```
df = ("/content/sample_data/Urban company data /uc.csv")
df
```

```
'/content/sample_data/Urban company data /uc.csv'
```

```
import pandas as pd

# Load the dataset into a DataFrame
df = pd.read_csv("/content/sample_data/Urban company data /uc.csv")

# Display the first few rows of the DataFrame
df.head()
```

	service	subservice_name	subservice_charge	city_name	country_name	source
0	ac_service_repair	Non-Inverter PCB repaired	['₹1800']	ahmedabad	India	UrbanCompany
1	ac_service_repair	Inverter PCB repaired	['₹4000']	ahmedabad	India	UrbanCompany
2	ac_service_repair	Replace LVT	['₹900', '₹499 (Labour)']	ahmedabad	India	UrbanCompany
3	ac_service_repair	Capacitor 2-5 mfd	['₹250', '₹349 (Labour)']	ahmedabad	India	UrbanCompany
4	ac_service_repair	Capacitor 10-25 mfd	['₹400', '₹349 (Labour)']	ahmedabad	India	UrbanCompany

```
# Handling missing values: Checking which columns have missing values
print(df.isnull().sum())

# Removing duplicates
df.drop_duplicates(inplace=True)
```

```
service              0
subservice_name      0
subservice_charge    0
city_name            0
country_name         0
source               0
dtype: int64
```

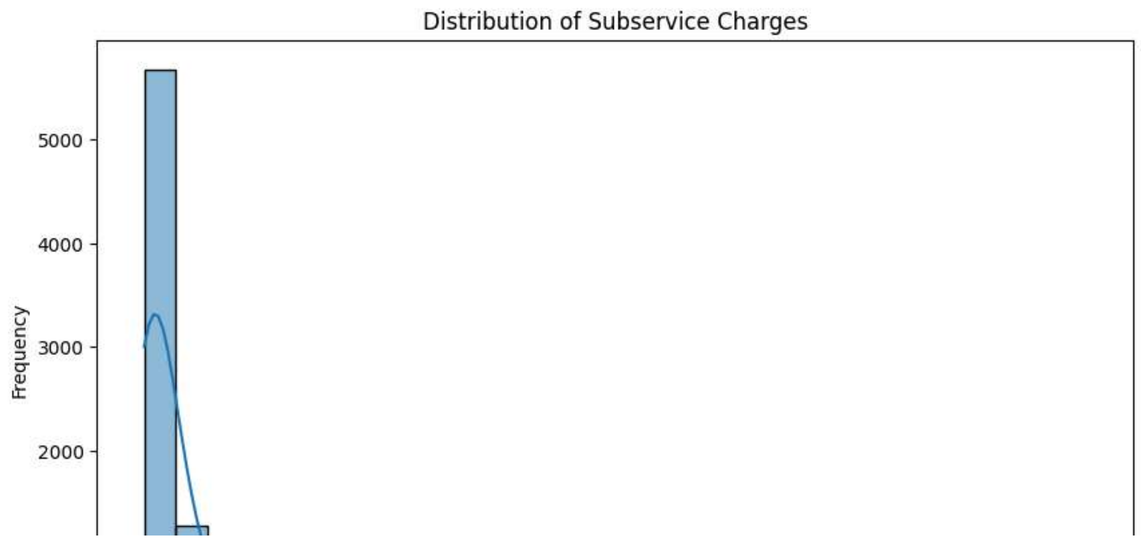
```
# Descriptive statistics of the dataset
df.describe()
```

	service	subservice_name	subservice_charge	city_name	country_name	source
count	8488	8488	8488	8488	8488	8488
unique	5	228	271	43	1	1
top	ac_service_repair	Material Procurement Charges	['₹119']	ahmedabad	India	UrbanCompany
freq	2622	84	353	199	8488	8488

```
# Importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset (make sure the path is correct)
df = pd.read_csv('/content/sample_data/Urban company data /uc.csv')

# Display the first few rows of the dataset
print(df.head())

# Basic Data Cleaning
# Check for missing values
print("Missing values in each column:")
print(df.isnull().sum())

# Fill missing values (using forward fill)
df.ffill(inplace=True)

# Inspect the 'subservice_charge' column to ensure proper format
print("Sample subservice_charge data:")
print(df['subservice_charge'].sample(5))
```

```
# Convert 'subservice_charge' from string representation to numeric values for analysis
def convert_charge(charge):
    # Safely parse the string representation into a list of floats
    try:
        return [float(i.replace('₹', '').replace(',', '').strip()) for i in eval(charge)]
    except:
        return [0]  # or handle this case as necessary

df['subservice_charge'] = df['subservice_charge'].apply(convert_charge)

# Flatten the list of charges into a single row for easier analysis
df_flat = df.explode('subservice_charge')

# Exploratory Data Analysis (EDA)
# Descriptive statistics of subservice charges
print("Descriptive statistics of subservice charges:")
print(df_flat['subservice_charge'].describe())

# Visualization 1: Distribution of Subservice Charges
plt.figure(figsize=(10, 6))
sns.histplot(df_flat['subservice_charge'], bins=30, kde=True)
plt.title('Distribution of Subservice Charges')
plt.xlabel('Subservice Charge (₹)')
plt.ylabel('Frequency')
plt.show()

# Visualization 2: Average Charge by Service Type
plt.figure(figsize=(12, 6))
average_charges = df_flat.groupby('service')['subservice_charge'].mean().reset_index()
sns.barplot(x='service', y='subservice_charge', data=average_charges)
plt.title('Average Subservice Charge by Service')
plt.xlabel('Service')
plt.ylabel('Average Charge (₹)')
plt.xticks(rotation=45)
plt.show()

# Correlation Heatmap (if applicable)
# Ensure there are numeric features to compute correlations
numeric_cols = df_flat.select_dtypes(include=['float64', 'int64']).columns
if len(numeric_cols) > 0:
    plt.figure(figsize=(10, 8))
    correlation = df_flat[numeric_cols].corr()
    sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f")
    plt.title('Correlation Heatmap')
    plt.show()
else:
    print("No numeric columns available for correlation analysis.")
```

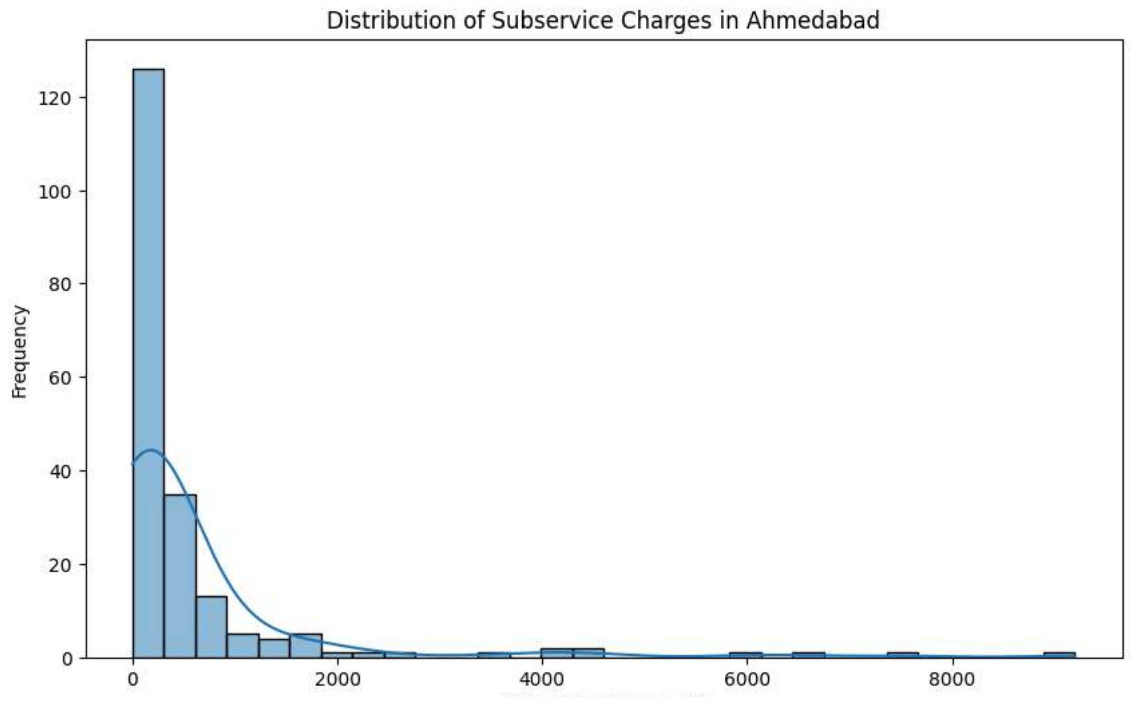
```
             service            subservice_name          subservice_charge  \
0  ac_service_repair  Non-Inverter PCB repaired                  ['₹1800']
1  ac_service_repair      Inverter PCB repaired                  ['₹4000']
2  ac_service_repair                Replace LVT  ['₹900', '₹499 (Labour)']
3  ac_service_repair           Capacitor 2-5 mfd  ['₹250', '₹349 (Labour)']
4  ac_service_repair         Capacitor 10-25 mfd  ['₹400', '₹349 (Labour)']

   city_name country_name        source
0  ahmedabad        India  UrbanCompany
1  ahmedabad        India  UrbanCompany
2  ahmedabad        India  UrbanCompany
3  ahmedabad        India  UrbanCompany
4  ahmedabad        India  UrbanCompany
Missing values in each column:
service              0
subservice_name      0
subservice_charge    0
city_name            0
country_name         0
source               0
dtype: int64
Sample subservice_charge data:
7200    ['₹750', '₹299 (Labour)']
543                     ['₹350']
5276                    ['₹269']
6937    ['₹350', '₹499 (Labour)']
1575                   ['₹1899']
Name: subservice_charge, dtype: object
Descriptive statistics of subservice charges:
count     8530
unique     202
top          0
freq      2364
Name: subservice_charge, dtype: int64
```



```
# Importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset (make sure the path is correct)
df = pd.read_csv('/content/sample_data/Urban company data /uc.csv')

# Display the first few rows of the dataset
print(df.head())

# Basic Data Cleaning
# Check for missing values
print("Missing values in each column:")
print(df.isnull().sum())

# Fill missing values (using forward fill)
df.ffill(inplace=True)

# Convert 'subservice_charge' from string representation to numeric values for analysis
def convert_charge(charge):
    try:
        return [float(i.replace('₹', '').replace(',', '').strip()) for i in eval(charge)]
    except:
        return [0]  # Handle the case as necessary

df['subservice_charge'] = df['subservice_charge'].apply(convert_charge)

# Flatten the list of charges into a single row for easier analysis
```

```
df_flat = df.explode('subservice_charge')

# Filter data for a specific city (e.g., "Ahmedabad")
city_name = "ahmedabad"  # Change this to the city you're interested in
df_city = df_flat[df_flat['city_name'].str.lower() == city_name]

# Check if the filtered DataFrame is empty
if df_city.empty:
    print(f"No data available for the city: {city_name}")
else:
    # Exploratory Data Analysis (EDA) for the specific city
    # Descriptive statistics of subservice charges for the city
    print("Descriptive statistics of subservice charges for", city_name)
    print(df_city['subservice_charge'].describe())

    # Visualization 1: Distribution of Subservice Charges for the city
    plt.figure(figsize=(10, 6))
    sns.histplot(df_city['subservice_charge'], bins=30, kde=True)
    plt.title(f'Distribution of Subservice Charges in {city_name.title()}')
    plt.xlabel('Subservice Charge (₹)')
    plt.ylabel('Frequency')
    plt.show()

    # Visualization 2: Average Charge by Service Type for the city
    plt.figure(figsize=(12, 6))
    average_charges = df_city.groupby('service')['subservice_charge'].mean().reset_index()
    sns.barplot(x='service', y='subservice_charge', data=average_charges)
    plt.title(f'Average Subservice Charge by Service in {city_name.title()}')
    plt.xlabel('Service')
    plt.ylabel('Average Charge (₹)')
    plt.xticks(rotation=45)
    plt.show()

    # Correlation Heatmap (if applicable)
    numeric_cols = df_city.select_dtypes(include=['float64', 'int64']).columns
    if len(numeric_cols) > 0:
        plt.figure(figsize=(10, 8))
        correlation = df_city[numeric_cols].corr()  # Calculate correlation on numeric columns
        sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f")
        plt.title(f'Correlation Heatmap for {city_name.title()}')
        plt.show()
    else:
        print("No numeric columns available for correlation analysis in", city_name)
```

```
              service            subservice_name          subservice_charge  \
0  ac_service_repair  Non-Inverter PCB repaired                  ['₹1800']
1  ac_service_repair      Inverter PCB repaired                  ['₹4000']
2  ac_service_repair                Replace LVT  ['₹900', '₹499 (Labour)']
3  ac_service_repair          Capacitor 2-5 mfd  ['₹250', '₹349 (Labour)']
4  ac_service_repair        Capacitor 10-25 mfd  ['₹400', '₹349 (Labour)']

   city_name country_name        source
0  ahmedabad        India  UrbanCompany
1  ahmedabad        India  UrbanCompany
2  ahmedabad        India  UrbanCompany
3  ahmedabad        India  UrbanCompany
4  ahmedabad        India  UrbanCompany
Missing values in each column:
service             0
subservice_name     0
subservice_charge   0
city_name           0
country_name        0
source              0
dtype: int64
Descriptive statistics of subservice charges for ahmedabad
count     200
unique     76
top         0
freq       55
Name: subservice_charge, dtype: int64
```



```
# Importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset (make sure the path is correct)
df = pd.read_csv('/content/sample_data/Urban company data /uc.csv')

# Display the first few rows of the dataset
print(df.head())

# Basic Data Cleaning
# Check for missing values
print("Missing values in each column:")
print(df.isnull().sum())

# Fill missing values (using forward fill)
df.ffill(inplace=True)

# Convert 'subservice_charge' from string representation to numeric values for analysis
def convert_charge(charge):
    try:
        return [float(i.replace('₹', '').replace(',', '').strip()) for i in eval(charge)]
    except:
        return [0]  # Handle the case as necessary

df['subservice_charge'] = df['subservice_charge'].apply(convert_charge)

# Flatten the list of charges into a single row for easier analysis
```

```
df_flat = df.explode('subservice_charge')

# Filter data for a specific city (e.g., "Ahmedabad")
city_name = "ahmedabad"  # Change this to the city you're interested in
df_city = df_flat[df_flat['city_name'].str.lower() == city_name]

# Check if the filtered DataFrame is empty
if df_city.empty:
    print(f"No data available for the city: {city_name}")
else:
    # Descriptive statistics of subservice charges for the city
    print("Descriptive statistics of subservice charges for", city_name)
    print(df_city['subservice_charge'].describe())

    # Count of services offered in the city
    service_counts = df_city['service'].value_counts()
    print("\nCount of Services Offered in", city_name)
    print(service_counts)

    # Visualization: Count of Services Offered
    plt.figure(figsize=(12, 6))
    sns.barplot(x=service_counts.index, y=service_counts.values)
    plt.title(f'Count of Services Offered in {city_name.title()}')
    plt.xlabel('Service')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.show()

    # Visualization: Distribution of Subservice Charges for the city
    plt.figure(figsize=(10, 6))
    sns.histplot(df_city['subservice_charge'], bins=30, kde=True)
    plt.title(f'Distribution of Subservice Charges in {city_name.title()}')
    plt.xlabel('Subservice Charge (₹)')
    plt.ylabel('Frequency')
    plt.show()

    # Visualization: Average Charge by Service Type for the city
    plt.figure(figsize=(12, 6))
    average_charges = df_city.groupby('service')['subservice_charge'].mean().reset_index()
    sns.barplot(x='service', y='subservice_charge', data=average_charges)
    plt.title(f'Average Subservice Charge by Service in {city_name.title()}')
    plt.xlabel('Service')
    plt.ylabel('Average Charge (₹)')
    plt.xticks(rotation=45)
    plt.show()

    # Correlation Heatmap (if applicable)
    numeric_cols = df_city.select_dtypes(include=['float64', 'int64']).columns
    if len(numeric_cols) > 0:
        plt.figure(figsize=(10, 8))
        correlation = df_city[numeric_cols].corr()  # Calculate correlation on numeric columns
        sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f")
        plt.title(f'Correlation Heatmap for {city_name.title()}')
        plt.show()
    else:
        print("No numeric columns available for correlation analysis in", city_name)
```

```
             service            subservice_name          subservice_charge  \
0  ac_service_repair  Non-Inverter PCB repaired                 ['₹1800']
1  ac_service_repair      Inverter PCB repaired                 ['₹4000']
```