ARTIFICIAL INTELLIGENCE & MACHINE LEARNING (EVENING BATCH)

ASSIGNMENT – 2

NAME – AYUSHI JAIN

REGISTRATION NO – 21BPS1275

CAMPUS – VIT CHENNAI

Assignment 2

Perform the Below Tasks to complete the assignment:-

Tasks:-

1. Download the dataset: Dataset
2. Load the dataset.
3. Perform the Below Visualizations.
    - Univariate Analysis
    - Bi - Variate Analysis
    - Multivariate Analysis
4. Perform descriptive statistics on the dataset.
5. Handle the Missing values.

1) **Downloaded the dataset insurance.csv**

2) **Loaded the dataset**

```
In [2]: import pandas as pd
        import numpy as np
        df = pd.read_csv(r"C:\Users\Ayushi Jain\dataset\insurance.csv")
```

```
In [3]: df.head()
```

Out[3]:

	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

---

3) **Performing the visualizations**

a) Univariant analysis:

//descriptive statistics

```
In [4]: df['charges'].describe()
```

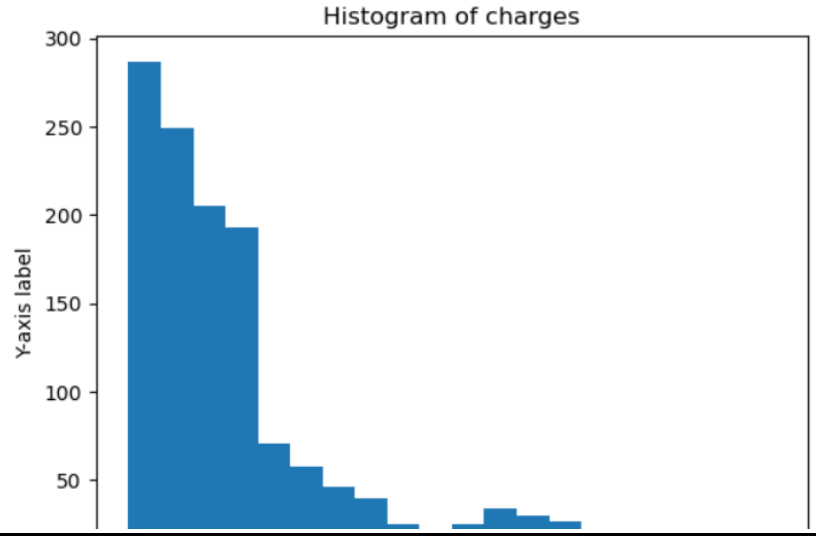
```
Out[4]: count     1338.000000
        mean     13270.422265
        std      12110.011237
        min       1121.873900
        25%       4740.287150
        50%       9382.033000
        75%      16639.912515
        max      63770.428010
        Name: charges, dtype: float64
```

```
In [5]: df['bmi'].describe()
```

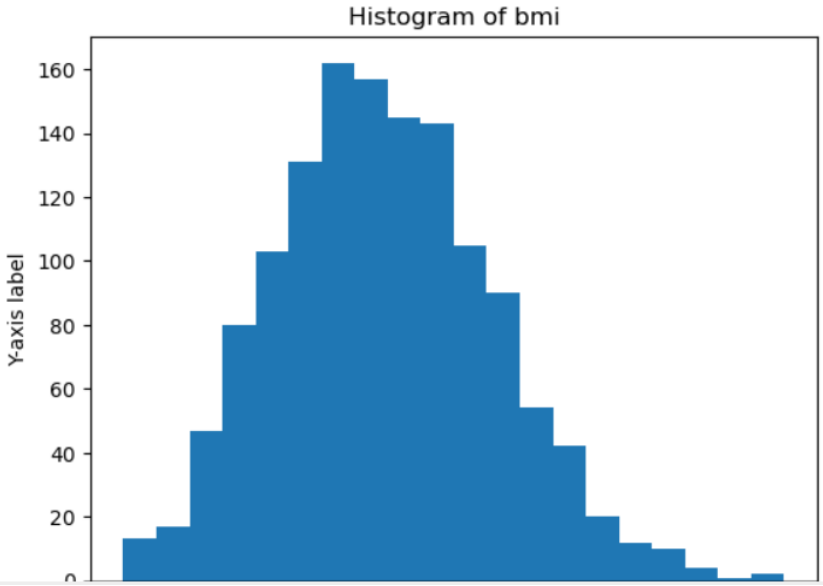
```
Out[5]: count    1338.000000
        mean       30.663397
        std         6.098187
        min        15.960000
        25%        26.296250
        50%        30.400000
        75%        34.693750
        max        53.130000
        Name: bmi, dtype: float64
```

//histrogram

In [6]:
```
import matplotlib.pyplot as plt
plt.hist(df['charges'], bins=20)  # Adjust the number of bins as needed
plt.xlabel('X-axis label')
plt.ylabel('Y-axis label')
plt.title('Histogram of charges')
plt.show()
```

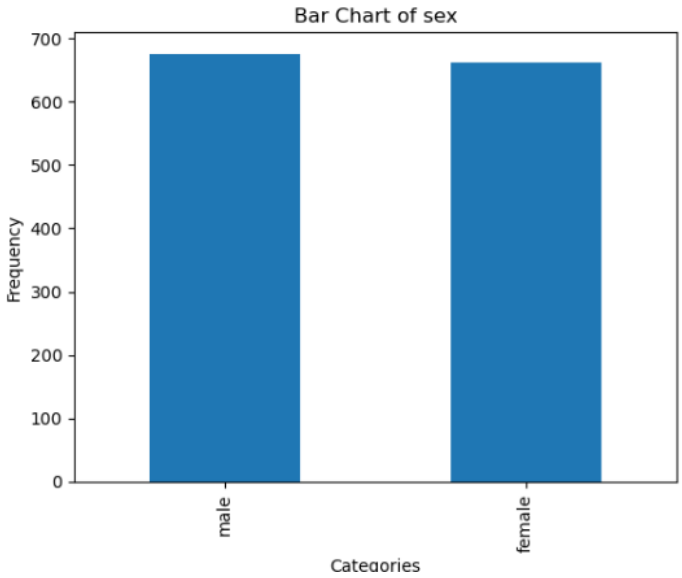


In [7]:
```
import matplotlib.pyplot as plt
plt.hist(df['bmi'], bins=20)  # Adjust the number of bins as needed
plt.xlabel('X-axis label')
plt.ylabel('Y-axis label')
plt.title('Histogram of bmi')
plt.show()
```



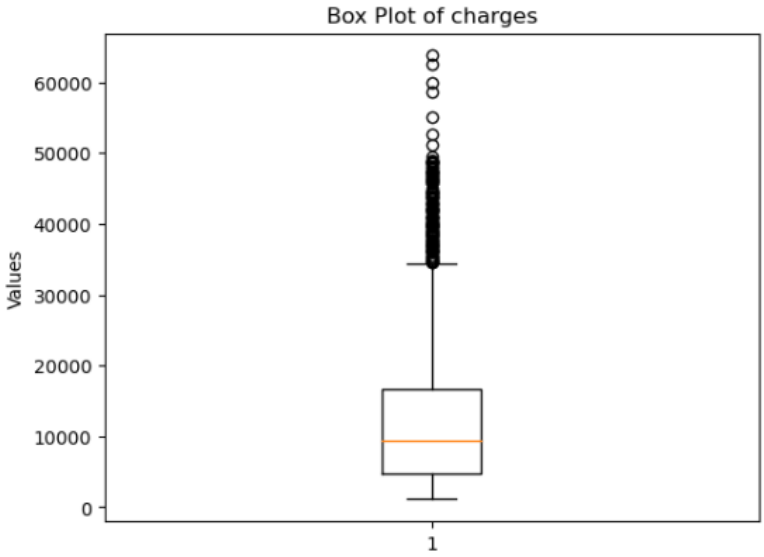
//bar plot

```
In [9]: import matplotlib.pyplot as plt

df['sex'].value_counts().plot(kind='bar')
plt.xlabel('Categories')
plt.ylabel('Frequency')
plt.title('Bar Chart of sex')
plt.show()
```



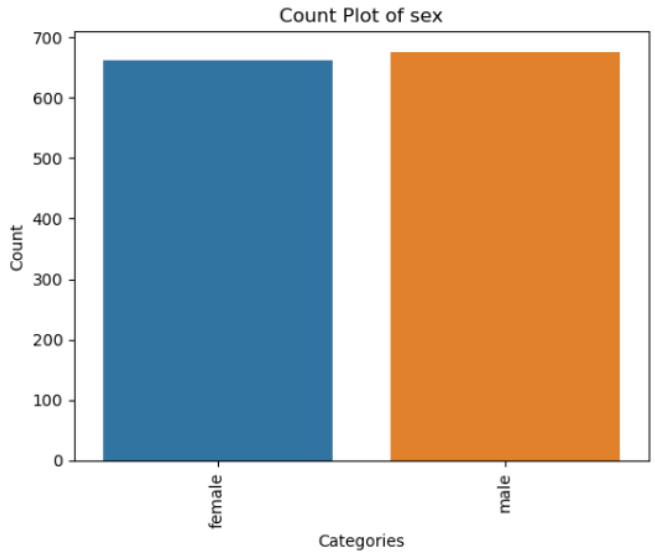
//box plot

```
In [10]: import matplotlib.pyplot as plt
plt.boxplot(df['charges'])
plt.ylabel('Values')
plt.title('Box Plot of charges')
plt.show()
```



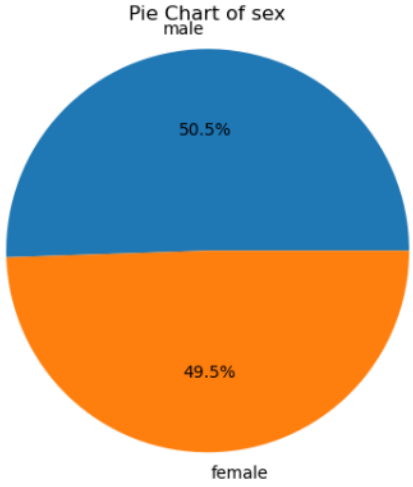
//count plot

```
In [12]: import seaborn as sns
         import matplotlib.pyplot as plt
         sns.countplot(data=df, x='sex')
         plt.xlabel('Categories')
         plt.ylabel('Count')
         plt.title('Count Plot of sex')
         plt.xticks(rotation=90)
         plt.show()
```



// pie chart

```
In [13]: import matplotlib.pyplot as plt

         labels = df['sex'].value_counts().index
         sizes = df['sex'].value_counts().values
         plt.pie(sizes, labels=labels, autopct='%1.1f%%')
         plt.title('Pie Chart of sex')
         plt.axis('equal')
         plt.show()
```

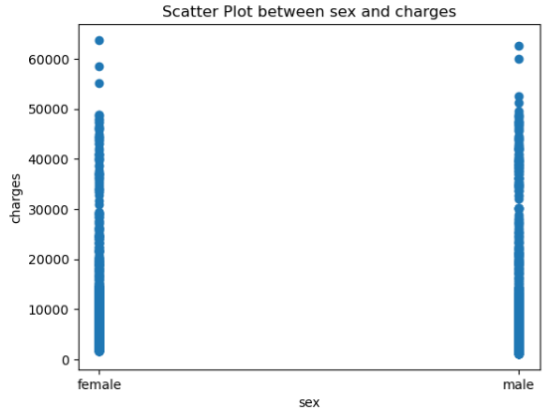



```
In [ ]:
```

b) Bivariant analysis

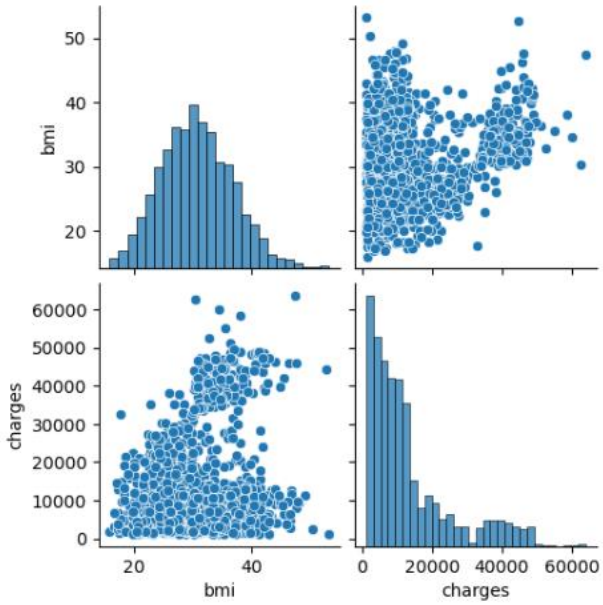
//scatter plot

```
In [17]: import matplotlib.pyplot as plt

plt.scatter(df['sex'], df['charges'])
plt.xlabel('sex')
plt.ylabel('charges')
plt.title('Scatter Plot between sex and charges')
plt.show()
```



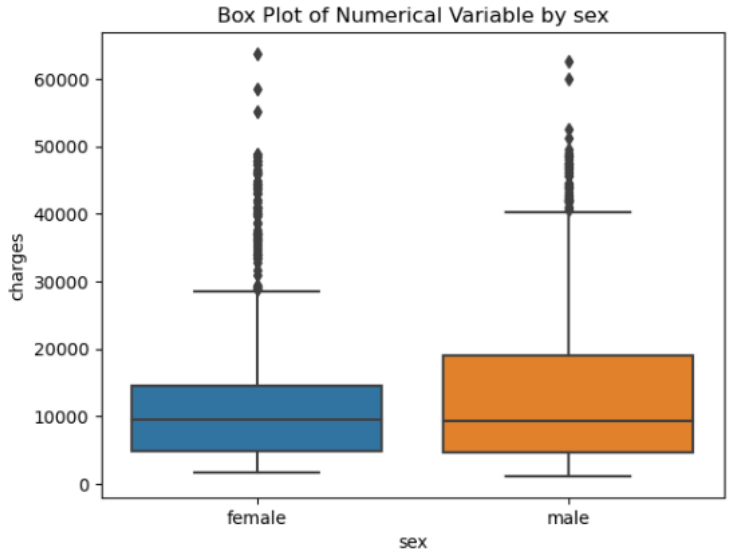
//pair plot

```
In [18]: import seaborn as sns

sns.pairplot(df[['sex', 'bmi', 'charges']])
plt.show()
```



// box plot

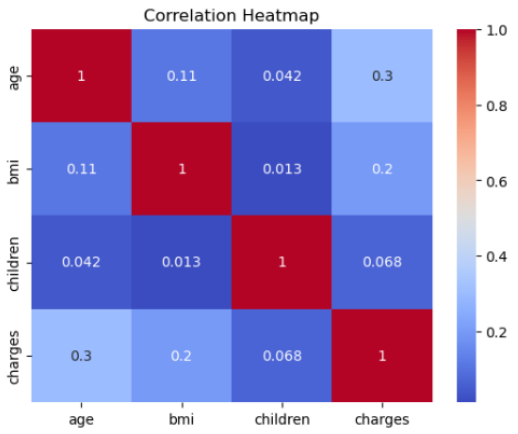
In [20]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(data=df, x='sex', y='charges')
plt.xlabel('sex')
plt.ylabel('charges')
plt.title('Box Plot of Numerical Variable by sex')
plt.show()
```



//heatmaps

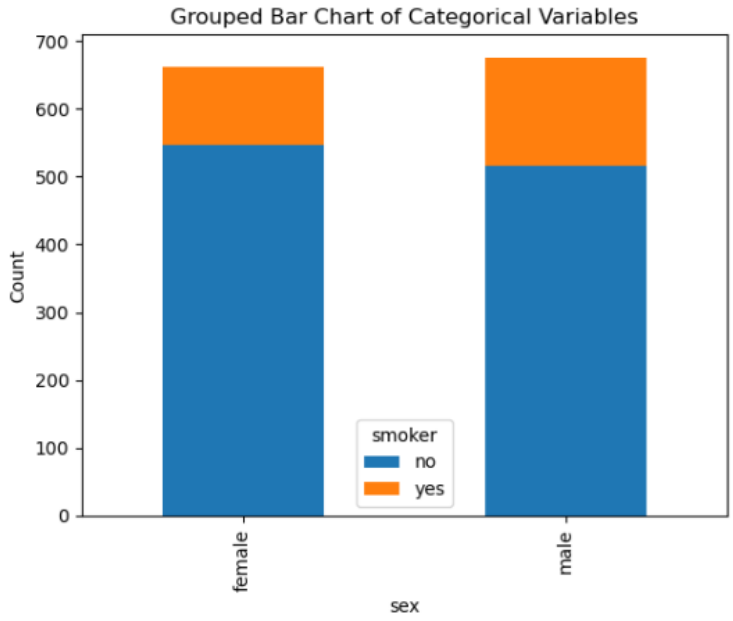
In [23]:
```
import seaborn as sns
import matplotlib.pyplot as plt

correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\Ayushi Jain\AppData\Local\Temp\ipykernel_25676\4079871336.py:4: FutureWarning: The default v
alue of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False.
Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = df.corr()
```



//grouped bar charts

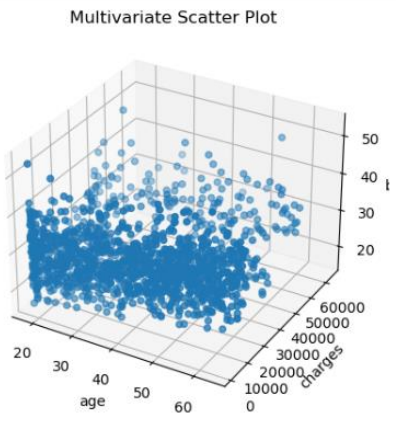
```
In [25]: import matplotlib.pyplot as plt

df.groupby(['sex', 'smoker']).size().unstack().plot(kind='bar', stacked=True)
plt.xlabel('sex')
plt.ylabel('Count')
plt.title('Grouped Bar Chart of Categorical Variables')
plt.show()
```



c) Multivariate analysis

//multivariate scatter plot

```
In [28]: import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['age'], df['charges'], df['bmi'])
ax.set_xlabel('age')
ax.set_ylabel('charges')
ax.set_zlabel('bmi')
plt.title('Multivariate Scatter Plot')
plt.show()
```



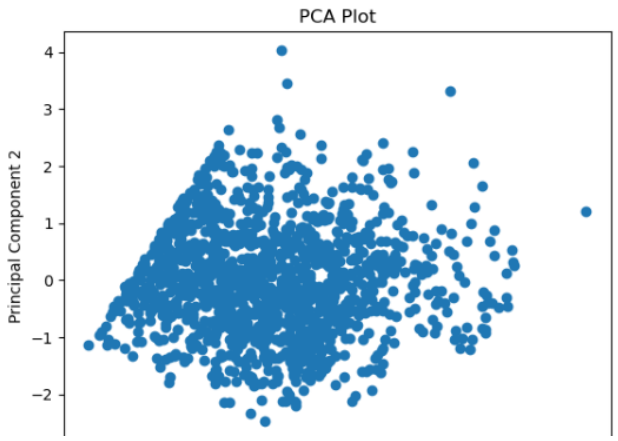
//MANOVA (multivariate analysis of variance)

In [31]:
```
import statsmodels.api as sm
from statsmodels.multivariate.manova import MANOVA



# Perform MANOVA
manova = MANOVA.from_formula('age + bmi + charges ~ sex', data=df)
print(manova.mv_test())
```

```
                    Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value  Num DF   Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0622 3.0000 1334.0000 6698.7816 0.0000
         Pillai's trace  0.9378 3.0000 1334.0000 6698.7816 0.0000
 Hotelling-Lawley trace 15.0647 3.0000 1334.0000 6698.7816 0.0000
    Roy's greatest root 15.0647 3.0000 1334.0000 6698.7816 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
             sex           Value  Num DF   Den DF  F Value Pr > F
----------------------------------------------------------------
            Wilks' lambda 0.9937 3.0000 1334.0000  2.8207 0.0378
           Pillai's trace 0.0063 3.0000 1334.0000  2.8207 0.0378
   Hotelling-Lawley trace 0.0063 3.0000 1334.0000  2.8207 0.0378
      Roy's greatest root 0.0063 3.0000 1334.0000  2.8207 0.0378
================================================================
```

//PCA

In [32]:
```
from sklearn.decomposition import PCA
import pandas as pd



X = df[['age', 'bmi', 'charges']]

# Standardize the data (optional but recommended)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X_scaled)

pca_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

# Visualize the results
import matplotlib.pyplot as plt
plt.scatter(pca_df['PC1'], pca_df['PC2'])
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Plot')
plt.show()
```



4) Descriptive statistics

In [4]:
```
df['charges'].describe()
```

Out[4]:
```
count     1338.000000
mean     13270.422265
std      12110.011237
min       1121.873900
25%       4740.287150
50%       9382.033000
75%      16639.912515
max      63770.428010
Name: charges, dtype: float64
```

In [5]:
```
df['bmi'].describe()
```

Out[5]:
```
count    1338.000000
mean       30.663397
std         6.098187
min        15.960000
25%        26.296250
50%        30.400000
75%        34.693750
max        53.130000
Name: bmi, dtype: float64
```

In [46]:
```
import pandas as pd
df = pd.read_csv(r"C:\Users\Ayushi Jain\dataset\insurance.csv")
# Calculate mean, median, and mode
mean = df['age'].mean()
median = df['age'].median()
mode = df['age'].mode().values[0]
print(median, mode,  mean)
```

```
39.0 18 39.20702541106129
```

In [48]:
```
import numpy as np


variance = np.var(df['age'], ddof=1)
std_dev = np.std(df['age'], ddof=1)
data_range = df['age'].max() - df['age'].min()
print( variance ,  std_dev ,  data_range)
```

```
197.40138665754424 14.049960379216172 46
```

In [ ]:
```

```

```
In [51]: # Calculate the 25th, 50th (median), and 75th percentiles
         first_quan = np.percentile(df['age'], 25)
         second_quan = np.percentile(df['age'], 50)
         third_quan = np.percentile(df['age'], 75)
         print (first_quan , second_quan , third_quan)

         27.0 39.0 51.0
```

```
In [ ]:
```

```
27.0 39.0 51.0
```

```
In [53]: summary_stats = df.describe()
         skewness = df['age'].skew()
         kurtosis = df['age'].kurtosis()
         frequency_counts = df['sex'].value_counts()
```

```
In [55]: print(summary_stats, skewness , kurtosis,  frequency_counts)

                age          bmi     children       charges
count  1338.000000  1338.000000  1338.000000   1338.000000
mean     39.207025    30.663397     1.094918  13270.422265
std      14.049960     6.098187     1.205493  12110.011237
min      18.000000    15.960000     0.000000   1121.873900
25%      27.000000    26.296250     0.000000   4740.287150
50%      39.000000    30.400000     1.000000   9382.033000
75%      51.000000    34.693750     2.000000  16639.912515
max      64.000000    53.130000     5.000000  63770.428010 0.05567251565299186 -1.2450876526418673 ma
le      676
female    662
Name: sex, dtype: int64
```

```
In [ ]:
```

5) Handling missing values

```
In [37]: import pandas as pd


         missing_values = df.isnull().sum()
         missing_values_column = df['charges'].isnull().sum()
```

```
In [38]: print(missing_values_column)

         0
```

It returns the value zero, hence it has no missing values

Visualizing using heat map

```
import seaborn as sns
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
```

Out[39]: <Axes: >