SCIENTIFIC DATA VISUALIZATION

Guidelines:-

1. Use Google Colab to complete this Activity.
2. Upload your dataset in Gist and copy the link.
3. For Questions, You will need to fill the "?" with appropriate attribute or keyword.
4. You just have to run the code for tutorial as it is and answer the questions accordingly.
5. In tutorial, you will need to load a dataset of your own and implement the code provided in the tutorial and make necessary changes to the code based on the Data.
6. Submit both IPYNB along with a PDF generated file of this IPYNB file.
7. Write Theoritical answers by just taking a new text cell below the question asked.
8. The Variable names must have last 2 digits of your student ID number.
9. You dont have to submit the Tutorial File.
10. You must Use your own dataset for the tutorials and perform the same as given in the Tutorial.

In [1]:

```
!pip install pandas seaborn matplotlib altair
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-pack
ages (2.2.2)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-pac
kages (0.13.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.8.0)
Requirement already satisfied: altair in /usr/local/lib/python3.10/dist-pack
ages (4.2.2)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/di
st-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyth
on3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dis
t-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/d
ist-packages (from pandas) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/
dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/di
st-packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist
-packages (from altair) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-pack
ages (from altair) (3.1.4)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/
dist-packages (from altair) (4.23.0)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packa
ges (from altair) (0.12.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/di
st-packages (from jsonschema>=3.0->altair) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/
local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair) (2024.10.
1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python
3.10/dist-packages (from jsonschema>=3.0->altair) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/d
ist-packages (from jsonschema>=3.0->altair) (0.20.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-pa
ckages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/
dist-packages (from jinja2->altair) (3.0.2)
```

Using my own Dataset to run the tutorial

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset from your gist link
df58= pd.read_csv("https://raw.githubusercontent.com/nikkiray309/sdv-act6/re

# Display the first few rows of the dataframe
df58.head()
```

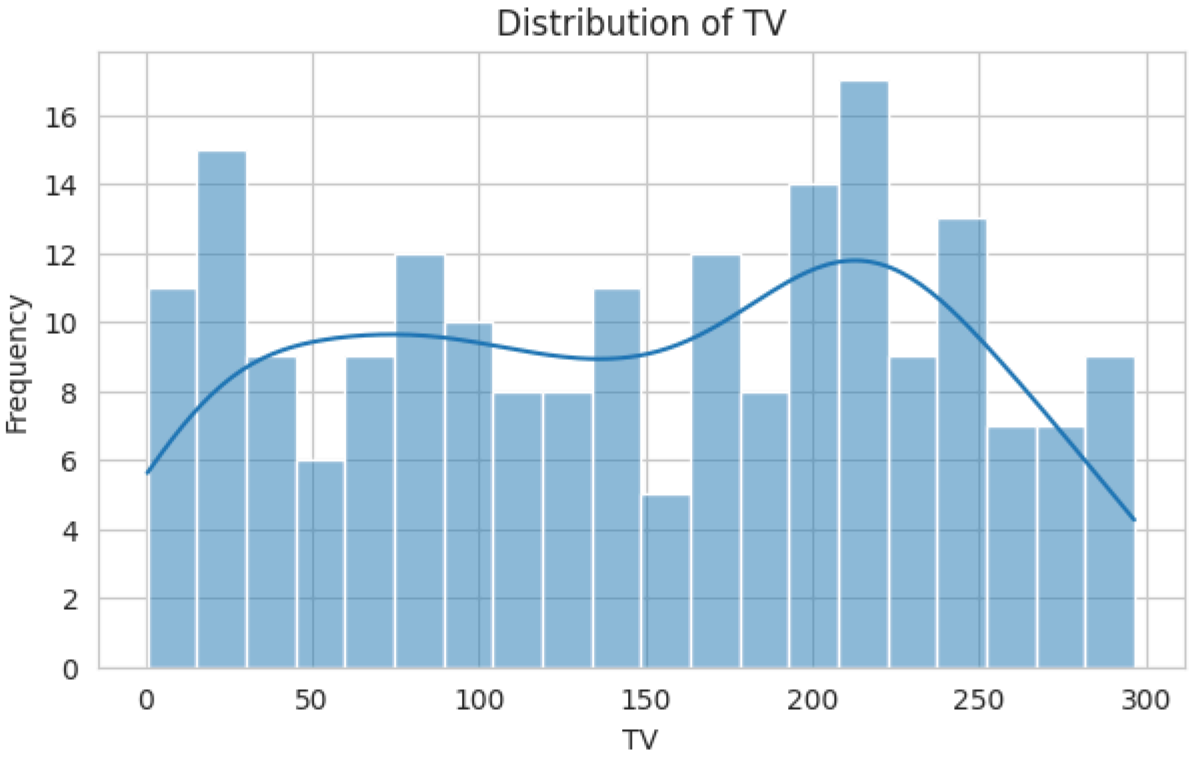
Out[2]:

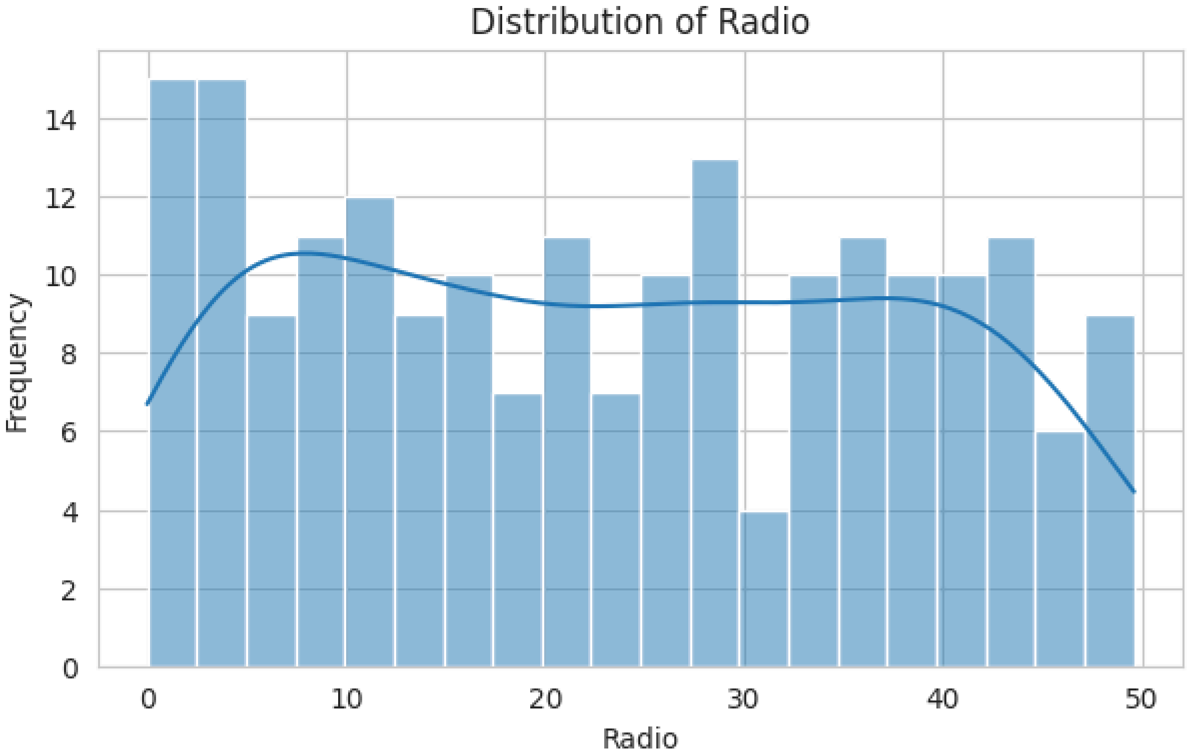
	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

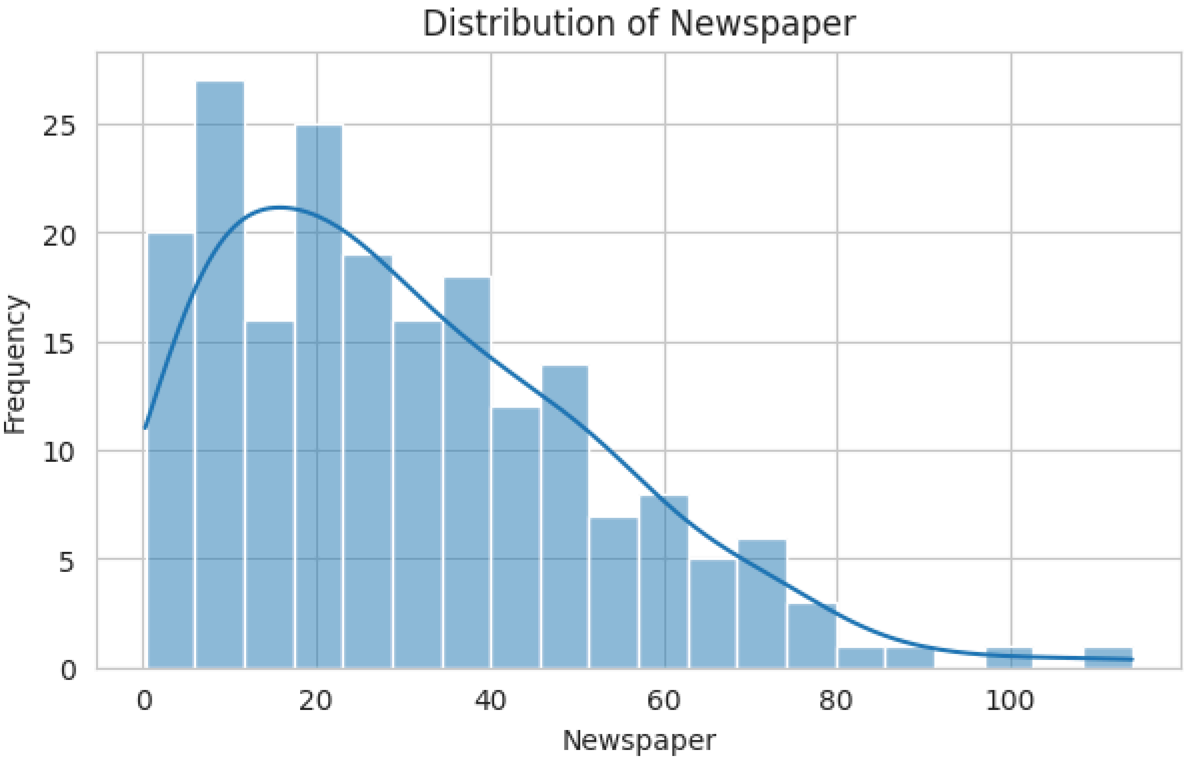
In [3]:
```
# Basic statistics of the dataset
print(df58.describe())

# Check for any missing values
print(df58.isnull().sum())
```

```
               TV       Radio   Newspaper       Sales
count  200.000000  200.000000  200.000000  200.000000
mean   147.042500   23.264000   30.554000   15.130500
std     85.854236   14.846809   21.778621    5.283892
min      0.700000    0.000000    0.300000    1.600000
25%     74.375000    9.975000   12.750000   11.000000
50%    149.750000   22.900000   25.750000   16.000000
75%    218.825000   36.525000   45.100000   19.050000
max    296.400000   49.600000  114.000000   27.000000
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

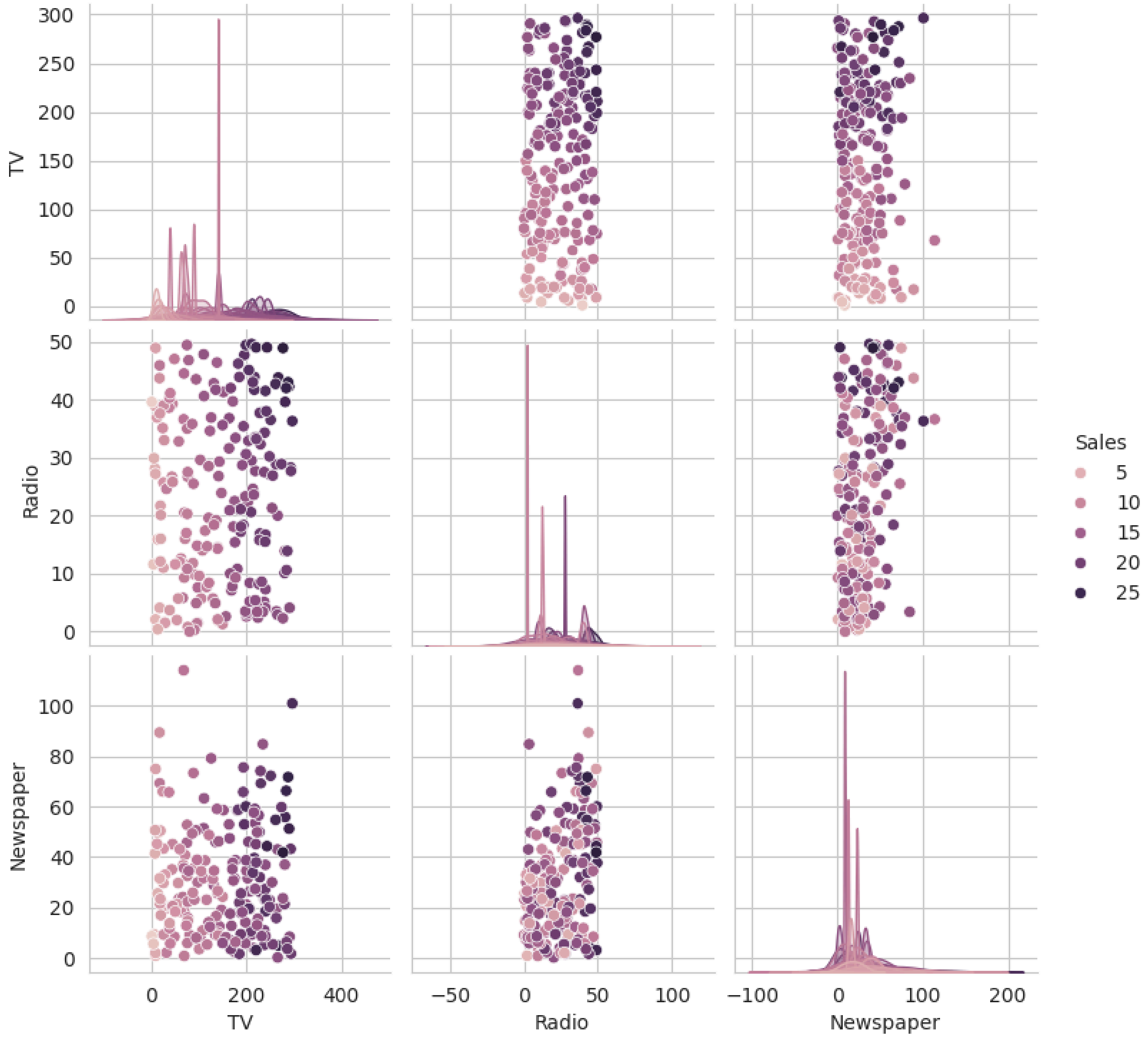
In [4]:
```
# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Plot the frequency distribution.
features = df58.columns[:-1]
for feature in features:
    plt.figure(figsize=(7, 4))
    sns.histplot(df58[feature], kde=True, bins=20)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')
    plt.show()
```


Distribution of TV
Frequency
0
2
4
6
8
10
12
14
16
0
50
100
150
200
250
300
TV


Distribution of Radio
Frequency
0
2
4
6
8
10
12
14
0
10
20
30
40
50
Radio

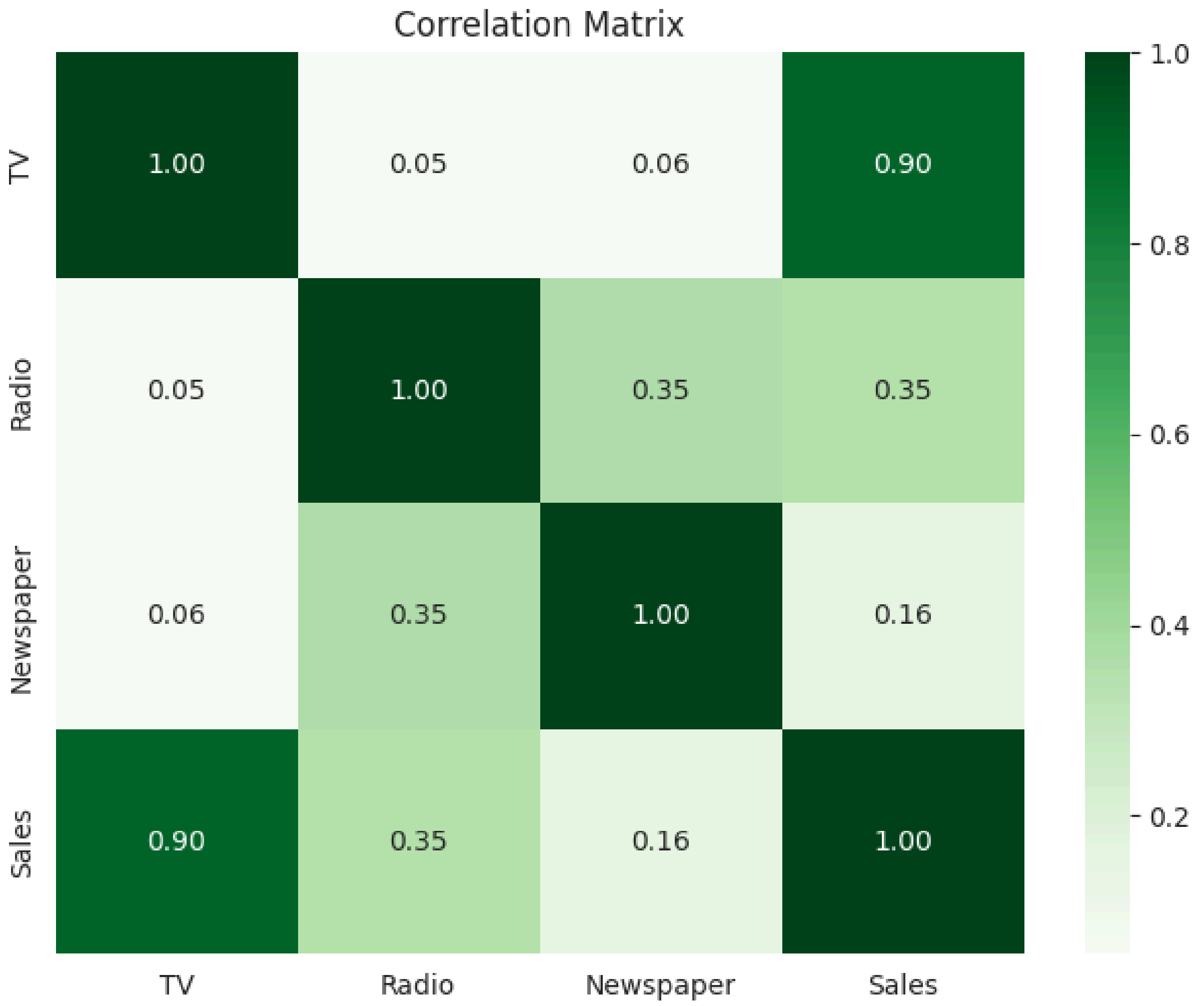


In [5]:
```
# Pairplot to visualize pairwise relationships in the dataset.
sns.pairplot(df58, hue='Sales', height=2.5)
plt.savefig('new.png')
plt.show()
```



In [6]:

```
# display the Correlation matrix heatmap below
plt.figure(figsize=(8, 6))
sns.heatmap(df58.corr(), annot=True, fmt=".2f", cmap='Greens')
plt.title('Correlation Matrix')
plt.show()
```



	TV	Radio	Newspaper	Sales
TV	1.00	0.05	0.06	0.90
Radio	0.05	1.00	0.35	0.35
Newspaper	0.06	0.35	1.00	0.16
Sales	0.90	0.35	0.16	1.00

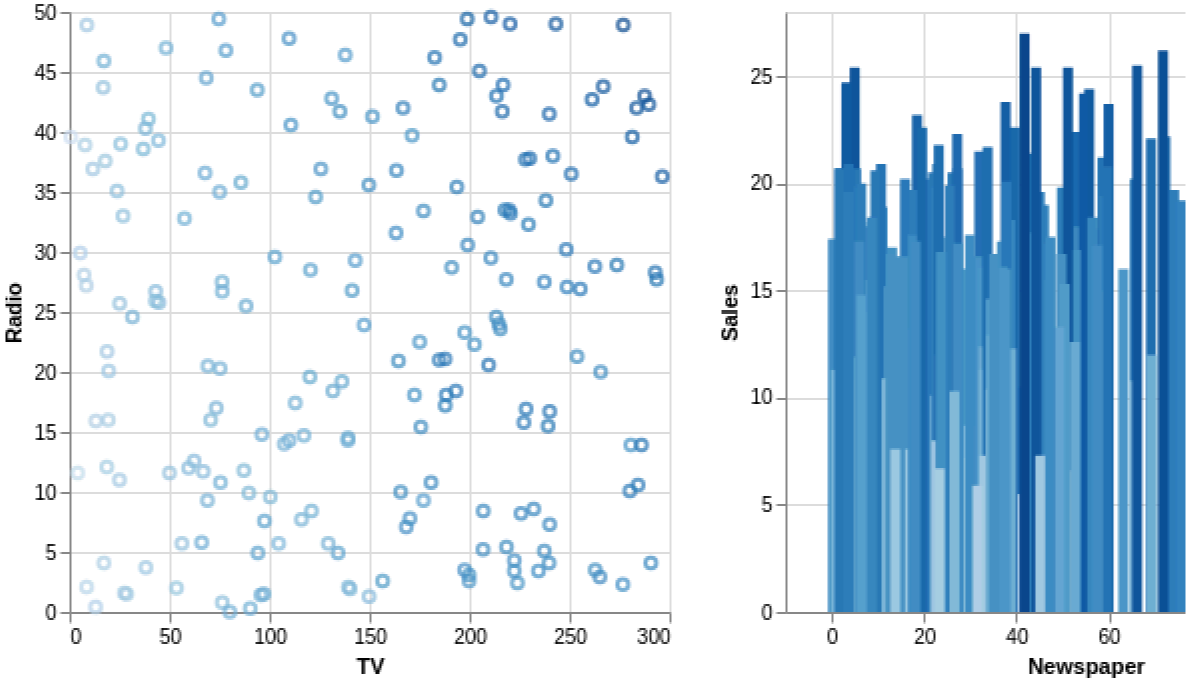
In [7]:

```
import altair as alt
```

In [8]:

```
# Scatter plot with reduced width and height
sp = alt.Chart(df58).mark_point().encode(
    x='TV',  # assigning the TV Budget on the x-axis
    y='Radio',  # assigning the Radio on the Y-axis
    color='Sales'  # assigning color to Sales
).properties(
    width=300,  # setting width
    height=300  # setting height
)

# Bar plot with reduced width and height
bp = alt.Chart(df58).mark_bar().encode(
    x='Newspaper',  # assign the Newspaper on the x-axis
    y='Sales',  # assign the sales on Y-axis
    color='Sales'  # assigning color to sales
).properties(
    width=300,  # setting width
    height=300  # setting height
).interactive()

# Concatenate two plots
alt.hconcat(sp, bp)
```

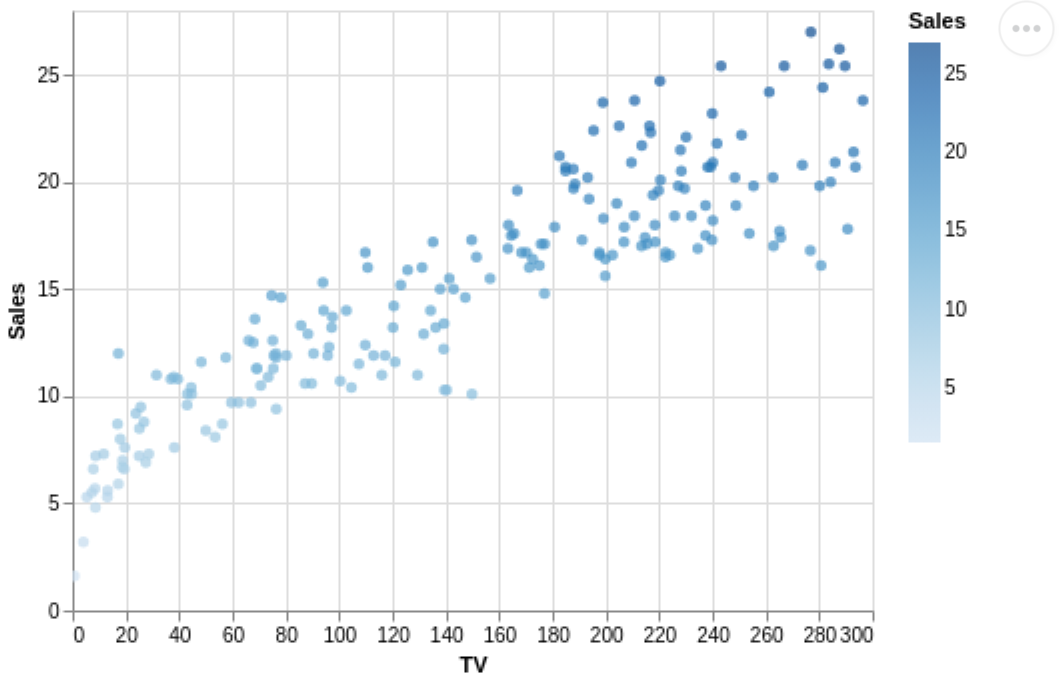
Out[8]:



In [9]:

```
sp= alt.Chart(df58).mark_circle().encode(
    x=('TV'), #assigning the TV Budget on the x-axis
    y=('Sales'), #assigning the Radio on the Y-axis
    color='Sales', #assigning color to Sales
).interactive()

sp
```

Out[9]:



You will need to fill the " ? " with the suitable values.

Question 1 (25 Points)

In [10]:

```
import pandas as pd

# Load the dataset from your gist link
df58= pd.read_csv("https://raw.githubusercontent.com/nikkiray309/sdv-act6/re

# Display the last few rows of the dataframe
print(df58.tail())
```

```
        TV  Radio  Newspaper  Sales
195   38.2    3.7       13.8    7.6
196   94.2    4.9        8.1   14.0
197  177.0    9.3        6.4   14.8
198  283.6   42.0       66.2   25.5
199  232.1    8.6        8.7   18.4
```

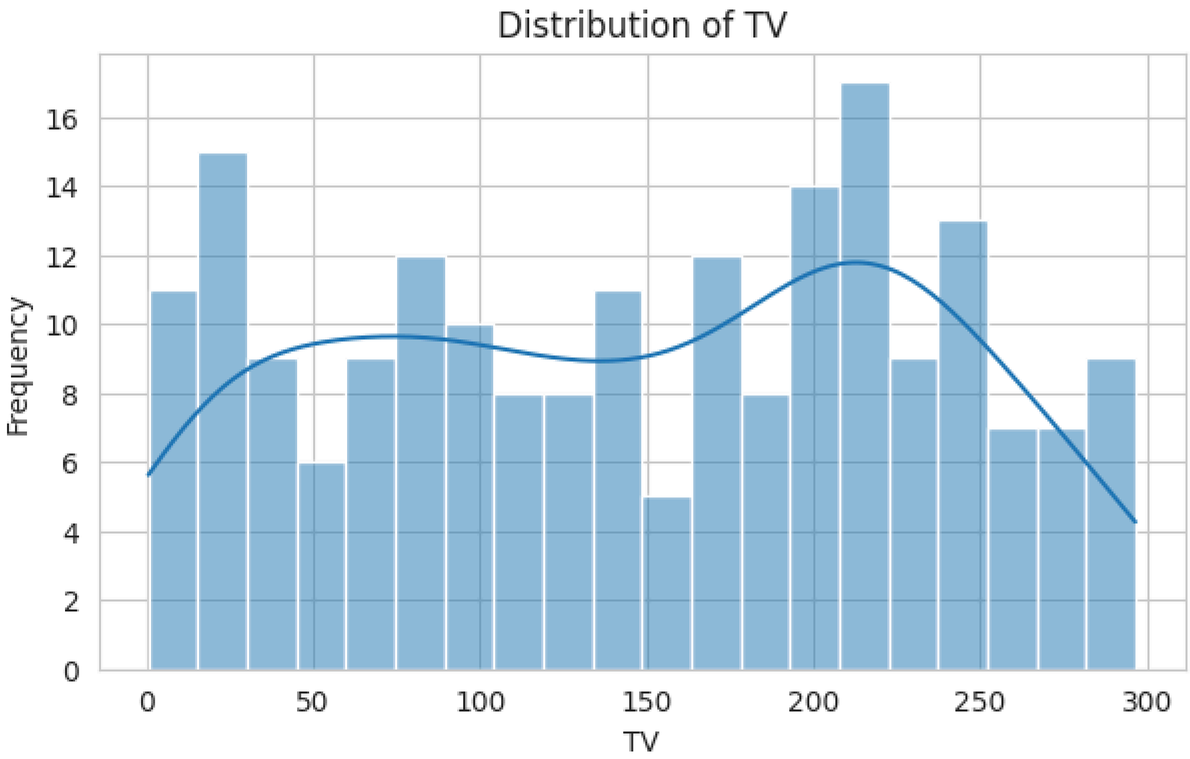
In [11]:

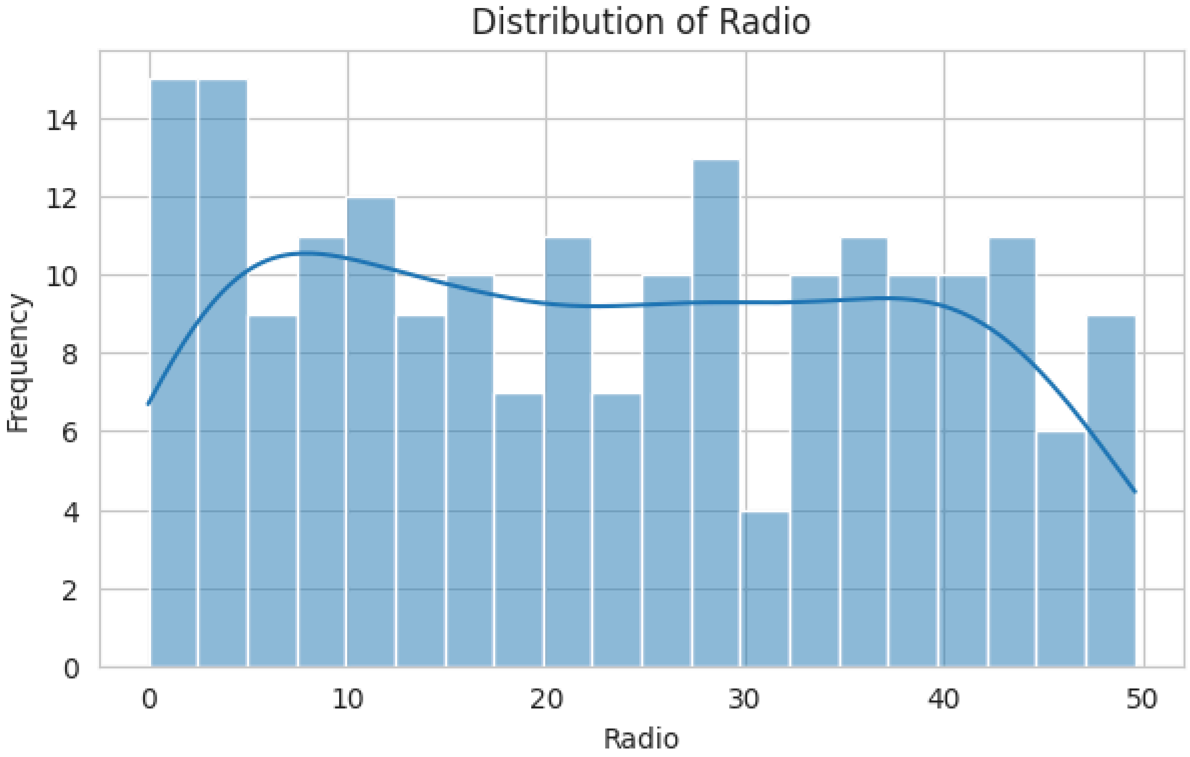
```
# Basic statistics of the dataset
print(df58.describe())

# Check for any missing values
print(df58.isnull().sum())
```

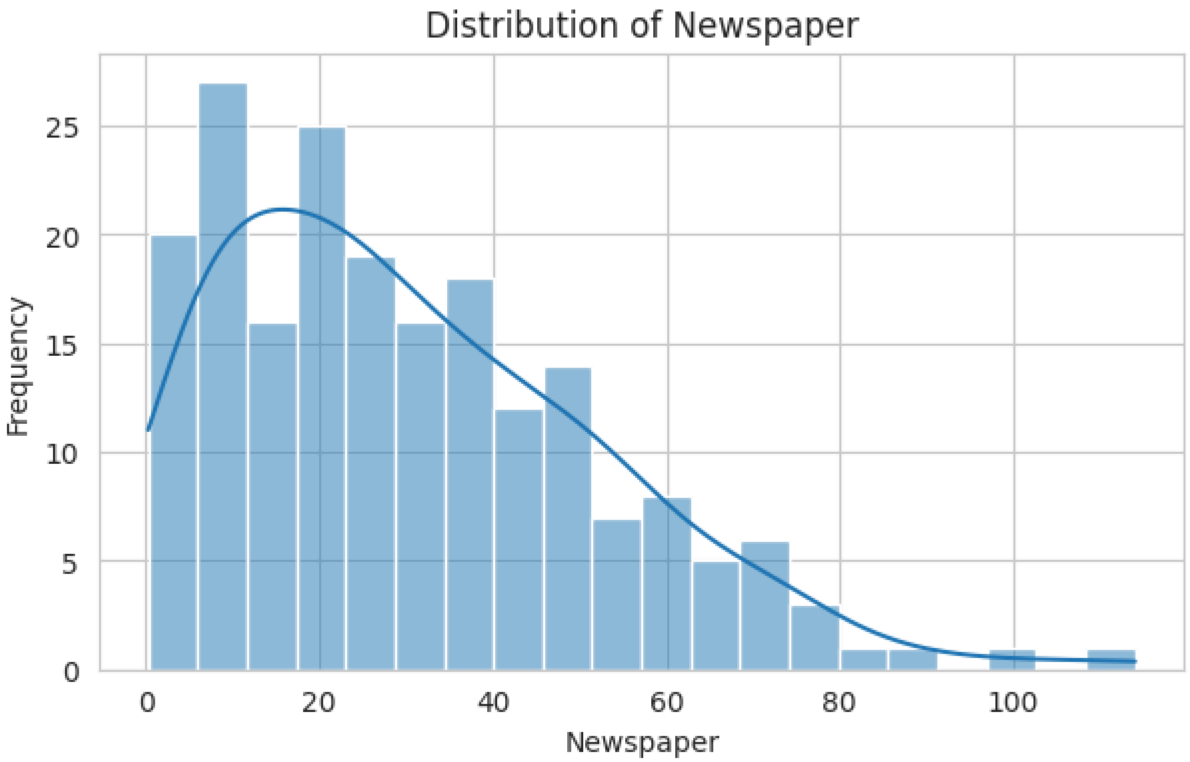
```
               TV       Radio   Newspaper       Sales
count  200.000000  200.000000  200.000000  200.000000
mean   147.042500   23.264000   30.554000   15.130500
std     85.854236   14.846809   21.778621    5.283892
min      0.700000    0.000000    0.300000    1.600000
25%     74.375000    9.975000   12.750000   11.000000
50%    149.750000   22.900000   25.750000   16.000000
75%    218.825000   36.525000   45.100000   19.050000
max    296.400000   49.600000  114.000000   27.000000
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [12]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Plot the frequency distribution.
features = df58.columns[:-1]
for feature in features:
    plt.figure(figsize=(7, 4))
    sns.histplot(df58[feature], kde=True, bins=20)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')
    plt.show()
```

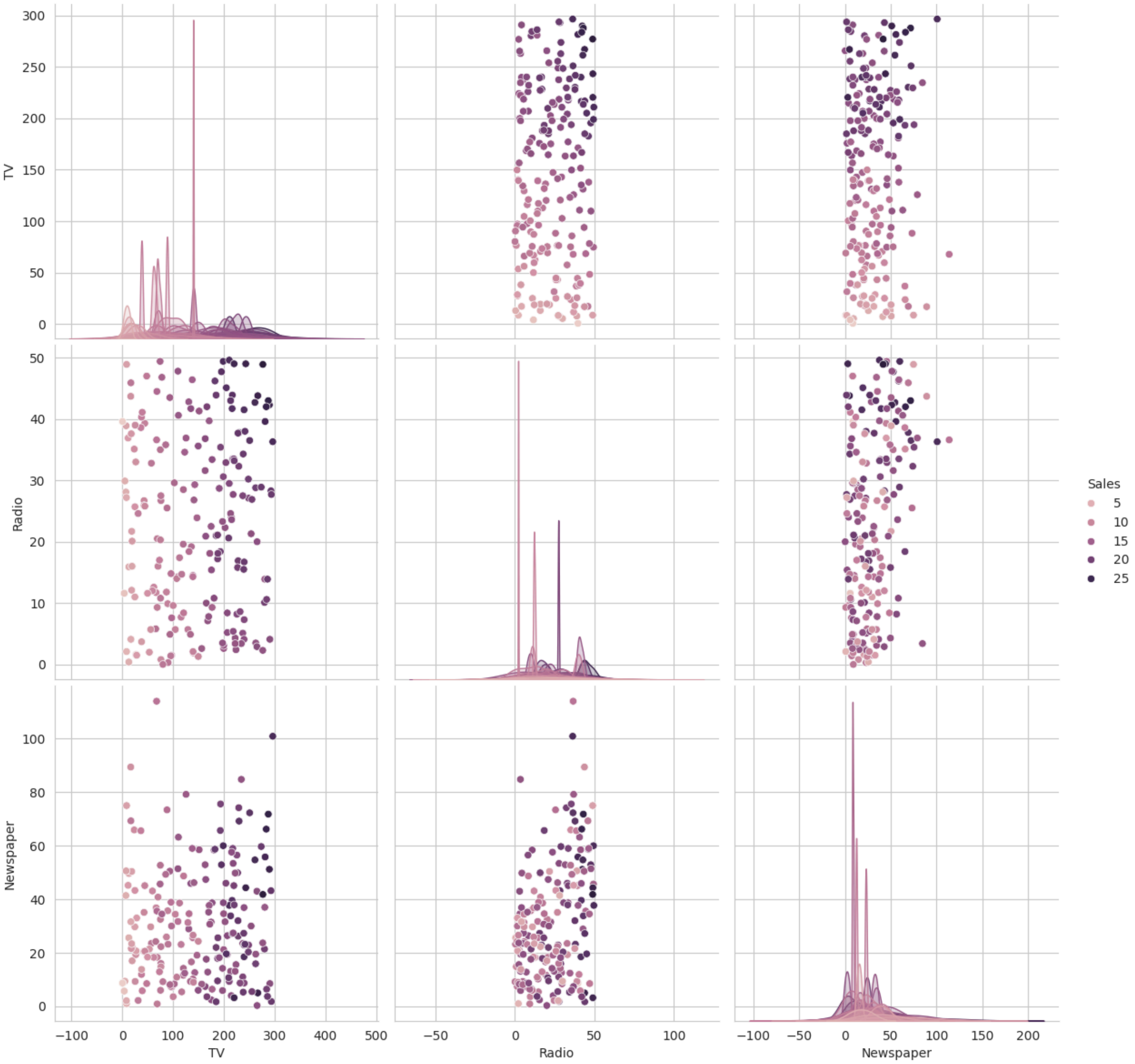

Distribution of TV
Frequency
0
2
4
6
8
10
12
14
16
0
50
100
150
200
250
300
TV


Distribution of Radio
Frequency
0
2
4
6
8
10
12
14
0
10
20
30
40
50
Radio

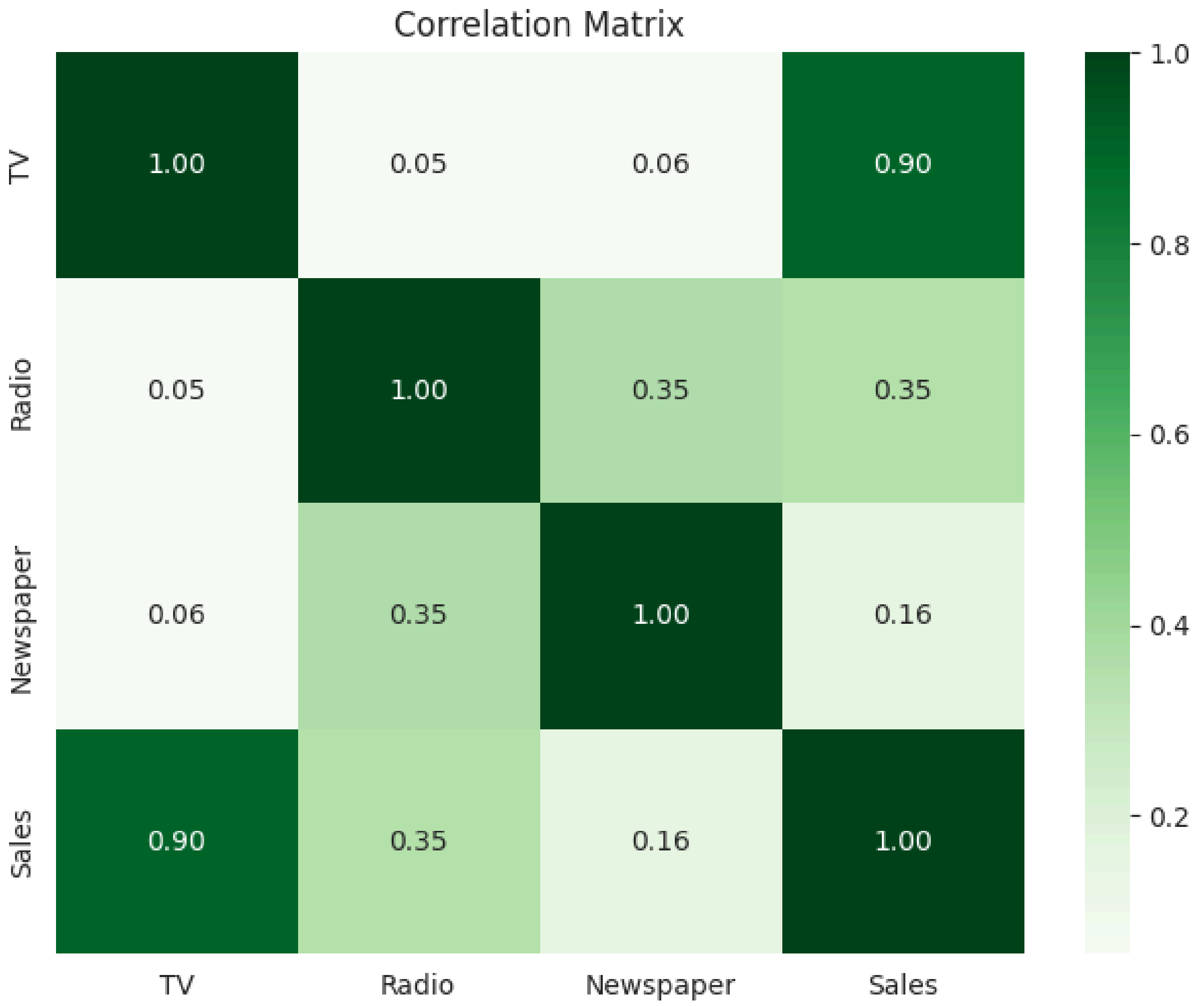


In [13]:

```
# display the Pairplot
sns.pairplot(df58, hue='Sales', height=4) # adjust the height based on your
plt.show()
```



```
In [14]: # display the Correlation matrix heatmap below
         plt.figure(figsize=(8, 6))
         sns.heatmap(df58.corr(), annot=True, fmt=".2f", cmap='Greens')
         plt.title('Correlation Matrix')
         plt.show()
```



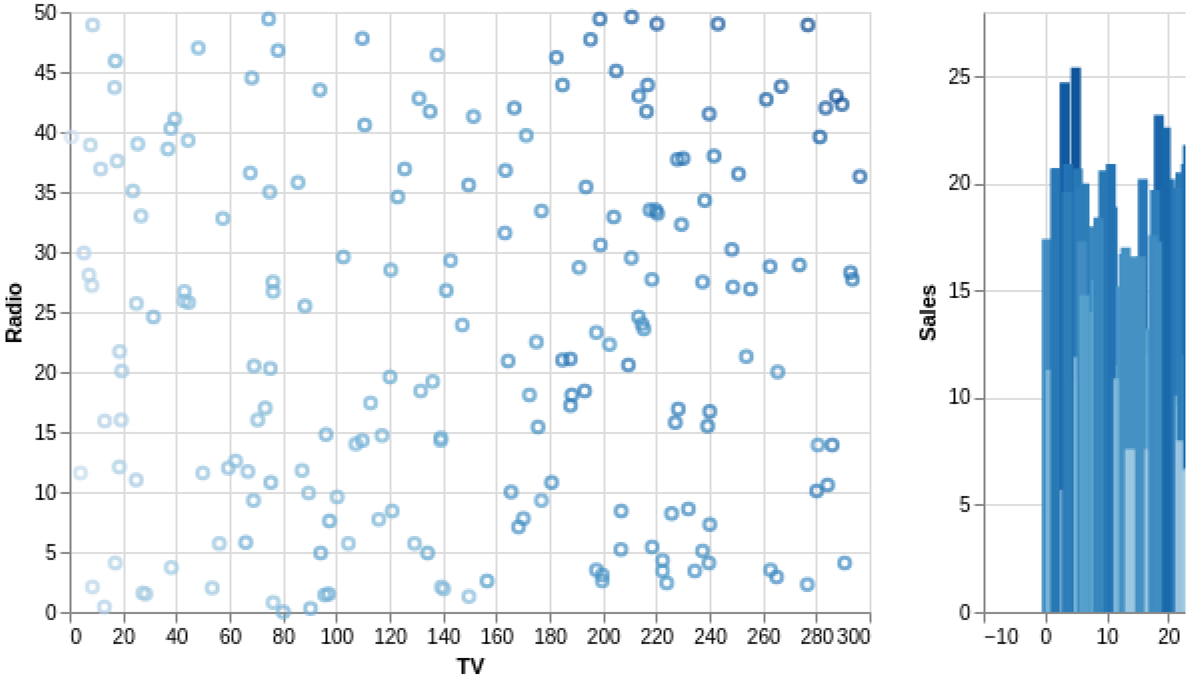
Correlation Matrix

	TV	Radio	Newspaper	Sales
TV	1.00	0.05	0.06	0.90
Radio	0.05	1.00	0.35	0.35
Newspaper	0.06	0.35	1.00	0.16
Sales	0.90	0.35	0.16	1.00

In [15]:

```
sp= alt.Chart(df58).mark_point().encode(
    x=('TV'), #assigning the TV Budget on the x-axis
    y=('Radio'), #assigning the Radio on the Y-axis
    color='Sales', #assigning color to Sales
)

bp=alt.Chart(df58).mark_bar().encode(
     x=('Newspaper'), #assign the Newspaper on the x-axis
    y=('Sales'),#assign the sales on Y-axis
    color='Sales' #assigning color to sales
).interactive()

alt.hconcat(sp, bp) #concatenate two plots
```

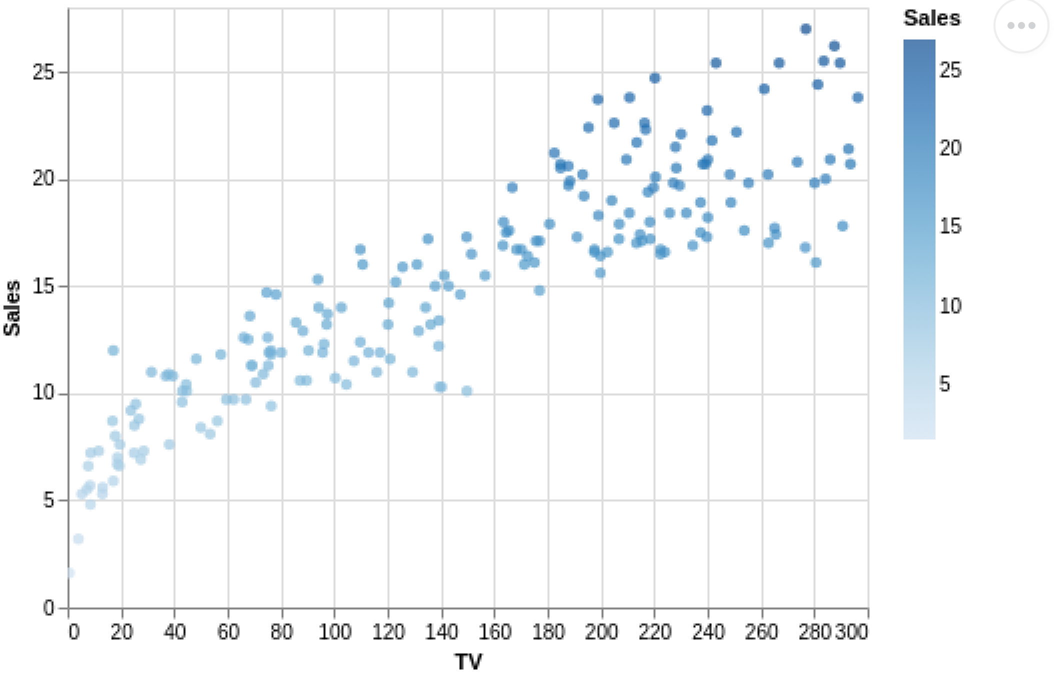
Out[15]:



In [16]:

```
sp= alt.Chart(df58).mark_circle().encode(
    x=('TV'), #assigning the TV Budget on the x-axis
    y=('Sales'), #assigning the Radio on the Y-axis
    color='Sales', #assigning color to Sales
).interactive()

sp
```

Out[16]:



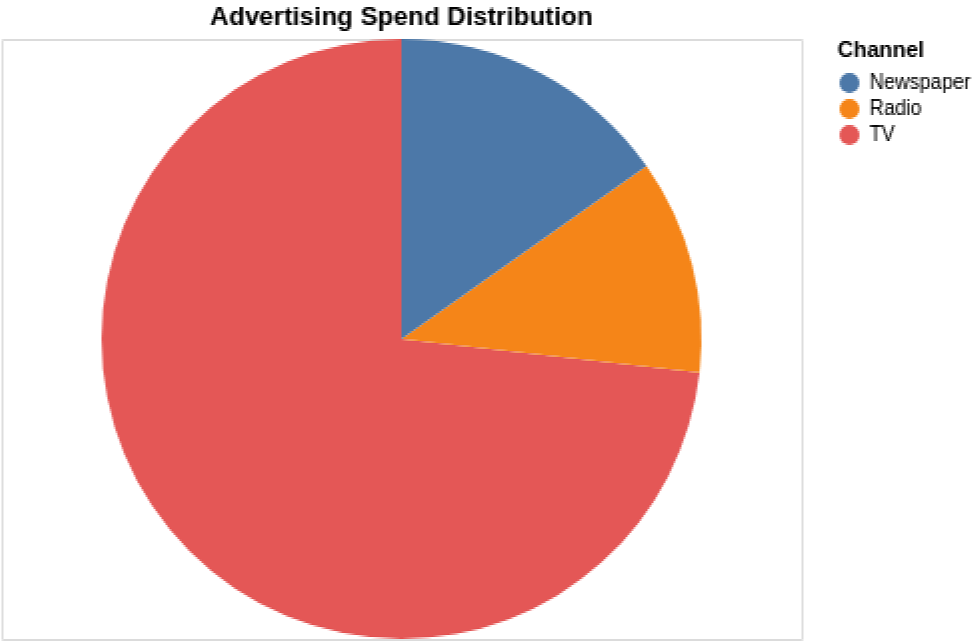
In [17]:

```
# Display 2-3 visualizations of your choice and attributes are your wish.
# Summing up the values for each advertising channel
spending = df58[['TV', 'Radio', 'Newspaper']].sum().reset_index()
spending.columns = ['Channel', 'Total']
```

```
# Create the pie chart using Altair
pie_chart = alt.Chart(spending).mark_arc().encode(
    theta=alt.Theta(field="Total", type="quantitative"),
    color=alt.Color(field="Channel", type="nominal"),
    tooltip=['Channel', 'Total']
).properties(
    title="Advertising Spend Distribution"
)

# Display the chart
pie_chart
```

```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

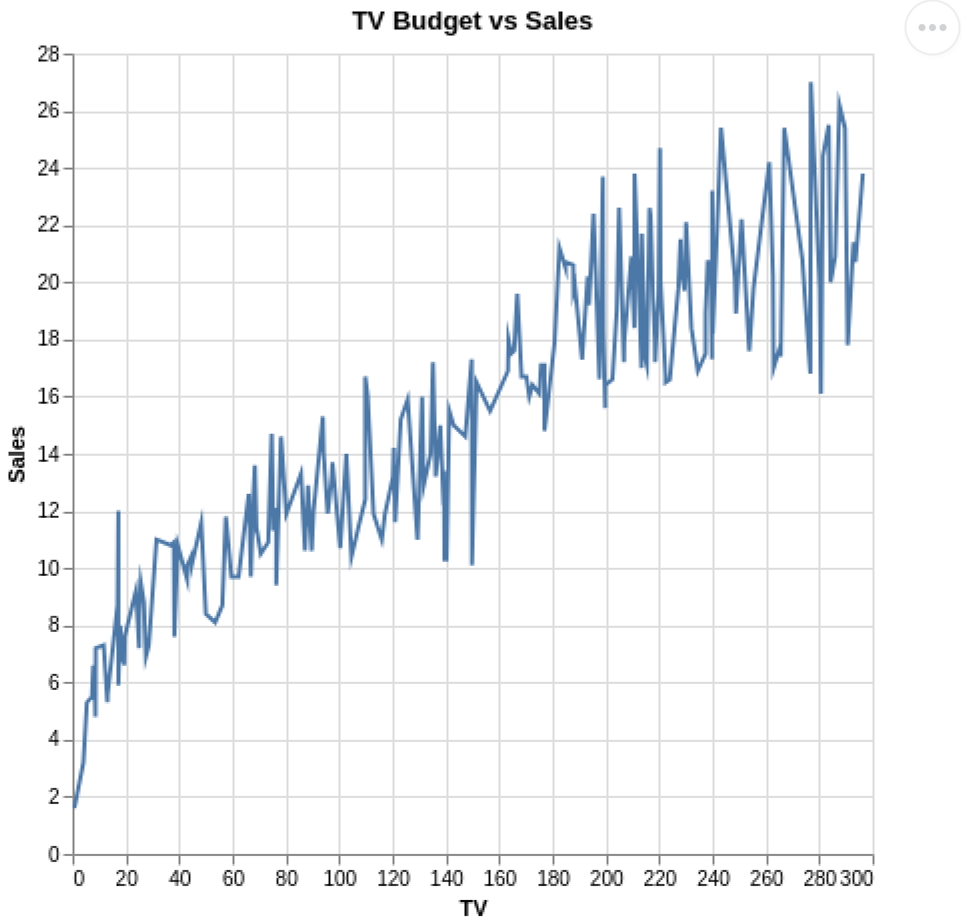
Out[17]:



In [18]:

```
# Create a line chart
line_chart = alt.Chart(df58).mark_line().encode(
    x='TV',
    y='Sales'
).properties(
    title='TV Budget vs Sales',
    width=400,
    height=400
)
line_chart
```

Out[18]:



Which library/ package are you going to use for interactive visualization in this lab? Simply describe them

Answer:

Altair is a Python library exclusively used for statistical data visualization. It is built on top of Vega and Vega-lite grams. Due to its simple, clear syntax, Altair is often used for tasks like Data exploration in Jupyter notebooks, Creating reports and dashboards with interactive elements, and Teaching data visualization concepts.

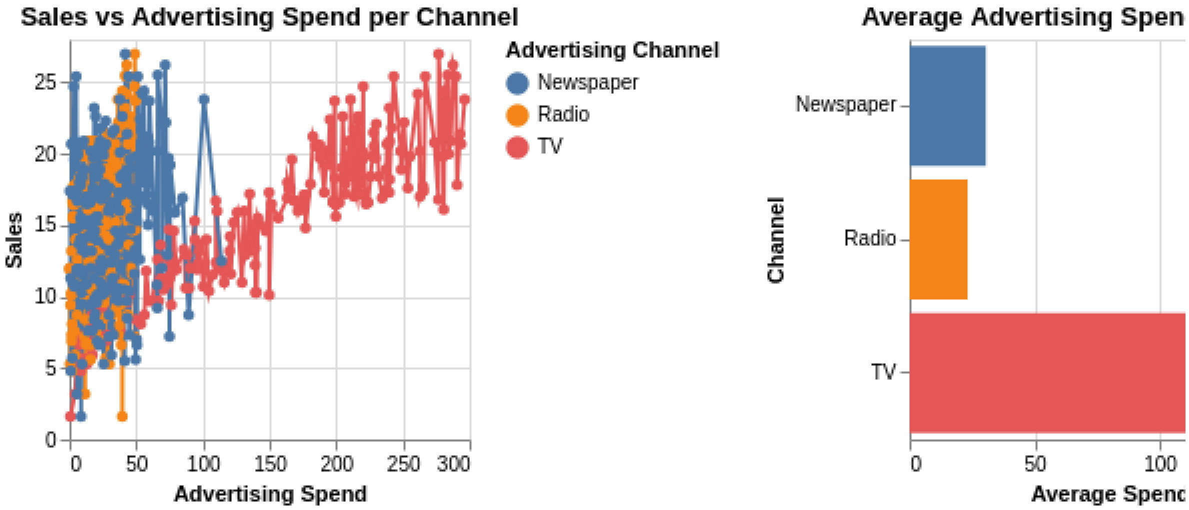
Create multiple plots which contains one interactive legend. Describe multiple plots and Analyze the data based on the plots.

In [33]:

```
# Prepare the data in long format for Altair
data_long = df58.melt(id_vars=['Sales'], value_vars=['TV', 'Radio', 'Newspap
                      var_name='Channel', value_name='Spend')

# Interactive selection for legend (shared across both charts)
highlight = alt.selection_multi(fields=['Channel'], bind='legend')

# Line chart for Spend vs Sales per Channel
line_chart = alt.Chart(data_long).mark_line(point=True).encode(
    x=alt.X('Spend:Q', title='Advertising Spend'),
    y=alt.Y('Sales:Q', title='Sales'),
    color=alt.Color('Channel:N', legend=alt.Legend(title="Advertising Channe
    opacity=alt.condition(highlight, alt.value(1), alt.value(0.1)),
    tooltip=['Channel', 'Spend', 'Sales']
).add_selection(
    highlight
).properties(
    width=200,
    height=200,
    title="Sales vs Advertising Spend per Channel"
)

# Bar chart of average spend by Channel
bar_chart = alt.Chart(data_long).mark_bar().encode(
    x=alt.X('average(Spend):Q', title='Average Spend'),
    y=alt.Y('Channel:N', title='Channel'),
    color=alt.Color('Channel:N'),
    opacity=alt.condition(highlight, alt.value(1), alt.value(0.1)),
    tooltip=['Channel', 'average(Spend):Q']
).properties(
    width=200,
    height=200,
    title="Average Advertising Spend per Channel"
)

# Combine all charts with shared interactive legend
combined_chart = alt.hconcat(line_chart, bar_chart).resolve_scale(
    color='independent'
)

combined_chart
```

```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

Out[33]:



Answer:

The scatterplot shows sales on the y-axis and advertising spending on the x-axis, with points color-coded by advertising channel (Newspaper, Radio, and TV). Investing in TV advertising appears to have the most direct impact on sales among the three channels. The bar chart compares the average advertising spend for each channel. Businesses are spending more on TV advertising, likely due to its stronger impact on sales. Meanwhile, Radio receives the least investment on average.

Explain your Understanding of this Tutorial in detail?

Answer:

In this tutorial, I have done some statistical analysis on my dataset which was loaded from the gist link. I have visualized the frequency distribution of different attributes along with pair plot and heat map. I have learned to concatenate multiple plots using the Altair library in Python. I have learned to create multiple plots with one interactive legend.

Create two different types of charts (scatter plot, bar chart, line chart, etc.) based on your dataset.

In [20]:

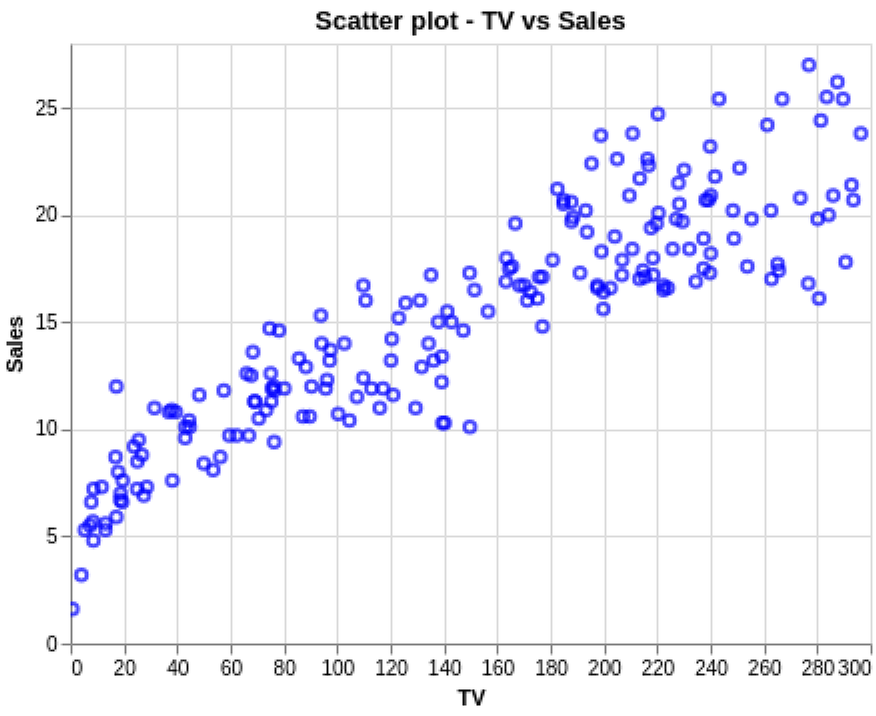
```
# Start coding here
import pandas as pd

# Load the dataset from your gist link
df58= pd.read_csv("https://raw.githubusercontent.com/nikkiray309/sdv-act6/re

# Display the last few rows of the dataframe
print(df58.tail())
```

```
        TV  Radio  Newspaper  Sales
195   38.2    3.7       13.8    7.6
196   94.2    4.9        8.1   14.0
197  177.0    9.3        6.4   14.8
198  283.6   42.0       66.2   25.5
199  232.1    8.6        8.7   18.4
```

In [21]:

```
alt.Chart(df58).mark_point(shape='circle', color = "blue").encode(
    x=alt.X('TV'),
    y=alt.Y('Sales'), tooltip = ['TV','Sales']
).properties(
     title = "Scatter plot - TV vs Sales"
 ).interactive()
```

Out[21]:

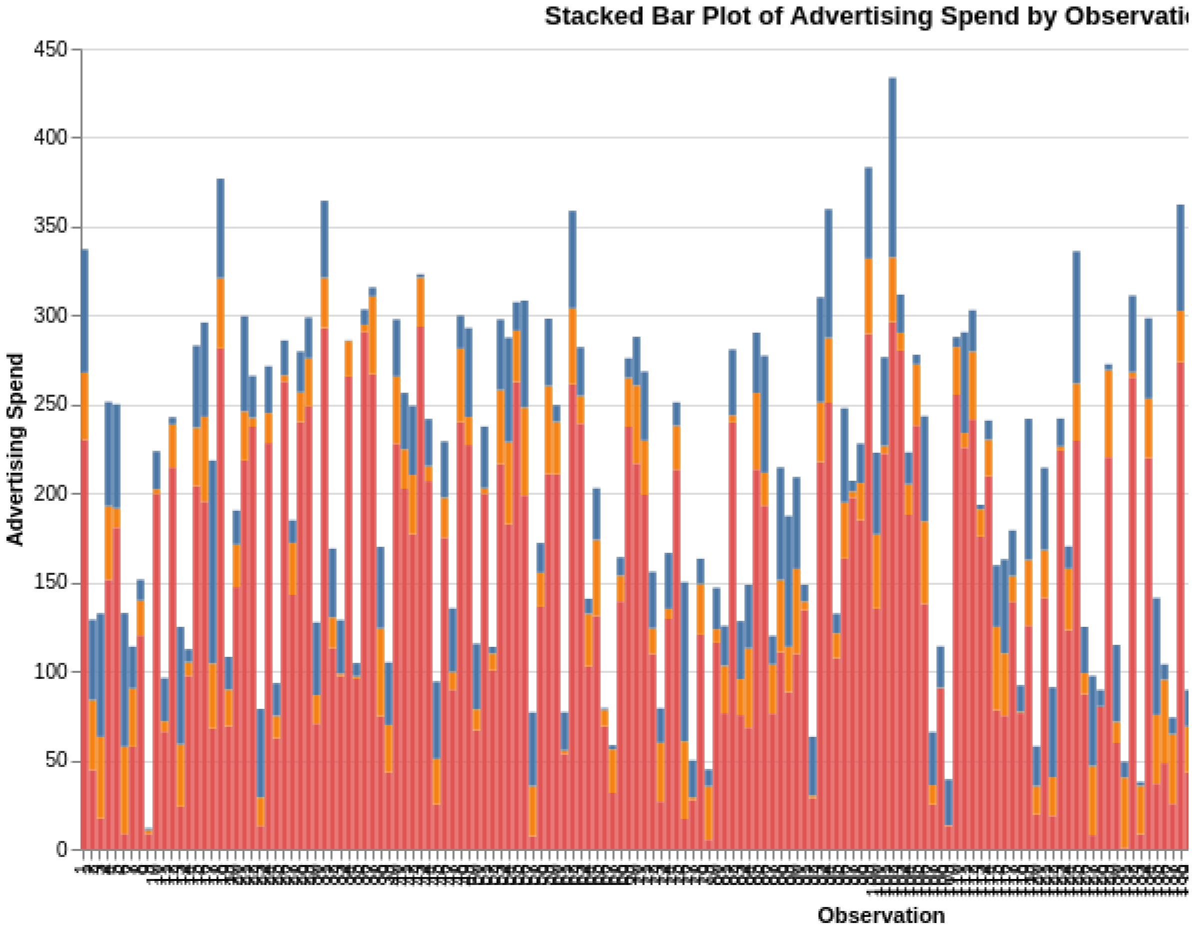


In [22]:

```
# Prepare the data in long format for Altair
data_long = df58.melt(id_vars=[], value_vars=['TV', 'Radio', 'Newspaper'],
                      var_name='Channel', value_name='Spend')

# Add an index to represent each observation
data_long['Observation'] = data_long.groupby('Channel').cumcount() + 1

# Create the stacked bar plot
stacked_bar = alt.Chart(data_long).mark_bar().encode(
    x=alt.X('Observation:O', title='Observation'),
    y=alt.Y('Spend:Q', stack='zero', title='Advertising Spend'),
    color=alt.Color('Channel:N', legend=alt.Legend(title="Advertising Channe
    tooltip=['Channel', 'Spend']
).properties(
    width=800,
    height=400,
    title="Stacked Bar Plot of Advertising Spend by Observation"
)

stacked_bar
```

```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

Out[22]:



Question 2 (25 Points)

In [23]:

```python
import altair as alt
import pandas as pd
```

In [24]:

```python
url="https://gist.githubusercontent.com/nikkiray309/cb912e3a70271ab1df380b95
data58=pd.read_csv(url)
#Import the dataset from your github account

# printing the data
print(data58)
```

```
      age     sex     bmi  children smoker     region      charges
0      19  female  27.900         0    yes  southwest  16884.92400
1      18    male  33.770         1     no  southeast   1725.55230
2      28    male  33.000         3     no  southeast   4449.46200
3      33    male  22.705         0     no  northwest  21984.47061
4      32    male  28.880         0     no  northwest   3866.85520
...   ...     ...     ...       ...    ...        ...          ...
1333   50    male  30.970         3     no  northwest  10600.54830
1334   18  female  31.920         0     no  northeast   2205.98080
1335   18  female  36.850         0     no  southeast   1629.83350
1336   21  female  25.800         0     no  southwest   2007.94500
1337   61  female  29.070         0    yes  northwest  29141.36030

[1338 rows x 7 columns]
```

In [25]:

```
# print the columns
data58.columns
```

Out[25]:

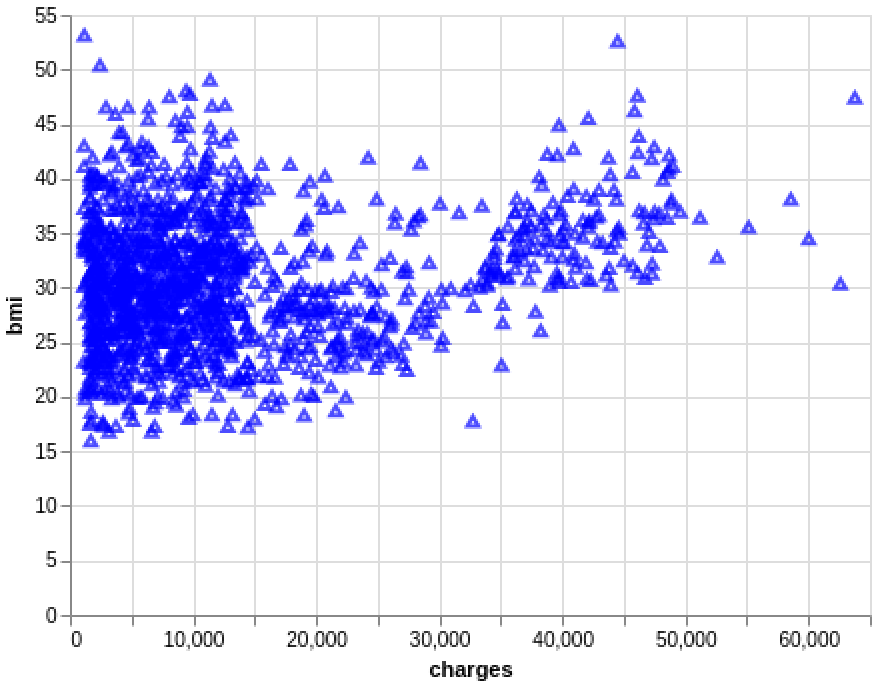
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dty
pe='object')
```

In [26]:

```
# plotting points on X and Y axis for with our desired column names
# I want the all the points to be in triangle
# plotting points on X and Y axis for with our desired column names
alt.Chart(data58).mark_point(shape="triangle-up", color="blue").encode(
    x=alt.X('charges'),
    y=alt.Y('bmi')
)
```

```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

Out[26]:

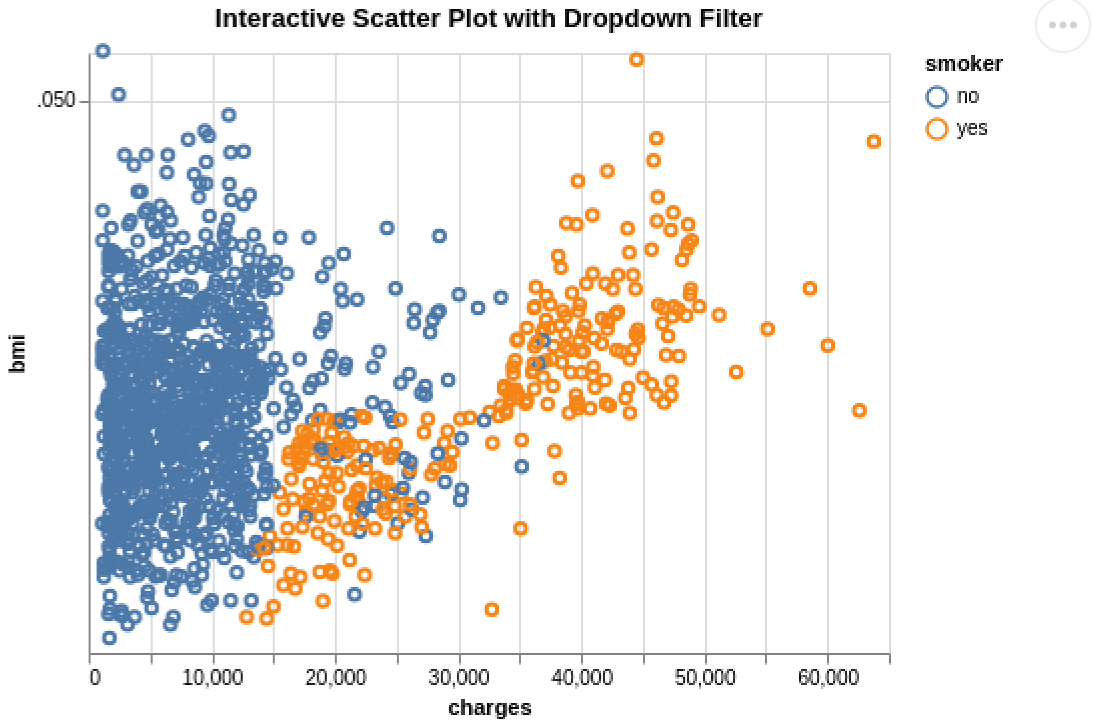


In [27]:

```
# Get unique values in the 'listing_id' column to use in the dropdown
reg_details58 = data58['region'].unique()
options = reg_details58.tolist()
labels = [str(option) + ' ' for option in options]

input_dropdown = alt.binding_radio(
    options=options + [None],
    labels=labels + ['All'],
    name='Region : '
)
selection = alt.selection_single(
    fields=['region'],
    bind=input_dropdown,
    empty='all'
)

chart_interactive = alt.Chart(data58).mark_point().encode(
    x='charges:Q',
    y='bmi:T',   # Use 'bmi' as the Y-axis
    color=alt.condition(selection, alt.Color('smoker:N'), alt.value('lightgr
    opacity=alt.condition(selection, alt.value(1), alt.value(0.2))
).add_selection(
    selection
).properties(
    title="Interactive Scatter Plot with Dropdown Filter"
)

chart_interactive
```

```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

Out[27]:



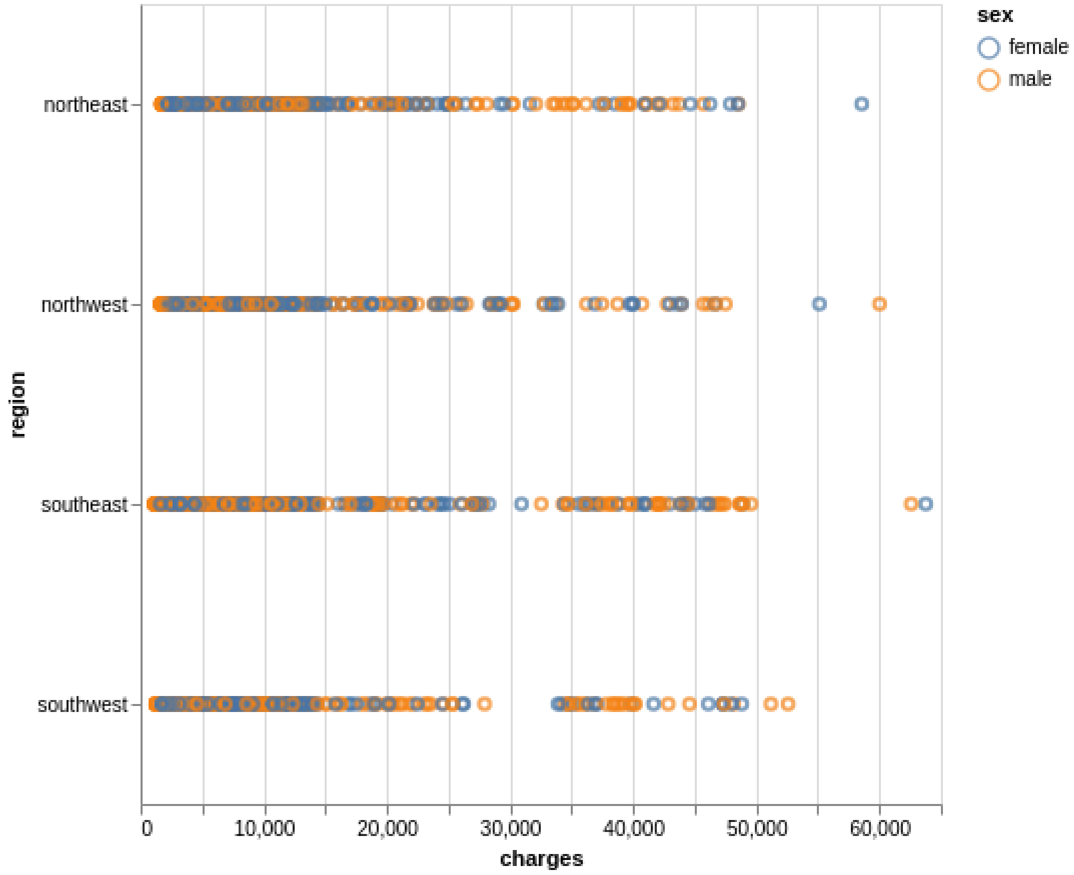
Region : ○ southwest ○ southeast ○ northwest ○ northeast ● All

In [34]:

```
alt.Chart(data58).mark_point().encode(
    x=alt.X('charges'),
    y=alt.Y('region'),
    color='sex', tooltip=['region','sex','charges']
).properties(
    width=400,
    height=400,
)
```

```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

Out[34]:

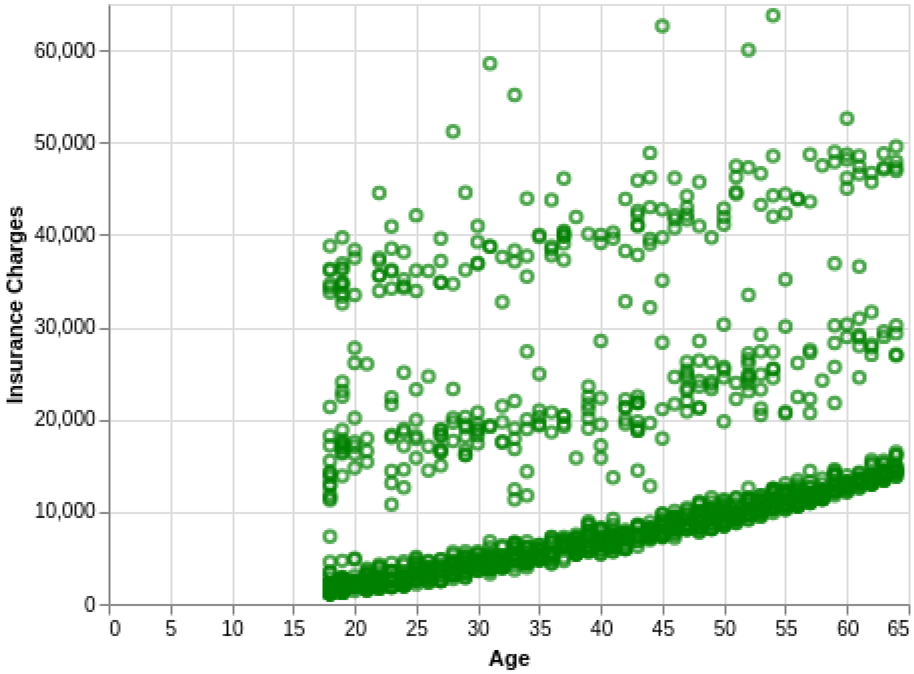


In [29]:

```
alt.Chart(data58).mark_point(shape='circle', color="green").encode(
    x=alt.X('age', title='Age'),
    y=alt.Y('charges', title='Insurance Charges'), tooltip=['age', 'charges'
)
```

```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

Out[29]:

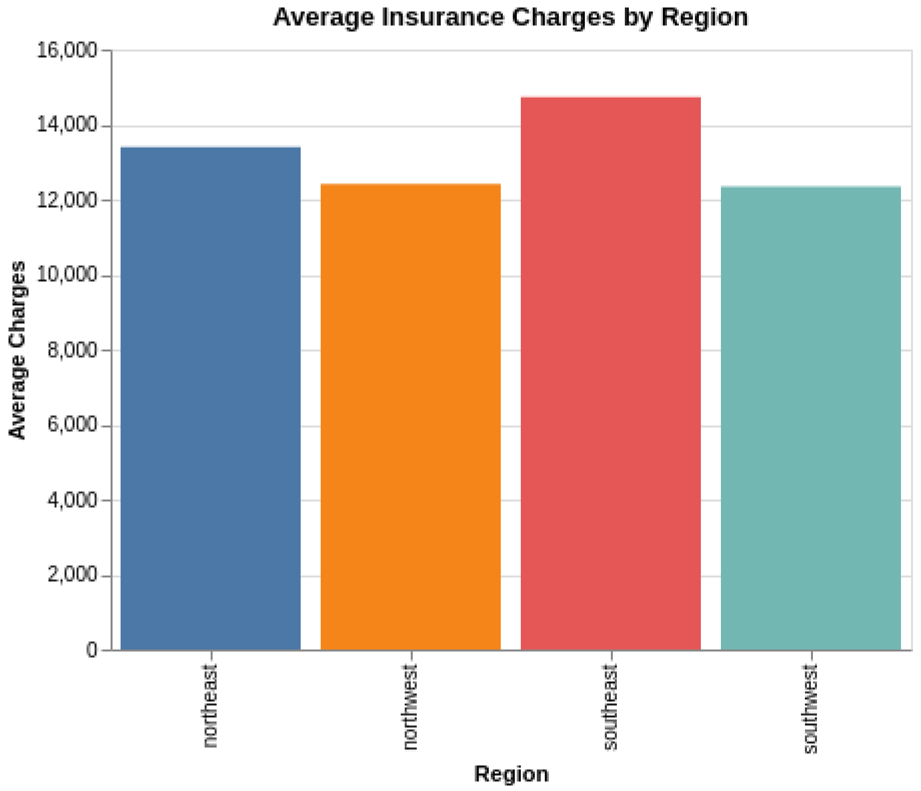


In [30]:

```
# Display 2-3 visualization of your choice

avg_charges = data58.groupby('region', as_index=False)['charges'].mean()

# Create the bar chart for average charges by region
chart = alt.Chart(avg_charges).mark_bar().encode(
    x=alt.X('region:N', title='Region'),
    y=alt.Y('charges:Q', title='Average Charges'),
    color=alt.Color('region:N', legend=None), tooltip = ['region','charges']
).properties(
    title='Average Insurance Charges by Region',
    width=400,
    height=300
)

chart
```

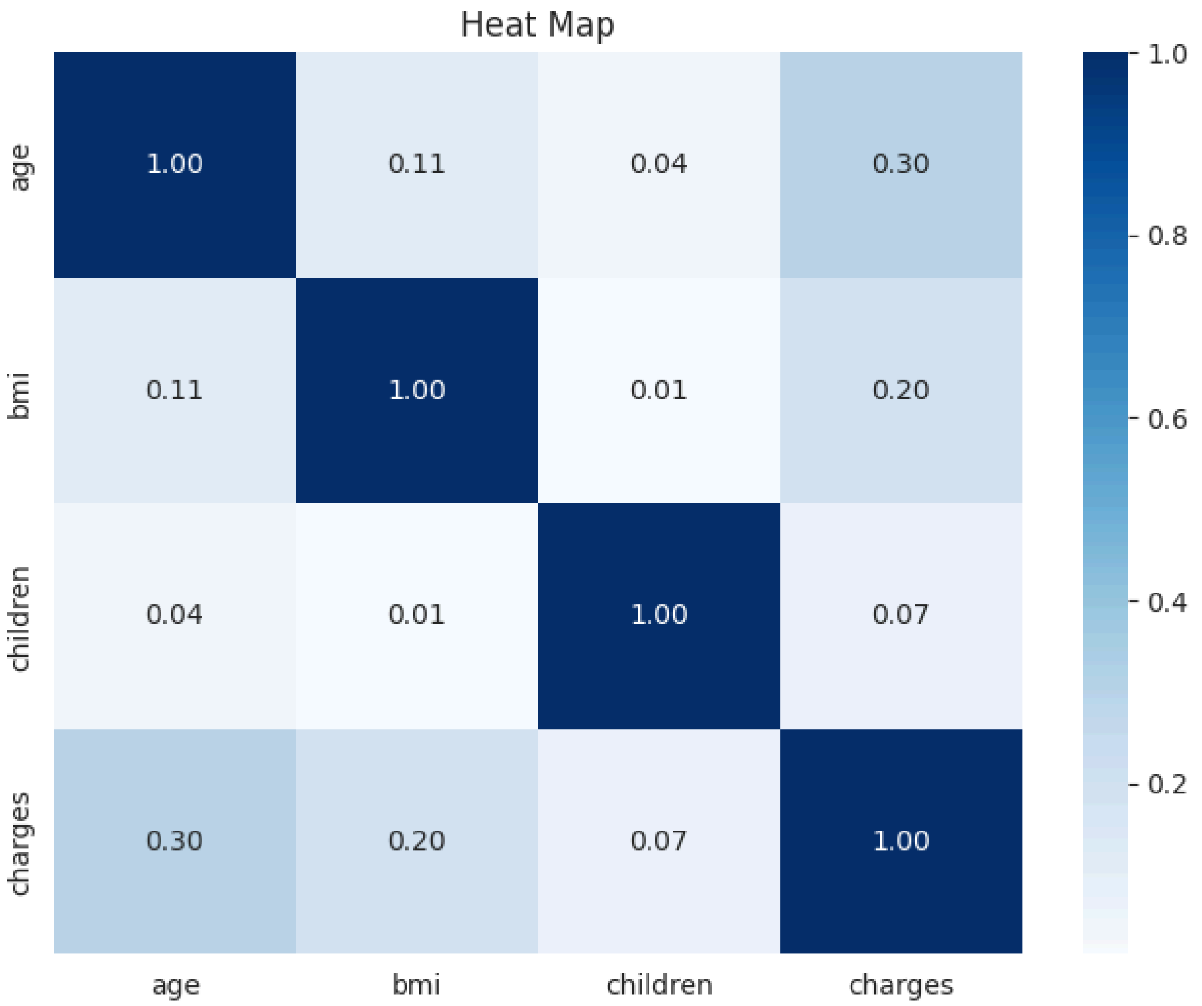
```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

Out[30]:



In [31]:

```python
# display the heatmap below
plt.figure(figsize=(8, 6))
sns.heatmap(data58.select_dtypes(include=['number']).corr(), annot=True, fmt
plt.title('Heat Map')
plt.show()
```



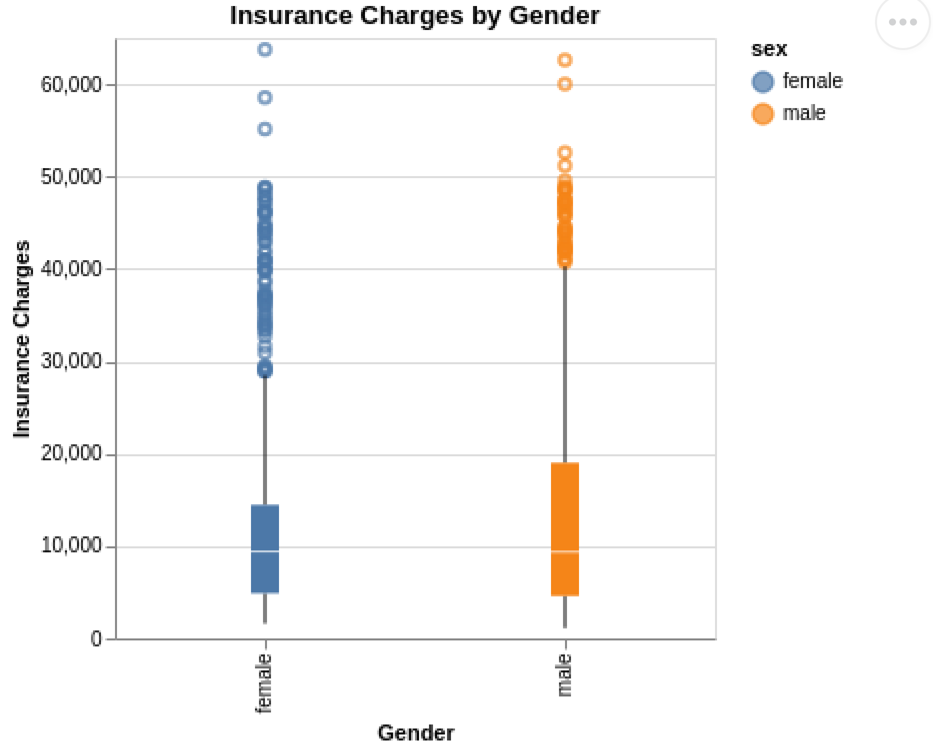
Heat Map

In [36]:

```
# Create a box plot for charges by gender
gender58 = alt.Chart(data58).mark_boxplot().encode(
    x=alt.X('sex:N', title='Gender'),
    y=alt.Y('charges:Q', title='Insurance Charges'),
    color='sex:N'
).properties(
    title='Insurance Charges by Gender',
    width=300,
    height=300
)

# Create a box plot for charges by smoker status
smoker58 = alt.Chart(data58).mark_boxplot().encode(
    x=alt.X('smoker:N', title='Smoker'),
    y=alt.Y('charges:Q', title='Insurance Charges'),
    color='smoker:N'
).properties(
    title='Insurance Charges by Smoking Status',
    width=300,
    height=300
)
gender58
```

```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

Out[36]:

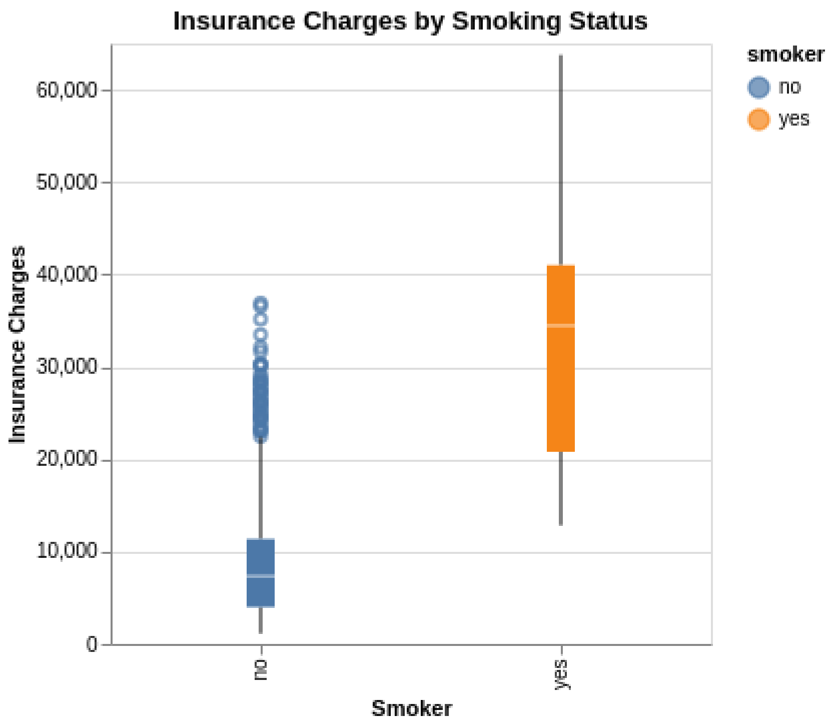


In [37]:

```
smoker58
```

```
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
/usr/local/lib/python3.10/dist-packages/altair/utils/core.py:384: FutureWarn
ing: the convert_dtype parameter is deprecated and will be removed in a futu
re version.  Do ``ser.astype(object).apply()`` instead if you want ``convert
_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

Out[37]:



Question:- How do you save a plot created with matplotlib to a file?

Answer:-

We use the `savefig()` function in matplotlib to save a plot. The syntax goes as:

```
plt.savefig('plot_name.png')
```

Question:- Describe a scenario where you would prefer seaborn over matplotlib.

Answer:-

Seaborn would be preferable over Matplotlib when you need to create statistically-oriented plots with simplified code. Taking my dataset as a scenario. For example, I want to plot a scatter plot for BMI vs charges, but I want to distinguish them by gender. With Seaborn, you can easily create a scatter plot with a categorical hue `(e.g., hue='gender')`, which will automatically color the points based on the gender value, giving you an insightful and easy-to-read visualization.

Why you are choosing these elements/ labels as tooltips. What are the advantages with or without the tooltips?

Answer:-

Choosing these elements or labels as tooltips in a visualization offers significant advantages for improving the user experience and understanding of the data.

Advantages: Tooltips allow users to access additional details without cluttering the plot. It enhances interactivity.

What makes Altair a good choice for creating interactive visualizations?

Answer:-

Altair uses a declarative syntax, which allows you to specify what you want to visualize without manually controlling the underlying details (such as axes, scales, or positions). It's easy to add interactivity to visualizations with built-in features like zooming, panning, and tooltips.

Provide your understanding of this Assignment in about 250 - 300 words.

Answer here:-

From this assignment, I have learned to load data using dataset gist links. The tutorial also walks us through basic EDA tasks like calculating basic statistics (mean, median, standard deviation, etc.), and understanding the dataset's structure. I have used the Altair library to build different visualizations in Python. This assignment also involved concatenating multiple plots and creating a single interactive legend for different plots. I have also demonstrated the importance of creating multiple chart types (e.g., scatter, bar, line, pie) to capture different perspectives on the data. Finally, the assignment emphasizes saving visualizations, an essential step for sharing results. Knowing how to export plots ensures that insights can be effectively communicated and presented.