Question 1

```
import numpy as np
import pandas as pd
import plotly.graph_objects as go
import plotly.express as px

data = {
'Company': ['Apple', 'Microsoft', 'Amazon', 'Google', 'Facebook'],
'Revenue_2022': [394, 198, 513, 280, 117], # Revenue in billion
dollars
'Revenue_2023': [420, 215, 540, 310, 130] # Revenue in billion dollars
}
df = pd.DataFrame(data)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=df['Company'],
    y=df['Revenue_2022'],
    name='2022',
    marker_color='royalblue'
))

fig.add_trace(go.Bar(
    x=df['Company'],
    y=df['Revenue_2023'],
    name='2023',
    marker_color='lightcoral'
))

fig.update_layout(
    title='Comparative Annual Revenue (2022 vs 2023)',
    title_x=0.5,
    xaxis_title='Company',
    yaxis_title='Revenue (Billion USD)',
    barmode='group',
    legend_title='Year',
    template='plotly_white',
    height=600,
    width=900
)

for trace in fig.data:
    y_data = trace.y
    x_data = trace.x
    for i, y in enumerate(y_data):
        fig.add_annotation(
            x=x_data[i],
```

```
            y=y,
            text=f"${y}B",
            showarrow=False,
            yshift=10,
            font=dict(size=12)
        )

fig.show()
```

Question 2

```
# Create DataFrame from the data
data = {
    'Advertising Budget': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
# in thousand dollars
    'Sales Revenue': [15, 25, 40, 50, 65, 80, 85, 100, 120, 140]
# in thousand dollars
}

df = pd.DataFrame(data)

fig = px.scatter(
    df,
    x='Advertising Budget',
    y='Sales Revenue',
    title='Advertising Budget vs Sales Revenue'
)

fig.update_traces(
    marker=dict(
        color='blue',
        size=12,
        symbol='circle',
        line=dict(width=2, color='darkblue')
    )
)

fig.update_layout(
    xaxis_title='Advertising Budget (thousand dollars)',
    yaxis_title='Sales Revenue (thousand dollars)',
    width=700,
    height=500
)

fig.show()
```

Question 3

```
# Create DataFrame from the data
data = {
    'Brand': ['Apple', 'Samsung', 'Xiaomi', 'Oppo', 'Vivo'],
    'Market Share': [30, 28, 17, 12, 13]  # Percentage of total market
}

df = pd.DataFrame(data)

fig = px.pie(
    df,
    values='Market Share',
    names='Brand',
    title='Smartphone Market Share',
    hover_data={'Market Share': ':.1f%'},
)

fig.update_traces(
    textposition='inside',
    textinfo='percent',
    hoverinfo='label+percent',
    marker=dict(line=dict(color='white', width=2))
)

fig.update_layout(
    showlegend=True,
    width=600,
    height=500
)

fig.show()
```

Question 4

```
# Create the dataset
data = {
    'Job Sector': ['IT', 'Finance', 'Healthcare', 'Education',
'Retail'] * 5,
    'Salary': [75000, 85000, 62000, 48000, 40000, 77000, 90000, 65000,
50000, 42000,
               78000, 87000, 67000, 52000, 43000, 80000, 89000, 69000,
54000, 45000,
               82000, 91000, 71000, 56000, 47000]  # Salary in USD
}

df = pd.DataFrame(data)
```

```
fig = px.box(
    df,
    x='Job Sector',
    y='Salary',
    color='Job Sector',
    points='all',
    title='Salary Distribution Across Job Sectors',
    labels={'Salary': 'Annual Salary (USD)'},
    height=600,
    width=800
)

fig.update_layout(
    xaxis_title='Job Sector',
    yaxis_title='Annual Salary (USD)',
    showlegend=False,
    plot_bgcolor='rgba(240, 240, 240, 0.5)',
    font=dict(size=12),
    boxmode='group'
)

fig.update_traces(
    boxpoints='outliers',
    jitter=0.3,
    pointpos=-1.8,
    marker=dict(
        size=8,
        line=dict(width=2, color='DarkSlateGrey')
    ),
    line=dict(width=2),
    fillcolor='rgba(255,255,255,0.8)'
)

fig.show()
```

Question 5

```
s = """GDP Inflation Unemployment Interest_Rate
19352.46582 0.592630224 10.34223474 6.467903667
47585.00101 4.864594335 4.673926328 2.534717113
36867.70315 4.245991884 6.505735782 1.585464337
30334.26573 1.455525998 7.39634212 9.539969835
8644.913382 1.318212352 8.472839811 9.690688298
8643.731496 1.325320294 12.42211154 8.275576133
3846.096996 1.869090093 5.396085386 3.741523923
43442.63114 2.861403942 9.170813261 1.879049026
30454.63558 2.443752584 10.10897483 7.158097239
35695.55631 1.810531131 3.557404953 4.961372444"""
```

```
l = [i.split(" ") for i in s.split('\n')]
df = pd.DataFrame(l[1:], columns=l[0])

corr_matrix = df.corr()

fig = px.imshow(
    corr_matrix,
    text_auto=True,
    color_continuous_scale='RdBu_r',
    aspect="auto",
    title='Correlation Matrix of Financial Indicators',
    zmin=-1,
    zmax=1
)

fig.update_layout(
    width=700,
    height=600,
    coloraxis_colorbar=dict(
        title='Correlation',
        thicknessmode="pixels", thickness=20,
        lenmode="pixels", len=300,
        tickvals=[-1, -0.5, 0, 0.5, 1],
        ticktext=['-1.0', '-0.5', '0.0', '0.5', '1.0']
    ),
    margin=dict(l=60, r=50, t=80, b=50)
)


fig.show()
```

Question 6

```
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Create the dataset
data = {
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun'],
    'Product A Sales': [500, 600, 700, 800, 750, 780],
    'Product B Sales': [400, 450, 470, 490, 520, 550],
    'Profit': [50, 80, 100, 120, 110, 130]  # Profit in thousand
dollars
}

# Convert to DataFrame
```

```
df = pd.DataFrame(data)

fig = make_subplots(
    rows=1,
    cols=3,
    subplot_titles=('Line Chart', 'Bar Chart', 'Scatter Plot')
)

fig.add_trace(
    go.Scatter(
        x=df['Month'],
        y=df['Product A Sales'],
        mode='lines+markers',
        name='Product A'
    ),
    row=1, col=1
)

fig.add_trace(
    go.Scatter(
        x=df['Month'],
        y=df['Product B Sales'],
        mode='lines+markers',
        name='Product B'
    ),
    row=1, col=1
)

fig.add_trace(
    go.Bar(
        x=df['Month'],
        y=df['Profit'],
        name='Profit'
    ),
    row=1, col=2
)

fig.add_trace(
    go.Scatter(
        x=df['Product A Sales'],
        y=df['Product B Sales'],
        mode='markers',
        name='Product A vs B'
    ),
    row=1, col=3
)

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

```
)

fig.show()
```

Question 7

```
import pandas as pd
import plotly.express as px

# Create the dataset
data = {
    'Country': ['USA', 'China', 'India', 'Germany', 'Brazil'] * 3,
    'Year': [2000, 2000, 2000, 2000, 2000, 2010, 2010, 2010, 2010,
2010, 2020, 2020, 2020, 2020, 2020],
    'GDP': [10, 5, 2, 3, 1, 15, 9, 5, 4, 2, 22, 14, 7, 5, 3],  # GDP
in trillion dollars
    'Population': [280, 1260, 1000, 83, 175, 310, 1350, 1200, 82, 190,
331, 1440, 1380, 80, 210],  # in million
    'Life_Expectancy': [77, 71, 65, 80, 68, 79, 74, 69, 82, 72, 81,
76, 72, 83, 75]  # in years
}

df = pd.DataFrame(data)

fig = px.scatter(
    df,
    x="Population",
    y="Life_Expectancy",
    size="GDP",
    color="Country",
    animation_frame="Year",
    size_max=60,
    range_x=[0, 1500],
    range_y=[60, 85],
    labels={
        "Population": "Population (million)",
        "Life_Expectancy": "Life Expectancy (years)",
        "GDP": "GDP (trillion $)"
    },
    title="Economic Growth: GDP, Population and Life Expectancy (2000-
2020)"
)

fig.update_layout(
    width=800,
    height=600
)
```

```
fig.show()
```

Question 8

```
import pandas as pd
import plotly.graph_objects as go
import networkx as nx

# Create the dataset
data = {
    'User A': ['Alice', 'Alice', 'Bob', 'Charlie', 'David', 'Eve',
'Frank', 'Grace', 'Hannah', 'Ivan'],
    'User B': ['Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace',
'Hannah', 'Ivan', 'Alice', 'Bob'],
    'Connection Strength': [1, 2, 3, 1, 4, 5, 2, 3, 4, 5]  #
Represents the strength of the relationship
}

df = pd.DataFrame(data)

G = nx.Graph()

for index, row in df.iterrows():
    G.add_edge(row['User A'], row['User B'], weight=row['Connection
Strength'])

pos = nx.spring_layout(G, seed=42)

edge_x = []
edge_y = []
edge_weights = []

for edge in G.edges():
    x0, y0 = pos[edge[0]]
    x1, y1 = pos[edge[1]]
    edge_x.extend([x0, x1, None])
    edge_y.extend([y0, y1, None])
    edge_weights.append(G.edges[edge]['weight'])

edges = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=1, color='#888'),
    hoverinfo='none',
    mode='lines')

node_x = []
node_y = []
```

```
for node in G.nodes():
    x, y = pos[node]
    node_x.append(x)
    node_y.append(y)

node_adjacencies = []
node_text = []
for node in G.nodes():
    adjacencies = list(G.neighbors(node))
    node_adjacencies.append(len(adjacencies))
    connections = ", ".join([f"{adj} (strength: {G.edges[node, adj]
['weight']})" for adj in adjacencies])
    node_text.append(f"{node}<br>Connections:
{len(adjacencies)}<br>{connections}")

nodes = go.Scatter(
    x=node_x, y=node_y,
    mode='markers',
    hoverinfo='text',
    text=node_text,
    marker=dict(
        showscale=True,
        colorscale='YlGnBu',
        color=[],
        size=20,
        colorbar=dict(
            thickness=15,
            title='Node Connections',
            xanchor='left',
            titleside='right'
        ),
        line=dict(width=2)))

nodes.marker.color = node_adjacencies

fig = go.Figure(data=[edges, nodes],
                layout=go.Layout(
                    title='User Connection Network',
                    titlefont=dict(size=16),
                    showlegend=False,
                    hovermode='closest',
                    margin=dict(b=20, l=5, r=5, t=40),
                    xaxis=dict(showgrid=False, zeroline=False,
showticklabels=False),
                    yaxis=dict(showgrid=False, zeroline=False,
showticklabels=False),
                    width=800,
                    height=600
                ))
```

```
for node, (x, y) in pos.items():
    fig.add_annotation(
        x=x,
        y=y,
        text=node,
        showarrow=False,
        font=dict(size=12),
        bgcolor="white",
        bordercolor="black",
        borderwidth=1,
        borderpad=1,
        xshift=0,
        yshift=20
    )

fig.show()
```

Question 9

```
from sklearn.linear_model import LinearRegression
from datetime import datetime, timedelta

s = """Date Stock_Price
01-01-2024 102.4836
02-01-2024 100.3188
03-01-2024 105.2586
04-01-2024 110.6455
05-01-2024 102.8696
06-01-2024 103.8798
07-01-2024 113.9567
08-01-2024 110.9079
09-01-2024 105.7334
10-01-2024 111.8037
11-01-2024 107.7839
12-01-2024 108.7825
13-01-2024 113.331
14-01-2024 103.5649
15-01-2024 105.5168
16-01-2024 112.3401
17-01-2024 111.0975
18-01-2024 118.743
19-01-2024 113.6417
20-01-2024 112.1304
21-01-2024 127.5303
22-01-2024 120.0832
23-01-2024 122.5599
24-01-2024 116.1086
25-01-2024 121.5205
```

```
26-01-2024 125.8071
27-01-2024 120.5077
28-01-2024 129.1512
29-01-2024 125.2796
30-01-2024 127.8345
31-01-2024 127.2945
01-02-2024 140.5745
02-02-2024 132.2557
03-02-2024 128.0448
04-02-2024 138.4562
05-02-2024 129.2493
06-02-2024 137.408
07-02-2024 127.5754
08-02-2024 131.7429
09-02-2024 140.3782
10-02-2024 144.0964
11-02-2024 142.271
12-02-2024 141.846
13-02-2024 141.9288
14-02-2024 137.0518
15-02-2024 141.8553
16-02-2024 144.1615
17-02-2024 152.7604
18-02-2024 150.2029
19-02-2024 140.6797
20-02-2024 152.1255
21-02-2024 149.5897
22-02-2024 149.1406
23-02-2024 156.5937
24-02-2024 159.7005
25-02-2024 160.212
26-02-2024 152.3696
27-02-2024 156.0297
28-02-2024 160.2422
29-02-2024 164.4737
01-03-2024 158.2102
02-03-2024 160.6879
03-03-2024 157.0946
04-03-2024 157.6553
05-03-2024 168.7091
06-03-2024 172.4378
07-03-2024 166.3066
08-03-2024 172.6944
09-03-2024 170.495
10-03-2024 166.4714
11-03-2024 172.514
12-03-2024 179.4074
13-03-2024 172.5481
14-03-2024 181.5606
```

```
15-03-2024 161.6487
16-03-2024 179.8671
17-03-2024 177.2029
18-03-2024 176.2827
19-03-2024 179.2467
20-03-2024 169.8601
21-03-2024 179.7097
22-03-2024 183.6037
23-03-2024 190.2178
24-03-2024 181.247
25-03-2024 180.806
26-03-2024 183.3498
27-03-2024 191.4457
28-03-2024 189.5225
29-03-2024 186.2401
30-03-2024 192.4653
31-03-2024 191.3945
01-04-2024 196.7624
02-04-2024 189.419
03-04-2024 192.3011
04-04-2024 192.989
05-04-2024 188.642
06-04-2024 198.4503
07-04-2024 199.2851
08-04-2024 199.0155
09-04-2024 198.8271"""

l = [i.split(" ") for i in s.split('\n')]
df = pd.DataFrame(l[1:], columns=l[0])
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')
df['Stock_Price'] = pd.to_numeric(df['Stock_Price'])

df['Days'] = (df['Date'] - df['Date'].min()).dt.days

X = df['Days'].values.reshape(-1, 1)
y = df['Stock_Price'].astype(float).values
model = LinearRegression()
model.fit(X, y)

last_date = df['Date'].max()
future_days = 30
future_dates = [last_date + timedelta(days=i+1) for i in
range(future_days)]
future_days_numeric = [df['Days'].max() + i + 1 for i in
range(future_days)]

df['Predicted'] = model.predict(X)
future_predictions =
model.predict(np.array(future_days_numeric).reshape(-1, 1))
```

```
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=df['Date'],
    y=df['Stock_Price'],
    mode='lines+markers',
    name='Historical Stock_Price',
    line=dict(color='blue'),
    marker=dict(size=4)
))

fig.add_trace(go.Scatter(
    x=df['Date'],
    y=df['Predicted'],
    mode='lines',
    name='Regression Line',
    line=dict(color='red', dash='dash')
))

fig.add_trace(go.Scatter(
    x=future_dates,
    y=future_predictions,
    mode='lines',
    name='Future Prediction',
    line=dict(color='green', dash='dash')
))

fig.update_layout(
    title='Stock_Price History with Linear Regression Prediction',
    xaxis_title='Date',
    yaxis_title='Stock_Price ($)',
    legend_title='Data Type',
    hovermode='x unified',
    width=1000,
    height=600
)

fig.add_shape(
    type="rect",
    xref="x",
    yref="paper",
    x0=last_date,
    y0=0,
    x1=future_dates[-1],
    y1=1,
    fillcolor="lightgreen",
    opacity=0.2,
    layer="below",
    line_width=0,
```

```
)

slope = model.coef_[0]
intercept = model.intercept_
r_squared = model.score(X, y)

fig.add_annotation(
    xref="paper",
    yref="paper",
    x=0.02,
    y=0.98,
    text=f"Model: y = {slope:.2f}x + {intercept:.2f}<br>R² =
{r_squared:.4f}",
    showarrow=False,
    bgcolor="white",
    bordercolor="black",
    borderwidth=1
)

fig.show()
```

Question 10

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = [iris.target_names[i] for i in iris.target]
df['species_id'] = iris.target

# 1. Scatter plot matrix
fig1 = px.scatter_matrix(
    df,
    dimensions=iris.feature_names,
    color="species",
    title="Scatter Matrix of Iris Dataset",
    labels={col: col.replace('(cm)', '').replace('_', ' ') for col in
iris.feature_names},
    height=800
)
fig1.update_layout(
```

```
    title_font_size=20,
    legend_title_text='Species'
)

# 2. Box plots for each feature by species
fig2 = make_subplots(
    rows=2,
    cols=2,
    subplot_titles=iris.feature_names
)

for i, feature in enumerate(iris.feature_names):
    row = i // 2 + 1
    col = i % 2 + 1

    for species in iris.target_names:
        fig2.add_trace(
            go.Box(
                y=df[df['species'] == species][feature],
                name=species,
                legendgroup=species,
                showlegend=True if i == 0 else False
            ),
            row=row,
            col=col
        )

fig2.update_layout(
    height=700,
    title_text="Box Plots of Iris Features by Species",
    title_font_size=20
)

# 3. 3D scatter plot
fig3 = px.scatter_3d(
    df,
    x=iris.feature_names[0],
    y=iris.feature_names[1],
    z=iris.feature_names[2],
    color='species',
    symbol='species',
    labels={
        iris.feature_names[0]: iris.feature_names[0].replace('(cm)',
'').replace('_', ' '),
        iris.feature_names[1]: iris.feature_names[1].replace('(cm)',
'').replace('_', ' '),
        iris.feature_names[2]: iris.feature_names[2].replace('(cm)',
'').replace('_', ' ')
    },
    title="3D Scatter Plot of Iris Dataset"
```

```
)
fig3.update_layout(
    height=700,
    title_font_size=20
)

# 4. Violin plots for each feature by species
fig4 = make_subplots(
    rows=2,
    cols=2,
    subplot_titles=iris.feature_names
)

for i, feature in enumerate(iris.feature_names):
    row = i // 2 + 1
    col = i % 2 + 1

    for species in iris.target_names:
        fig4.add_trace(
            go.Violin(
                y=df[df['species'] == species][feature],
                name=species,
                legendgroup=species,
                showlegend=True if i == 0 else False,
                box_visible=True,
                meanline_visible=True
            ),
            row=row,
            col=col
        )

fig4.update_layout(
    height=700,
    title_text="Violin Plots of Iris Features by Species",
    title_font_size=20
)

fig1.show()
fig2.show()
fig3.show()
fig4.show()
```