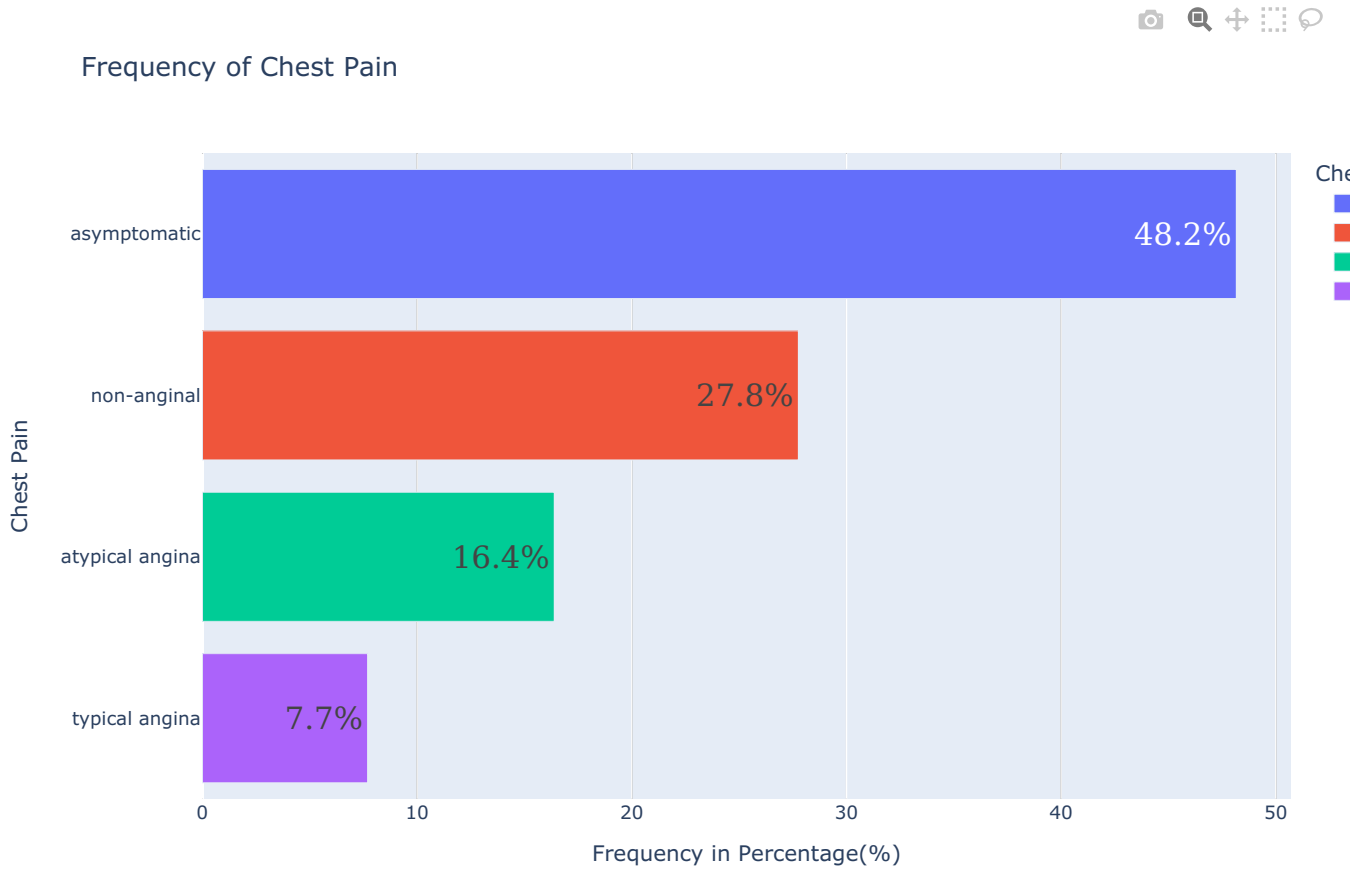


In [24]:
```
from plotly.offline import iplot

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)
```



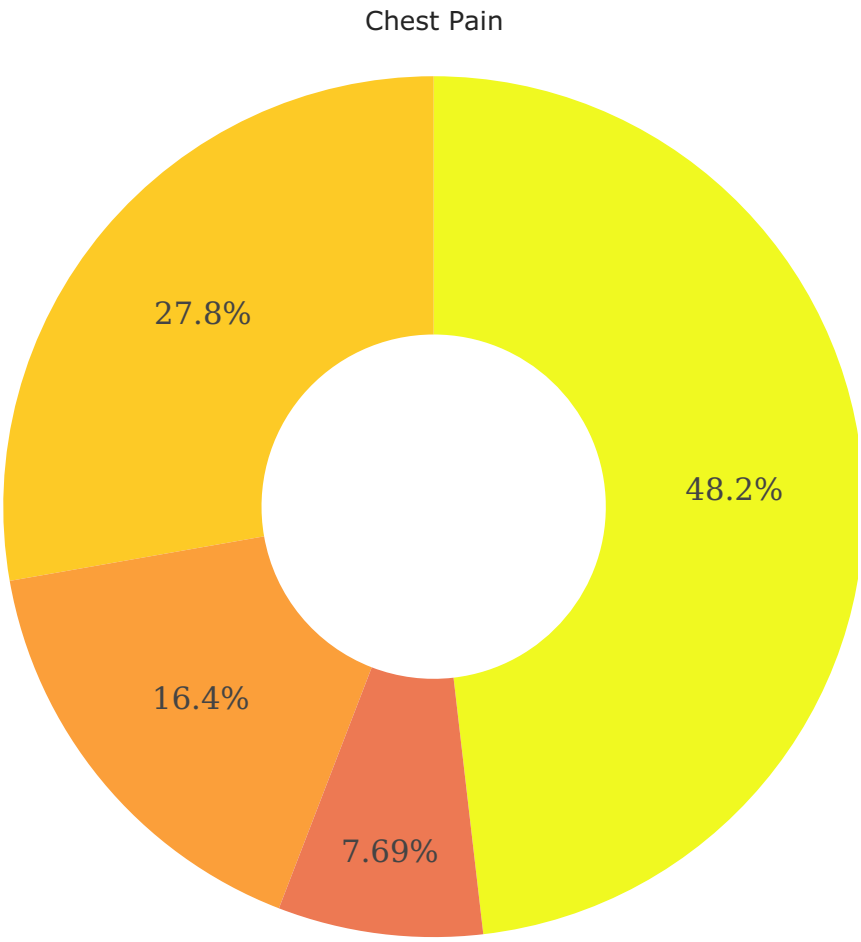
In [25]:
```
cp = df["cp"].value_counts()
```

```
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)
```



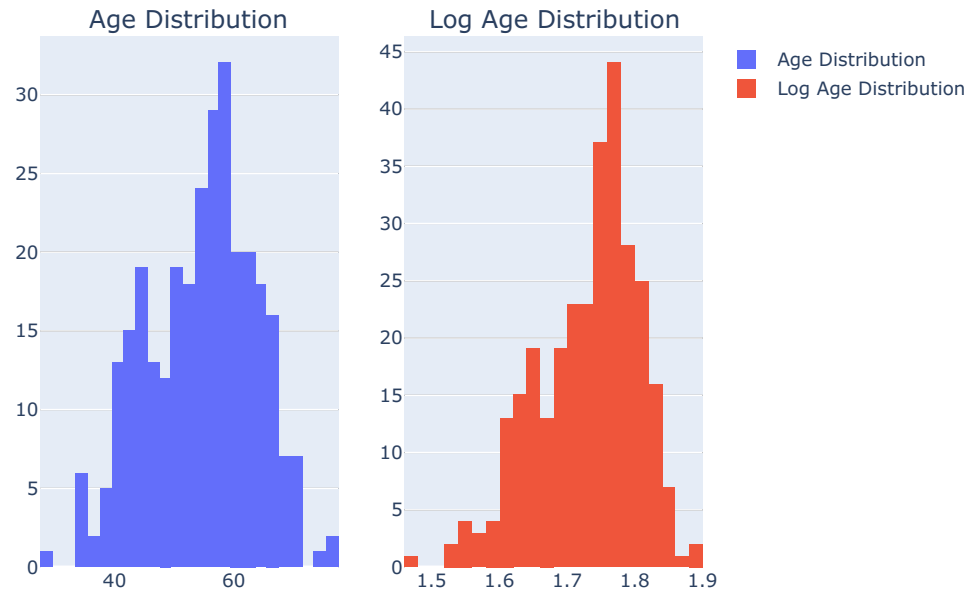
In [26]:

```
fig=px.pie(df.groupby('cp',as_index=False)['sex'].count().sort_values(by='sex',ascending=False).reset_index(dro
           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
                  title_x=0.5, showlegend=True)

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

fig.show()
```



In [27]:

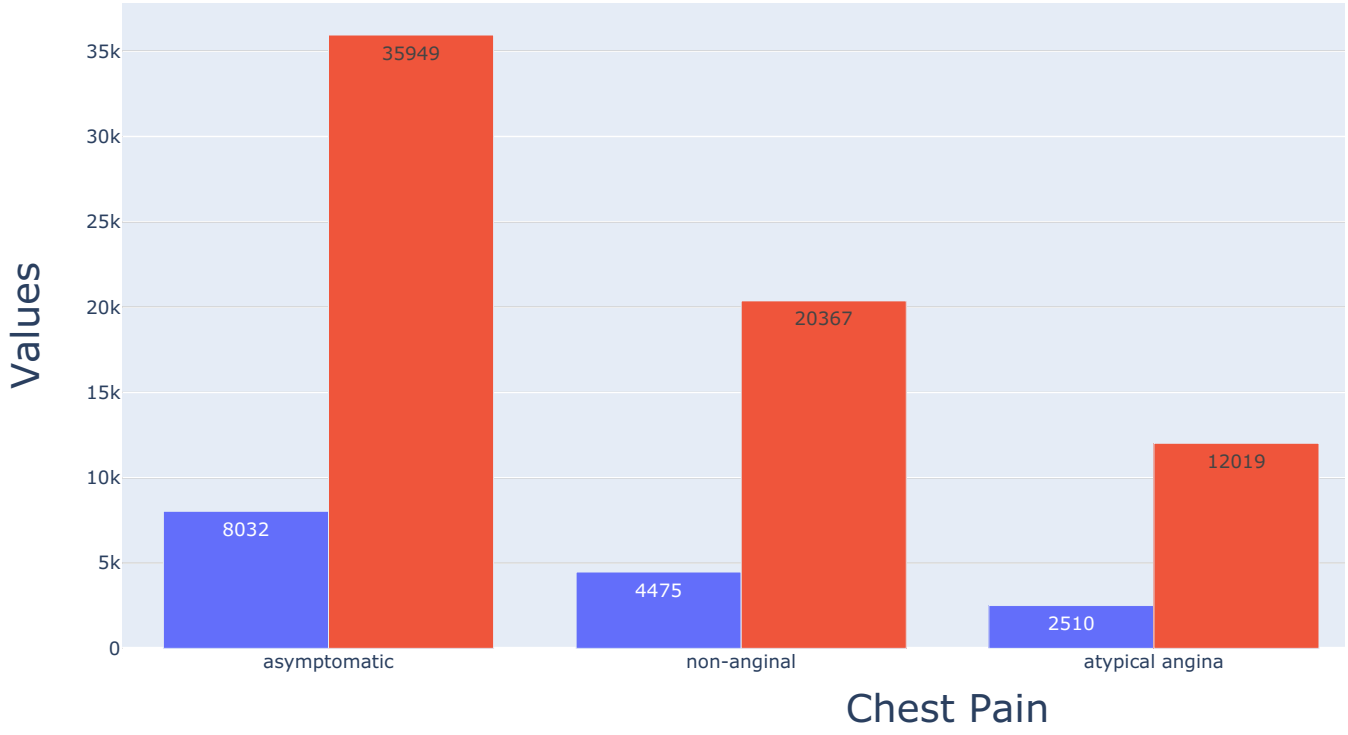
```
import plotly.express as px
from plotly.offline import iplot
import plotly.graph_objects as go
from plotly.subplots import make_subplots

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]:

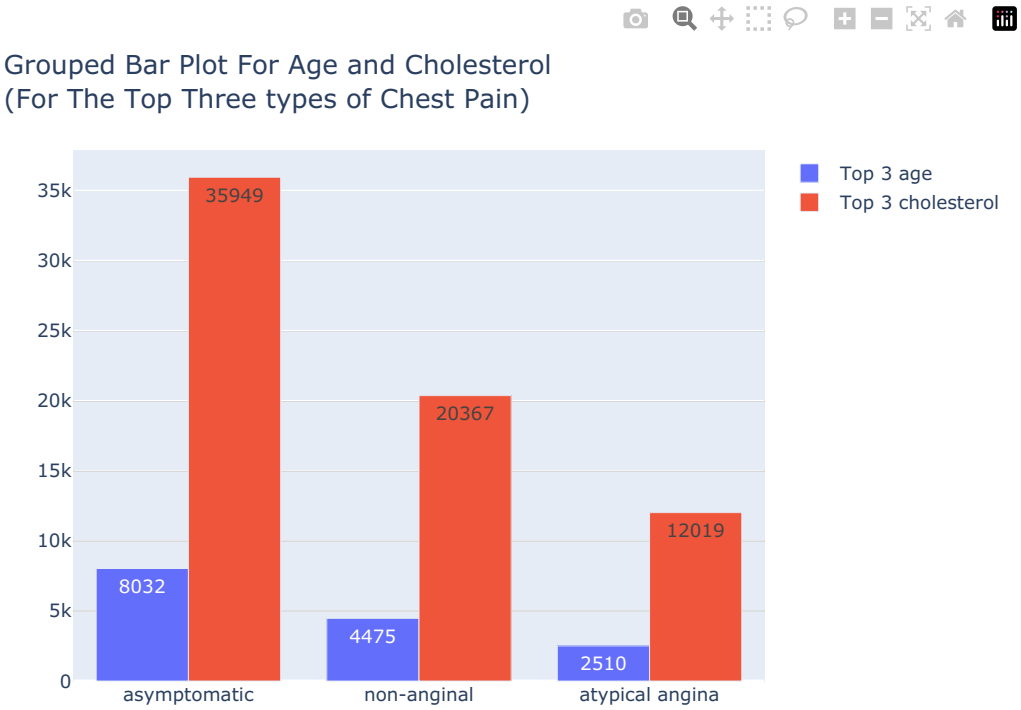
```
import numpy as np
import plotly.graph_objs as go
from plotly.offline import iplot

# 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
```



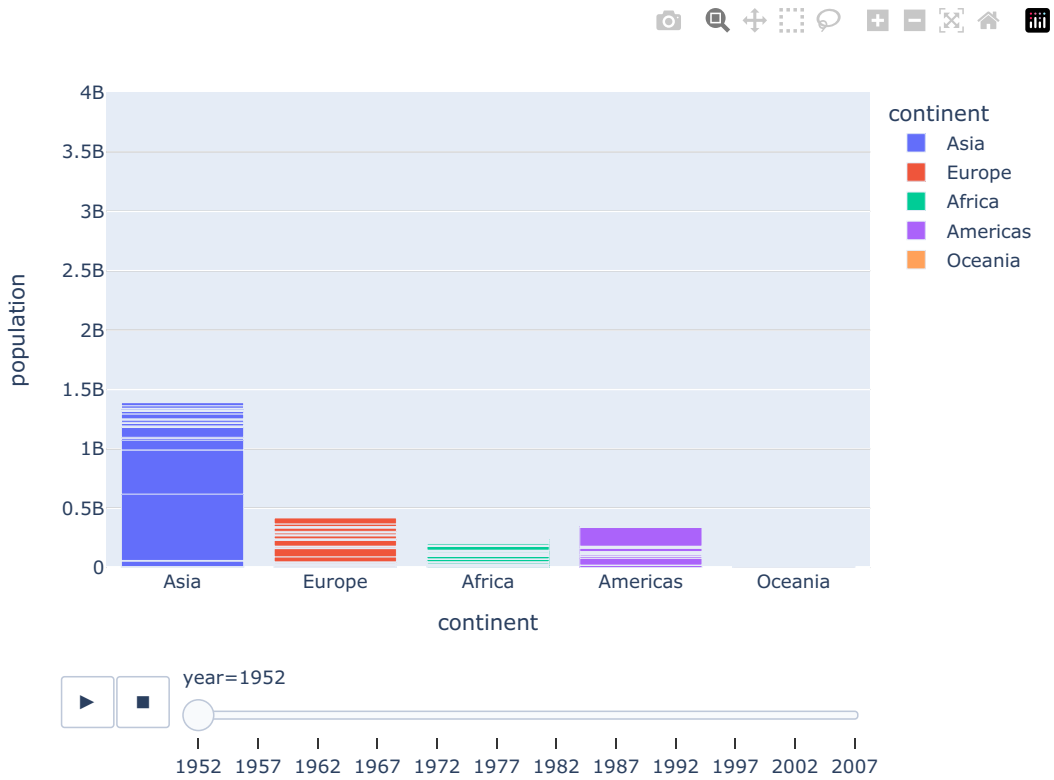
In [29]:

```
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))
```



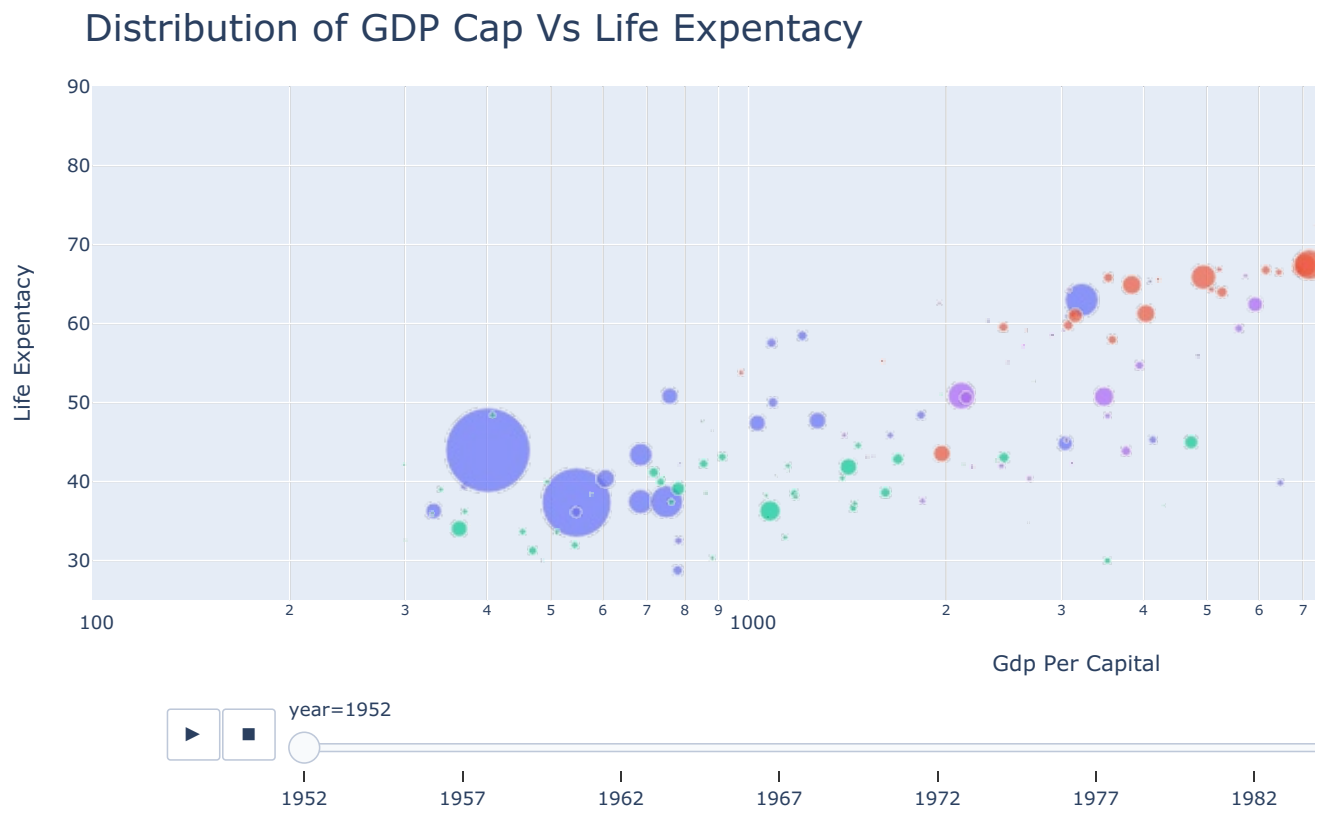
In [37]:
```
gap_df = pd.read_csv("gapminder_full.csv")

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()
```

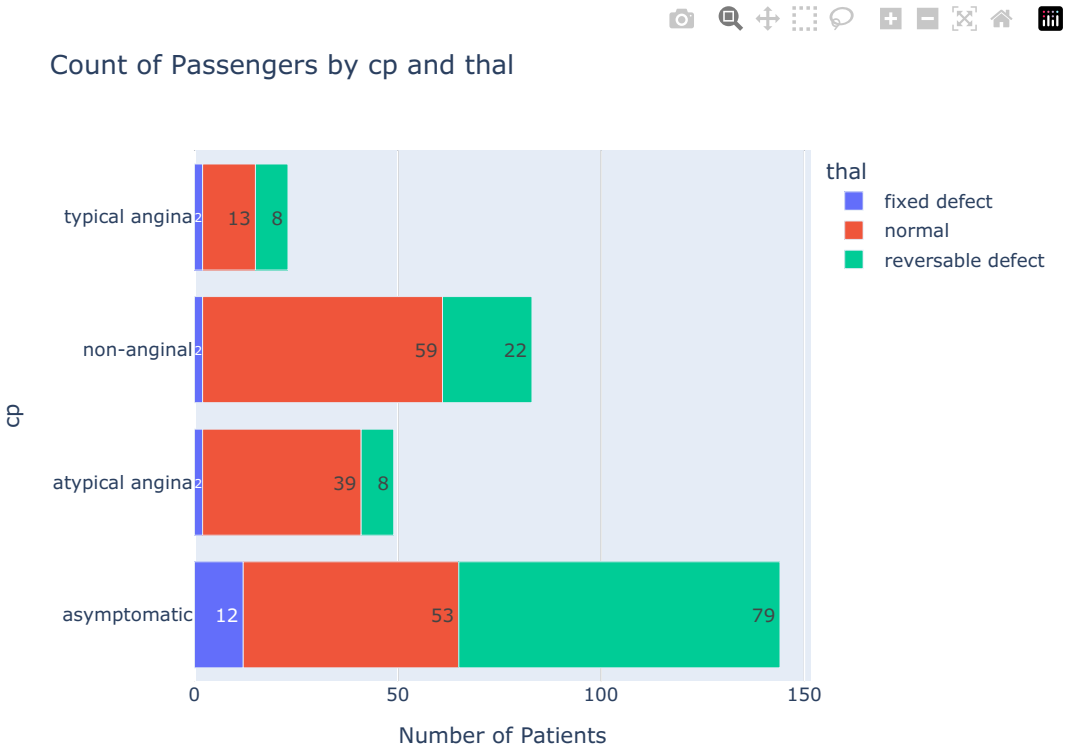


In [41]:
```
#Grouping the data by state
df1 = df[['cp','age','chol','num']]
df1.groupby('cp').sum().head(10).style.background_gradient(cmap='Blues')
```

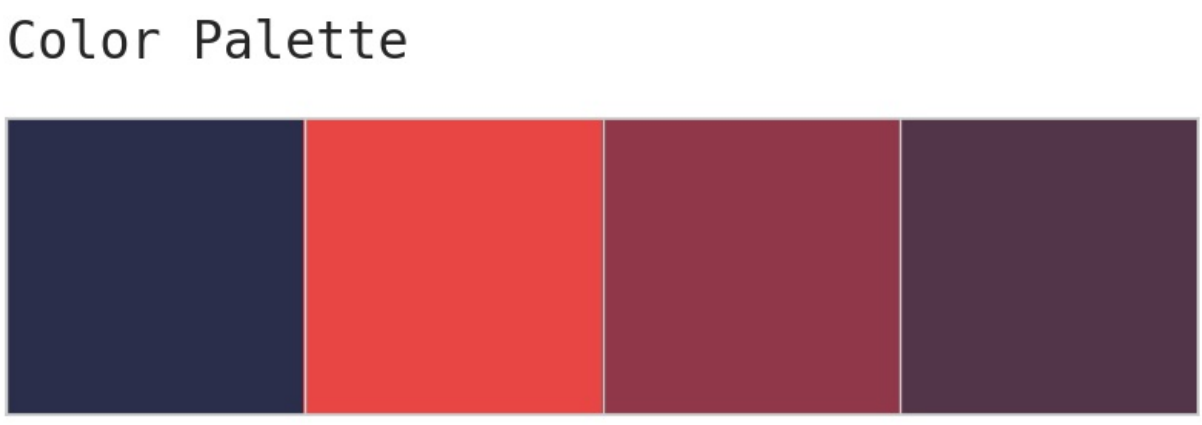
Out[41]:

cp	age	chol	num
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()
```



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()
```

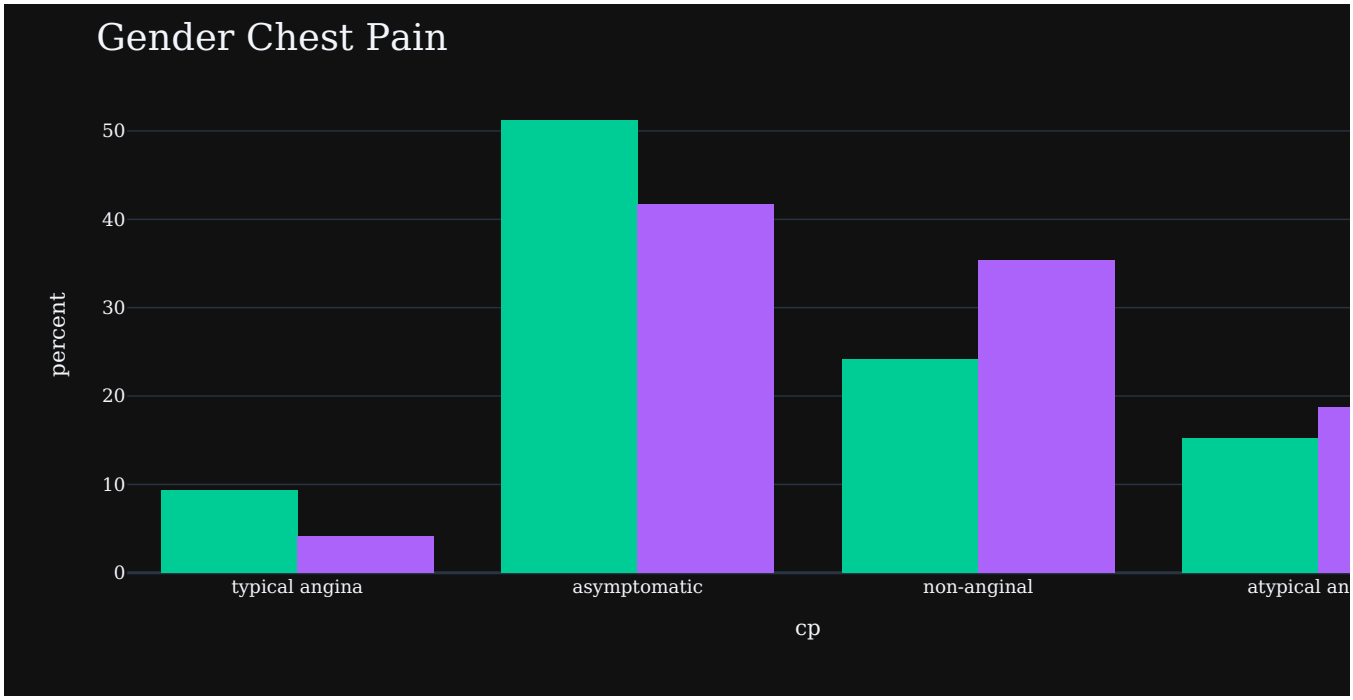


In [44]:
```
def format_title(title, subtitle=None, subtitle_font=None, subtitle_font_size=None):
    title = f'<b>{title}</b>'
    if not subtitle:
        return title
    subtitle = f'<span style="font-family: {subtitle_font}; font-size: {subtitle_font_size}px;">{subtitle}</spa
    return f'{title}<br>{subtitle}'


import plotly.figure_factory as ff

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

fig = ff.create_annotated_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()
```

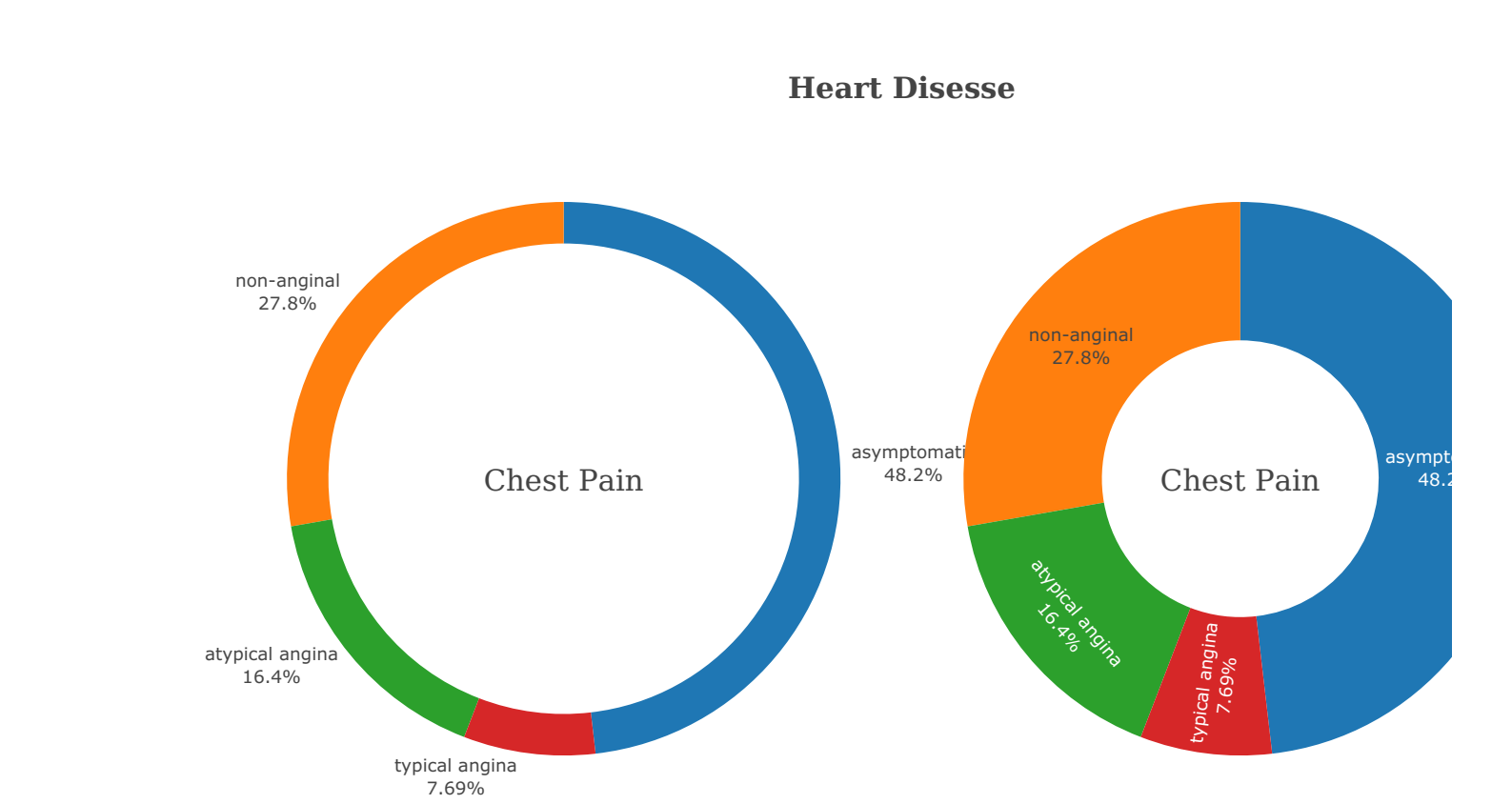
In [45]:

```
# 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()
```



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="<b> Heart Disesse <b>",
                  titlefont={'size':20, 'family': 'Serif',},
                  showlegend=False,
                  height=600,
                  width=1000,
                  template=None,
                 )

fig.show()
```

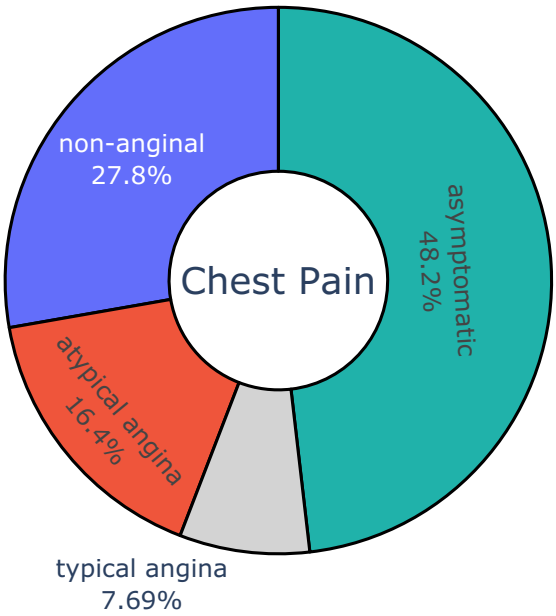


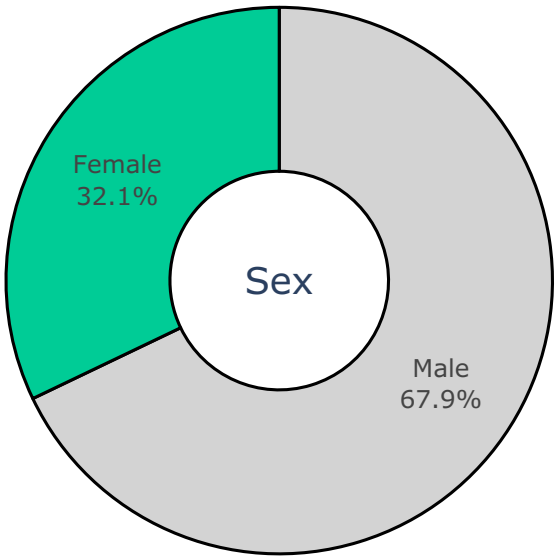
In [47]:

```
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="<b> Heart Desies <b>",
```

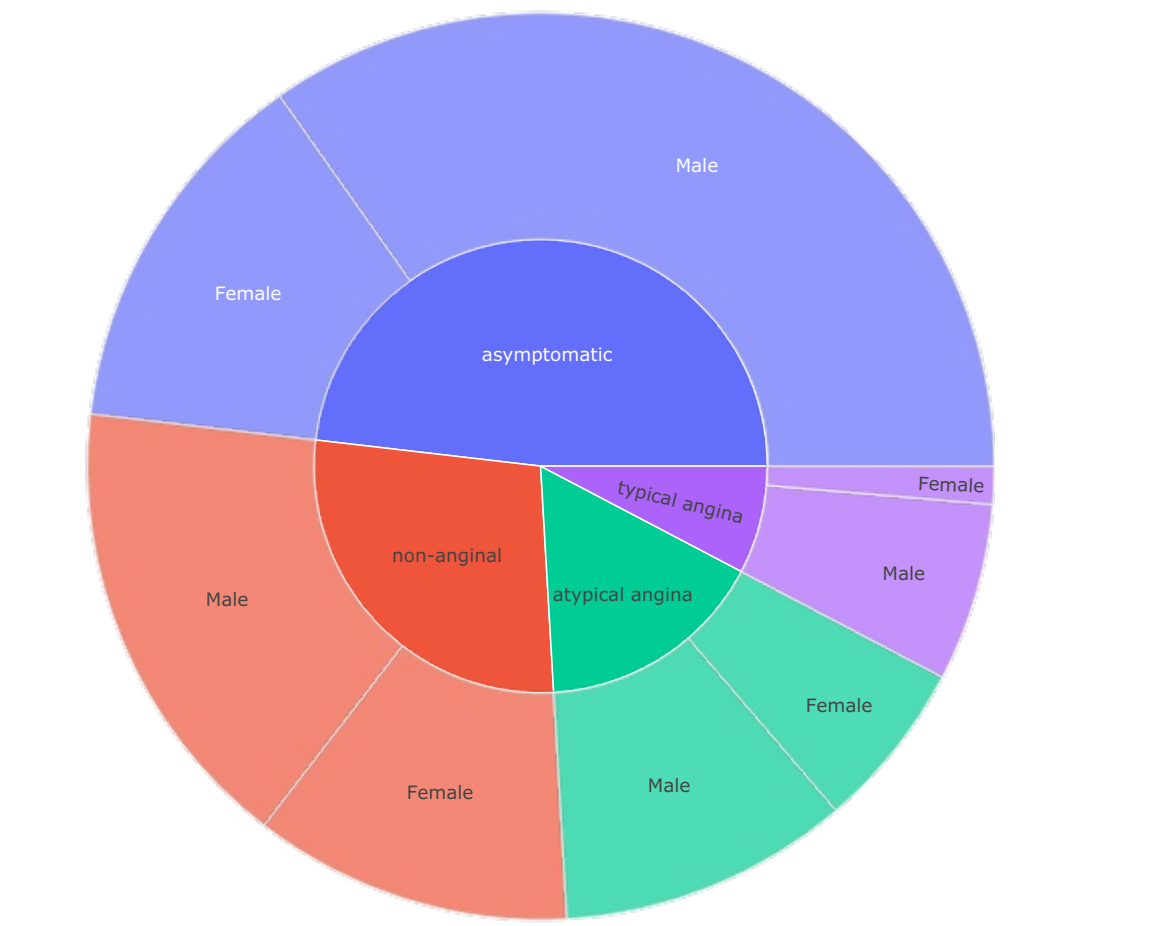
```
                  titlefont={'color':None, 'size': 24, 'family': 'San-Serif'},
                  showlegend=False,
                  height=600,
                  width=950,
                  )
fig.show()
```

Heart Desies

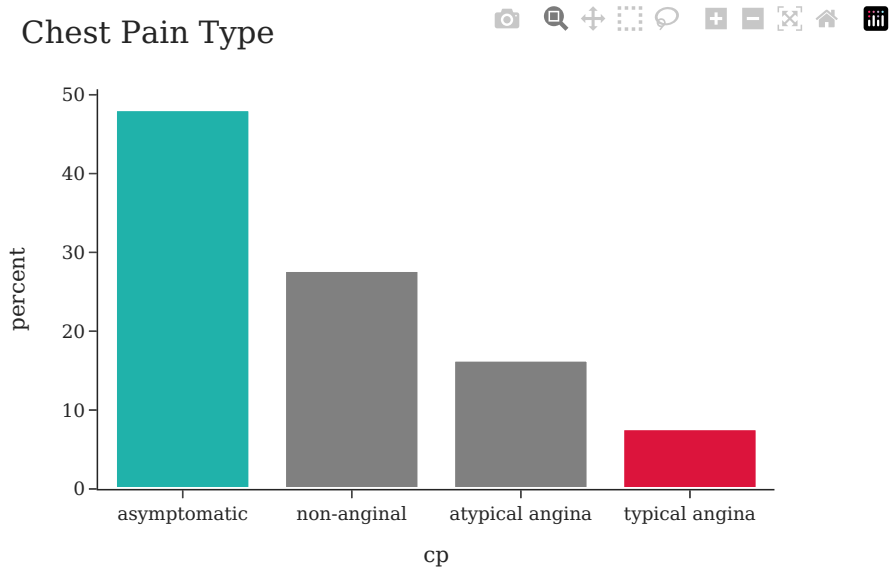




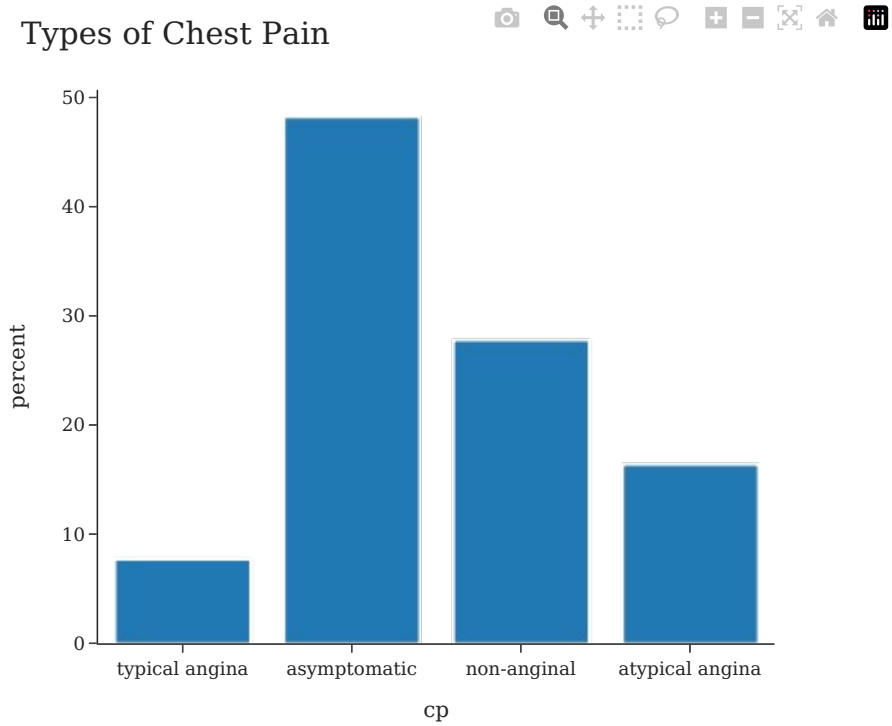
In [48]:
```
# data students performance
fig = px.sunburst(df,
                  path=['cp', 'sex'])
fig.update_layout(title_text="<b>Chest Pain vs Gender<b>",
                  titlefont={'size': 24, 'family':'Serif'},
                  width=750,
                  height=750,
                 )
fig.show()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



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()
```

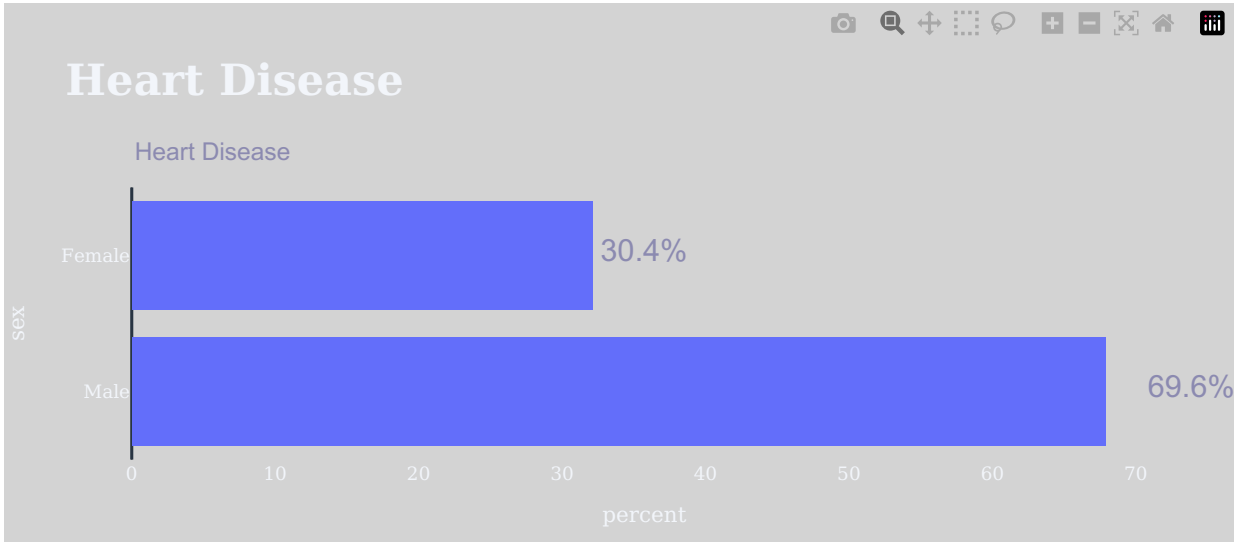


In [50]:
```
fig = px.histogram(df, x="cp",
                   width=600,
                   height=500,
                   histnorm='percent',
                   template="simple_white",
                   )
fig.update_layout(title="Types of Chest Pain",
                  font_family="San Serif",
                  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()
```



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",
```

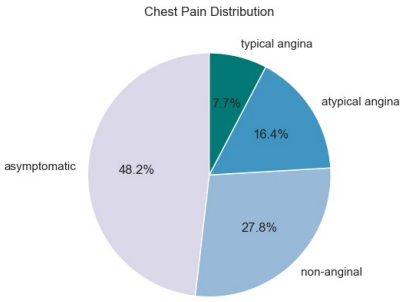
```
                    orientation='h',
                    width=800,
                    height=350,
                    histnorm='percent',
                    template="plotly_dark"
                    )
fig.update_layout(title="<b>Heart Disease<b>",
                  font_family="San Serif",
                  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()
```

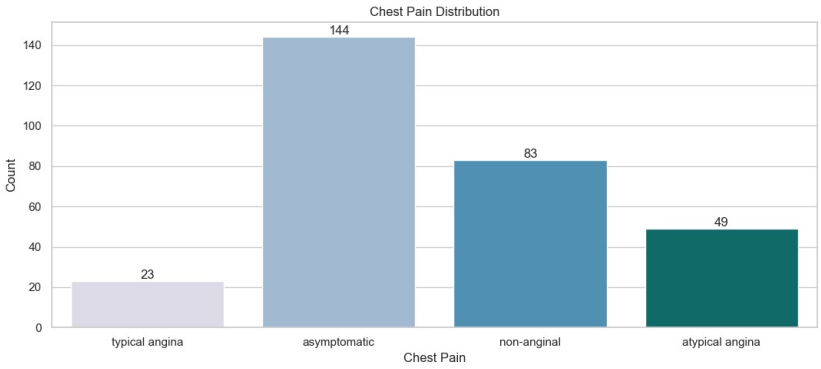


In [52]:

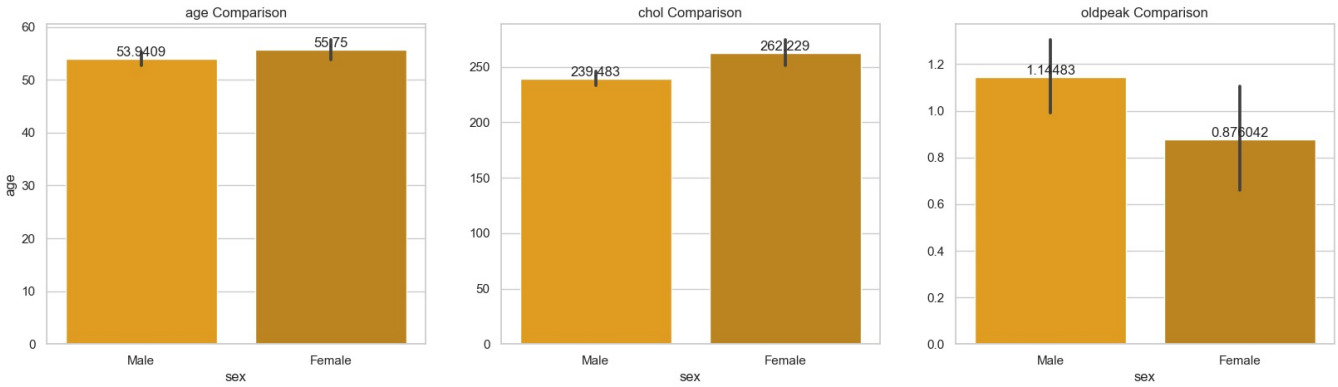
```
# Plotting the pie chart
plt.figure(figsize=(20, 5))

# 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)), a
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()
```



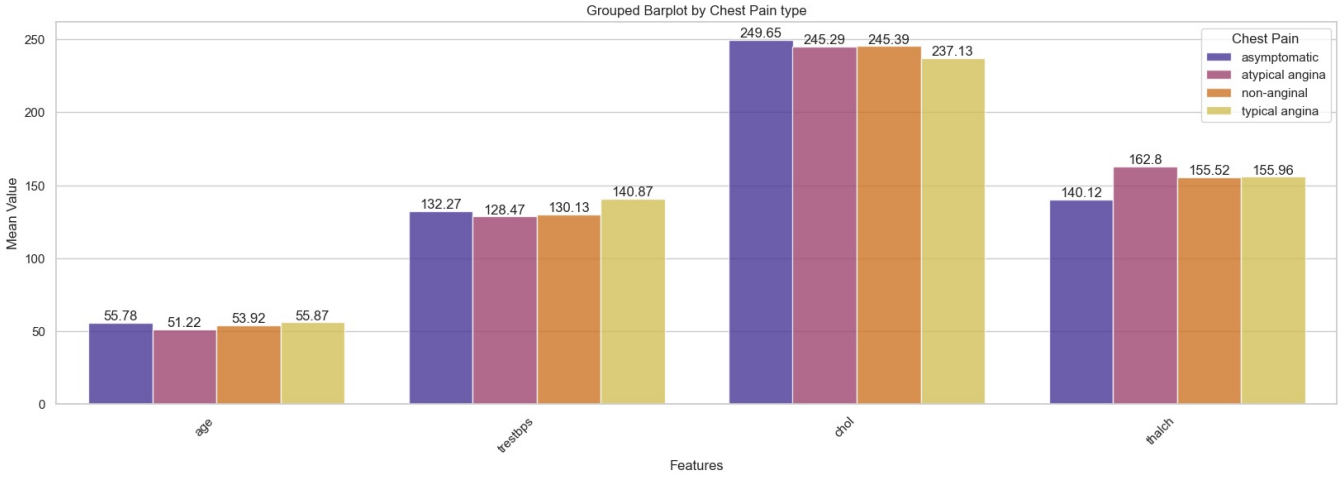


In [53]:
```
plt.figure(figsize=(20, 5))

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()
```

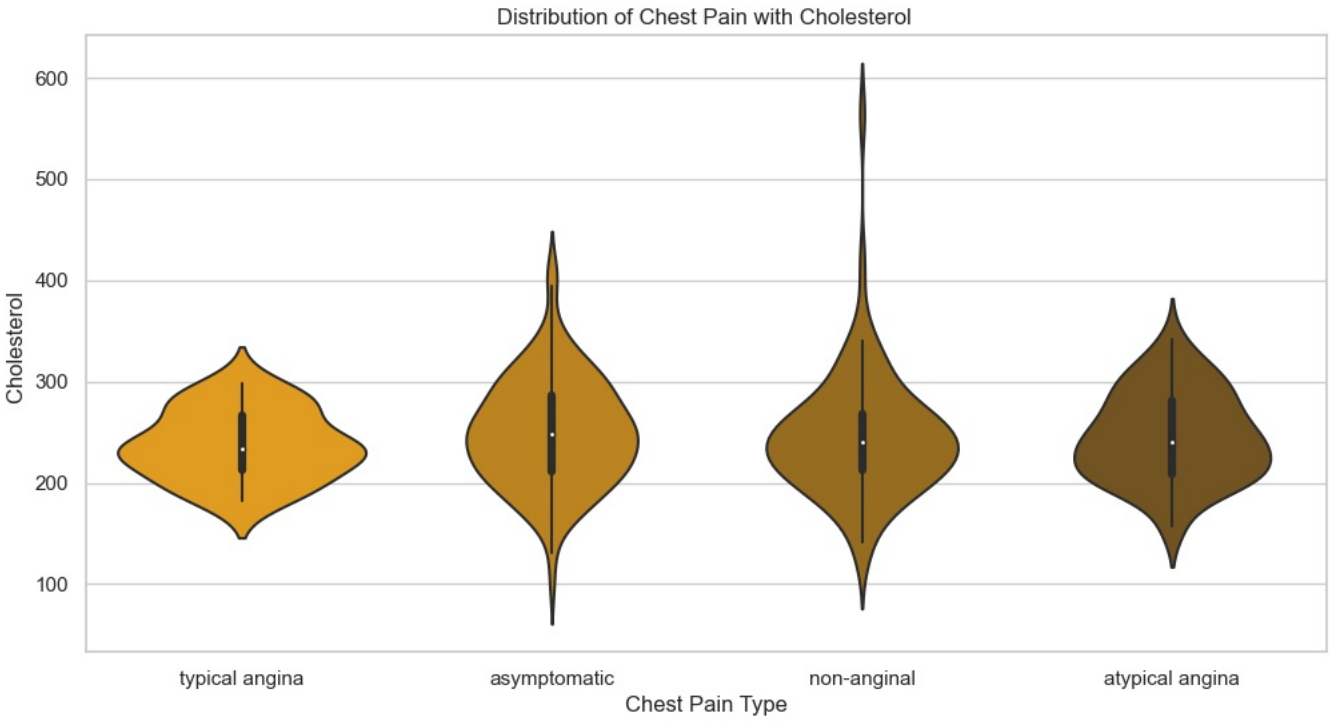


In [54]:
```
sns.pairplot(df[['cp','age','chol','thalch']], hue='cp', aspect=1.5,dropna=True,palette='bright')
plt.show()
```

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()
```

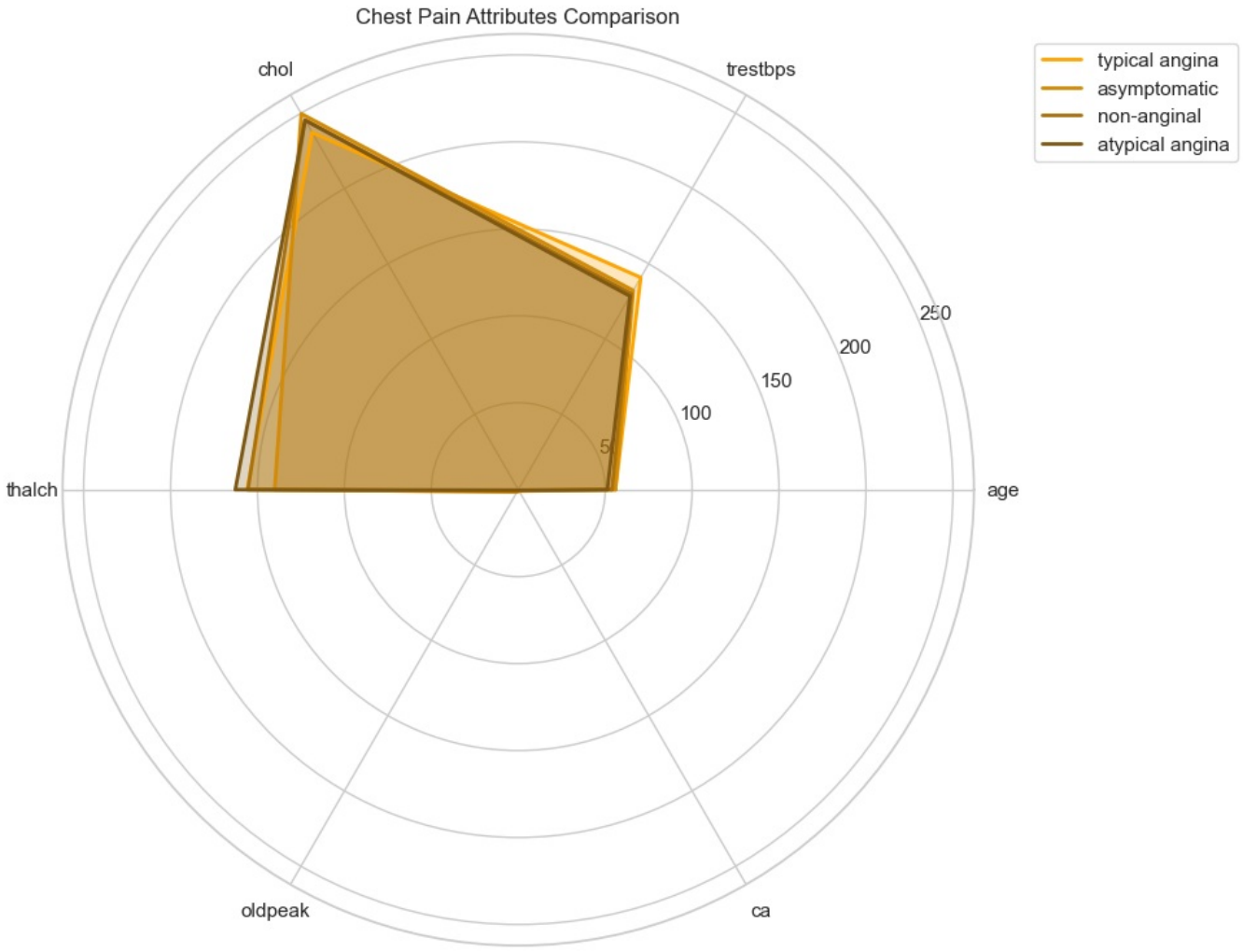


In [56]:
```
# Visualization 8: Violin Plot - Skill Moves Distribution
plt.figure(figsize=(12, 6))
sns.violinplot(x='cp', y='chol', data=df)
plt.title('Distribution of Chest Pain with Cholesterol ')
plt.xlabel('Chest Pain Type')
plt.ylabel('Cholesterol ')
plt.show()
```



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 angi
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()
```



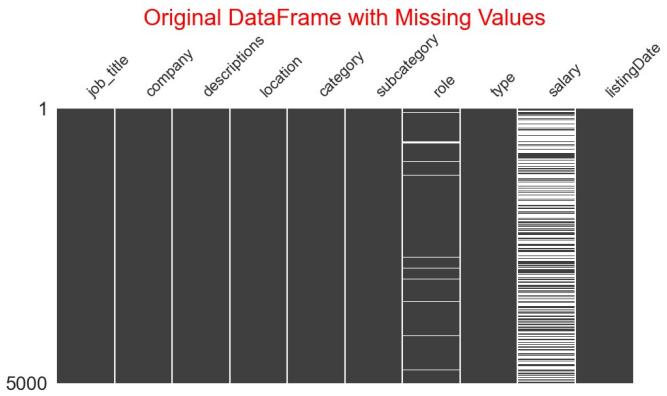
In [58]:
```
jobs = pd.read_csv("jobstreet_all_job_dataset.csv")
jobs = jobs.sample(5000)
jobs = jobs.drop(columns=['job_id'], axis=1)
jobs = jobs.reset_index()
jobs = jobs.drop(columns=['index'], axis=1)
display(jobs.shape)
jobs.head(2)
```

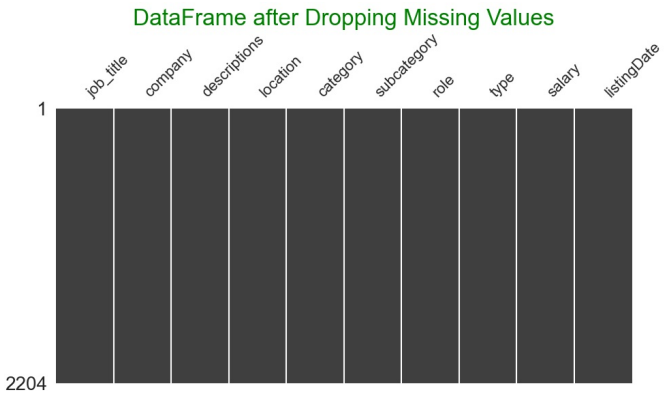
```
(5000, 10)
```

Out[58]:

	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	2(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	2(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)
```

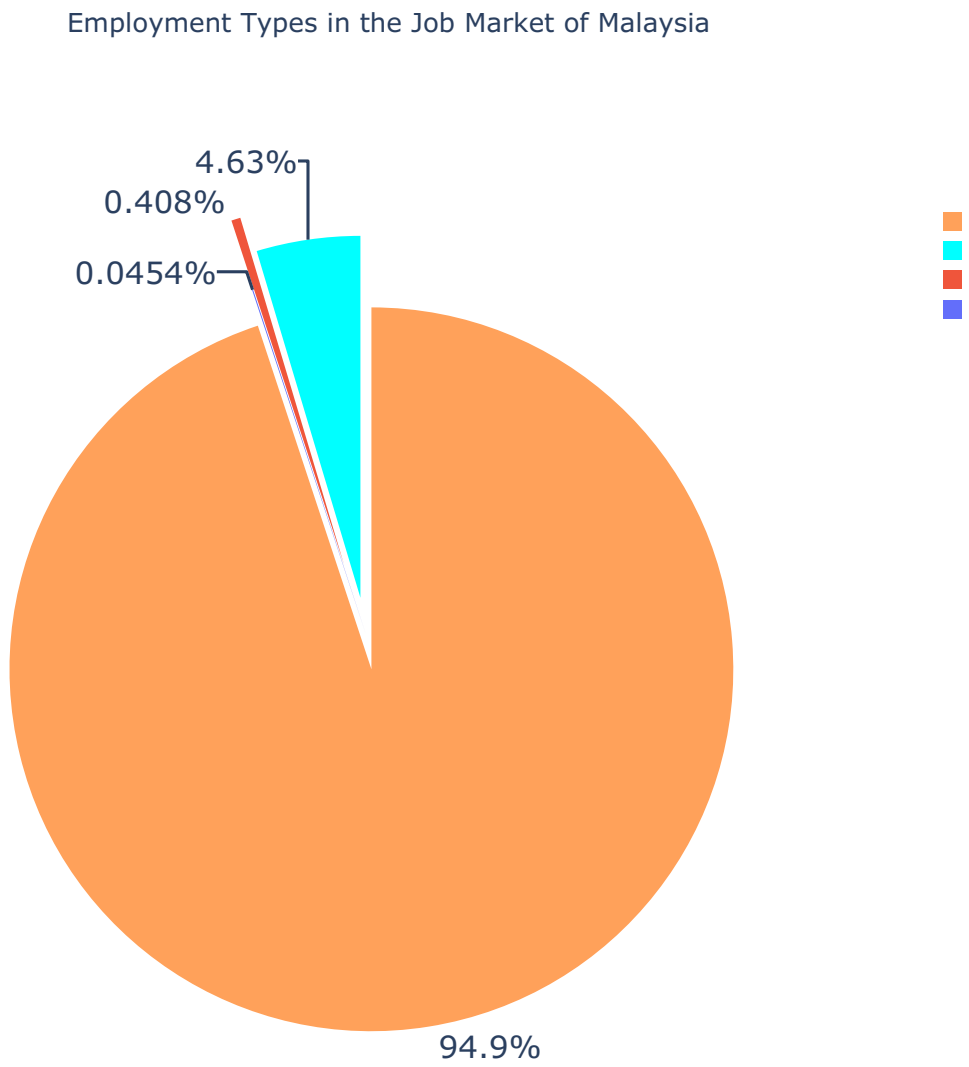
Out[60]:

	job_title	company	descriptions	location	category	subcategory	role	type	listingDate	Sa
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,
```

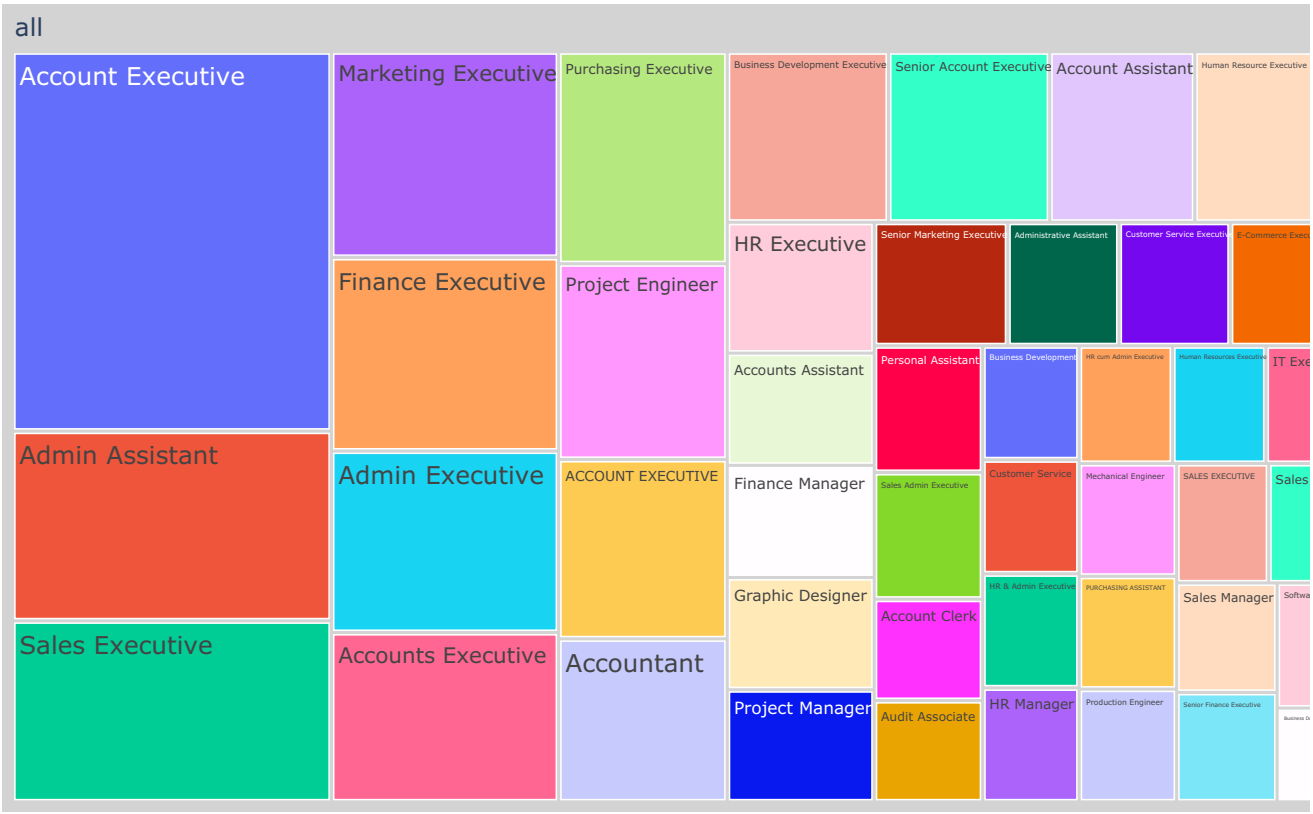
```
                  title_y=0.98)

fig.show()
```



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()
```

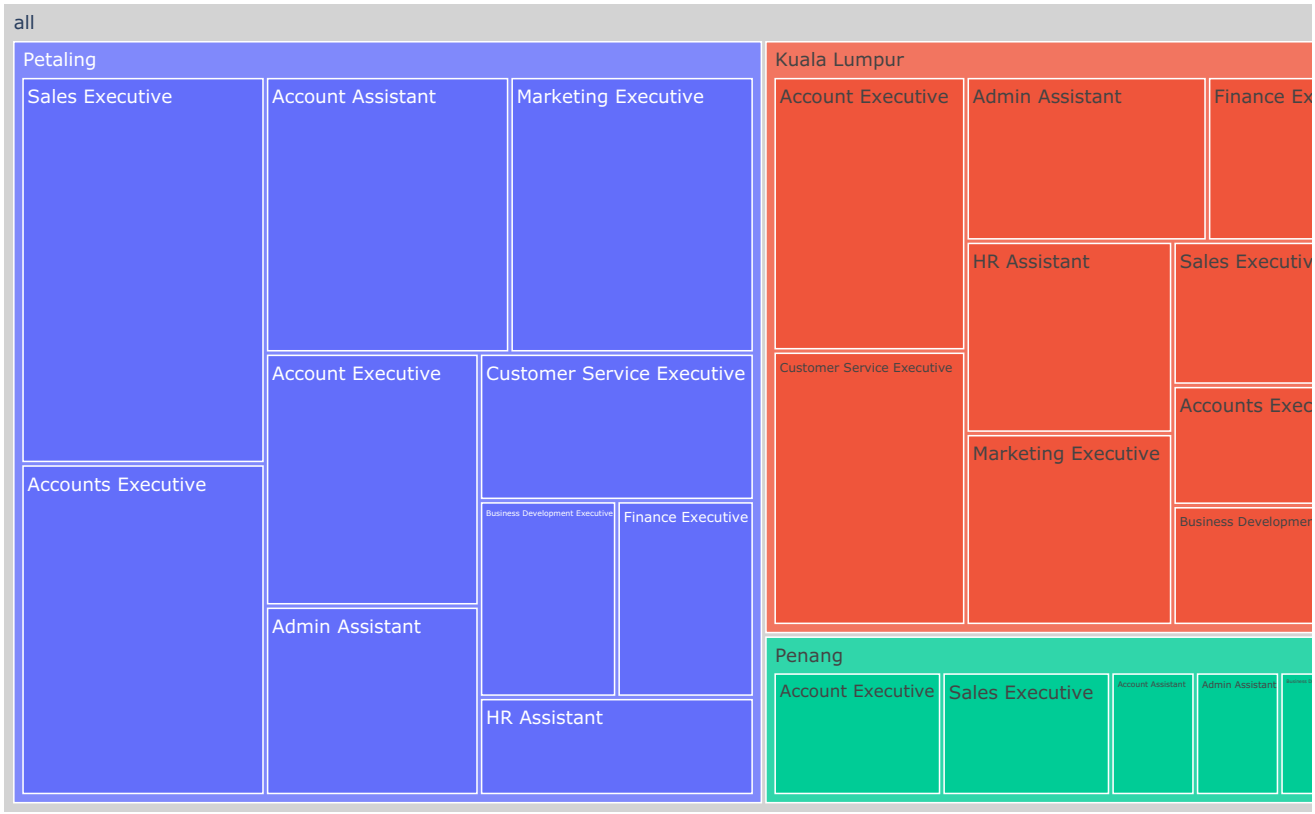
Top Job Openings: Job Roles in Malaysia



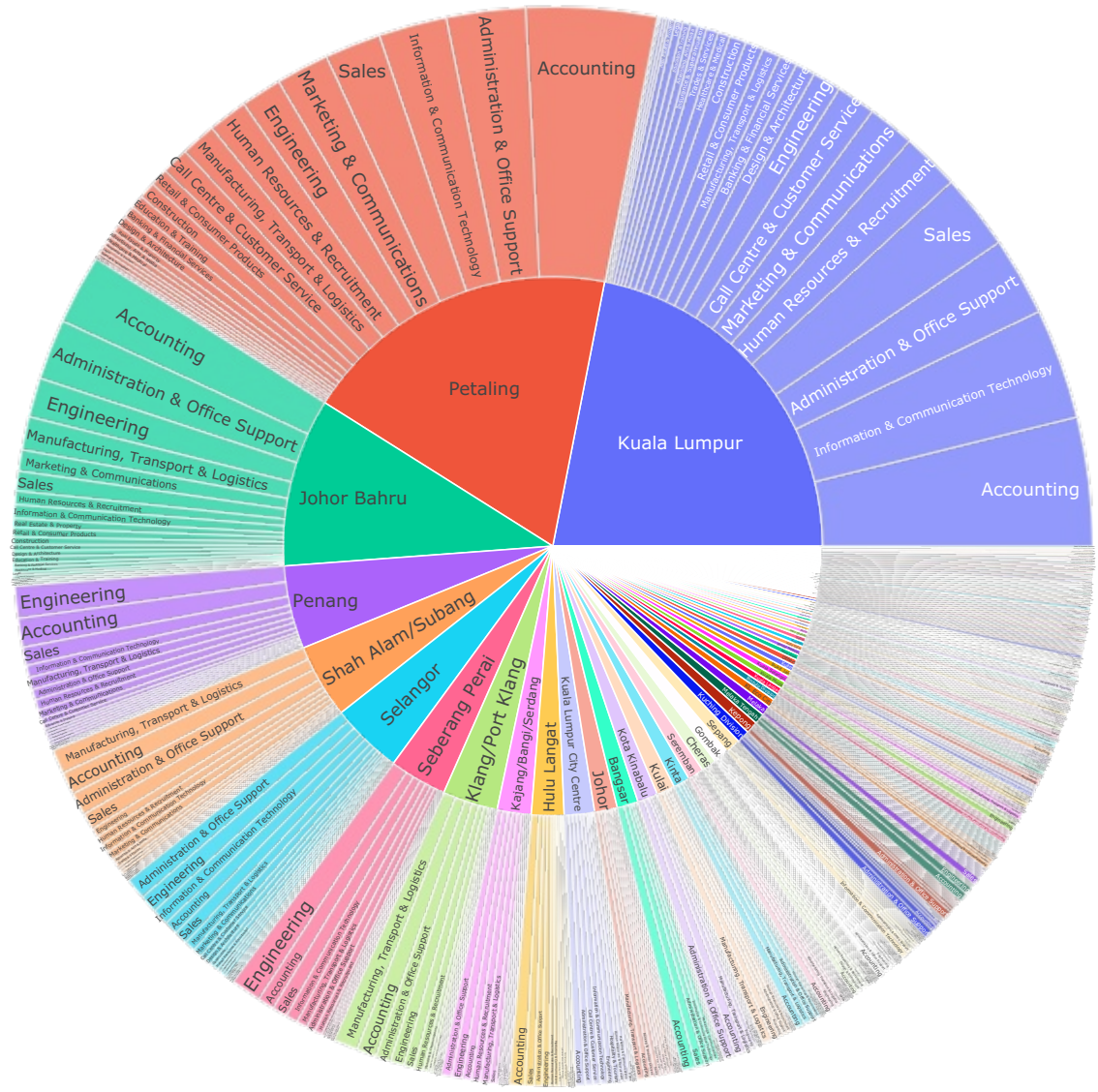
In [63]:

```
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



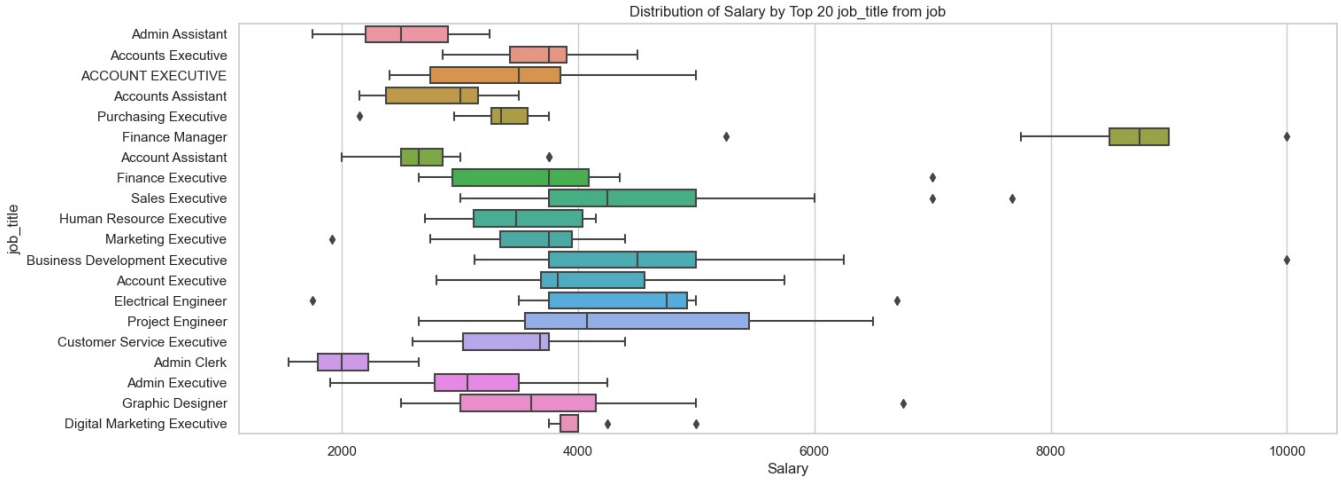
In [64]:
```
# plot a sunburst chart
fig = px.sunburst(job, path=['location','category'])

# 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()
```

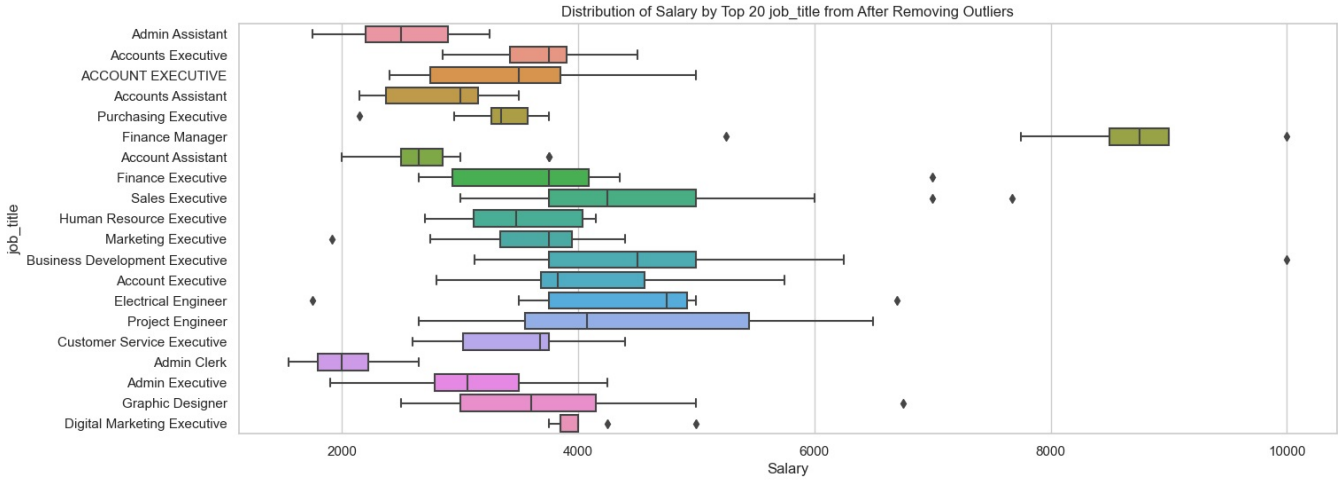


In [65]:

```
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')
```

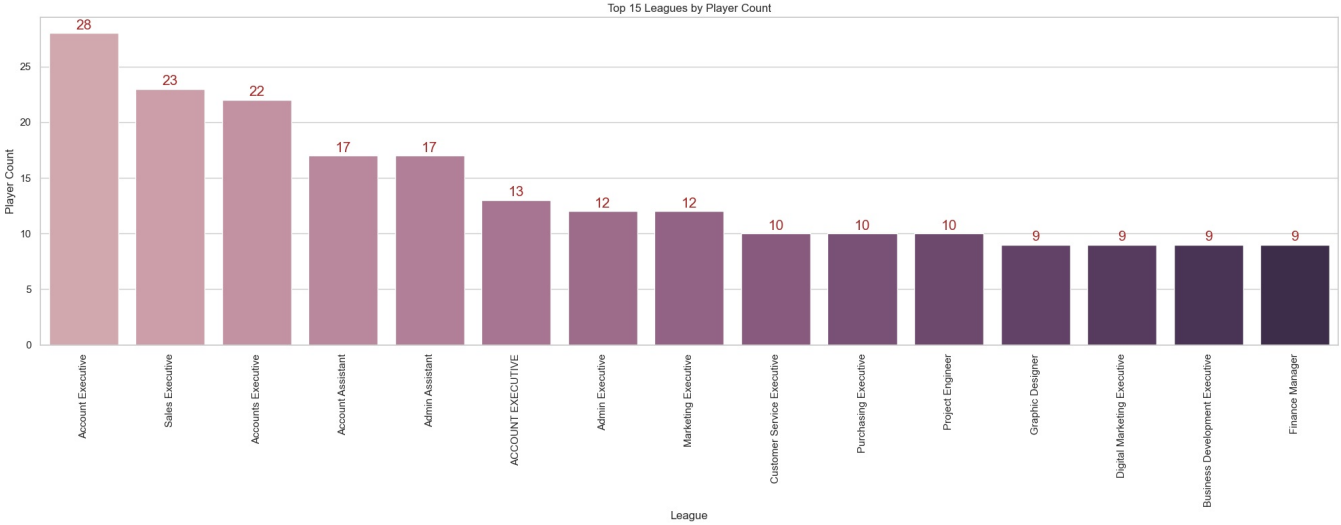
```
plt.ylabel('job_title')
plt.show()
```





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:,}"
    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]:

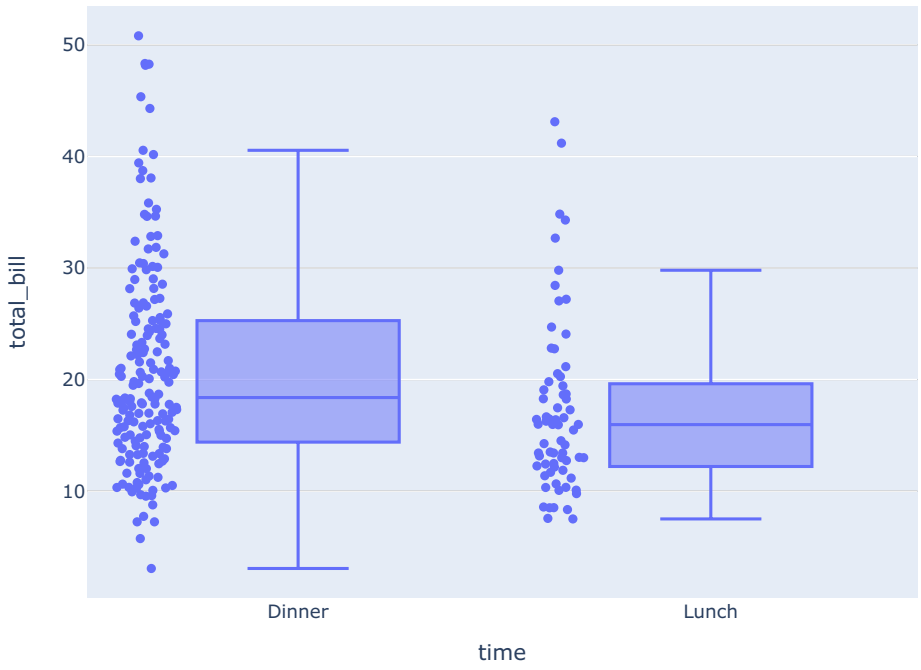
```
tips_df = pd.read_csv("tip.csv")

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()
```

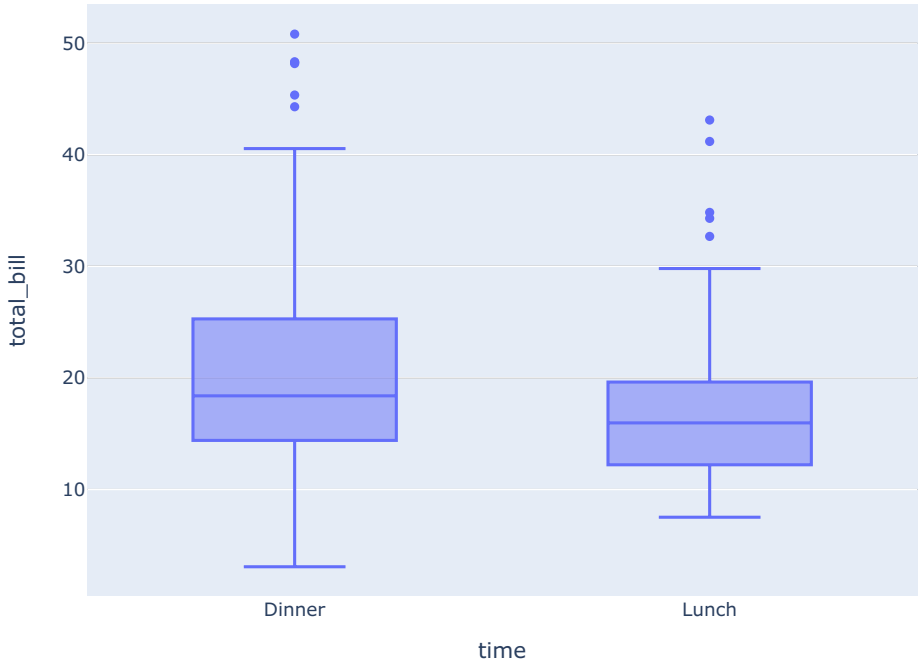
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3

In [68]:

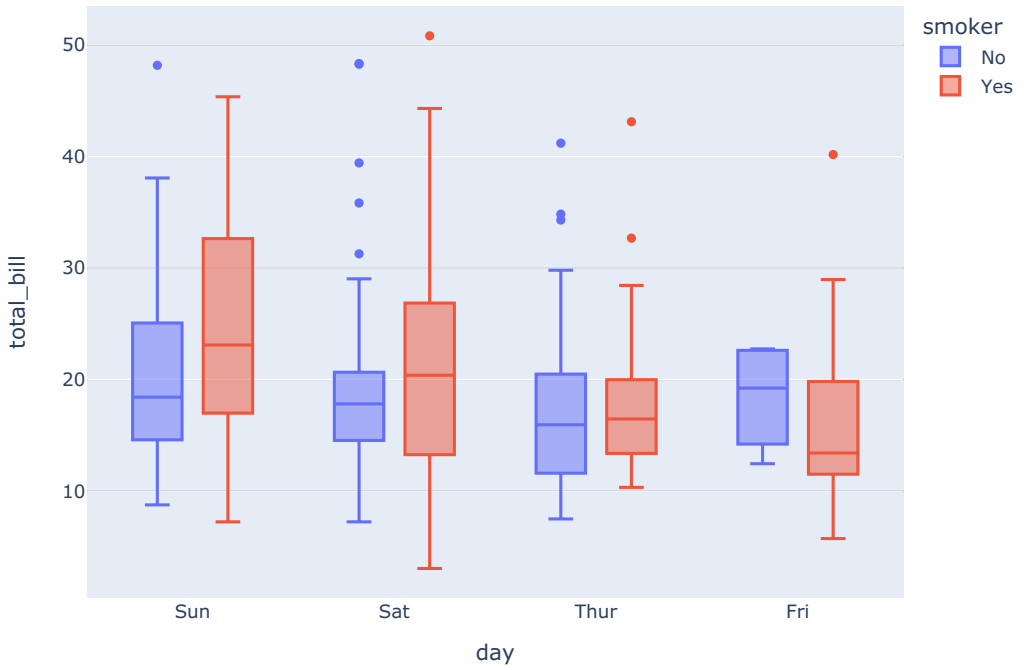
```
fig = px.box(tips_df,
             x="time",
             y="total_bill",
             points="all")
fig.show()
```



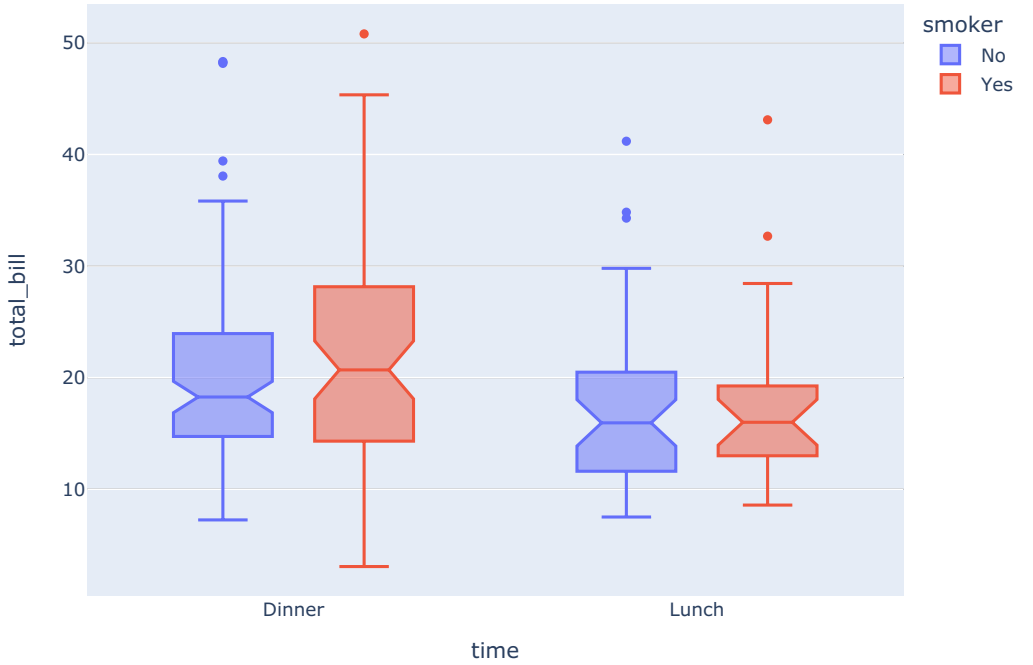
In [69]:
```
fig = px.box(tips_df,
             x="time",
             y="total_bill",
             points="outliers")
fig.show()
```



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

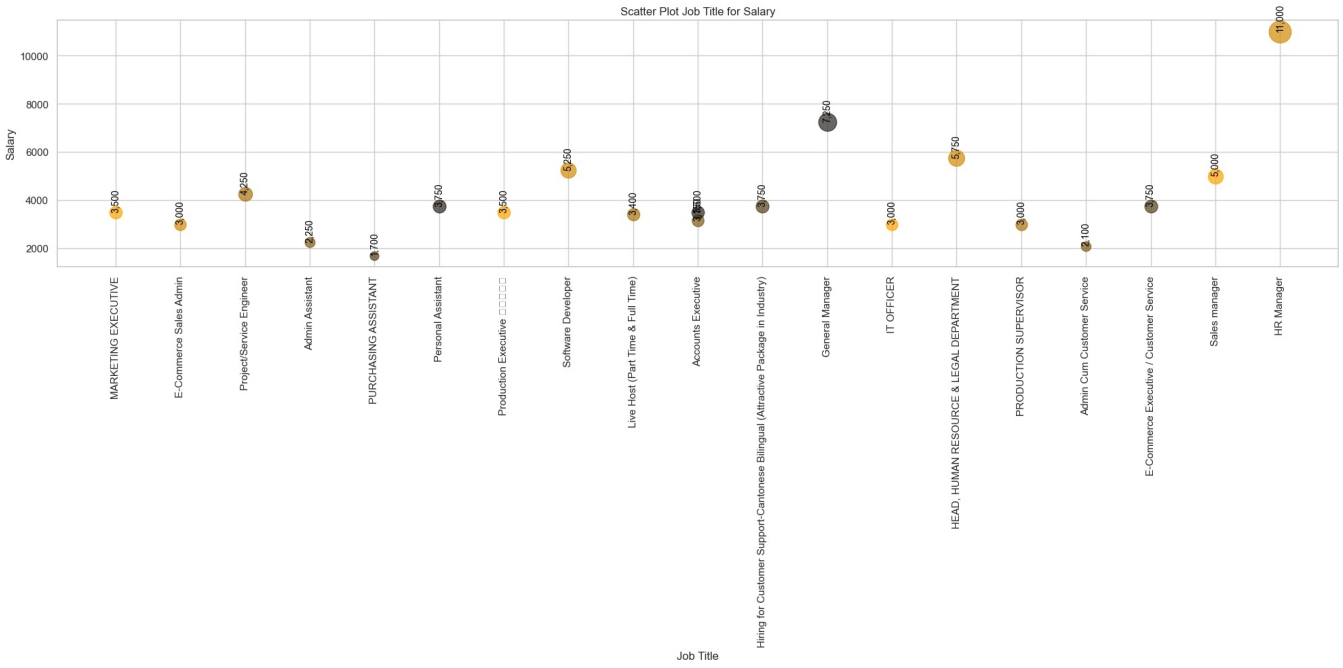


In [71]:
```
fig = px.box(tips_df,
             x="time",
             y="total_bill",
             color="smoker",
             notched=True,
             hover_data=["day"] # add day column to hover data
            )
fig.show()
```



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)
```

Out[74]:

	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/ emai...	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()
```

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

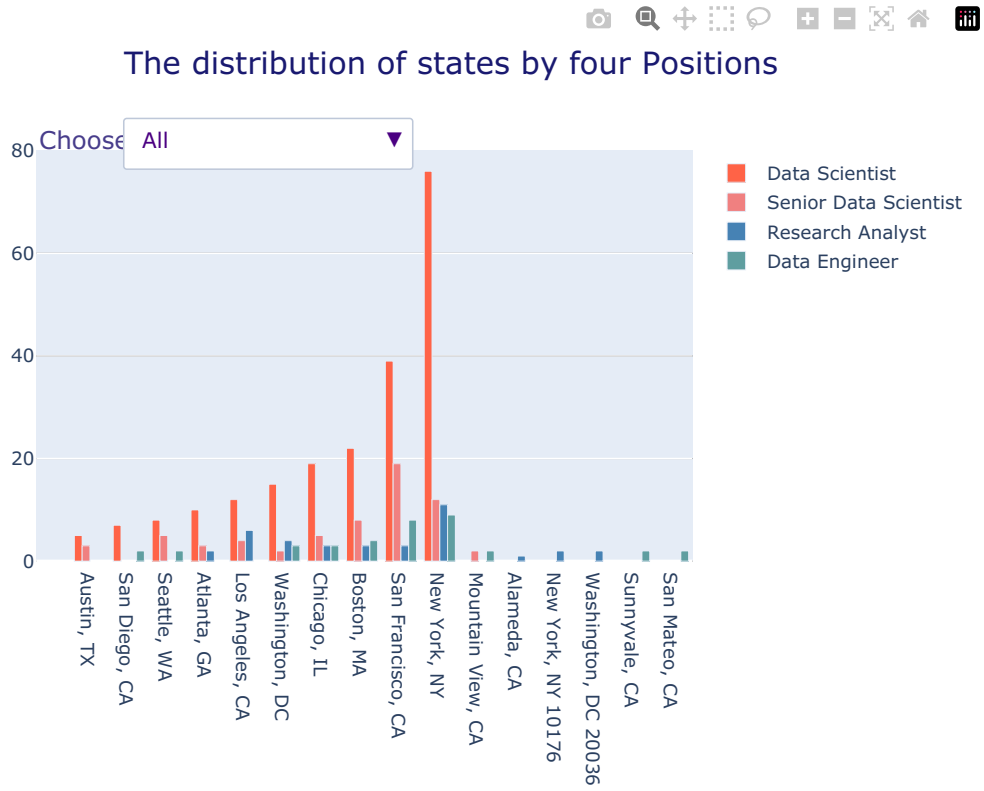
Out[76]:

	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()
```

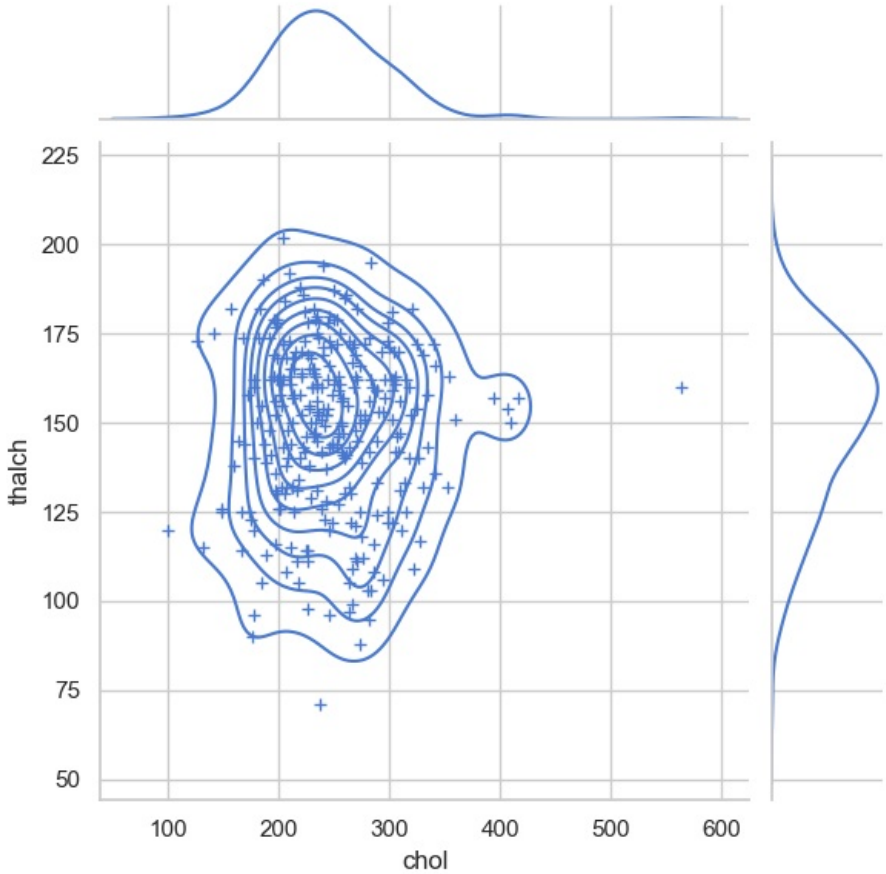


In [81]:
```
df =  pd.read_csv("heart_disease_uci.csv")
df = df.dropna()
df.head(2)
```

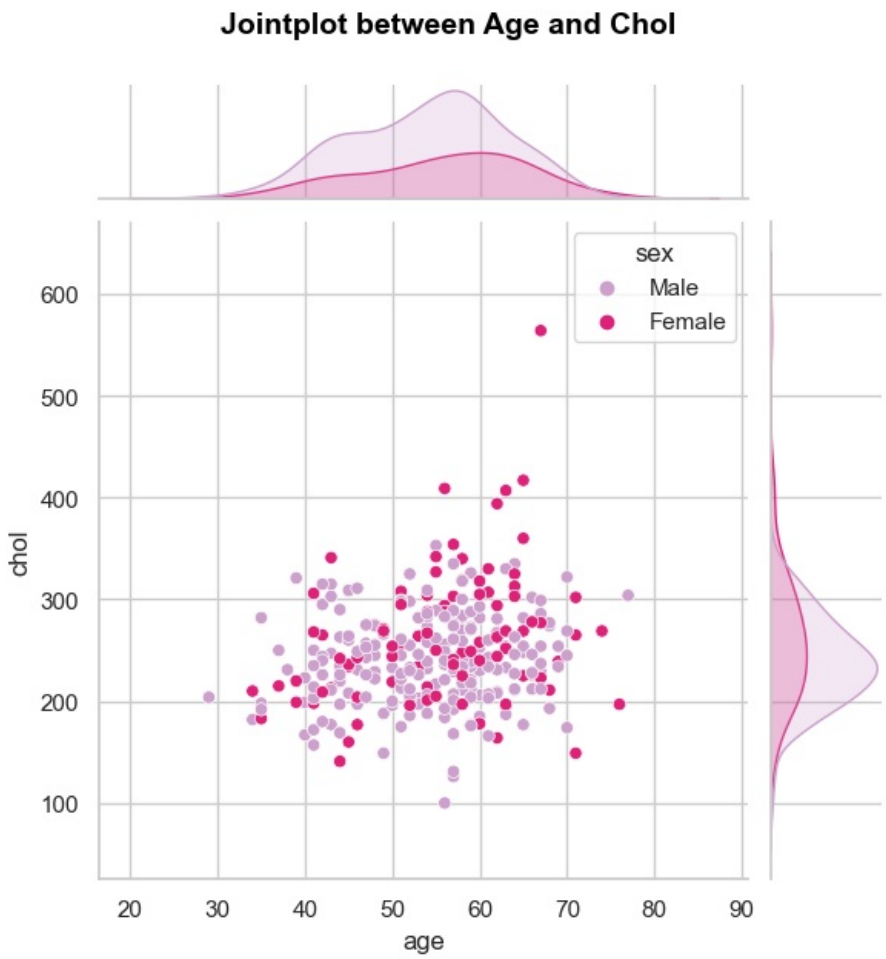
Out[81]:

	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

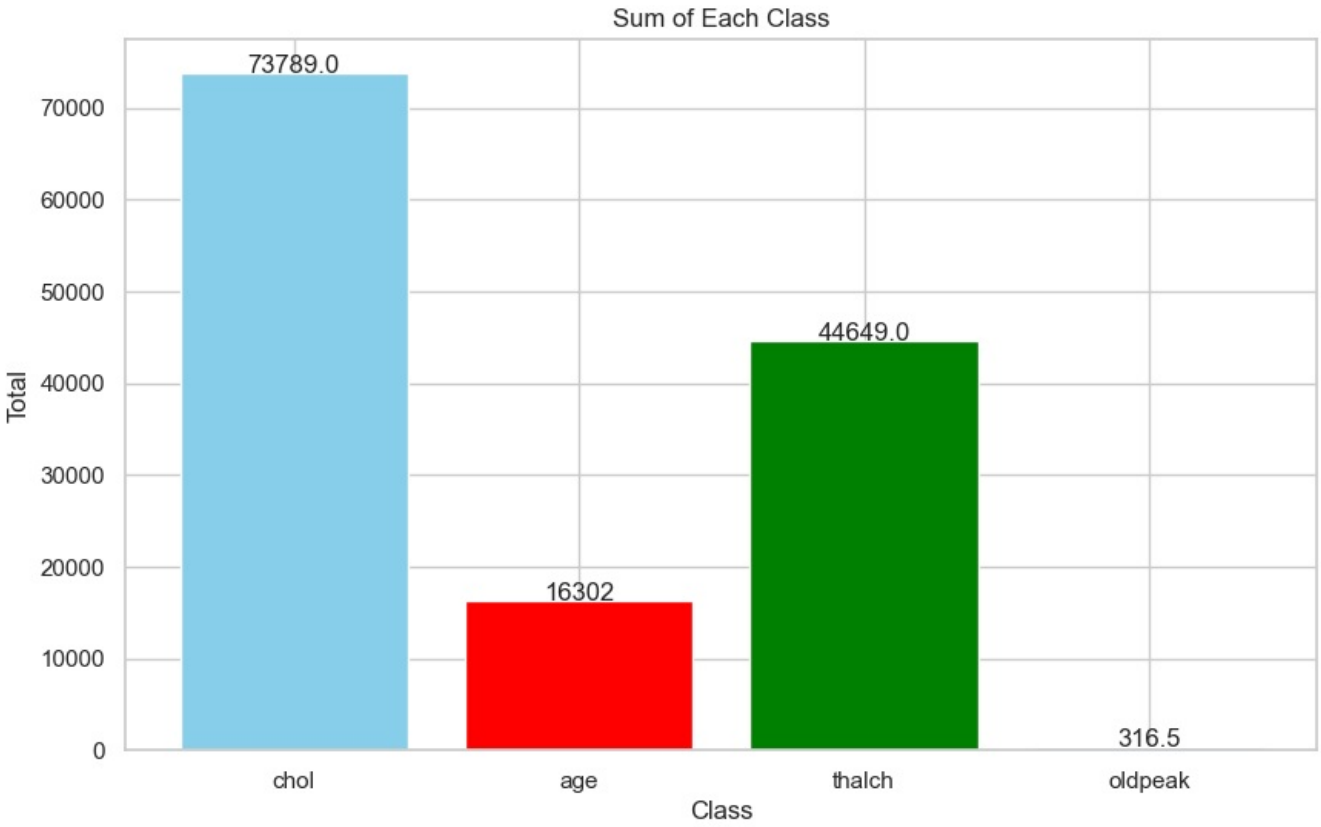
In [82]:
```
g = sns.jointplot(x="chol", y="thalch", data=df, kind="kde", color="b")
g.plot_joint(plt.scatter, c="b", s=30, linewidth=1, marker="+")
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("chol", "thalch");
```



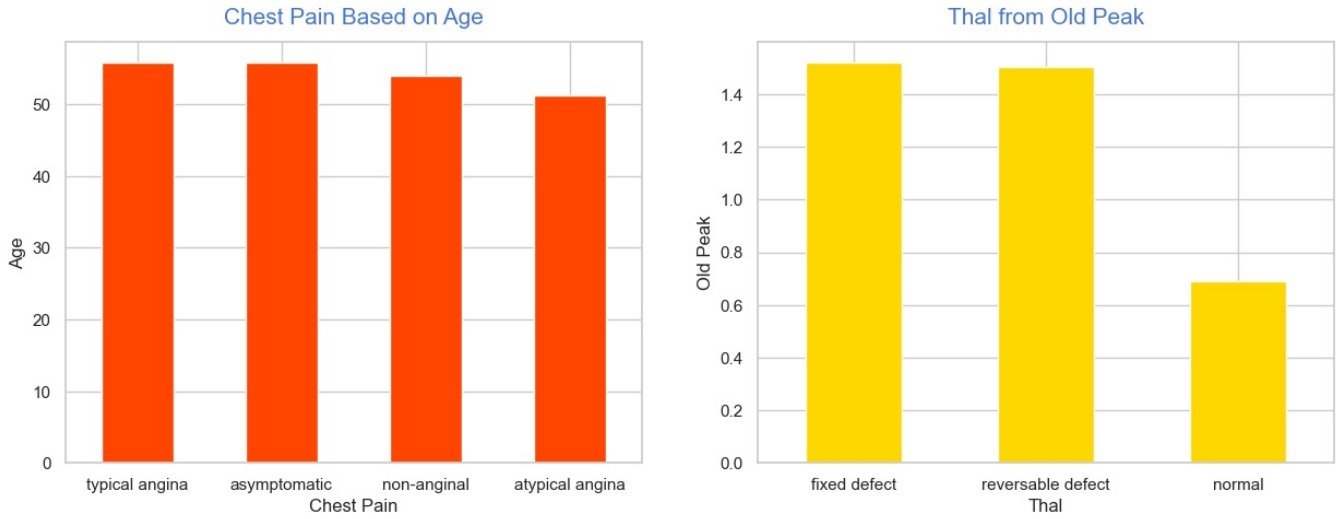
In [83]:
```
# --- Create Jointplot ---
jointplot = sns.jointplot(x = 'age', y = 'chol', data = df, hue = 'sex', palette = 'PuRd')


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



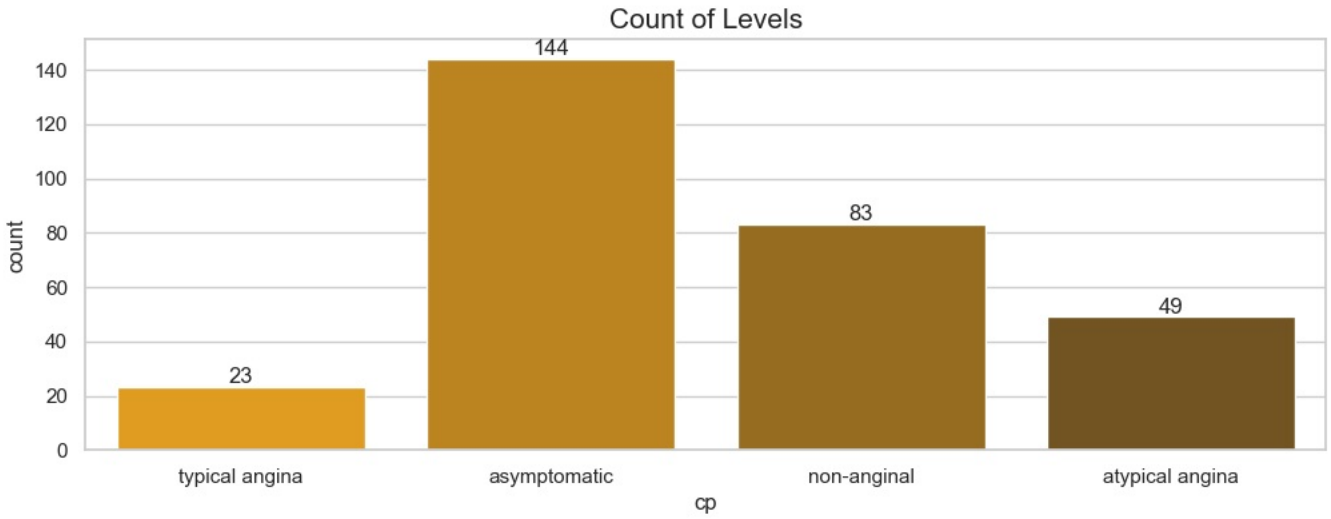
In [84]:
```
import matplotlib.pyplot as plt

# 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()
```



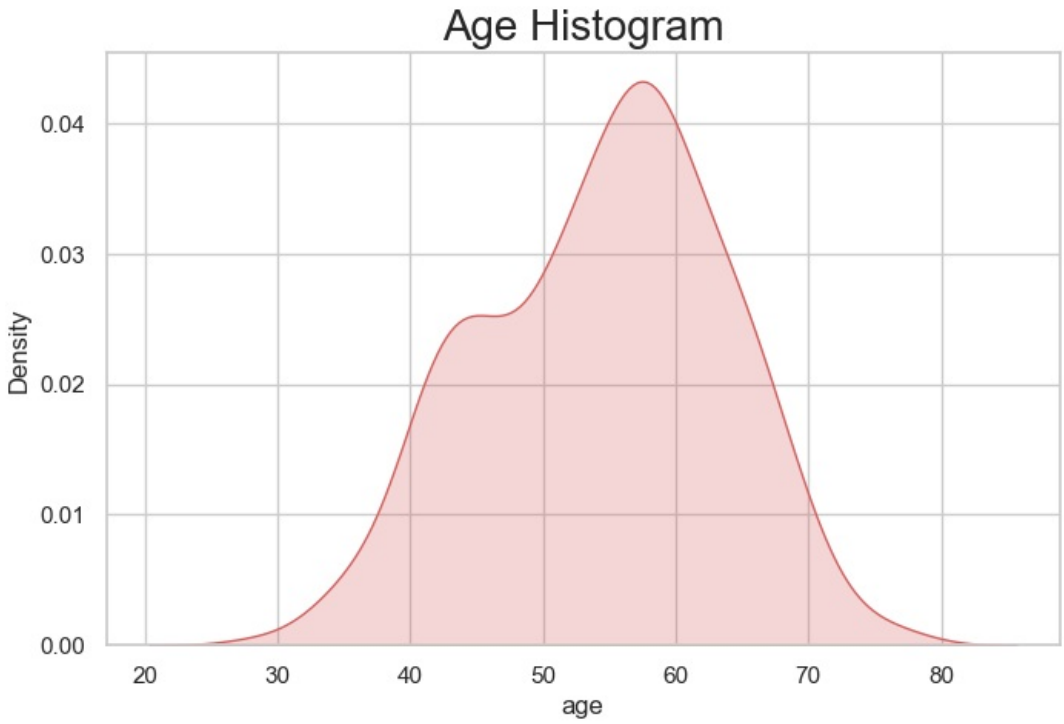
In [85]:
```
plt.rcParams['figure.figsize'] = (15,5)
plt.subplot(1, 2, 1)
chart = df.groupby('cp')['age'].mean().sort_values(ascending = False).plot(kind = 'bar', color = 'orangered')
chart.set_xticklabels(chart.get_xticklabels(), rotation = 0)
plt.title('Chest Pain Based on Age', fontsize = 15, color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

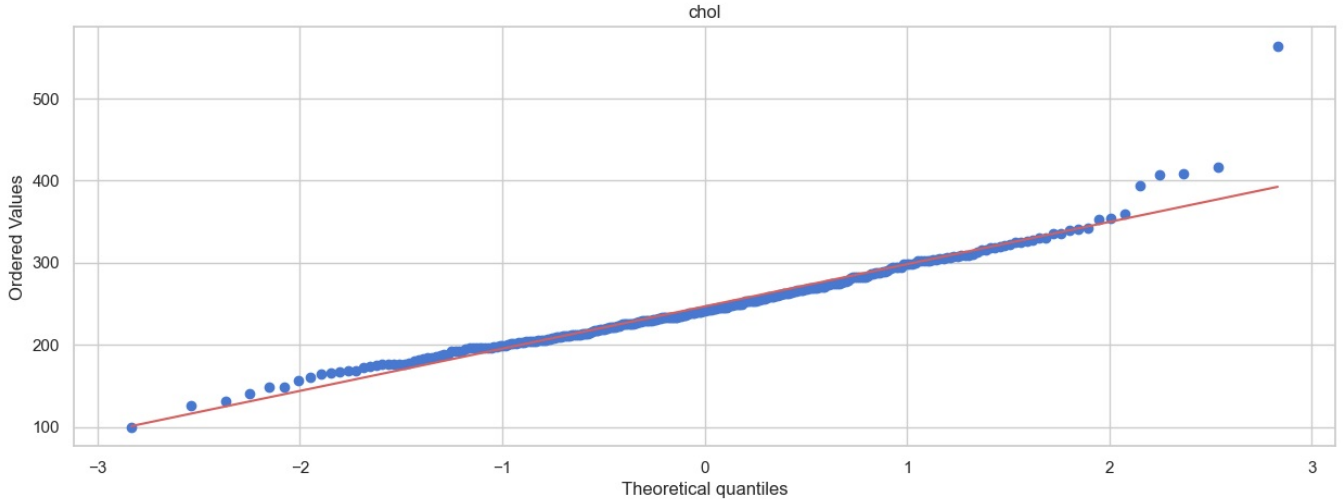
In [88]:
```
import scipy.stats as stats

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
```

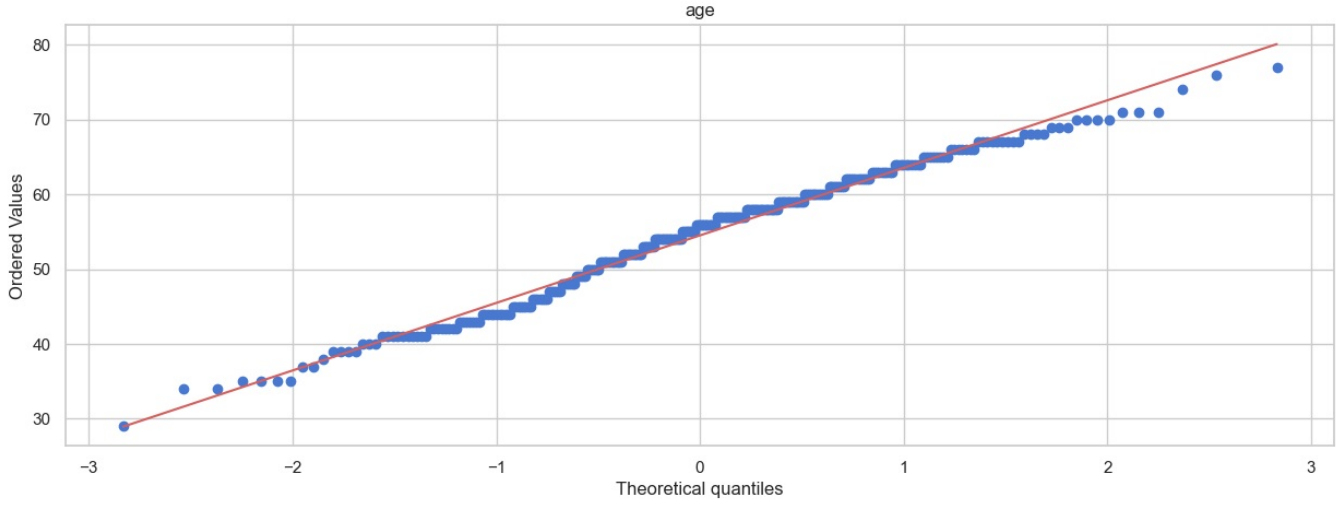
Out[88]:

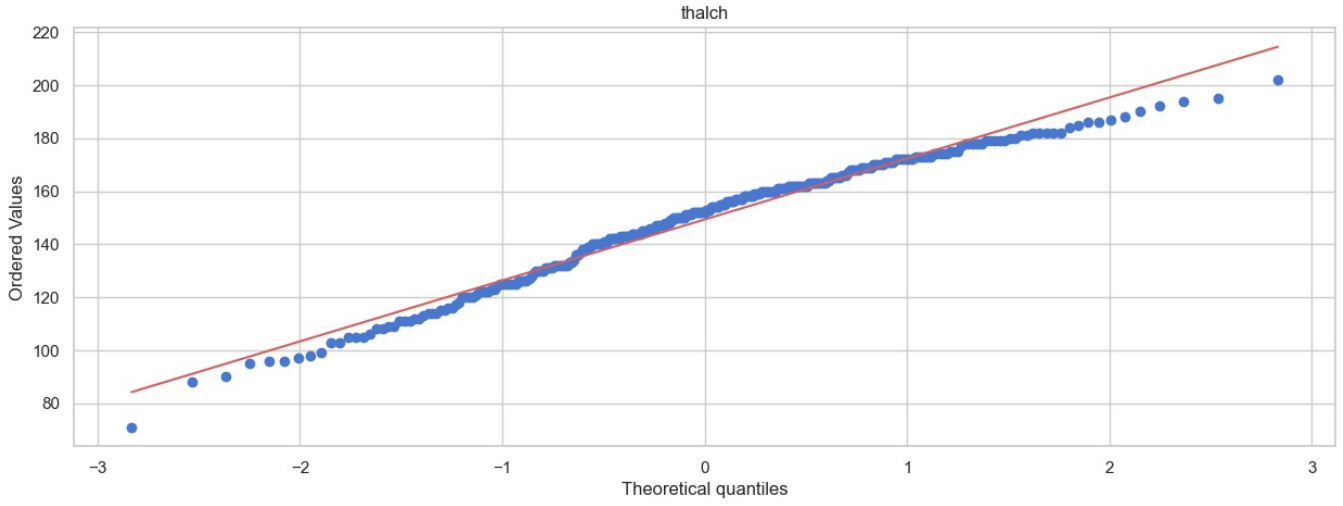
	Column	Skewness	Kurtosis
0	id	0.90	3.95
1	age	-0.21	-0.52
2	trestbps	0.70	0.80
3	chol	1.03	4.35
4	thalch	-0.53	-0.09
5	oldpeak	1.24	1.52
6	ca	1.19	0.26
7	num	1.05	-0.16

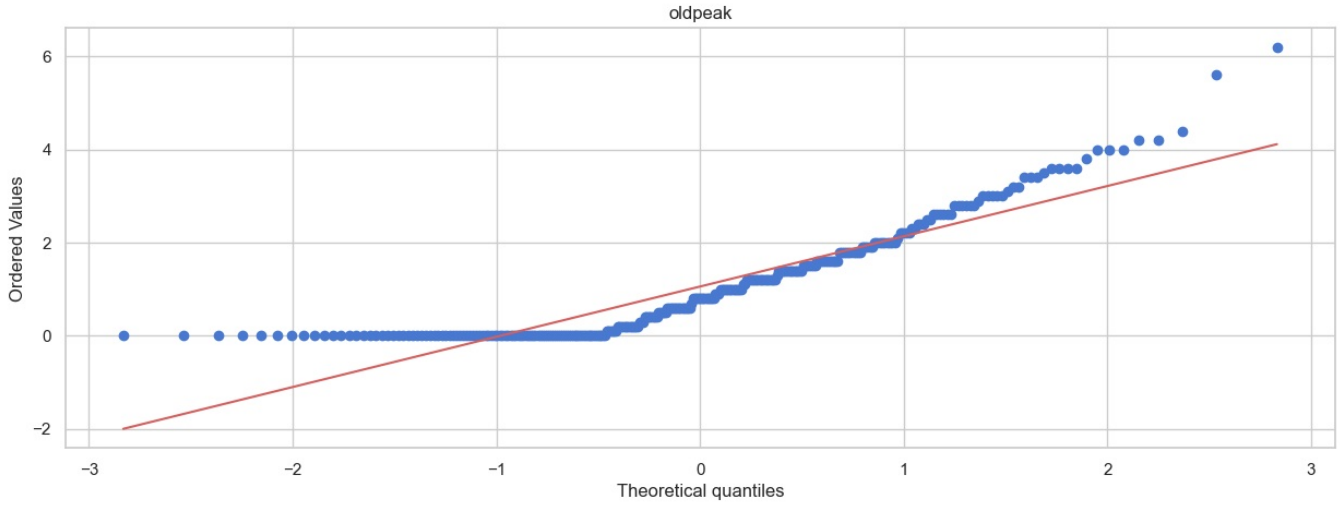
In [89]:

```
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();
```

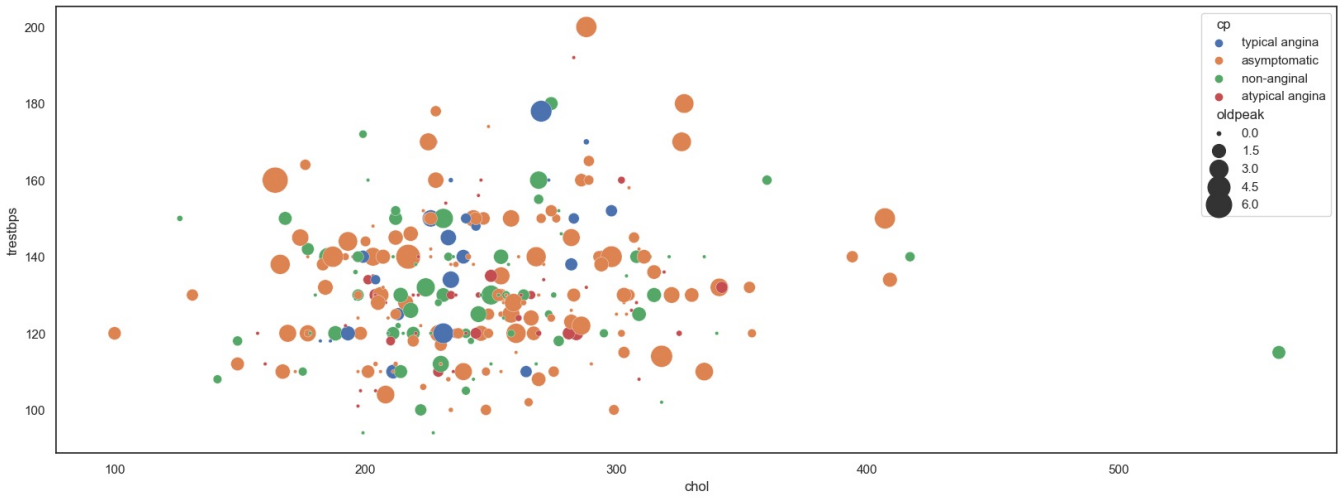




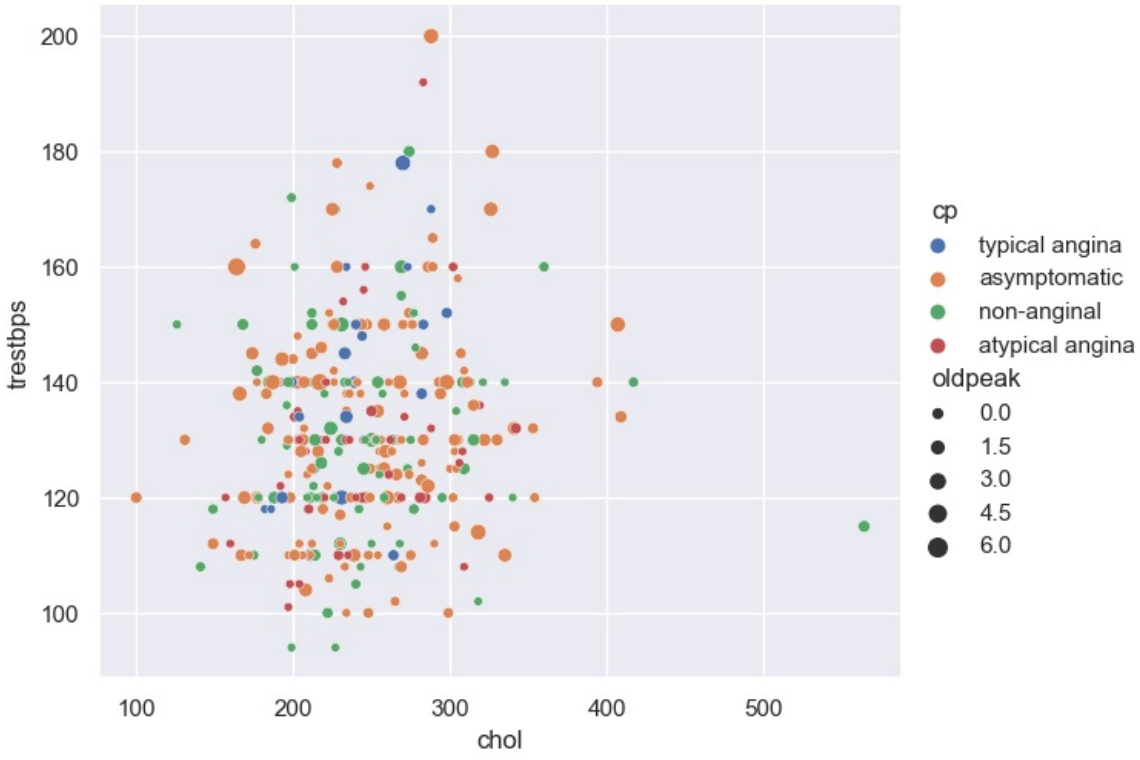




In [90]:
```
sns.set(rc={'figure.figsize':(20,7)})
sns.set_style("white")
sns.scatterplot(data=df, x="chol", y="trestbps", size="oldpeak", hue='cp',legend=True, sizes=(10, 500));
```

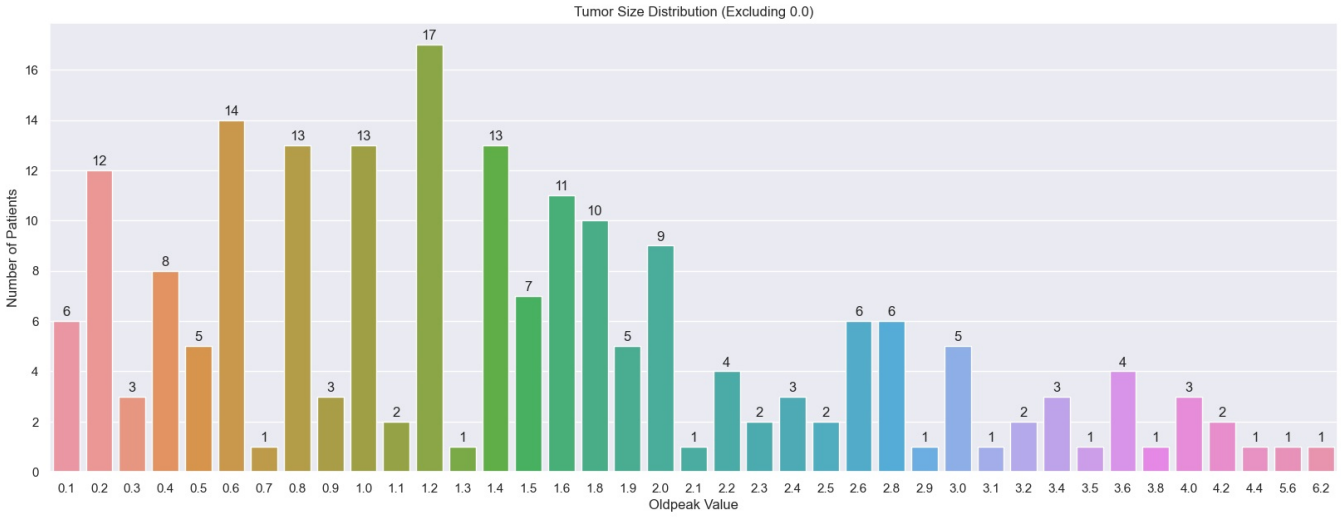


In [91]:
```
sns.set(rc={'figure.figsize':(20,7)})
sns.relplot(y='trestbps',x='chol',data=df,kind='scatter',size='oldpeak',hue='cp',aspect=1.2);
```



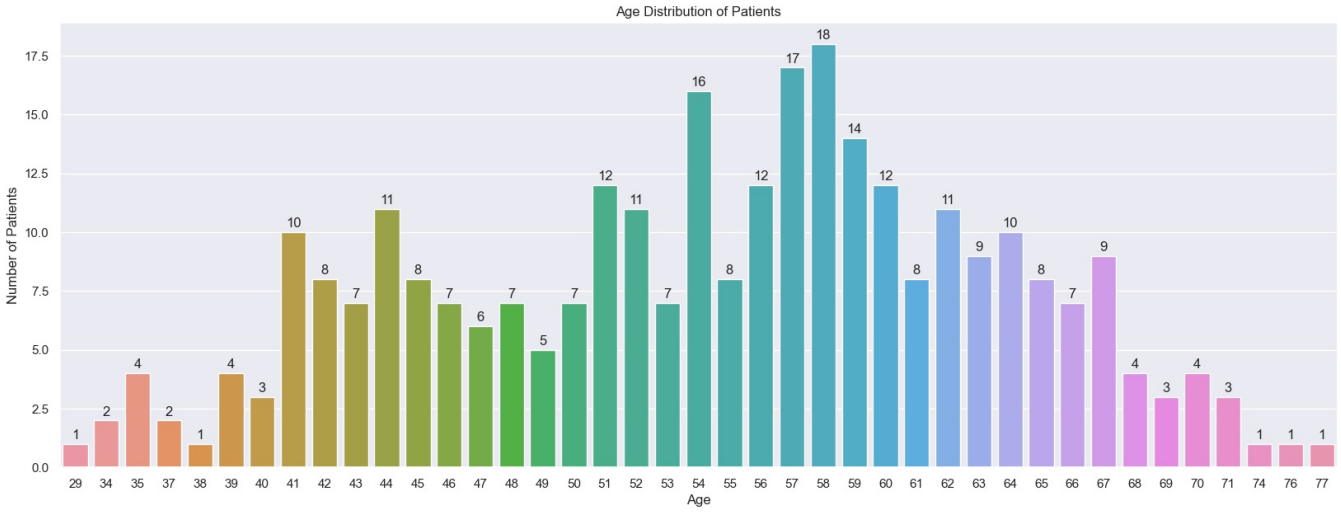
In [92]:
```
# Filter out rows with 'oldpeak' equal to 0.0
df_filtered = df[df['oldpeak'] != 0.0]

# 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=

# 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]:

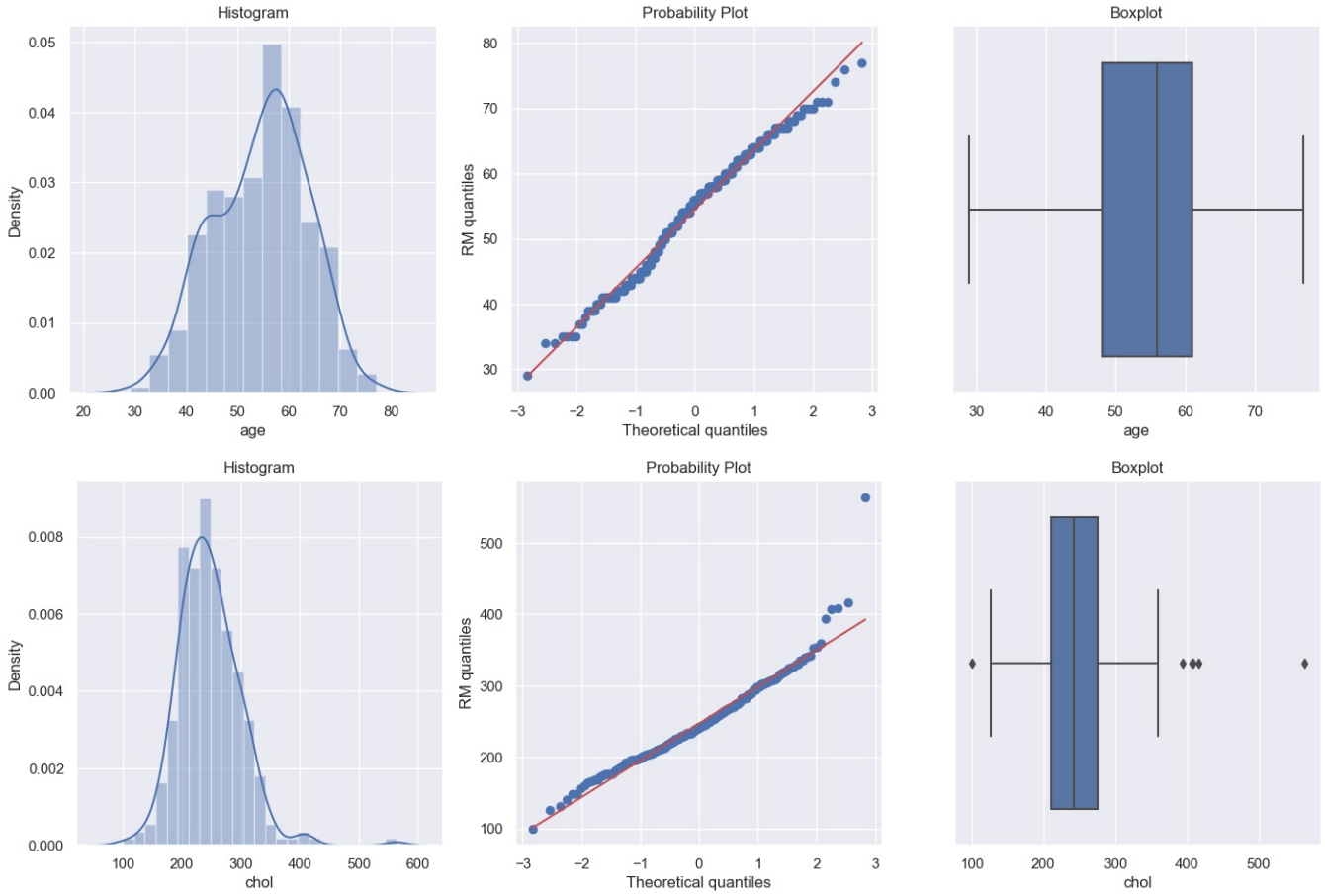
```
plt.figure(figsize=(20, 7))
# Create the countplot
sns.countplot(data=df, x='age', order=sorted(df['age'].unique()))

# 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=

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



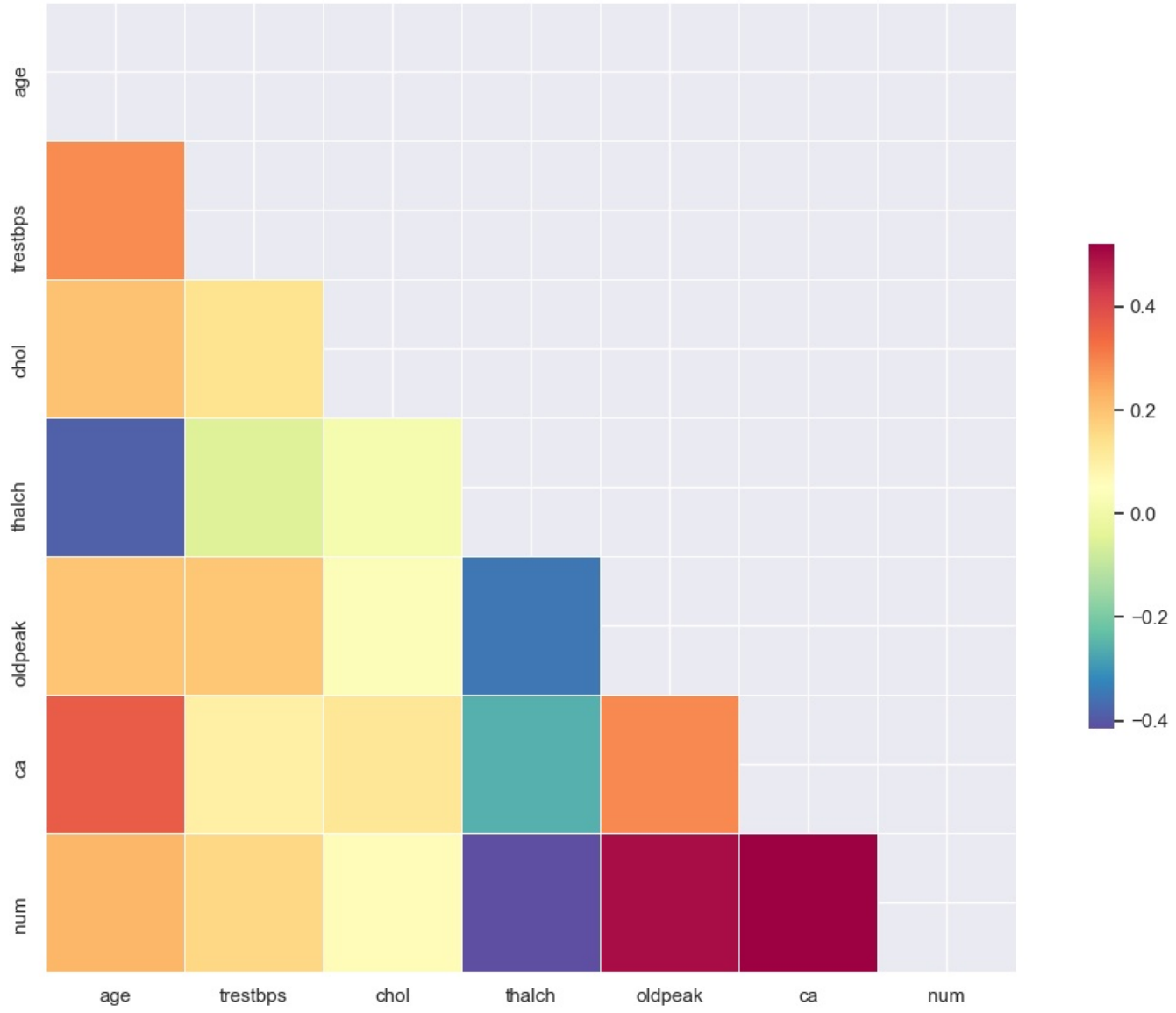
In [94]:

```
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="O").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.
```

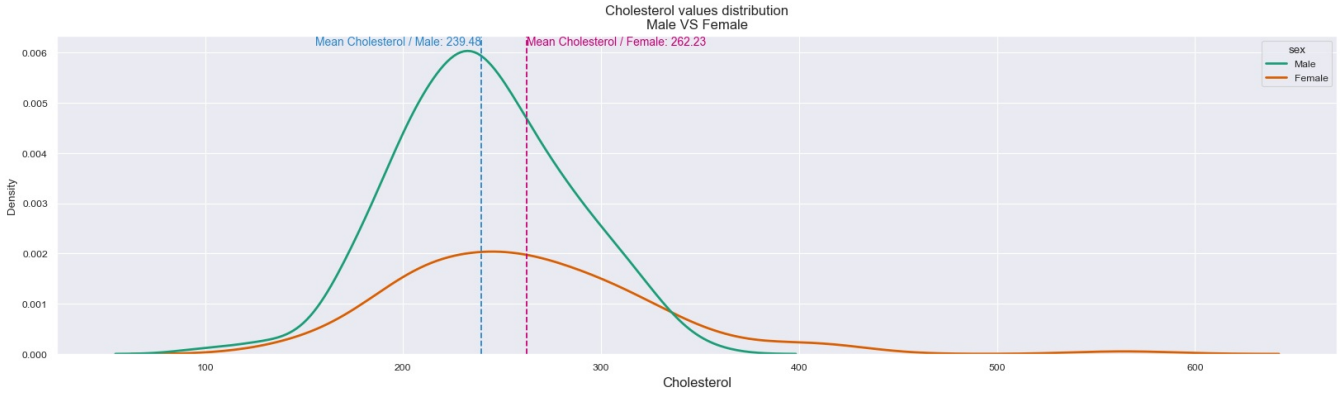


In [96]:
```
df = df.drop('id',axis=1)
df.head(2)
```

Out[96]:

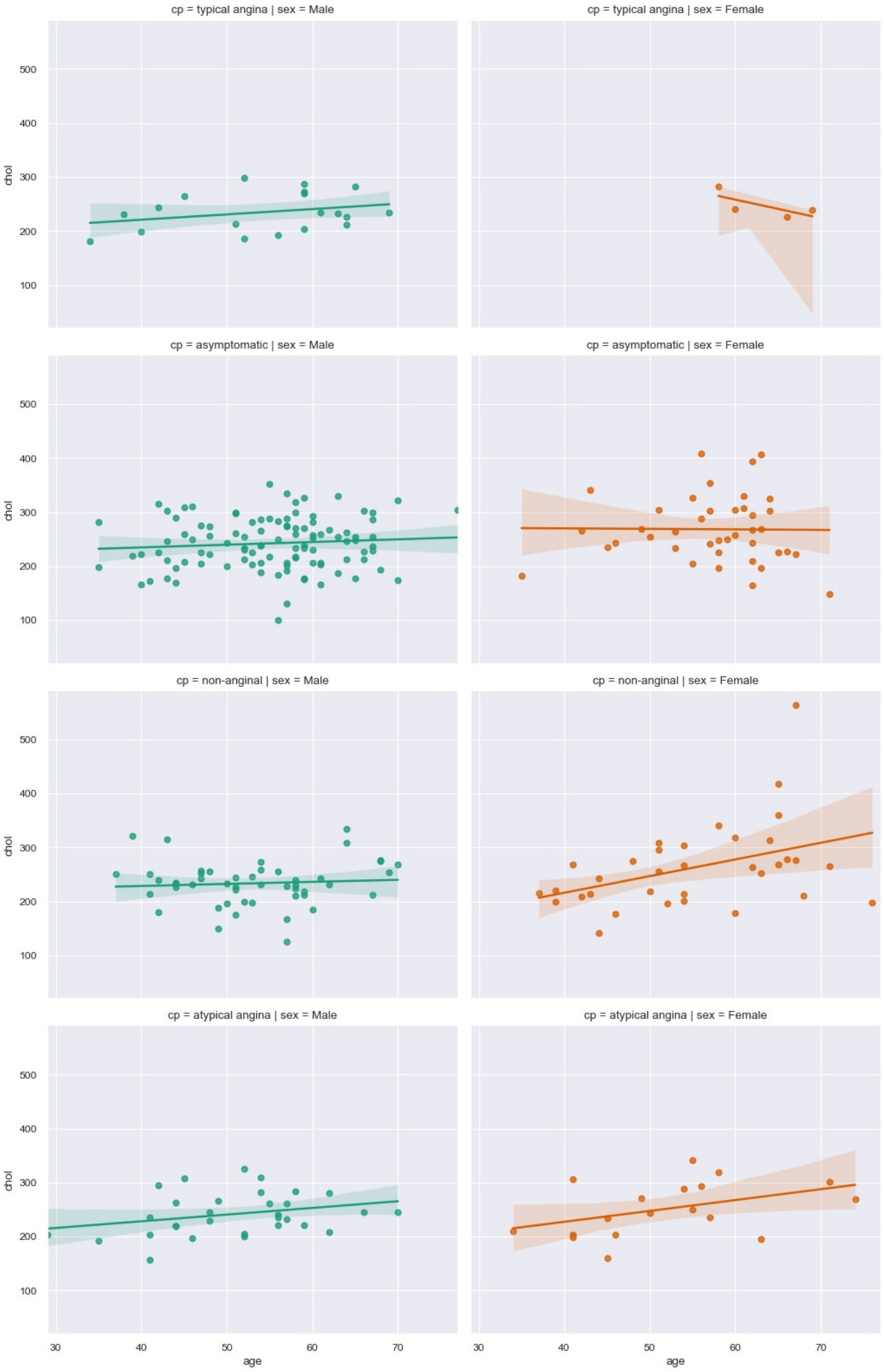
	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2

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()
```



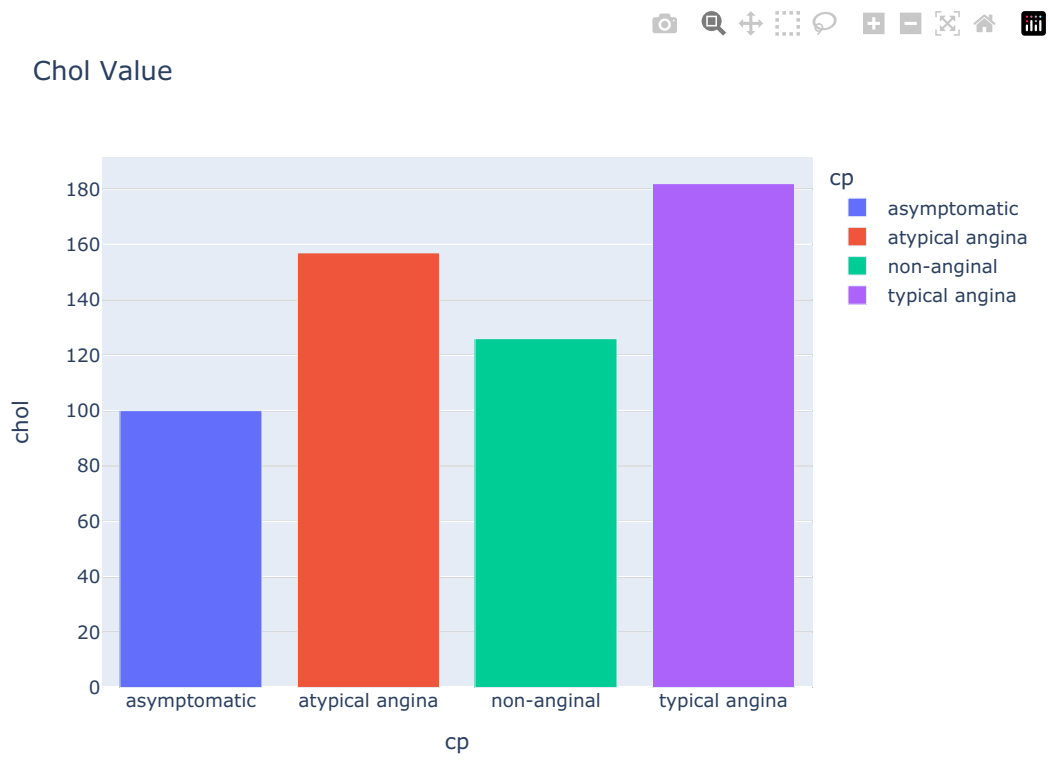
In [98]:

```
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()
```



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()
```

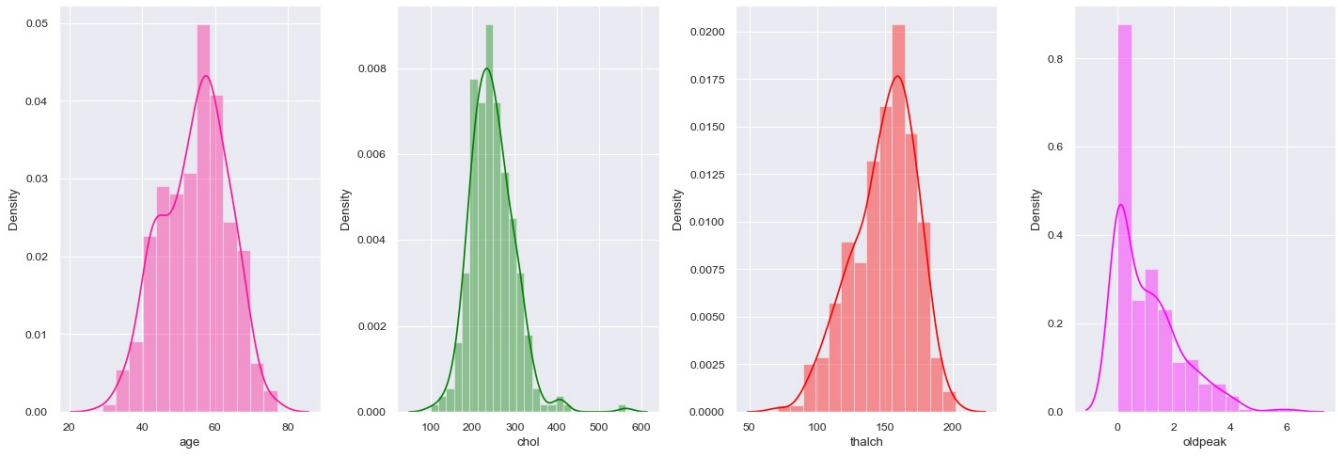


In [100…
```
df =  pd.read_csv("heart_disease_uci.csv")
df = df.dropna()
df = df.drop('id',axis =1 )
df.head(2)
```

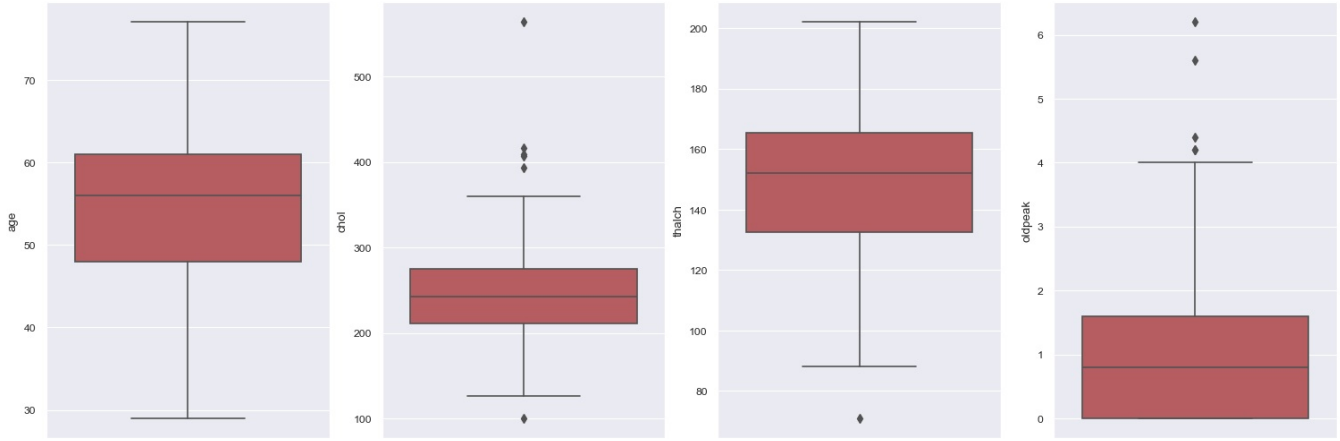
Out[100]:

	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

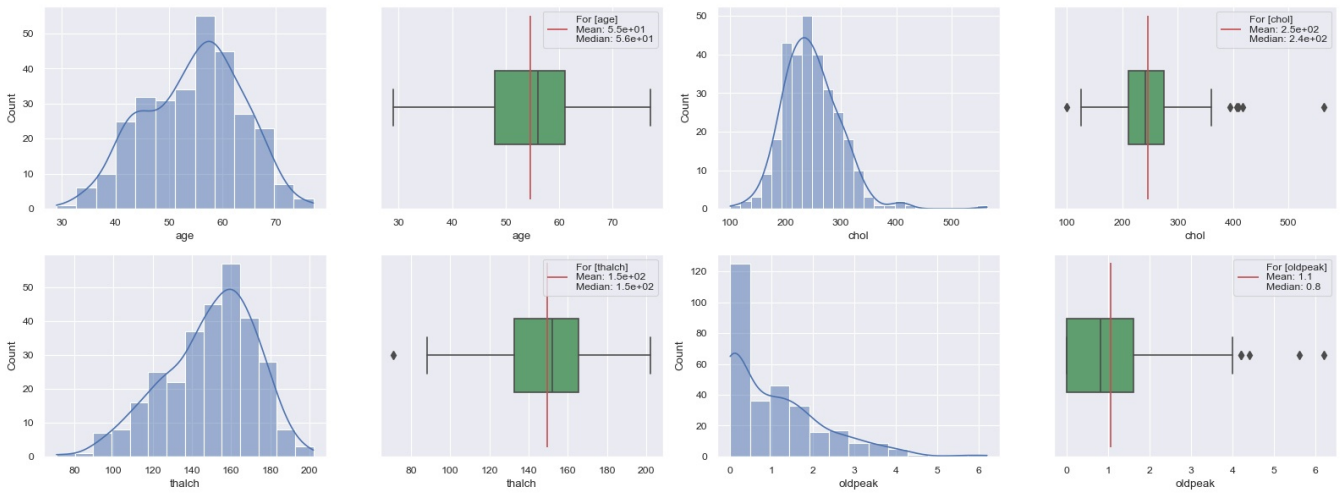
In [101…
```
plt.figure(figsize=(18,5))

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()
```



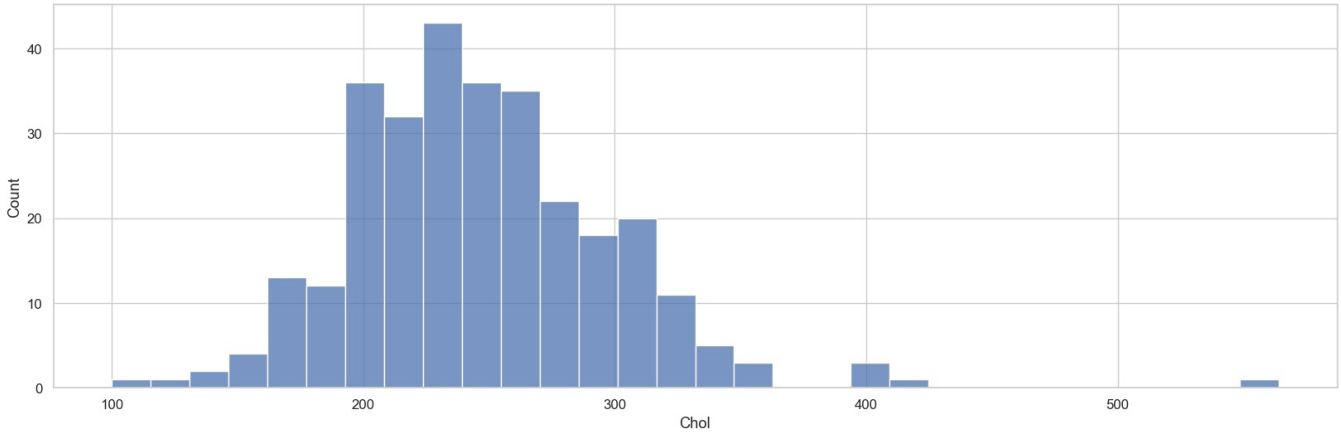
In [102…
```
df_cpy = df.copy("Deep")
df_cpy = df_cpy.select_dtypes("number")
df_cpy = df_cpy[['age','chol','thalch','oldpeak']]

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}\nMe
    axes.flatten()[2*i+1].legend()

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

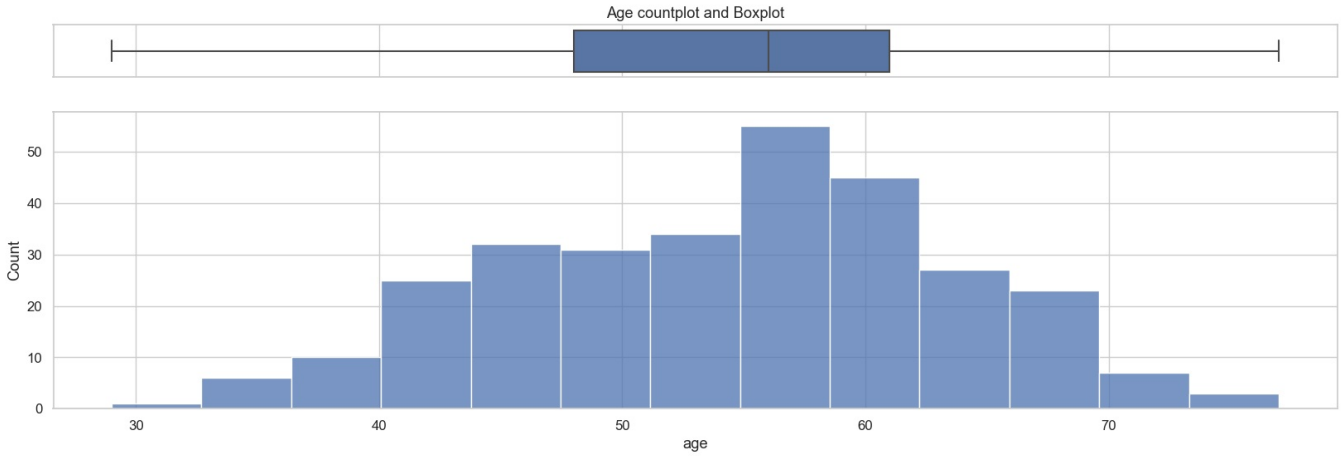


In [104…
```
sns.set(style='whitegrid', palette="deep", font_scale=1.1, rc={"figure.figsize": [20, 6]})

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



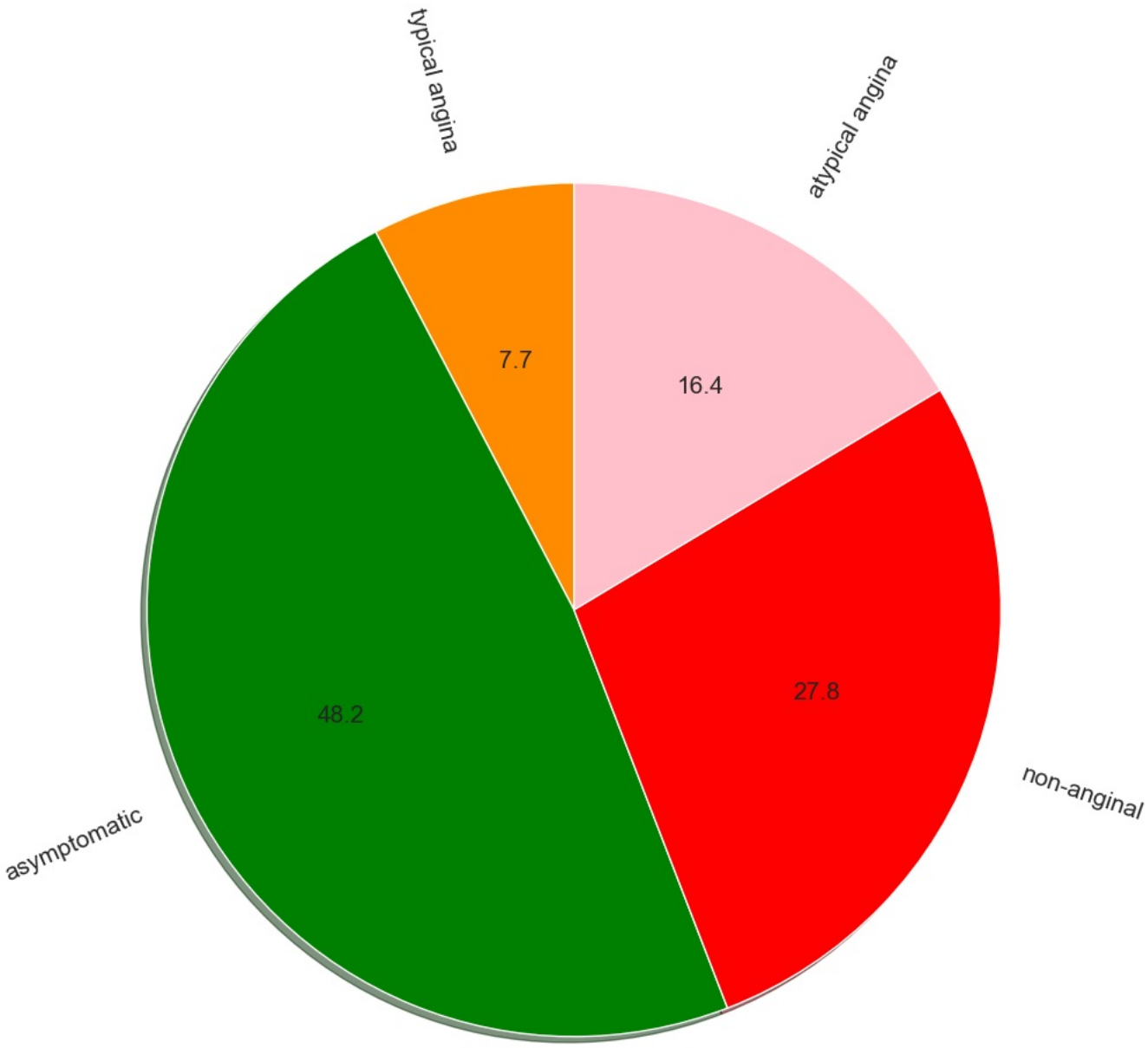
In [105…

```
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) ;
```

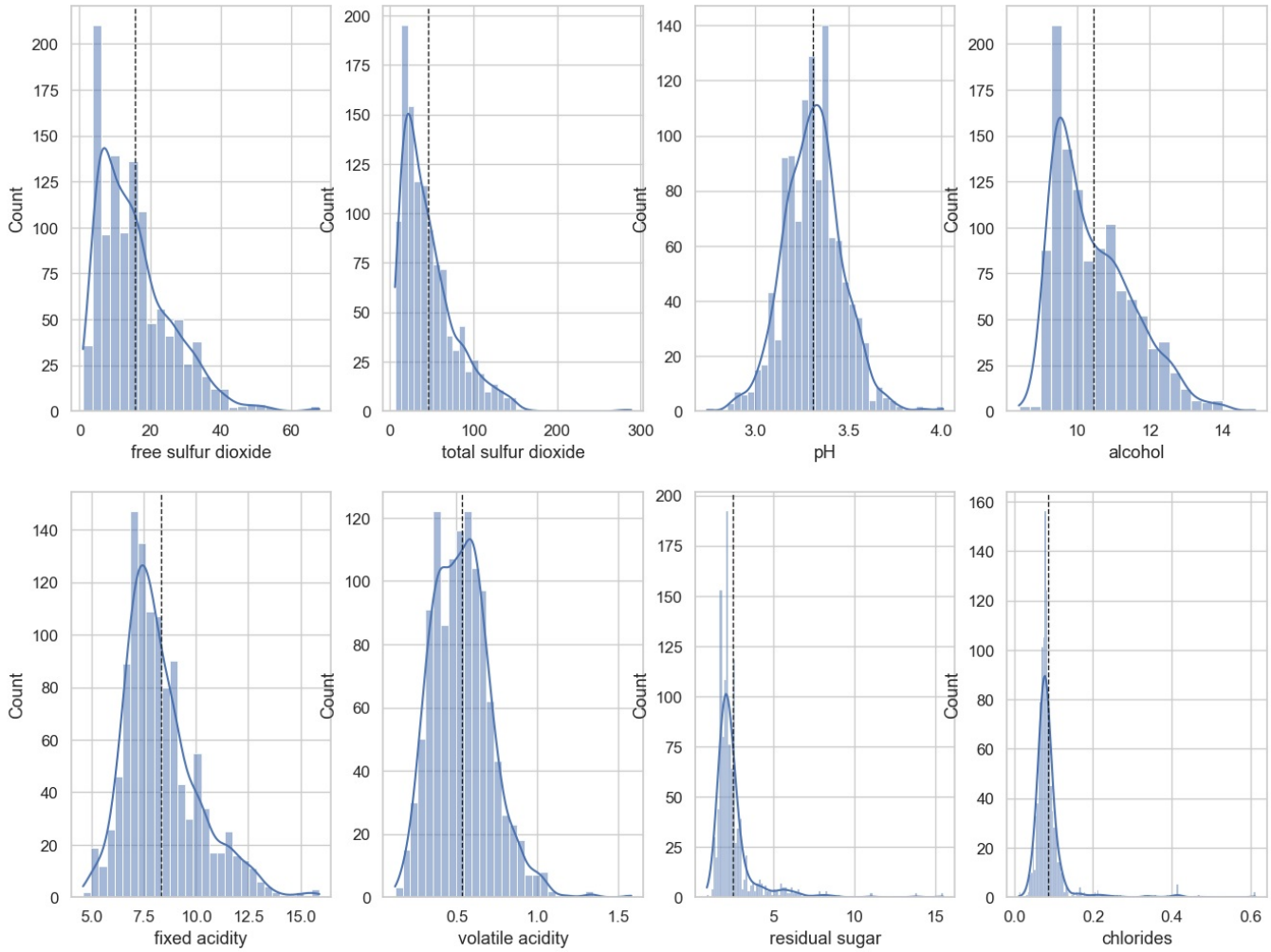


In [107…
```
wine = pd.read_csv("WineQT.csv")
wine.head(2)
```

Out[107]:

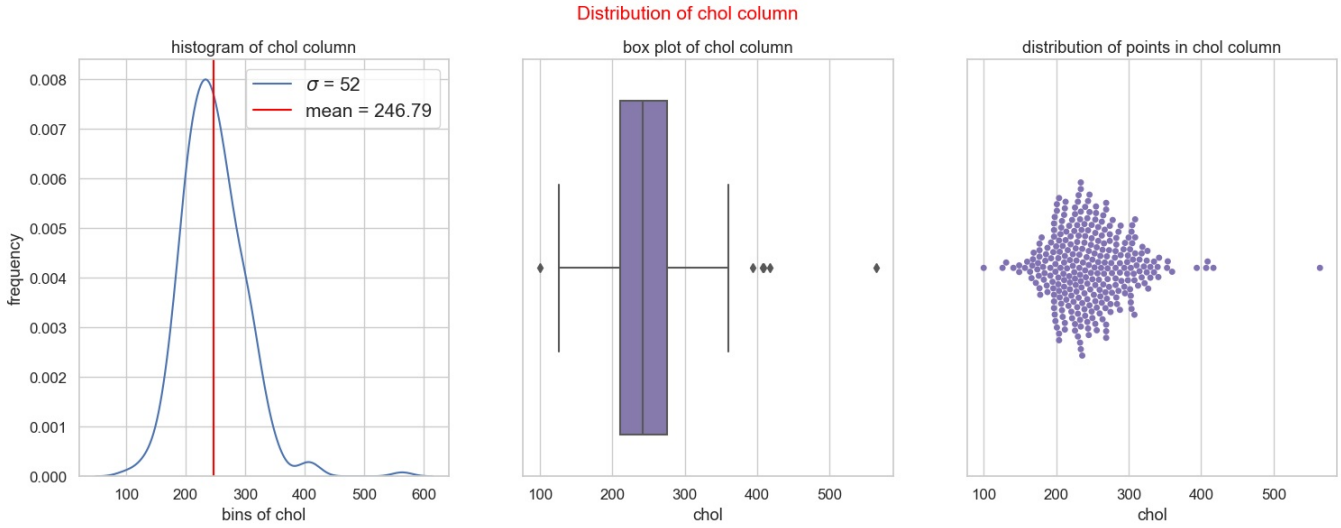
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.0	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	0
1	7.8	0.88	0.0	2.6	0.10	25.0	67.0	1.0	3.20	0.68	9.8	5	1

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)
```

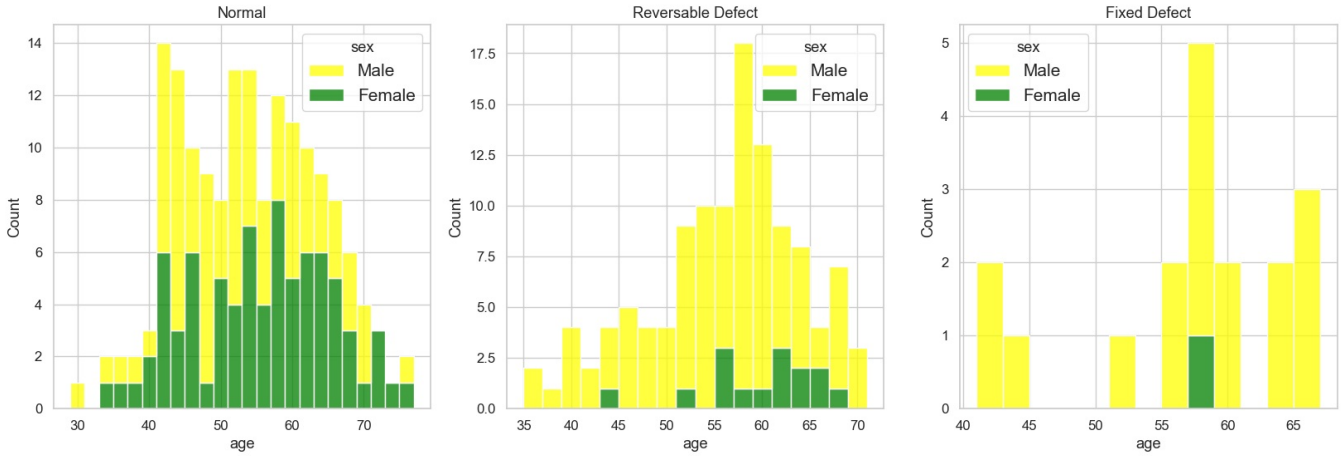


In [110…

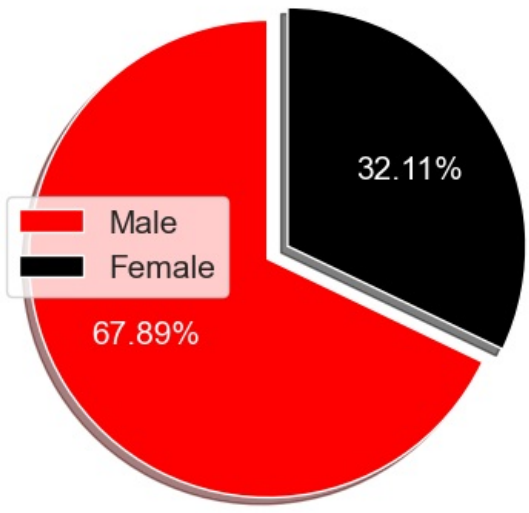
```
#set configuration for charts
plt.rcParams["figure.figsize"]=[18 , 6]
plt.rcParams["font.size"]=15
plt.rcParams["legend.fontsize"]="medium"
plt.rcParams["figure.titlesize"]="medium"

def plot_disribution(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"Ditstribution 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_disribution(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"]== 'reversable 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()
```

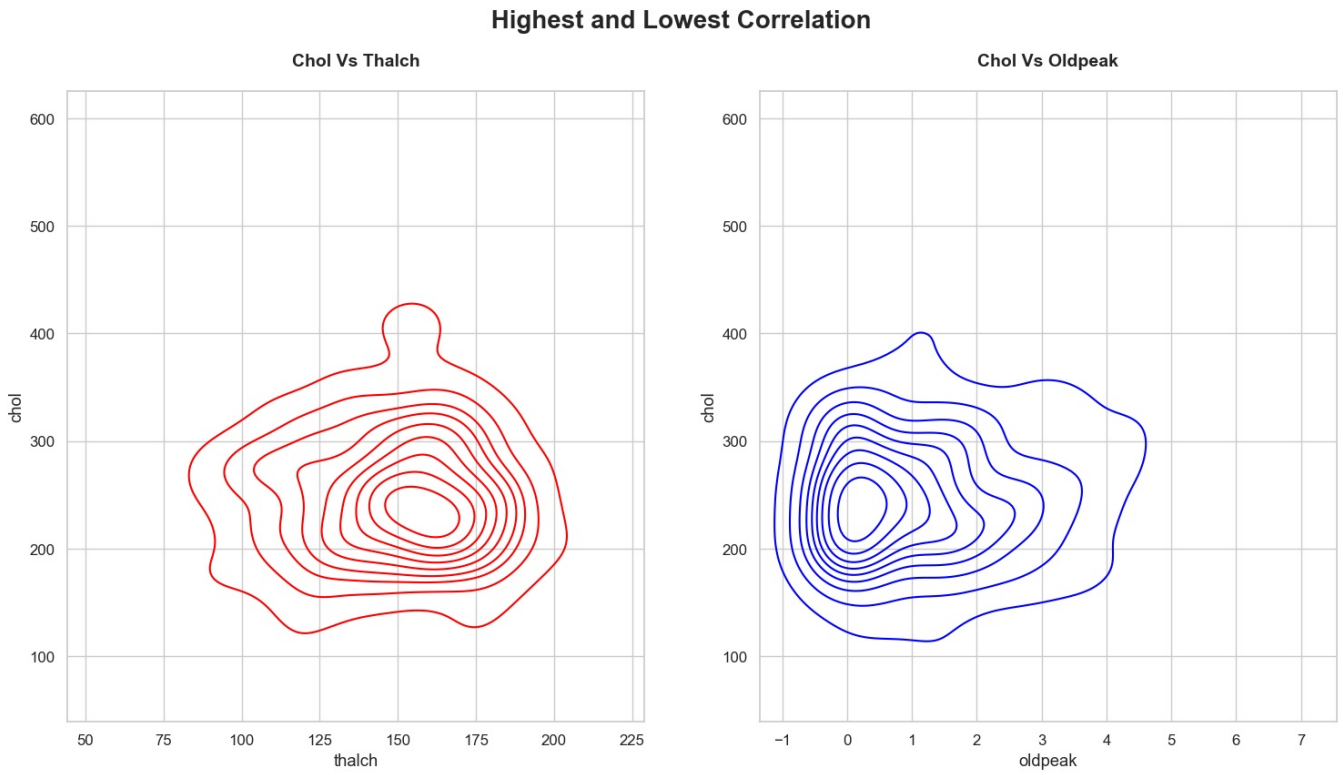


In [112…
```
sex = ["Male", "Female"]
values = df["sex"].value_counts()
color = ["#FF0000", "#000000"]

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);
```

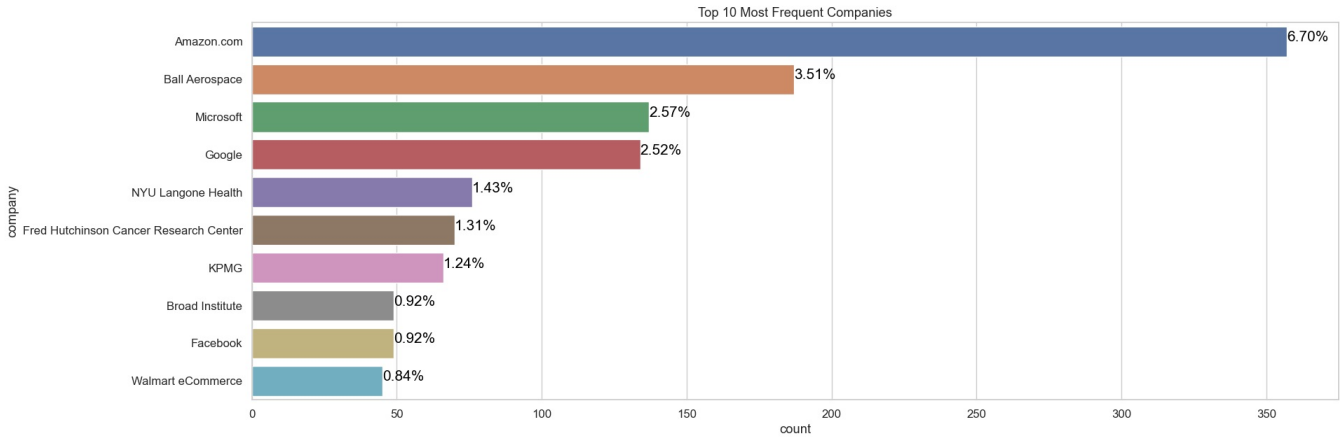


In [114…
```
df1 = pd.read_csv('US_Job_Market.csv')
df1 = df1.dropna().reset_index()
df1 = df1.drop('index',axis=1)
df1.head(2)
```

Out[114]:

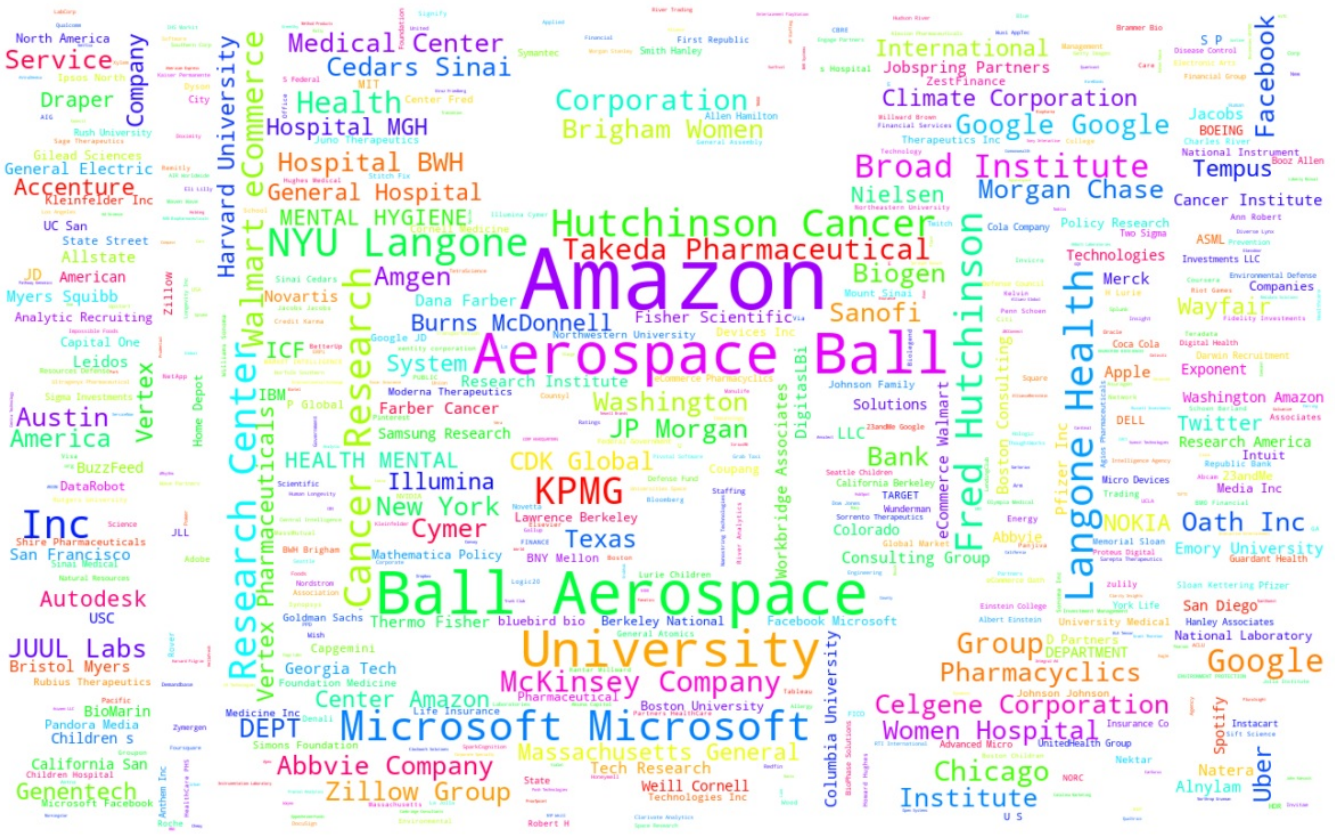
	position	company	description	reviews	location
0	Data Analyst	Operation HOPE	DEPARTMENT: Program OperationsPOSITION LOCATIO...	44.0	Atlanta, GA 30303
1	Assistant Professor -TT - Signal Processing & ...	Emory University	DESCRIPTION\nThe Emory University Department o...	550.0	Atlanta, GA

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=0)
plt.title('Top 10 Most Frequent Companies')
plt.show()
```



In [116…

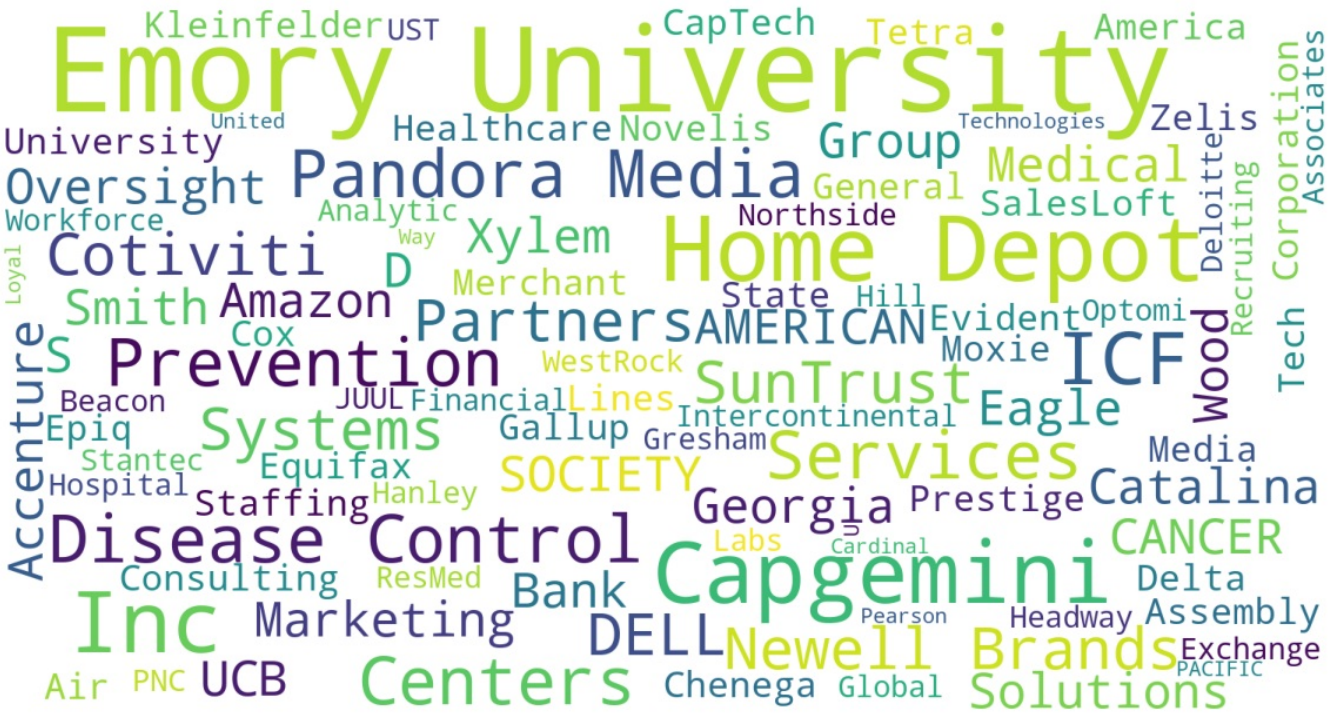
```
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
from wordcloud import WordCloud, STOPWORDS
import pandas as pd
import requests
from io import BytesIO

# 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…

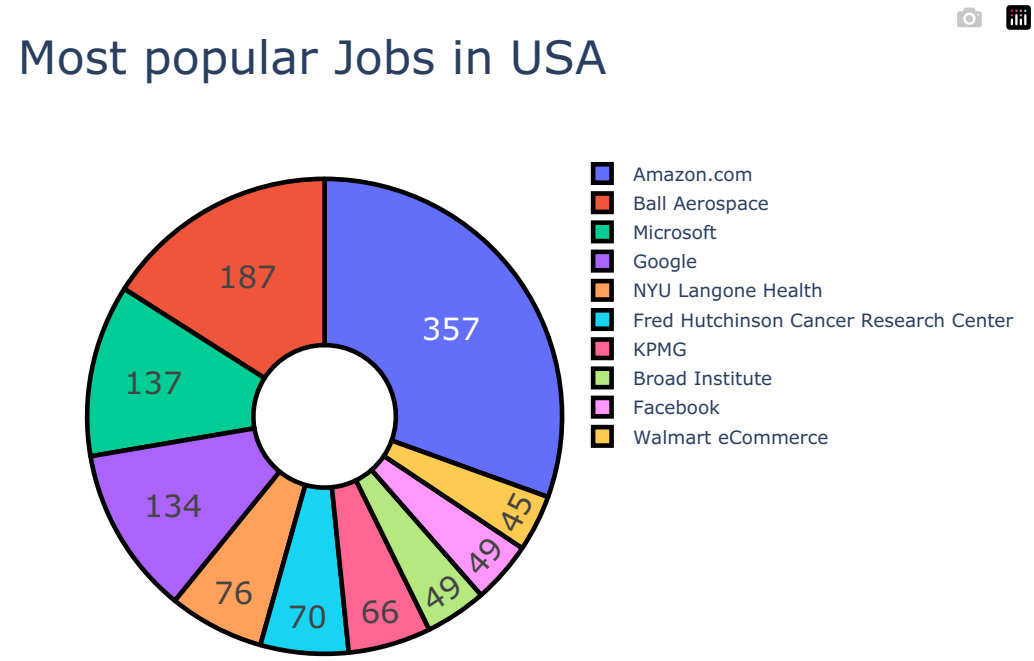
```
from wordcloud import WordCloud

# 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()
```

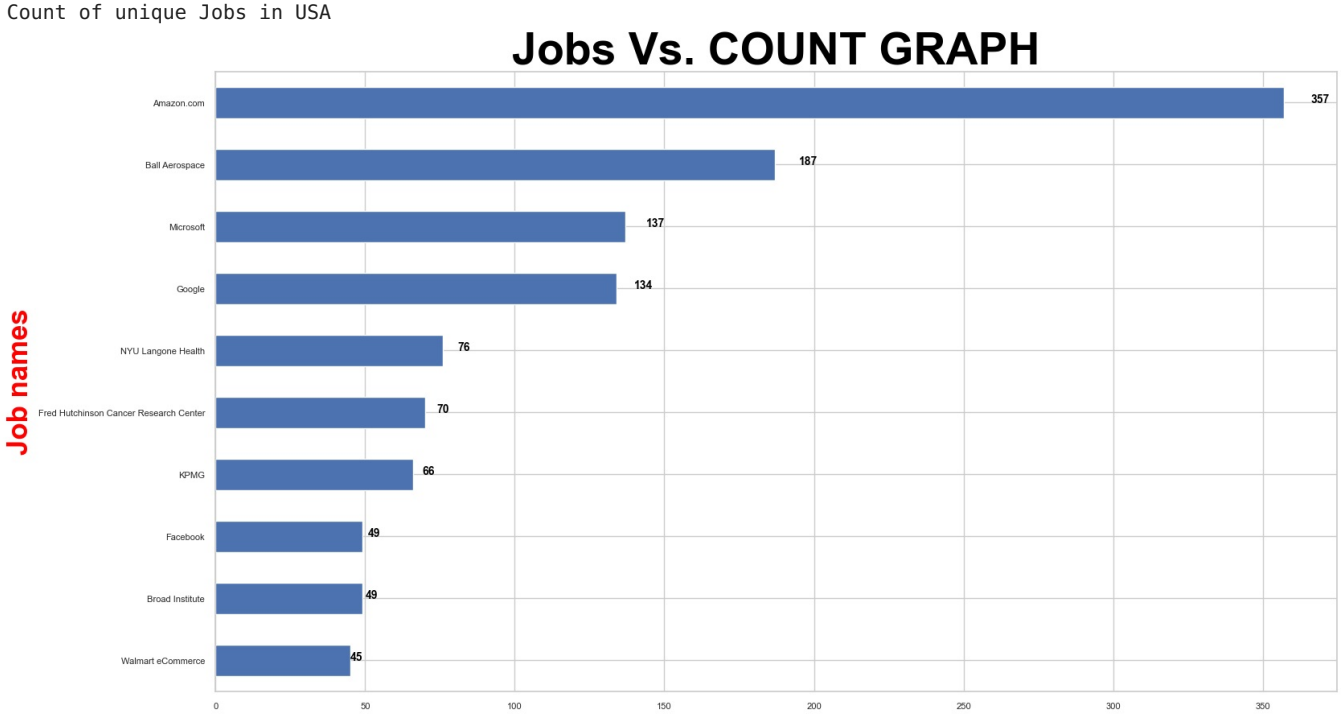


In [119…

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

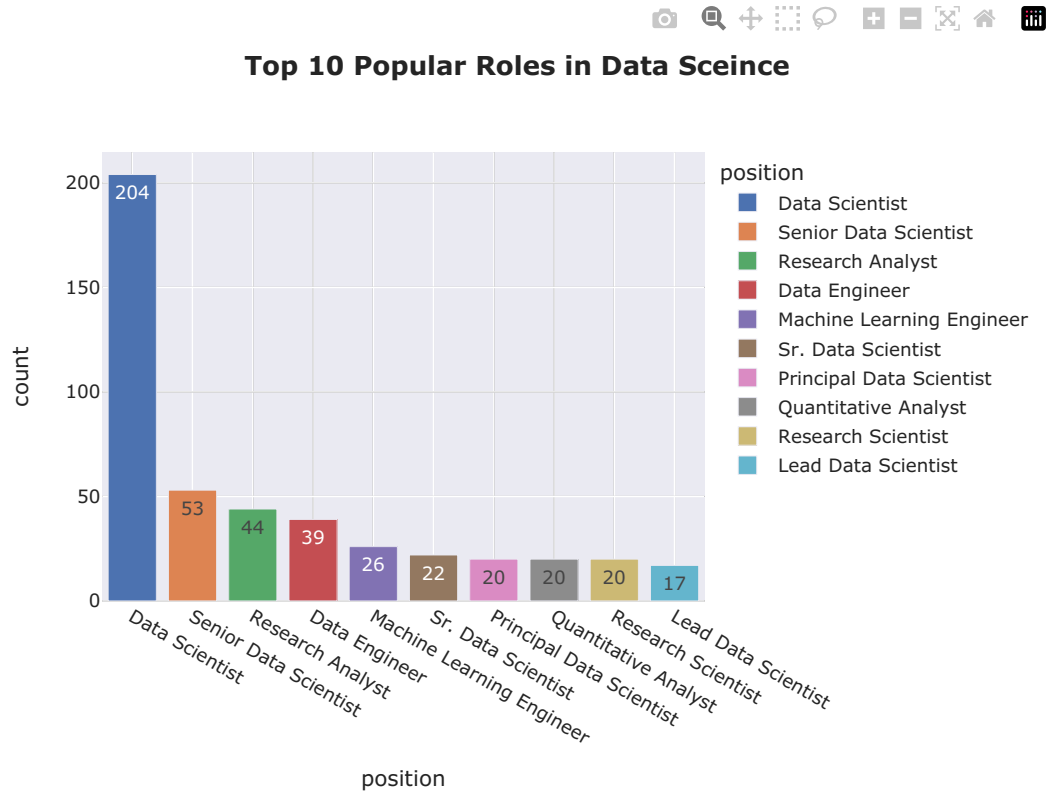
```
fig=plt.figure(figsize=(18,10))
locationCount.plot(kind="barh",fontsize=8)
plt.ylabel("Job names",fontsize=25,color="red",fontweight='bold')
plt.title("Jobs Vs. COUNT GRAPH",fontsize=40,color="BLACK",fontweight='bold')
for v in range(len(locationCount)):
    plt.text(v+locationCount[v],v,locationCount[v],fontsize=10,color="BLACK",fontweight='bold')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

```
Count of unique Jobs in USA
```



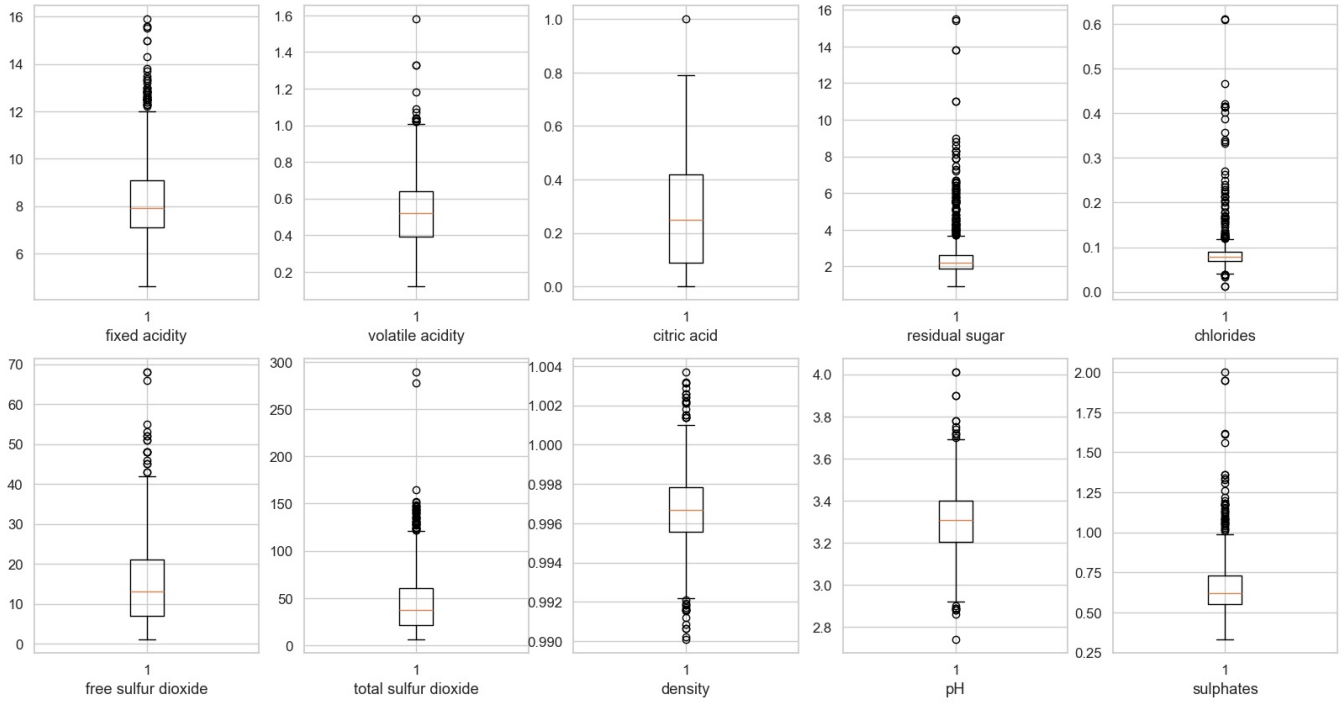
In [120…

```
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':'c
fig.show()
```

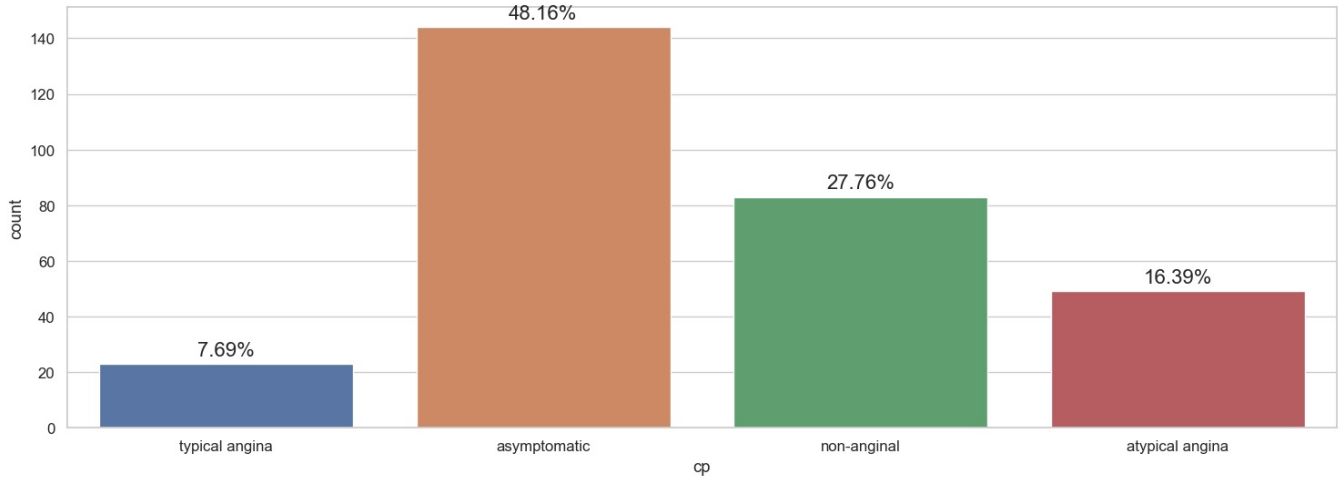


In [121…

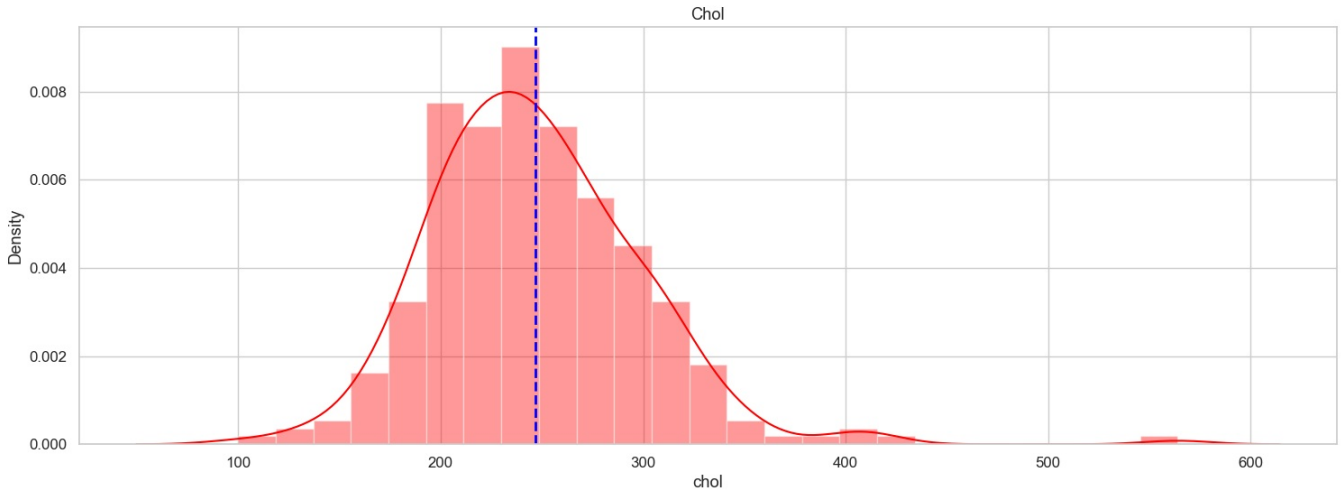
```
# Plotting Outliers
col = 1
plt.figure(figsize = (20, 10))
for i in wine.columns:
    if col < 11:
        plt.subplot(2, 5, col)
        plt.boxplot(wine[i])
        plt.xlabel(i)
    col = col + 1
```



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)\
                    .style.background_gradient(cmap='GnBu')\
```

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