OwusuSefah540ProjectMilestone2

August 7, 2024

0.1 Final Project Milestone 2

Bernard Owusu Sefah

DSC540

Data Cleaning for Meteorite Landings

06/12/2024 Imports

[2]:
```
import pandas as pd
```

Load the dataset

[3]:
```
# Specify the file path of the CSV file
file_path = 'Meteorite_Landings_20240614.csv'
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(file_path)
```

Step 1: Replace Headers

Description: Standardize the column headers to lowercase and replace spaces with underscores for consistency

[4]:
```
# Clean and standardize the column names
# - Strip leading/trailing whitespace
# - Convert to lowercase
# - Replace spaces with underscores
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')
```

Step 2: Handle Missing Values

Description: Identify and handle missing values in critical columns (e.g., mass_(g), year, reclat, reclong)

[5]:
```
# Replace missing values in the 'mass_(g)' column with 0
df['mass_(g)'] = df['mass_(g)'].fillna(0)
# Forward fill missing values in the 'year' column
# This propagates the last known year to subsequent missing values
df['year'] = df['year'].ffill()
# Drop rows with missing values in the 'reclat' and 'reclong' columns
```

```
df.dropna(subset=['reclat', 'reclong'], inplace=True)
```

Step 3: Convert Data Types

Description: Ensure that columns are in appropriate data types, such as converting year to integer

[6]:
```
# Convert the 'year' column to integer data type
df['year'] = df['year'].astype(int)
# Convert the 'mass_(g)' column to float data type
df['mass_(g)'] = df['mass_(g)'].astype(float)
```

Step 4: Remove Duplicates

Description: Remove any duplicate records to ensure data quality

[7]:
```
# Drop duplicate rows from the DataFrame
df.drop_duplicates(inplace=True)
```

Step 5: Outlier Detection and Handling

Description: Identify and handle outliers in the mass_(g) column by capping the mass values at a reasonable threshold

[8]:
```
# Calculate the upper limit for mass values (99th percentile)
mass_upper_limit = df['mass_(g)'].quantile(0.99)
# Cap the mass values at the upper limit
# If a mass value exceeds the upper limit, replace it with the upper limit
df['mass_(g)'] = df['mass_(g)'].apply(lambda x: mass_upper_limit if x >␣
↪mass_upper_limit else x)
```

[9]:
```
# Print the first few rows of the cleaned DataFrame
print(df.head())
```

```
       name   id nametype     recclass  mass_(g)  fall  year    reclat  \
0    Aachen    1    Valid           L5      21.0  Fell  1880  50.77500
1    Aarhus    2    Valid           H6     720.0  Fell  1951  56.18333
2      Abee    6    Valid          EH4   63000.0  Fell  1952  54.21667
3  Acapulco   10    Valid  Acapulcoite    1914.0  Fell  1976  16.88333
4   Achiras  370    Valid           L6     780.0  Fell  1902 -33.16667

     reclong           geolocation
0    6.08333     (50.775, 6.08333)
1   10.23333  (56.18333, 10.23333)
2 -113.00000    (54.21667, -113.0)
3  -99.90000     (16.88333, -99.9)
4  -64.95000    (-33.16667, -64.95)
```

[13]:
```
# Print the summary information of the cleaned DataFrame
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 38401 entries, 0 to 45715
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         38401 non-null  object
 1   id           38401 non-null  int64
 2   nametype     38401 non-null  object
 3   recclass     38401 non-null  object
 4   mass_(g)     38401 non-null  float64
 5   fall         38401 non-null  object
 6   year         38401 non-null  int32
 7   reclat       38401 non-null  float64
 8   reclong      38401 non-null  float64
 9   geolocation  38401 non-null  object
dtypes: float64(3), int32(1), int64(1), object(5)
memory usage: 3.1+ MB
None
```

Save the cleaned dataset

[16]:
```
# Specify the file path for the cleaned CSV file
cleaned_file_path = 'Cleaned_Meteorite_Landings.csv'
# Save the cleaned DataFrame to a new CSV file
df.to_csv(cleaned_file_path, index=False)
```

0.2 Ethical Implications of Data Wrangling

During the data cleaning process, several changes were made to the dataset, including standardizing column headers, handling missing values, converting data types, removing duplicates, and managing outliers. These transformations were necessary to ensure data consistency, accuracy, and usability for analysis. There are no specific legal or regulatory guidelines for the meteorite landing data itself, but general data handling best practices were followed. One risk is the potential misrepresentation of data after handling outliers, which could lead to incorrect conclusions. Assumptions were made to fill missing values and cap extreme mass values, which might affect the integrity of the data. The dataset was sourced from a reputable open data portal, ensuring its credibility and ethical acquisition. To mitigate ethical implications, transparent documentation of all data transformations was maintained, and original data was preserved for reference to avoid misinterpretation.

0.3 Milestone 3

Cleaning/Formatting Website Data Imports

[2]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

Step 1: Scrape the data from the website

Description: Fetch the HTML content and parse it using BeautifulSoup

[3]:

```
url = 'https://worldpopulationreview.com/world-cities'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')
```

Step 2: Extract the table data

Description: Locate the table in the HTML and extract its rows and columns

[74]:

```
table = soup.find('table')

# Verify that the table is correctly found
if table:
    print("Table found")
else:
    print("Table not found")
    exit()

# Extract headers
headers = ['Rank', 'City', 'Country', '2024 Population', '2023 Population',␣
↪'Growth Rate']
print("Headers extracted:", headers)

# Extract rows and focus only on relevant columns
rows = []
for i, row in enumerate(table.find_all('tr')[1:]):  # Skip the header row
    columns = row.find_all('td')
    if not columns:  # Skip rows without columns
        print(f"Skipping Row {i} because it is empty")
        continue
    if len(columns) >= 5:  # Ensure the row has at least 5 columns
        row_data = [
            i,  # Rank (sequential number)
            columns[0].text.strip() if columns[0].find('a') is None else␣
↪columns[0].find('a').text.strip(),  # City
            columns[1].text.strip() if columns[1].find('a') is None else␣
↪columns[1].find('a').text.strip(),  # Country
            columns[2].text.strip().replace(',', ''),  # 2024 Population
            columns[3].text.strip().replace(',', ''),  # 2023 Population
            columns[4].text.strip().replace('%', '').replace(',', '')  # Growth␣
↪Rate
        ]
        print(f"Row {i} data: {row_data}")  # Debug: Print the extracted row␣
↪data
        rows.append(row_data)
    else:
        print(f"Skipping Row {i} due to insufficient columns (found␣
↪{len(columns)})")
```

```python
# Convert to DataFrame
df = pd.DataFrame(rows, columns=headers)

# Print initial data
print("Initial data extracted:")
print(df.head())
```

```
Table found
Headers extracted: ['Rank', 'City', 'Country', '2024 Population', '2023
Population', 'Growth Rate']
Row 0 data: [0, 'Tokyo', 'Japan', '37115035', '37194105', '-0.21']
Row 1 data: [1, 'Delhi', 'India', '33807403', '32941309', '2.63']
Row 2 data: [2, 'Shanghai', 'China', '29867918', '29210808', '2.25']
Row 3 data: [3, 'Dhaka', 'Bangladesh', '23935652', '23209616', '3.13']
Row 4 data: [4, 'Sao Paulo', 'Brazil', '22806704', '22619736', '0.83']
Row 5 data: [5, 'Cairo', 'Egypt', '22623874', '22183201', '1.99']
Row 6 data: [6, 'Mexico City', 'Mexico', '22505315', '22281442', '1']
Row 7 data: [7, 'Beijing', 'China', '22189082', '21766214', '1.94']
Row 8 data: [8, 'Mumbai', 'India', '21673149', '21296517', '1.77']
Row 9 data: [9, 'Osaka', 'Japan', '18967459', '19013434', '-0.24']
Row 10 data: [10, 'Chongqing', 'China', '17773923', '17340704', '2.5']
Row 11 data: [11, 'Karachi', 'Pakistan', '17648555', '17236230', '2.39']
Row 12 data: [12, 'Kinshasa', 'DR Congo', '17032322', '16315534', '4.39']
Row 13 data: [13, 'Lagos', 'Nigeria', '16536018', '15945912', '3.7']
Row 14 data: [14, 'Istanbul', 'Turkey', '16047350', '15847768', '1.26']
Row 15 data: [15, 'Buenos Aires', 'Argentina', '15618288', '15490415', '0.83']
Row 16 data: [16, 'Kolkata', 'India', '15570786', '15332793', '1.55']
Row 17 data: [17, 'Manila', 'Philippines', '14941953', '14667089', '1.87']
Row 18 data: [18, 'Guangzhou', 'China', '14590096', '14284353', '2.14']
Row 19 data: [19, 'Tianjin', 'China', '14470873', '14238643', '1.63']
Row 20 data: [20, 'Lahore', 'Pakistan', '14407074', '13979390', '3.06']
Row 21 data: [21, 'Bangalore', 'India', '14008262', '13607800', '2.94']
Row 22 data: [22, 'Rio De Janeiro', 'Brazil', '13824347', '13727720', '0.7']
Row 23 data: [23, 'Shenzhen', 'China', '13311855', '13072633', '1.83']
Row 24 data: [24, 'Moscow', 'Russia', '12712305', '12680389', '0.25']
Row 25 data: [25, 'Chennai', 'India', '12053697', '11776147', '2.36']
Row 26 data: [26, 'Bogota', 'Colombia', '11658211', '11507960', '1.31']
Row 27 data: [27, 'Jakarta', 'Indonesia', '11436004', '11248839', '1.66']
Row 28 data: [28, 'Lima', 'Peru', '11361938', '11204382', '1.41']
Row 29 data: [29, 'Paris', 'France', '11276701', '11208440', '0.61']
Row 30 data: [30, 'Bangkok', 'Thailand', '11233869', '11069982', '1.48']
Row 31 data: [31, 'Hyderabad', 'India', '11068877', '10801163', '2.48']
Row 32 data: [32, 'Seoul', 'South Korea', '10004840', '9988049', '0.17']
Row 33 data: [33, 'Nanjing', 'China', '9947548', '9698464', '2.57']
Row 34 data: [34, 'Chengdu', 'China', '9828110', '9653772', '1.81']
Row 35 data: [35, 'London', 'United Kingdom', '9748033', '9648110', '1.04']
Row 36 data: [36, 'Luanda', 'Angola', '9651032', '9292336', '3.86']
Row 37 data: [37, 'Tehran', 'Iran', '9616007', '9499781', '1.22']
```

```
Row 38 data: [38, 'Ho Chi Minh City', 'Vietnam', '9567656', '9320866', '2.65']
Row 39 data: [39, 'Nagoya', 'Japan', '9556879', '9569328', '-0.13']
Row 40 data: [40, 'Xi An Shaanxi', 'China', '9013837', '8785174', '2.6']
Row 41 data: [41, 'Ahmedabad', 'India', '8854444', '8650605', '2.36']
Row 42 data: [42, 'Wuhan', 'China', '8850850', '8718250', '1.52']
Row 43 data: [43, 'Kuala Lumpur', 'Malaysia', '8815630', '8621724', '2.25']
Row 44 data: [44, 'Hangzhou', 'China', '8419842', '8237206', '2.22']
Row 45 data: [45, 'Suzhou', 'China', '8350625', '8074031', '3.43']
Row 46 data: [46, 'Surat', 'India', '8330528', '8064949', '3.29']
Row 47 data: [47, 'Dar Es Salaam', 'Tanzania', '8161231', '7775865', '4.96']
Row 48 data: [48, 'New York', 'United States', '8097282', '8258035', '-1.95']
Row 49 data: [49, 'Baghdad', 'Iraq', '7921134', '7711305', '2.72']
Row 50 data: [50, 'Shenyang', 'China', '7830377', '7680967', '1.95']
Row 51 data: [51, 'Riyadh', 'Saudi Arabia', '7820551', '7682430', '1.8']
Row 52 data: [52, 'Hong Kong', 'Hong Kong', '7725859', '7684801', '0.53']
Row 53 data: [53, 'Foshan', 'China', '7704935', '7597386', '1.42']
Row 54 data: [54, 'Dongguan', 'China', '7675146', '7587049', '1.16']
Row 55 data: [55, 'Pune', 'India', '7345848', '7166374', '2.5']
Row 56 data: [56, 'Santiago', 'Chile', '6950952', '6903392', '0.69']
Row 57 data: [57, 'Haerbin', 'China', '6938008', '6803811', '1.97']
Row 58 data: [58, 'Madrid', 'Spain', '6783241', '6751374', '0.47']
Row 59 data: [59, 'Khartoum', 'Sudan', '6542070', '6344348', '3.12']
Row 60 data: [60, 'Toronto', 'Canada', '6431430', '6371958', '0.93']
Row 61 data: [61, 'Johannesburg', 'South Africa', '6324351', '6198016', '2.04']
Row 62 data: [62, 'Belo Horizonte', 'Brazil', '6300409', '6247889', '0.84']
Row 63 data: [63, 'Dalian', 'China', '6217487', '6077995', '2.3']
Row 64 data: [64, 'Singapore', 'Singapore', '6119203', '6080859', '0.63']
Row 65 data: [65, 'Qingdao', 'China', '6104597', '5986525', '1.97']
Row 66 data: [66, 'Zhengzhou', 'China', '6014887', '5859272', '2.66']
Row 67 data: [67, 'Ji Nan Shandong', 'China', '5940698', '5806031', '2.32']
Row 68 data: [68, 'Abidjan', 'Ivory Coast', '5866704', '5686350', '3.17']
Row 69 data: [69, 'Barcelona', 'Spain', '5711917', '5687356', '0.43']
Row 70 data: [70, 'Yangon', 'Myanmar', '5709678', '5610241', '1.77']
Row 71 data: [71, 'Addis Ababa', 'Ethiopia', '5703628', '5460591', '4.45']
Row 72 data: [72, 'Alexandria', 'Egypt', '5696131', '5588477', '1.93']
Row 73 data: [73, 'Saint Petersburg', 'Russia', '5581707', '5561294', '0.37']
Row 74 data: [74, 'Nairobi', 'Kenya', '5541172', '5325160', '4.06']
Row 75 data: [75, 'Chittagong', 'Bangladesh', '5513609', '5379660', '2.49']
Row 76 data: [76, 'Guadalajara', 'Mexico', '5499678', '5419880', '1.47']
Row 77 data: [77, 'Fukuoka', 'Japan', '5478076', '5490271', '-0.22']
Row 78 data: [78, 'Ankara', 'Turkey', '5477087', '5397098', '1.48']
Row 79 data: [79, 'Hanoi', 'Vietnam', '5431801', '5253385', '3.4']
Row 80 data: [80, 'Melbourne', 'Australia', '5315600', '5235407', '1.53']
Row 81 data: [81, 'Monterrey', 'Mexico', '5195355', '5116647', '1.54']
Row 82 data: [82, 'Sydney', 'Australia', '5184896', '5120894', '1.25']
Row 83 data: [83, 'Changsha', 'China', '5027975', '4921487', '2.16']
Row 84 data: [84, 'Urumqi', 'China', '5005964', '4865038', '2.9']
Row 85 data: [85, 'Cape Town', 'South Africa', '4977833', '4890280', '1.79']
```

Row 86 data: [86, 'Jiddah', 'Saudi Arabia', '4943210', '4862941', '1.65']
Row 87 data: [87, 'Brasilia', 'Brazil', '4935274', '4873048', '1.28']
Row 88 data: [88, 'Kunming', 'China', '4861079', '4761284', '2.1']
Row 89 data: [89, 'Changchun', 'China', '4802447', '4710382', '1.95']
Row 90 data: [90, 'Kabul', 'Afghanistan', '4728384', '4588666', '3.04']
Row 91 data: [91, 'Hefei', 'China', '4727290', '4615758', '2.42']
Row 92 data: [92, 'Yaounde', 'Cameroon', '4681768', '4509287', '3.83']
Row 93 data: [93, 'Ningbo', 'China', '4659830', '4537901', '2.69']
Row 94 data: [94, 'Shantou', 'China', '4656525', '4573713', '1.81']
Row 95 data: [95, 'New Taipei', 'Taiwan', '4534877', '4504147', '0.68']
Row 96 data: [96, 'Tel Aviv', 'Israel', '4495727', '4420855', '1.69']
Row 97 data: [97, 'Kano', 'Nigeria', '4490734', '4348481', '3.27']
Row 98 data: [98, 'Shijiazhuang', 'China', '4454132', '4370473', '1.91']
Row 99 data: [99, 'Montreal', 'Canada', '4341638', '4307958', '0.78']
Row 100 data: [100, 'Rome', 'Italy', '4331974', '4315671', '0.38']
Row 101 data: [101, 'Jaipur', 'India', '4308510', '4207084', '2.41']
Row 102 data: [102, 'Recife', 'Brazil', '4305127', '4263940', '0.97']
Row 103 data: [103, 'Nanning', 'China', '4291463', '4191890', '2.38']
Row 104 data: [104, 'Fortaleza', 'Brazil', '4246399', '4206240', '0.95']
Row 105 data: [105, 'Kozhikode', 'India', '4243962', '4088555', '3.8']
Row 106 data: [106, 'Porto Alegre', 'Brazil', '4239867', '4211933', '0.66']
Row 107 data: [107, 'Taiyuan Shanxi', 'China', '4226782', '4145010', '1.97']
Row 108 data: [108, 'Douala', 'Cameroon', '4203108', '4063200', '3.44']
Row 109 data: [109, 'Ekurhuleni', 'South Africa', '4190832', '4118327', '1.76']
Row 110 data: [110, 'Malappuram', 'India', '4184921', '4009087', '4.39']
Row 111 data: [111, 'Medellin', 'Colombia', '4137386', '4102308', '0.86']
Row 112 data: [112, 'Changzhou', 'China', '4085502', '3981658', '2.61']
Row 113 data: [113, 'Kampala', 'Uganda', '4050826', '3846102', '5.32']
Row 114 data: [114, 'Antananarivo', 'Madagascar', '4048666', '3872264', '4.56']
Row 115 data: [115, 'Lucknow', 'India', '4038214', '3945409', '2.35']
Row 116 data: [116, 'Abuja', 'Nigeria', '4025735', '3839646', '4.85']
Row 117 data: [117, 'Nanchang', 'China', '4016037', '3920379', '2.44']
Row 118 data: [118, 'Wenzhou', 'China', '4009531', '3919724', '2.29']
Row 119 data: [119, 'Xiamen', 'China', '4007468', '3935484', '1.83']
Row 120 data: [120, 'Ibadan', 'Nigeria', '4004316', '3874908', '3.34']
Row 121 data: [121, 'Fuzhou Fujian', 'China', '3998754', '3922202', '1.95']
Row 122 data: [122, 'Salvador', 'Brazil', '3994982', '3958384', '0.92']
Row 123 data: [123, 'Casablanca', 'Morocco', '3950408', '3892837', '1.48']
Row 124 data: [124, 'Tangshan Hebei', 'China', '3925206', '3814702', '2.9']
Row 125 data: [125, 'Kumasi', 'Ghana', '3903481', '3768239', '3.59']
Row 126 data: [126, 'Curitiba', 'Brazil', '3852459', '3813082', '1.03']
Row 127 data: [127, 'Bekasi', 'Indonesia', '3830678', '3729351', '2.72']
Row 128 data: [128, 'Faisalabad', 'Pakistan', '3800193', '3710845', '2.41']
Row 129 data: [129, 'Los Angeles', 'United States', '3795936', '3820914', '-0.65']
Row 130 data: [130, 'Guiyang', 'China', '3661446', '3580904', '2.25']
Row 131 data: [131, 'Port Harcourt', 'Nigeria', '3636547', '3480101', '4.5']
Row 132 data: [132, 'Thrissur', 'India', '3605238', '3482456', '3.53']

Row 133 data: [133, 'Santo Domingo', 'Dominican Republic', '3587402', '3523890', '1.8']
Row 134 data: [134, 'Berlin', 'Germany', '3576873', '3573938', '0.08']
Row 135 data: [135, 'Asuncion', 'Paraguay', '3568830', '3510511', '1.66']
Row 136 data: [136, 'Dakar', 'Senegal', '3540462', '3429536', '3.23']
Row 137 data: [137, 'Kochi', 'India', '3507053', '3406055', '2.97']
Row 138 data: [138, 'Wuxi', 'China', '3498740', '3437346', '1.79']
Row 139 data: [139, 'Busan', 'South Korea', '3477419', '3471949', '0.16']
Row 140 data: [140, 'Campinas', 'Brazil', '3458441', '3422796', '1.04']
Row 141 data: [141, 'Mashhad', 'Iran', '3415532', '3367852', '1.42']
Row 142 data: [142, 'Sanaa', 'Yemen', '3407814', '3292497', '3.5']
Row 143 data: [143, 'Puebla', 'Mexico', '3394342', '3344761', '1.48']
Row 144 data: [144, 'Indore', 'India', '3393380', '3302077', '2.77']
Row 145 data: [145, 'Lanzhou', 'China', '3365910', '3297528', '2.07']
Row 146 data: [146, 'Ouagadougou', 'Burkina Faso', '3358934', '3203923', '4.84']
Row 147 data: [147, 'Kuwait City', 'Kuwait', '3353602', '3297759', '1.69']
Row 148 data: [148, 'Lusaka', 'Zambia', '3324219', '3181250', '4.49']
Row 149 data: [149, 'Kanpur', 'India', '3286142', '3234160', '1.61']
Row 150 data: [150, 'Durban', 'South Africa', '3262128', '3228003', '1.06']
Row 151 data: [151, 'Guayaquil', 'Ecuador', '3193267', '3142466', '1.62']
Row 152 data: [152, 'Pyongyang', 'North Korea', '3183135', '3157538', '0.81']
Row 153 data: [153, 'Milan', 'Italy', '3160631', '3154570', '0.19']
Row 154 data: [154, 'Guatemala City', 'Guatemala', '3159631', '3095099', '2.08']
Row 155 data: [155, 'Athens', 'Greece', '3154591', '3154463', '0']
Row 156 data: [156, 'Depok', 'Indonesia', '3133298', '3041229', '3.03']
Row 157 data: [157, 'Izmir', 'Turkey', '3120340', '3088414', '1.03']
Row 158 data: [158, 'Nagpur', 'India', '3106340', '3046687', '1.96']
Row 159 data: [159, 'Surabaya', 'Indonesia', '3088748', '3044413', '1.46']
Row 160 data: [160, 'Handan', 'China', '3085998', '3005409', '2.68']
Row 161 data: [161, 'Coimbatore', 'India', '3083721', '3009047', '2.48']
Row 162 data: [162, 'Huaian', 'China', '3071048', '2979893', '3.06']
Row 163 data: [163, 'Port Au Prince', 'Haiti', '3060169', '2987455', '2.43']
Row 164 data: [164, 'Zhongshan', 'China', '3051065', '3010685', '1.34']
Row 165 data: [165, 'Dubai', 'United Arab Emirates', '3051016', '3007583', '1.44']
Row 166 data: [166, 'Bamako', 'Mali', '3050570', '2929373', '4.14']
Row 167 data: [167, 'Mbuji Mayi', 'DR Congo', '3022855', '2891746', '4.53']
Row 168 data: [168, 'Kiev', 'Ukraine', '3020228', '3016789', '0.11']
Row 169 data: [169, 'Lisbon', 'Portugal', '3014607', '3000536', '0.47']
Row 170 data: [170, 'Weifang', 'China', '2994537', '2917819', '2.63']
Row 171 data: [171, 'Caracas', 'Venezuela', '2991727', '2972145', '0.66']
Row 172 data: [172, 'Thiruvananthapuram', 'India', '2984154', '2891119', '3.22']
Row 173 data: [173, 'Algiers', 'Algeria', '2952115', '2901810', '1.73']
Row 174 data: [174, 'Shizuoka', 'Japan', '2935527', '2937359', '-0.06']
Row 175 data: [175, 'Lubumbashi', 'DR Congo', '2933962', '2811959', '4.34']
Row 176 data: [176, 'Cali', 'Colombia', '2890433', '2863730', '0.93']
Row 177 data: [177, 'Goiania', 'Brazil', '2890418', '2848473', '1.47']
Row 178 data: [178, 'Pretoria', 'South Africa', '2889899', '2818100', '2.55']

Row 179 data: [179, 'Shaoxing', 'China', '2882171', '2805654', '2.73']
Row 180 data: [180, 'Incheon', 'South Korea', '2861686', '2848557', '0.46']
Row 181 data: [181, 'Yantai', 'China', '2834508', '2764584', '2.53']
Row 182 data: [182, 'Zibo', 'China', '2828435', '2780142', '1.74']
Row 183 data: [183, 'Huizhou', 'China', '2827610', '2758593', '2.5']
Row 184 data: [184, 'Manchester', 'United Kingdom', '2811756', '2791005', '0.74']
Row 185 data: [185, 'Taipei', 'Taiwan', '2766334', '2754196', '0.44']
Row 186 data: [186, 'Mogadishu', 'Somalia', '2726815', '2610483', '4.46']
Row 187 data: [187, 'Brazzaville', 'Republic of the Congo', '2724566', '2637733', '3.29']
Row 188 data: [188, 'Accra', 'Ghana', '2721165', '2660072', '2.3']
Row 189 data: [189, 'Bandung', 'Indonesia', '2714215', '2674000', '1.5']
Row 190 data: [190, 'Damascus', 'Syria', '2685361', '2584771', '3.89']
Row 191 data: [191, 'Birmingham', 'United Kingdom', '2684807', '2665100', '0.74']
Row 192 data: [192, 'Vancouver', 'Canada', '2682509', '2657088', '0.96']
Row 193 data: [193, 'Toluca De Lerdo', 'Mexico', '2674336', '2626368', '1.83']
Row 194 data: [194, 'Luoyang', 'China', '2666744', '2602793', '2.46']
Row 195 data: [195, 'Sapporo', 'Japan', '2660947', '2666112', '-0.19']
Row 196 data: [196, 'Chicago', 'United States', '2638159', '2664452', '-0.99']
Row 197 data: [197, 'Tashkent', 'Uzbekistan', '2633661', '2603243', '1.17']
Row 198 data: [198, 'Patna', 'India', '2633243', '2579762', '2.07']
Row 199 data: [199, 'Bhopal', 'India', '2624865', '2564502', '2.35']
Row 200 data: [200, 'Tangerang', 'Indonesia', '2570980', '2514077', '2.26']
Row 201 data: [201, 'Nantong', 'China', '2555722', '2492230', '2.55']
Row 202 data: [202, 'Brisbane', 'Australia', '2536449', '2504505', '1.28']
Row 203 data: [203, 'Tunis', 'Tunisia', '2510673', '2475446', '1.42']
Row 204 data: [204, 'Peshawar', 'Pakistan', '2480546', '2411785', '2.85']
Row 205 data: [205, 'Medan', 'Indonesia', '2479070', '2439054', '1.64']
Row 206 data: [206, 'Gujranwala', 'Pakistan', '2479058', '2415416', '2.63']
Row 207 data: [207, 'Baku', 'Azerbaijan', '2464162', '2432304', '1.31']
Row 208 data: [208, 'Hohhot', 'China', '2443686', '2380636', '2.65']
Row 209 data: [209, 'San Juan', 'Puerto Rico', '2436620', '2439564', '-0.12']
Row 210 data: [210, 'Belem', 'Brazil', '2432177', '2409409', '0.94']
Row 211 data: [211, 'Rawalpindi', 'Pakistan', '2430388', '2377325', '2.23']
Row 212 data: [212, 'Agra', 'India', '2422342', '2367554', '2.31']
Row 213 data: [213, 'Manaus', 'Brazil', '2406854', '2375636', '1.31']
Row 214 data: [214, 'Kannur', 'India', '2405664', '2346137', '2.54']
Row 215 data: [215, 'Beirut', 'Lebanon', '2402485', '2421354', '-0.78']
Row 216 data: [216, 'Maracaibo', 'Venezuela', '2400826', '2367626', '1.4']
Row 217 data: [217, 'Liuzhou', 'China', '2397410', '2343452', '2.3']
Row 218 data: [218, 'Visakhapatnam', 'India', '2385110', '2330928', '2.32']
Row 219 data: [219, 'Baotou', 'China', '2381242', '2334878', '1.99']
Row 220 data: [220, 'Vadodara', 'India', '2373365', '2324084', '2.12']
Row 221 data: [221, 'Barranquilla', 'Colombia', '2373302', '2349400', '1.02']
Row 222 data: [222, 'Phnom Penh', 'Cambodia', '2352680', '2281198', '3.13']
Row 223 data: [223, 'Sendai', 'Japan', '2341433', '2342302', '-0.04']

Row 224 data: [224, 'Taoyuan', 'Taiwan', '2338724', '2318850', '0.86']
Row 225 data: [225, 'Xuzhou', 'China', '2332531', '2287166', '1.98']
Row 226 data: [226, 'Houston', 'United States', '2319119', '2314157', '0.21']
Row 227 data: [227, 'Aleppo', 'Syria', '2317650', '2203025', '5.2']
Row 228 data: [228, 'Tijuana', 'Mexico', '2297216', '2259787', '1.66']
Row 229 data: [229, 'Esfahan', 'Iran', '2294589', '2258396', '1.6']
Row 230 data: [230, 'Nashik', 'India', '2294299', '2237369', '2.54']
Row 231 data: [231, 'Vijayawada', 'India', '2290785', '2229765', '2.74']
Row 232 data: [232, 'Amman', 'Jordan', '2252688', '2232240', '0.92']
Row 233 data: [233, 'Putian', 'China', '2250104', '2176022', '3.4']
Row 234 data: [234, 'Multan', 'Pakistan', '2205407', '2154600', '2.36']
Row 235 data: [235, 'Grande Vitoria', 'Brazil', '2196818', '2170513', '1.21']
Row 236 data: [236, 'Wuhu Anhui', 'China', '2195262', '2124797', '3.32']
Row 237 data: [237, 'Mecca', 'Saudi Arabia', '2184560', '2149928', '1.61']
Row 238 data: [238, 'Kollam', 'India', '2181940', '2106606', '3.58']
Row 239 data: [239, 'Naples', 'Italy', '2180027', '2179384', '0.03']
Row 240 data: [240, 'Daegu', 'South Korea', '2179929', '2180997', '-0.05']
Row 241 data: [241, 'Conakry', 'Guinea', '2178596', '2110937', '3.21']
Row 242 data: [242, 'Yangzhou', 'China', '2175102', '2131291', '2.06']
Row 243 data: [243, 'Havana', 'Cuba', '2152518', '2148930', '0.17']
Row 244 data: [244, 'Taizhou Zhejiang', 'China', '2152226', '2102143', '2.38']
Row 245 data: [245, 'Baoding', 'China', '2149703', '2107539', '2']
Row 246 data: [246, 'Perth', 'Australia', '2143491', '2117997', '1.2']
Row 247 data: [247, 'Brussels', 'Belgium', '2132178', '2121992', '0.48']
Row 248 data: [248, 'Linyi Shandong', 'China', '2120049', '2076364', '2.1']
Row 249 data: [249, 'Bursa', 'Turkey', '2115513', '2086324', '1.4']
Row 250 data: [250, 'Rajkot', 'India', '2096981', '2043107', '2.64']
Row 251 data: [251, 'Minsk', 'Belarus', '2064733', '2057257', '0.36']
Row 252 data: [252, 'Hiroshima', 'Japan', '2062884', '2067591', '-0.23']
Row 253 data: [253, 'Haikou', 'China', '2056646', '2016092', '2.01']
Row 254 data: [254, 'Daqing', 'China', '2044371', '2000656', '2.19']
Row 255 data: [255, 'Lome', 'Togo', '2042734', '1981615', '3.08']
Row 256 data: [256, 'Lianyungang', 'China', '2035631', '1986439', '2.48']
Row 257 data: [257, 'Yancheng Jiangsu', 'China', '2034326', '1993463', '2.05']
Row 258 data: [258, 'Panama City', 'Panama', '2015735', '1976866', '1.97']
Row 259 data: [259, 'Almaty', 'Kazakhstan', '2015209', '1987301', '1.4']
Row 260 data: [260, 'Semarang', 'Indonesia', '2013571', '1975306', '1.94']
Row 261 data: [261, 'Hyderabad', 'Pakistan', '2011964', '1967684', '2.25']
Row 262 data: [262, 'Valencia', 'Venezuela', '2007265', '1983445', '1.2']
Row 263 data: [263, 'Davao City', 'Philippines', '1991457', '1949400', '2.16']
Row 264 data: [264, 'Vienna', 'Austria', '1990487', '1975271', '0.77']
Row 265 data: [265, 'Rabat', 'Morocco', '1989197', '1959388', '1.52']
Row 266 data: [266, 'Ludhiana', 'India', '1988438', '1951085', '1.91']
Row 267 data: [267, 'Quito', 'Ecuador', '1986667', '1956995', '1.52']
Row 268 data: [268, 'Benin City', 'Nigeria', '1972558', '1904631', '3.57']
Row 269 data: [269, 'La Paz', 'Bolivia', '1965570', '1935619', '1.55']
Row 270 data: [270, 'Baixada Santista', 'Brazil', '1965110', '1947785', '0.89']
Row 271 data: [271, 'West Yorkshire', 'United Kingdom', '1942470', '1928661',

'0.72']
Row 272 data: [272, 'Can Tho', 'Vietnam', '1938915', '1865172', '3.95']
Row 273 data: [273, 'Zhuhai', 'China', '1930373', '1889829', '2.15']
Row 274 data: [274, 'Leon De Los Aldamas', 'Mexico', '1924435', '1898749', '1.35']
Row 275 data: [275, 'Quanzhou', 'China', '1920733', '1869994', '2.71']
Row 276 data: [276, 'Matola', 'Mozambique', '1915035', '1852405', '3.38']
Row 277 data: [277, 'Datong', 'China', '1913674', '1873693', '2.13']
Row 278 data: [278, 'Sharjah', 'United Arab Emirates', '1872199', '1830858', '2.26']
Row 279 data: [279, 'Madurai', 'India', '1871912', '1834279', '2.05']
Row 280 data: [280, 'Raipur', 'India', '1871107', '1816813', '2.99']
Row 281 data: [281, 'Adana', 'Turkey', '1856638', '1835895', '1.13']
Row 282 data: [282, 'Santa Cruz', 'Bolivia', '1855732', '1820114', '1.96']
Row 283 data: [283, 'Palembang', 'Indonesia', '1852673', '1818421', '1.88']
Row 284 data: [284, 'Mosul', 'Iraq', '1847691', '1792020', '3.11']
Row 285 data: [285, 'Cixi', 'China', '1840039', '1788871', '2.86']
Row 286 data: [286, 'Meerut', 'India', '1835403', '1797805', '2.09']
Row 287 data: [287, 'Gaziantep', 'Turkey', '1833006', '1804704', '1.57']
Row 288 data: [288, 'La Laguna', 'Mexico', '1826135', '1780592', '2.56']
Row 289 data: [289, 'Batam', 'Indonesia', '1806147', '1748142', '3.32']
Row 290 data: [290, 'Turin', 'Italy', '1805727', '1801944', '0.21']
Row 291 data: [291, 'Warsaw', 'Poland', '1799451', '1797516', '0.11']
Row 292 data: [292, 'Jiangmen', 'China', '1797469', '1768511', '1.64']
Row 293 data: [293, 'Varanasi', 'India', '1789047', '1754425', '1.97']
Row 294 data: [294, 'Hamburg', 'Germany', '1787280', '1787520', '-0.01']
Row 295 data: [295, 'Montevideo', 'Uruguay', '1781363', '1774396', '0.39']
Row 296 data: [296, 'Budapest', 'Hungary', '1780391', '1778052', '0.13']
Row 297 data: [297, 'Lyon', 'France', '1774395', '1761188', '0.75']
Row 298 data: [298, 'Xiangyang', 'China', '1772318', '1742706', '1.7']
Row 299 data: [299, 'Bucharest', 'Romania', '1767520', '1776385', '-0.5']
Row 300 data: [300, 'Yichang', 'China', '1758100', '1711244', '2.74']
Row 301 data: [301, 'Yinchuan', 'China', '1757699', '1716591', '2.39']
Row 302 data: [302, 'Shiraz', 'Iran', '1742750', '1720954', '1.27']
Row 303 data: [303, 'Kananga', 'DR Congo', '1738716', '1664442', '4.46']
Row 304 data: [304, 'Srinagar', 'India', '1737502', '1698277', '2.31']
Row 305 data: [305, 'Monrovia', 'Liberia', '1735365', '1678020', '3.42']
Row 306 data: [306, 'Tiruppur', 'India', '1731862', '1677173', '3.26']
Row 307 data: [307, 'Jamshedpur', 'India', '1730521', '1695060', '2.09']
Row 308 data: [308, 'Suqian', 'China', '1726327', '1664968', '3.69']
Row 309 data: [309, 'Aurangabad', 'India', '1725283', '1683389', '2.49']
Row 310 data: [310, 'Qinhuangdao', 'China', '1723728', '1680962', '2.54']
Row 311 data: [311, 'Stockholm', 'Sweden', '1719604', '1700066', '1.15']
Row 312 data: [312, 'Anshan', 'China', '1713452', '1689499', '1.42']
Row 313 data: [313, 'Glasgow', 'United Kingdom', '1708147', '1698088', '0.59']
Row 314 data: [314, 'Xining', 'China', '1705078', '1666605', '2.31']
Row 315 data: [315, 'Makassar', 'Indonesia', '1704930', '1673094', '1.9']
Row 316 data: [316, 'Hengyang', 'China', '1702012', '1661854', '2.42']

Row 317 data: [317, 'Novosibirsk', 'Russia', '1701510', '1694765', '0.4']
Row 318 data: [318, 'Ulaanbaatar', 'Mongolia', '1699363', '1672627', '1.6']
Row 319 data: [319, 'Onitsha', 'Nigeria', '1694913', '1623382', '4.41']
Row 320 data: [320, 'Jilin', 'China', '1693701', '1668785', '1.49']
Row 321 data: [321, 'Anyang', 'China', '1693375', '1642347', '3.11']
Row 322 data: [322, 'Auckland', 'New Zealand', '1692770', '1673220', '1.17']
Row 323 data: [323, 'Tabriz', 'Iran', '1678028', '1660737', '1.04']
Row 324 data: [324, 'Muscat', 'Oman', '1676167', '1650319', '1.57']
Row 325 data: [325, 'Calgary', 'Canada', '1665023', '1639613', '1.55']
Row 326 data: [326, 'Phoenix', 'United States', '1662607', '1650070', '0.76']
Row 327 data: [327, 'Qiqihaer', 'China', '1662407', '1635458', '1.65']
Row 328 data: [328, 'N Djamena', 'Chad', '1655618', '1592324', '3.97']
Row 329 data: [329, 'Marseille', 'France', '1635707', '1627549', '0.5']
Row 330 data: [330, 'Cordoba', 'Argentina', '1625937', '1611651', '0.89']
Row 331 data: [331, 'Jodhpur', 'India', '1625325', '1586547', '2.44']
Row 332 data: [332, 'Kathmandu', 'Nepal', '1621642', '1571010', '3.22']
Row 333 data: [333, 'Rosario', 'Argentina', '1613041', '1594096', '1.19']
Row 334 data: [334, 'Tegucigalpa', 'Honduras', '1609261', '1568025', '2.63']
Row 335 data: [335, 'Ciudad Juarez', 'Mexico', '1604085', '1582313', '1.38']
Row 336 data: [336, 'Harare', 'Zimbabwe', '1603201', '1578128', '1.59']
Row 337 data: [337, 'Karaj', 'Iran', '1603011', '1593608', '0.59']
Row 338 data: [338, 'Medina', 'Saudi Arabia', '1598976', '1572571', '1.68']
Row 339 data: [339, 'Jining Shandong', 'China', '1595963', '1569560', '1.68']
Row 340 data: [340, 'Abu Dhabi', 'United Arab Emirates', '1593284', '1566999', '1.68']
Row 341 data: [341, 'Munich', 'Germany', '1584507', '1576416', '0.51']
Row 342 data: [342, 'Ranchi', 'India', '1584237', '1547258', '2.39']
Row 343 data: [343, 'Daejon', 'South Korea', '1581705', '1577436', '0.27']
Row 344 data: [344, 'Zhangjiakou', 'China', '1568122', '1536171', '2.08']
Row 345 data: [345, 'Edmonton', 'Canada', '1567615', '1544448', '1.5']
Row 346 data: [346, 'Mandalay', 'Myanmar', '1563021', '1531860', '2.03']
Row 347 data: [347, 'Gaoxiong', 'Taiwan', '1559085', '1552899', '0.4']
Row 348 data: [348, 'Kota', 'India', '1558468', '1516795', '2.75']
Row 349 data: [349, 'Natal', 'Brazil', '1556413', '1535316', '1.37']
Row 350 data: [350, 'Nouakchott', 'Mauritania', '1552146', '1491958', '4.03']
Row 351 data: [351, 'Jabalpur', 'India', '1551004', '1522121', '1.9']
Row 352 data: [352, 'Huainan', 'China', '1538603', '1512668', '1.71']
Row 353 data: [353, 'Grande Sao Luis', 'Brazil', '1536017', '1523629', '0.81']
Row 354 data: [354, 'Asansol', 'India', '1534081', '1505033', '1.93']
Row 355 data: [355, 'Philadelphia', 'United States', '1533828', '1550542', '-1.08']
Row 356 data: [356, 'Yekaterinburg', 'Russia', '1532970', '1527728', '0.34']
Row 357 data: [357, 'Gwangju', 'South Korea', '1532902', '1529472', '0.22']
Row 358 data: [358, 'Yiwu', 'China', '1525749', '1483320', '2.86']
Row 359 data: [359, 'Chaozhou', 'China', '1520628', '1496298', '1.63']
Row 360 data: [360, 'San Antonio', 'United States', '1513974', '1495295', '1.25']
Row 361 data: [361, 'Gwalior', 'India', '1508846', '1475016', '2.29']

Row 362 data: [362, 'Ganzhou', 'China', '1508037', '1459678', '3.31']
Row 363 data: [363, 'Homs', 'Syria', '1499603', '1443429', '3.89']
Row 364 data: [364, 'Niamey', 'Niger', '1496258', '1437233', '4.11']
Row 365 data: [365, 'Mombasa', 'Kenya', '1495223', '1440396', '3.81']
Row 366 data: [366, 'Allahabad', 'India', '1493346', '1465152', '1.92']
Row 367 data: [367, 'Basra', 'Iraq', '1485156', '1448124', '2.56']
Row 368 data: [368, 'Kisangani', 'DR Congo', '1483513', '1423395', '4.22']
Row 369 data: [369, 'San Jose', 'Costa Rica', '1482460', '1461989', '1.4']
Row 370 data: [370, 'Amritsar', 'India', '1480470', '1451748', '1.98']
Row 371 data: [371, 'Taizhou Jiangsu', 'China', '1477600', '1446810', '2.13']
Row 372 data: [372, 'Chon Buri', 'Thailand', '1472709', '1454222', '1.27']
Row 373 data: [373, 'Jiaxing', 'China', '1470917', '1424982', '3.22']
Row 374 data: [374, 'Weihai', 'China', '1465926', '1429315', '2.56']
Row 375 data: [375, 'Hai Phong', 'Vietnam', '1463650', '1422974', '2.86']
Row 376 data: [376, 'Ottawa', 'Canada', '1451571', '1437188', '1']
Row 377 data: [377, 'Zurich', 'Switzerland', '1443349', '1431538', '0.83']
Row 378 data: [378, 'Taian Shandong', 'China', '1441911', '1416064', '1.83']
Row 379 data: [379, 'Queretaro', 'Mexico', '1436818', '1413474', '1.65']
Row 380 data: [380, 'Joao Pessoa', 'Brazil', '1435125', '1421827', '0.94']
Row 381 data: [381, 'Kaifeng', 'China', '1434463', '1398262', '2.59']
Row 382 data: [382, 'Cochabamba', 'Bolivia', '1430688', '1400250', '2.17']
Row 383 data: [383, 'Konya', 'Turkey', '1429935', '1407632', '1.58']
Row 384 data: [384, 'Liuyang', 'China', '1428802', '1373523', '4.02']
Row 385 data: [385, 'Liuan', 'China', '1427894', '1378077', '3.61']
Row 386 data: [386, 'Rizhao', 'China', '1426583', '1391258', '2.54']
Row 387 data: [387, 'Kharkiv', 'Ukraine', '1418978', '1421052', '-0.15']
Row 388 data: [388, 'Dhanbad', 'India', '1414532', '1389776', '1.78']
Row 389 data: [389, 'Nanchong', 'China', '1412714', '1377572', '2.55']
Row 390 data: [390, 'Dongying', 'China', '1410791', '1379377', '2.28']
Row 391 data: [391, 'Belgrade', 'Serbia', '1410697', '1408144', '0.18']
Row 392 data: [392, 'Zunyi', 'China', '1406091', '1376453', '2.15']
Row 393 data: [393, 'Zhanjiang', 'China', '1399310', '1371998', '1.99']
Row 394 data: [394, 'Bucaramanga', 'Colombia', '1396632', '1381498', '1.1']
Row 395 data: [395, 'Uyo', 'Nigeria', '1393453', '1329284', '4.83']
Row 396 data: [396, 'Copenhagen', 'Denmark', '1391205', '1381005', '0.74']
Row 397 data: [397, 'San Diego', 'United States', '1388996', '1388320', '0.05']
Row 398 data: [398, 'Shiyan', 'China', '1387377', '1354025', '2.46']
Row 399 data: [399, 'Taizhong', 'Taiwan', '1381855', '1369066', '0.93']
Row 400 data: [400, 'Bareilly', 'India', '1380715', '1348664', '2.38']
Row 401 data: [401, 'Pointe Noire', 'Republic of the Congo', '1379368',
'1336387', '3.22']
Row 402 data: [402, 'Adelaide', 'Australia', '1379280', '1366783', '0.91']
Row 403 data: [403, 'Suweon', 'South Korea', '1378229', '1365352', '0.94']
Row 404 data: [404, 'Mwanza', 'Tanzania', '1378014', '1310754', '5.13']
Row 405 data: [405, 'Mianyang Sichuan', 'China', '1376449', '1351383', '1.85']
Row 406 data: [406, 'Samut Prakan', 'Thailand', '1376146', '1358871', '1.27']
Row 407 data: [407, 'Maceio', 'Brazil', '1375984', '1363510', '0.91']
Row 408 data: [408, 'Qom', 'Iran', '1373800', '1354174', '1.45']

Row 409 data: [409, 'Antalya', 'Turkey', '1372400', '1347240', '1.87']
Row 410 data: [410, 'Joinville', 'Brazil', '1361992', '1348521', '1']
Row 411 data: [411, 'Tengzhou', 'China', '1360779', '1322916', '2.86']
Row 412 data: [412, 'Yingkou', 'China', '1355564', '1323174', '2.45']
Row 413 data: [413, 'Ad Dammam', 'Saudi Arabia', '1352912', '1329291', '1.78']
Row 414 data: [414, 'Suzhou', 'China', '1349994', '1310433', '3.02']
Row 415 data: [415, 'Tanger', 'Morocco', '1348848', '1314178', '2.64']
Row 416 data: [416, 'Freetown', 'Sierra Leone', '1347559', '1309168', '2.93']
Row 417 data: [417, 'Helsinki', 'Finland', '1346810', '1337786', '0.67']
Row 418 data: [418, 'Aligarh', 'India', '1346018', '1312369', '2.56']
Row 419 data: [419, 'Moradabad', 'India', '1335966', '1301740', '2.63']
Row 420 data: [420, 'Pekan Baru', 'Indonesia', '1334532', '1303355', '2.39']
Row 421 data: [421, 'Maoming', 'China', '1333930', '1303900', '2.3']
Row 422 data: [422, 'Lilongwe', 'Malawi', '1333096', '1276316', '4.45']
Row 423 data: [423, 'Porto', 'Portugal', '1329301', '1324652', '0.35']
Row 424 data: [424, 'Prague', 'Czech Republic', '1327947', '1323339', '0.35']
Row 425 data: [425, 'Astana', 'Kazakhstan', '1324111', '1291280', '2.54']
Row 426 data: [426, 'Jieyang', 'China', '1320779', '1282479', '2.99']
Row 427 data: [427, 'Fushun Liaoning', 'China', '1317276', '1304565', '0.97']
Row 428 data: [428, 'Mysore', 'India', '1316461', '1288245', '2.19']
Row 429 data: [429, 'Abomey Calavi', 'Benin', '1314916', '1252890', '4.95']
Row 430 data: [430, 'Ruian', 'China', '1314784', '1281790', '2.57']
Row 431 data: [431, 'Fes', 'Morocco', '1313311', '1290039', '1.8']
Row 432 data: [432, 'Port Elizabeth', 'South Africa', '1312631', '1295928',
'1.29']
Row 433 data: [433, 'Florianopolis', 'Brazil', '1309895', '1294486', '1.19']
Row 434 data: [434, 'Ahvaz', 'Iran', '1309372', '1293619', '1.22']
Row 435 data: [435, 'Bukavu', 'DR Congo', '1308469', '1248783', '4.78']
Row 436 data: [436, 'Dallas', 'United States', '1302753', '1302868', '-0.01']
Row 437 data: [437, 'Nnewi', 'Nigeria', '1300993', '1239186', '4.99']
Row 438 data: [438, 'Kazan', 'Russia', '1296232', '1291884', '0.34']
Row 439 data: [439, 'Jinhua', 'China', '1296065', '1257941', '3.03']
Row 440 data: [440, 'San Luis Potosi', 'Mexico', '1292133', '1273325', '1.48']
Row 441 data: [441, 'Baoji', 'China', '1290231', '1261077', '2.31']
Row 442 data: [442, 'Durg Bhilainagar', 'India', '1289673', '1266034', '1.87']
Row 443 data: [443, 'Bhubaneswar', 'India', '1289254', '1257642', '2.51']
Row 444 data: [444, 'Kigali', 'Rwanda', '1287952', '1247551', '3.24']
Row 445 data: [445, 'Sofia', 'Bulgaria', '1287540', '1288114', '-0.04']
Row 446 data: [446, 'Pingdingshan Henan', 'China', '1285750', '1256581', '2.32']
Row 447 data: [447, 'Dublin', 'Ireland', '1284551', '1270172', '1.13']
Row 448 data: [448, 'Puning', 'China', '1282756', '1254160', '2.28']
Row 449 data: [449, 'Chifeng', 'China', '1279544', '1252414', '2.17']
Row 450 data: [450, 'Zhuzhou', 'China', '1277238', '1254519', '1.81']
Row 451 data: [451, 'Bujumbura', 'Burundi', '1277050', '1206767', '5.82']
Row 452 data: [452, 'Zhenjiang Jiangsu', 'China', '1275395', '1250668', '1.98']
Row 453 data: [453, 'Liupanshui', 'China', '1272872', '1230644', '3.43']
Row 454 data: [454, 'Barquisimeto', 'Venezuela', '1267872', '1254192', '1.09']
Row 455 data: [455, 'Islamabad', 'Pakistan', '1266792', '1232447', '2.79']

```
Row 456 data: [456, 'Huaibei', 'China', '1264856', '1236271', '2.31']
Row 457 data: [457, 'Tasikmalaya', 'Indonesia', '1258124', '1213778', '3.65']
Row 458 data: [458, 'Maracay', 'Venezuela', '1256553', '1242945', '1.09']
Row 459 data: [459, 'Bogor', 'Indonesia', '1256155', '1231445', '2.01']
Row 460 data: [460, 'Da Nang', 'Vietnam', '1253228', '1220634', '2.67']
Row 461 data: [461, 'Nizhniy Novgorod', 'Russia', '1250302', '1251332', '-0.08']
Row 462 data: [462, 'Nanyang Henan', 'China', '1250300', '1224629', '2.1']
Row 463 data: [463, 'Xiangtan Hunan', 'China', '1249380', '1223912', '2.08']
Row 464 data: [464, 'Pizhou', 'China', '1247177', '1207275', '3.31']
Row 465 data: [465, 'Tiruchirappalli', 'India', '1244978', '1221960', '1.88']
Row 466 data: [466, 'Chelyabinsk', 'Russia', '1243883', '1240926', '0.24']
Row 467 data: [467, 'Mendoza', 'Argentina', '1242319', '1226427', '1.3']
Row 468 data: [468, 'Luohe', 'China', '1241152', '1197935', '3.61']
Row 469 data: [469, 'Xiongan', 'China', '1240158', '1183042', '4.83']
Row 470 data: [470, 'Chandigarh', 'India', '1239699', '1214775', '2.05']
Row 471 data: [471, 'Merida', 'Mexico', '1239654', '1220603', '1.56']
Row 472 data: [472, 'Jinzhou', 'China', '1238377', '1215376', '1.89']
Row 473 data: [473, 'Benxi', 'China', '1237550', '1216852', '1.7']
Row 474 data: [474, 'Binzhou', 'China', '1233584', '1200222', '2.78']
Row 475 data: [475, 'Aba', 'Nigeria', '1230407', '1188803', '3.5']
Row 476 data: [476, 'Chiang Mai', 'Thailand', '1228773', '1213348', '1.27']
Row 477 data: [477, 'Bazhong', 'China', '1225591', '1175989', '4.22']
Row 478 data: [478, 'Quetta', 'Pakistan', '1221495', '1190348', '2.62']
Row 479 data: [479, 'Kaduna', 'Nigeria', '1221451', '1187398', '2.87']
Row 480 data: [480, 'Guilin', 'China', '1218067', '1196841', '1.77']
Row 481 data: [481, 'Saharanpur', 'India', '1207856', '1171689', '3.09']
Row 482 data: [482, 'Hubli Dharwad', 'India', '1205428', '1181194', '2.05']
Row 483 data: [483, 'Yueqing', 'China', '1201400', '1169864', '2.7']
Row 484 data: [484, 'Guwahati', 'India', '1199455', '1176330', '1.97']
Row 485 data: [485, 'Mexicali', 'Mexico', '1196982', '1178477', '1.57']
Row 486 data: [486, 'Salem', 'India', '1194757', '1169953', '2.12']
Row 487 data: [487, 'Maputo', 'Mozambique', '1193253', '1162793', '2.62']
Row 488 data: [488, 'Tripoli', 'Libya', '1192436', '1183292', '0.77']
Row 489 data: [489, 'Haifa', 'Israel', '1186475', '1174429', '1.03']
Row 490 data: [490, 'Bandar Lampung', 'Indonesia', '1186233', '1162242', '2.06']
Row 491 data: [491, 'Bobo Dioulasso', 'Burkina Faso', '1185053', '1128646', '5']
Row 492 data: [492, 'Amsterdam', 'Netherlands', '1181817', '1174025', '0.66']
Row 493 data: [493, 'Shimkent', 'Kazakhstan', '1181020', '1155073', '2.25']
Row 494 data: [494, 'Omsk', 'Russia', '1180677', '1180745', '-0.01']
Row 495 data: [495, 'Aguascalientes', 'Mexico', '1179301', '1161448', '1.54']
Row 496 data: [496, 'Hargeysa', 'Somalia', '1176617', '1127198', '4.38']
Row 497 data: [497, 'Krasnoyarsk', 'Russia', '1173095', '1166792', '0.54']
Row 498 data: [498, 'Xinxiang', 'China', '1160840', '1140599', '1.77']
Row 499 data: [499, 'Siliguri', 'India', '1159371', '1126249', '2.94']
Row 500 data: [500, 'Wenling', 'China', '1159259', '1131361', '2.47']
Row 501 data: [501, 'Samara', 'Russia', '1154451', '1155732', '-0.11']
Row 502 data: [502, 'Zaozhuang', 'China', '1150859', '1133629', '1.52']
Row 503 data: [503, 'Cologne', 'Germany', '1149014', '1143715', '0.46']
```

```
Row 504 data: [504, 'Yongin', 'South Korea', '1147967', '1135671', '1.08']
Row 505 data: [505, 'Ufa', 'Russia', '1146786', '1144799', '0.17']
Row 506 data: [506, 'Fuyang', 'China', '1146343', '1120694', '2.29']
Row 507 data: [507, 'Ikorodu', 'Nigeria', '1145224', '1093308', '4.75']
Row 508 data: [508, 'Bien Hoa', 'Vietnam', '1142997', '1110824', '2.9']
Row 509 data: [509, 'Jalandhar', 'India', '1142682', '1118826', '2.13']
Row 510 data: [510, 'Panjin', 'China', '1142542', '1120675', '1.95']
Row 511 data: [511, "Ma'anshan", 'China', '1140264', '1112680', '2.48']
Row 512 data: [512, 'Cuernavaca', 'Mexico', '1140169', '1123843', '1.45']
Row 513 data: [513, 'Rostov on Don', 'Russia', '1139641', '1138854', '0.07']
Row 514 data: [514, 'Chihuahua', 'Mexico', '1135342', '1116295', '1.71']
Row 515 data: [515, 'Fuzhou Jiangxi', 'China', '1131718', '1100271', '2.86']
Row 516 data: [516, 'Tshikapa', 'DR Congo', '1131226', '1077108', '5.02']
Row 517 data: [517, 'Shangrao', 'China', '1130887', '1106736', '2.18']
Row 518 data: [518, 'Samarinda', 'Indonesia', '1128176', '1100324', '2.53']
Row 519 data: [519, 'Bishkek', 'Kyrgyzstan', '1127721', '1104742', '2.08']
Row 520 data: [520, 'Zhaoqing', 'China', '1127601', '1103202', '2.21']
Row 521 data: [521, 'San Salvador', 'El Salvador', '1123376', '1116052', '0.66']
Row 522 data: [522, 'Yichun Jiangxi', 'China', '1122041', '1078630', '4.02']
Row 523 data: [523, 'Chenzhou', 'China', '1119791', '1084969', '3.21']
Row 524 data: [524, 'Sekondi Takoradi', 'Ghana', '1119534', '1077929', '3.86']
Row 525 data: [525, 'Leshan', 'China', '1116497', '1087409', '2.67']
Row 526 data: [526, 'Aden', 'Yemen', '1116193', '1079670', '3.38']
Row 527 data: [527, 'Goyang', 'South Korea', '1114079', '1105283', '0.8']
Row 528 data: [528, 'Diyarbakir', 'Turkey', '1113333', '1096937', '1.49']
Row 529 data: [529, 'Asmara', 'Eritrea', '1111748', '1072666', '3.64']
Row 530 data: [530, 'Dezhou', 'China', '1108356', '1080680', '2.56']
Row 531 data: [531, 'Jingzhou Hubei', 'China', '1108067', '1089886', '1.67']
Row 532 data: [532, 'Managua', 'Nicaragua', '1107118', '1094510', '1.15']
Row 533 data: [533, 'Johor Bahru', 'Malaysia', '1107001', '1086214', '1.91']
Row 534 data: [534, 'Kermanshah', 'Iran', '1101611', '1084623', '1.57']
Row 535 data: [535, 'Nyala', 'Sudan', '1101314', '1056952', '4.2']
Row 536 data: [536, 'Oslo', 'Norway', '1100868', '1085992', '1.37']
Row 537 data: [537, 'Kirkuk', 'Iraq', '1100390', '1074884', '2.37']
Row 538 data: [538, 'Yerevan', 'Armenia', '1097542', '1094813', '0.25']
Row 539 data: [539, 'Cartagena', 'Colombia', '1096463', '1087599', '0.82']
Row 540 data: [540, 'Changshu', 'China', '1093687', '1059902', '3.19']
Row 541 data: [541, 'Huzhou', 'China', '1091811', '1067651', '2.26']
Row 542 data: [542, 'Xuchang', 'China', '1091744', '1063876', '2.62']
Row 543 data: [543, 'Solapur', 'India', '1088131', '1070322', '1.66']
Row 544 data: [544, 'Lille', 'France', '1085199', '1079120', '0.56']
Row 545 data: [545, 'Mersin', 'Turkey', '1084817', '1069402', '1.44']
Row 546 data: [546, 'Tbilisi', 'Georgia', '1084471', '1082245', '0.21']
Row 547 data: [547, 'Perm', 'Russia', '1084120', '1081768', '0.22']
Row 548 data: [548, 'Voronezh', 'Russia', '1083724', '1080808', '0.27']
Row 549 data: [549, 'Denpasar', 'Indonesia', '1075244', '1053041', '2.11']
Row 550 data: [550, 'Toulouse', 'France', '1070746', '1060460', '0.97']
Row 551 data: [551, 'Blantyre Limbe', 'Malawi', '1070625', '1030974', '3.85']
```

```
Row 552 data: [552, 'Aracaju', 'Brazil', '1070122', '1056986', '1.24']
Row 553 data: [553, 'Marrakech', 'Morocco', '1067172', '1049690', '1.67']
Row 554 data: [554, 'Qujing', 'China', '1066897', '1038964', '2.69']
Row 555 data: [555, 'Yueyang', 'China', '1064166', '1048889', '1.46']
Row 556 data: [556, 'Ilorin', 'Nigeria', '1063713', '1030498', '3.22']
Row 557 data: [557, 'Tampico', 'Mexico', '1062567', '1047251', '1.46']
Row 558 data: [558, 'Antwerp', 'Belgium', '1061089', '1057215', '0.37']
Row 559 data: [559, 'Teresina', 'Brazil', '1059657', '1050459', '0.88']
Row 560 data: [560, 'Guiping', 'China', '1055921', '1020399', '3.48']
Row 561 data: [561, 'Warangal', 'India', '1055604', '1031090', '2.38']
Row 562 data: [562, 'Changwon', 'South Korea', '1055373', '1054442', '0.09']
Row 563 data: [563, 'Padang', 'Indonesia', '1052849', '1033645', '1.86']
Row 564 data: [564, 'Saltillo', 'Mexico', '1050320', '1034126', '1.57']
Row 565 data: [565, 'Xintai', 'China', '1049486', '1023756', '2.51']
Row 566 data: [566, 'Cancun', 'Mexico', '1045005', '1022604', '2.19']
Row 567 data: [567, 'Cebu City', 'Philippines', '1042613', '1024945', '1.72']
Row 568 data: [568, 'San Miguel De Tucuman', 'Argentina', '1039226', '1026767',
'1.21']
Row 569 data: [569, 'Hamah', 'Syria', '1034498', '995747', '3.89']
Row 570 data: [570, 'Acapulco De Juarez', 'Mexico', '1032772', '1018976',
'1.35']
Row 571 data: [571, 'Warri', 'Nigeria', '1031425', '986921', '4.51']
Row 572 data: [572, 'Kayseri', 'Turkey', '1025507', '1013240', '1.21']
Row 573 data: [573, 'Chengde', 'China', '1024521', '993097', '3.16']
Row 574 data: [574, 'Owerri', 'Nigeria', '1022922', '983352', '4.02']
Row 575 data: [575, 'Rotterdam', 'Netherlands', '1021919', '1018012', '0.38']
Row 576 data: [576, 'Pingxiang Jiangxi', 'China', '1021575', '999721', '2.19']
Row 577 data: [577, 'Zhucheng', 'China', '1019600', '991208', '2.86']
Row 578 data: [578, 'Songkhla', 'Thailand', '1017784', '1005007', '1.27']
Row 579 data: [579, 'Valparaiso', 'Chile', '1016585', '1008599', '0.79']
Row 580 data: [580, 'Dehradun', 'India', '1016402', '991876', '2.47']
Row 581 data: [581, 'Nonthaburi', 'Thailand', '1013672', '1000947', '1.27']
Row 582 data: [582, 'Leiyang', 'China', '1013191', '978327', '3.56']
Row 583 data: [583, 'Dushanbe', 'Tajikistan', '1012794', '986899', '2.62']
Row 584 data: [584, 'Nampula', 'Mozambique', '1012582', '969322', '4.46']
Row 585 data: [585, 'Misratah', 'Libya', '1011119', '984193', '2.74']
Row 586 data: [586, 'Krasnodar', 'Russia', '1010552', '1001155', '0.94']
Row 587 data: [587, 'Laiwu', 'China', '1009727', '986248', '2.38']
Row 588 data: [588, 'Bordeaux', 'France', '1009594', '1000475', '0.91']
Row 589 data: [589, 'Jixi Heilongjiang', 'China', '1008989', '993175', '1.59']
Row 590 data: [590, 'San Pedro Sula', 'Honduras', '1008220', '982021', '2.67']
Row 591 data: [591, 'Odesa', 'Ukraine', '1007596', '1007716', '-0.01']
Row 592 data: [592, 'Jiujiang', 'China', '1004247', '982777', '2.18']
Row 593 data: [593, 'Lubango', 'Angola', '1003016', '958548', '4.64']
Row 594 data: [594, 'Morelia', 'Mexico', '1002461', '988744', '1.39']
Row 595 data: [595, 'Jos', 'Nigeria', '1001155', '970129', '3.2']
Row 596 data: [596, 'Sylhet', 'Bangladesh', '999374', '964291', '3.64']
Row 597 data: [597, 'Agadir', 'Morocco', '998146', '979248', '1.93']
```

Row 598 data: [598, 'Jacksonville', 'United States', '997164', '985843', '1.15']
Row 599 data: [599, 'Fort Worth', 'United States', '996756', '978468', '1.87']
Row 600 data: [600, 'Volgograd', 'Russia', '993352', '995078', '-0.17']
Row 601 data: [601, 'Mudanjiang', 'China', '992414', '975348', '1.75']
Row 602 data: [602, 'Guigang', 'China', '990310', '967428', '2.37']
Row 603 data: [603, 'Najaf', 'Iraq', '987814', '958487', '3.06']
Row 604 data: [604, 'Bangui', 'Central African Republic', '985965', '958335', '2.88']
Row 605 data: [605, 'Austin', 'United States', '984567', '979882', '0.48']
Row 606 data: [606, 'Rajshahi', 'Bangladesh', '983707', '961991', '2.26']
Row 607 data: [607, 'Hengshui', 'China', '983694', '955663', '2.93']
Row 608 data: [608, 'Jerusalem', 'Israel', '983097', '969804', '1.37']
Row 609 data: [609, 'Zhangzhou', 'China', '980054', '955519', '2.57']
Row 610 data: [610, 'Xinyu', 'China', '979303', '952278', '2.84']
Row 611 data: [611, 'Linfen', 'China', '978614', '951822', '2.81']
Row 612 data: [612, 'Tianmen', 'China', '978317', '953768', '2.57']
Row 613 data: [613, 'Ciudad Guayana', 'Venezuela', '978202', '964266', '1.45']
Row 614 data: [614, 'Zamboanga City', 'Philippines', '977081', '960349', '1.74']
Row 615 data: [615, 'Yangjiang', 'China', '976606', '956079', '2.15']
Row 616 data: [616, 'Taiz', 'Yemen', '974518', '940600', '3.61']
Row 617 data: [617, 'Cucuta', 'Colombia', '972485', '963045', '0.98']
Row 618 data: [618, 'Arequipa', 'Peru', '971296', '958998', '1.28']
Row 619 data: [619, 'Liling', 'China', '970907', '936997', '3.62']
Row 620 data: [620, 'Antipolo', 'Philippines', '968288', '946653', '2.29']
Row 621 data: [621, 'Veracruz', 'Mexico', '968070', '955226', '1.34']
Row 622 data: [622, 'Reynosa', 'Mexico', '967627', '951417', '1.7']
Row 623 data: [623, 'Khulna', 'Bangladesh', '965483', '955104', '1.09']
Row 624 data: [624, 'Deyang', 'China', '963160', '939166', '2.55']
Row 625 data: [625, 'Pathum Thani', 'Thailand', '962126', '950048', '1.27']
Row 626 data: [626, 'Bengbu', 'China', '959784', '944465', '1.62']
Row 627 data: [627, 'Jiangyin', 'China', '959250', '933395', '2.77']
Row 628 data: [628, 'Southampton', 'United Kingdom', '959202', '951531', '0.81']
Row 629 data: [629, 'Villahermosa', 'Mexico', '958118', '943530', '1.55']
Row 630 data: [630, 'Baishan', 'China', '956976', '927532', '3.17']
Row 631 data: [631, 'San Jose', 'United States', '956433', '969655', '-1.36']
Row 632 data: [632, 'Nice', 'France', '951808', '948149', '0.39']
Row 633 data: [633, 'Oran', 'Algeria', '950768', '935947', '1.58']
Row 634 data: [634, 'West Rand', 'South Africa', '947734', '934341', '1.43']
Row 635 data: [635, 'Cabinda', 'Angola', '947634', '904676', '4.75']
Row 636 data: [636, 'Umuahia', 'Nigeria', '947460', '904139', '4.79']
Row 637 data: [637, 'Bogra', 'Bangladesh', '944877', '905814', '4.31']
Row 638 data: [638, 'Bahawalpur', 'Pakistan', '944812', '919654', '2.74']
Row 639 data: [639, 'Seongnam', 'South Korea', '942159', '941508', '0.07']
Row 640 data: [640, 'Guntur', 'India', '940205', '918324', '2.38']
Row 641 data: [641, 'Dnipro', 'Ukraine', '936766', '941586', '-0.51']
Row 642 data: [642, 'Campo Grande', 'Brazil', '934936', '926095', '0.95']
Row 643 data: [643, 'Malang', 'Indonesia', '933392', '919443', '1.52']
Row 644 data: [644, 'Londrina', 'Brazil', '932892', '922425', '1.13']

Row 645 data: [645, 'Dandong', 'China', '932648', '917936', '1.6']
Row 646 data: [646, 'Changzhi', 'China', '930620', '910743', '2.18']
Row 647 data: [647, 'Hermosillo', 'Mexico', '929961', '915060', '1.63']
Row 648 data: [648, 'Bhiwandi', 'India', '924670', '906581', '2']
Row 649 data: [649, 'La Plata', 'Argentina', '923715', '914036', '1.06']
Row 650 data: [650, 'Charlotte', 'United States', '923164', '911311', '1.3']
Row 651 data: [651, 'Liverpool', 'United Kingdom', '922871', '917032', '0.64']
Row 652 data: [652, 'Ashgabat', 'Turkmenistan', '921601', '902353', '2.13']
Row 653 data: [653, 'Concepcion', 'Chile', '920916', '911862', '0.99']
Row 654 data: [654, 'Puducherry', 'India', '919280', '897983', '2.37']
Row 655 data: [655, 'Changde', 'China', '919026', '898338', '2.3']
Row 656 data: [656, 'Bergamo', 'Italy', '918675', '913376', '0.58']
Row 657 data: [657, 'Firozabad', 'India', '918218', '894388', '2.66']
Row 658 data: [658, 'Erbil', 'Iraq', '917639', '896716', '2.33']
Row 659 data: [659, 'Tyumen', 'Russia', '916862', '903815', '1.44']
Row 660 data: [660, 'Trujillo', 'Peru', '916632', '903896', '1.41']
Row 661 data: [661, 'Liaoyang', 'China', '916238', '900692', '1.73']
Row 662 data: [662, 'Shangqiu', 'China', '915823', '895111', '2.31']
Row 663 data: [663, 'Columbus', 'United States', '915427', '913175', '0.25']
Row 664 data: [664, 'Ulsan', 'South Korea', '914300', '912734', '0.17']
Row 665 data: [665, 'Tuxtla Gutierrez', 'Mexico', '913075', '897694', '1.71']
Row 666 data: [666, 'Kuerle', 'China', '906765', '875496', '3.57']
Row 667 data: [667, 'Soshanguve', 'South Africa', '905868', '892254', '1.53']
Row 668 data: [668, 'Xingtai', 'China', '904070', '886381', '2']
Row 669 data: [669, 'Culiacan', 'Mexico', '903910', '888620', '1.72']
Row 670 data: [670, 'Quzhou', 'China', '902621', '871457', '3.58']
Row 671 data: [671, 'Cherthala', 'India', '901820', '870465', '3.6']
Row 672 data: [672, 'Huangshi', 'China', '893107', '876930', '1.84']
Row 673 data: [673, 'Fuxin', 'China', '889767', '876131', '1.56']
Row 674 data: [674, 'Lokoja', 'Nigeria', '885882', '839046', '5.58']
Row 675 data: [675, 'Hufuf Mubarraz', 'Saudi Arabia', '884753', '872438', '1.41']
Row 676 data: [676, 'Libreville', 'Gabon', '883920', '869773', '1.63']
Row 677 data: [677, 'Yongzhou', 'China', '883326', '860522', '2.65']
Row 678 data: [678, 'Xinghua', 'China', '882986', '861978', '2.44']
Row 679 data: [679, 'Donetsk', 'Ukraine', '882209', '887716', '-0.62']
Row 680 data: [680, 'Yibin', 'China', '881480', '864910', '1.92']
Row 681 data: [681, 'Indianapolis (balance)', 'United States', '876665', '879293', '-0.3']
Row 682 data: [682, 'Enugu', 'Nigeria', '875552', '846560', '3.42']
Row 683 data: [683, 'Tainan', 'Taiwan', '875392', '869625', '0.66']
Row 684 data: [684, 'Xinyang', 'China', '873268', '855191', '2.11']
Row 685 data: [685, 'Ipoh', 'Malaysia', '872424', '857225', '1.77']
Row 686 data: [686, 'Luzhou', 'China', '870714', '857683', '1.52']
Row 687 data: [687, 'Banghazi', 'Libya', '870502', '859209', '1.31']
Row 688 data: [688, 'Maiduguri', 'Nigeria', '870201', '844747', '3.01']
Row 689 data: [689, 'Yangquan', 'China', '869585', '851630', '2.11']
Row 690 data: [690, 'Huaihua', 'China', '869575', '843754', '3.06']

Row 691 data: [691, 'Xiaogan', 'China', '869479', '849580', '2.34']
Row 692 data: [692, 'Tianshui', 'China', '858672', '837479', '2.53']
Row 693 data: [693, 'Bunia', 'DR Congo', '856339', '812090', '5.45']
Row 694 data: [694, 'Bozhou', 'China', '854946', '830125', '2.99']
Row 695 data: [695, 'Kottayam', 'India', '853635', '818628', '4.28']
Row 696 data: [696, 'Zhuji', 'China', '852608', '834782', '2.14']
Row 697 data: [697, 'Kunshan', 'China', '851399', '826414', '3.02']
Row 698 data: [698, 'Quebec City', 'Canada', '851061', '844249', '0.81']
Row 699 data: [699, 'Palermo', 'Italy', '850233', '849687', '0.06']
Row 700 data: [700, 'Winnipeg', 'Canada', '849251', '841108', '0.97']
Row 701 data: [701, 'Orumiyeh', 'Iran', '848443', '835900', '1.5']
Row 702 data: [702, 'Eskisehir', 'Turkey', '848002', '834065', '1.67']
Row 703 data: [703, 'Benguela', 'Angola', '843207', '809468', '4.17']
Row 704 data: [704, 'Jincheng', 'China', '841928', '818057', '2.92']
Row 705 data: [705, 'Valencia', 'Spain', '839770', '838301', '0.18']
Row 706 data: [706, 'Heze', 'China', '838928', '826777', '1.47']
Row 707 data: [707, 'Saratov', 'Russia', '837687', '838377', '-0.08']
Row 708 data: [708, 'Nellore', 'India', '837660', '816293', '2.62']
Row 709 data: [709, 'Huludao', 'China', '836344', '821132', '1.85']
Row 710 data: [710, 'Zanzibar', 'Tanzania', '835850', '800010', '4.48']
Row 711 data: [711, 'Barcelona Puerto La Cruz', 'Venezuela', '835805', '825581', '1.24']
Row 712 data: [712, 'Bikaner', 'India', '835802', '818566', '2.11']
Row 713 data: [713, 'Haicheng', 'China', '834639', '821192', '1.64']
Row 714 data: [714, 'Gebze', 'Turkey', '831360', '813394', '2.21']
Row 715 data: [715, 'Taixing', 'China', '830623', '811451', '2.36']
Row 716 data: [716, 'Liaocheng', 'China', '830004', '813369', '2.05']
Row 717 data: [717, 'Zhumadian', 'China', '829361', '804537', '3.09']
Row 718 data: [718, 'Newcastle Upon Tyne', 'United Kingdom', '828712', '823431', '0.64']
Row 719 data: [719, 'Langfang', 'China', '827967', '807833', '2.49']
Row 720 data: [720, 'Bucheon', 'South Korea', '826919', '826981', '-0.01']
Row 721 data: [721, 'Sulaimaniya', 'Iraq', '823199', '800793', '2.8']
Row 722 data: [722, 'Xalapa', 'Mexico', '822863', '811041', '1.46']
Row 723 data: [723, 'Malanje', 'Angola', '822471', '783243', '5.01']
Row 724 data: [724, 'Anqiu', 'China', '821765', '791931', '3.77']
Row 725 data: [725, 'Sorocaba', 'Brazil', '821435', '813320', '1']
Row 726 data: [726, 'Gaomi', 'China', '821154', '797964', '2.91']
Row 727 data: [727, 'Dasmarinas', 'Philippines', '820886', '802600', '2.28']
Row 728 data: [728, 'Cagayan De Oro City', 'Philippines', '820297', '803194', '2.13']
Row 729 data: [729, 'Hanchuan', 'China', '818761', '795835', '2.88']
Row 730 data: [730, 'Meishan', 'China', '818037', '796756', '2.67']
Row 731 data: [731, 'Bologna', 'Italy', '816848', '814332', '0.31']
Row 732 data: [732, 'Ar Rayyan', 'Qatar', '815869', '798382', '2.19']
Row 733 data: [733, 'Thessaloniki', 'Greece', '814980', '814524', '0.06']
Row 734 data: [734, 'Muzaffarnagar', 'India', '814491', '791214', '2.94']
Row 735 data: [735, 'Kayamkulam', 'India', '813379', '786192', '3.46']

Row 736 data: [736, 'Nottingham', 'United Kingdom', '813078', '806757', '0.78']
Row 737 data: [737, 'Nakhon Ratchasima', 'Thailand', '811446', '801853', '1.2']
Row 738 data: [738, 'Danyang', 'China', '810450', '789755', '2.62']
Row 739 data: [739, 'Ibb', 'Yemen', '810149', '771514', '5.01']
Row 740 data: [740, 'Amravati', 'India', '808263', '792620', '1.97']
Row 741 data: [741, 'Jiaozuo', 'China', '808079', '796509', '1.45']
Row 742 data: [742, 'Vereeniging', 'South Africa', '802900', '793927', '1.13']
Row 743 data: [743, 'Gorakhpur', 'India', '801634', '788276', '1.69']
Row 744 data: [744, 'Gaza', 'Palestine', '800636', '778187', '2.88']
Row 745 data: [745, 'Frankfurt', 'Germany', '800529', '796437', '0.51']
Row 746 data: [746, 'Anqing', 'China', '795987', '779535', '2.11']
Row 747 data: [747, 'Niigata', 'Japan', '795916', '797865', '-0.24']
Row 748 data: [748, 'Oshogbo', 'Nigeria', '795808', '771515', '3.15']
Row 749 data: [749, 'Linhai', 'China', '795777', '776076', '2.54']
Row 750 data: [750, 'Shaoguan', 'China', '794279', '784230', '1.28']
Row 751 data: [751, 'Erduosi Ordoss', 'China', '794143', '774321', '2.56']
Row 752 data: [752, 'Merca', 'Somalia', '793545', '760129', '4.4']
Row 753 data: [753, "Bur Sa'id", 'Egypt', '792925', '778280', '1.88']
Row 754 data: [754, 'Kitwe', 'Zambia', '792350', '762981', '3.85']
Row 755 data: [755, "Yan'an", 'China', '791839', '767188', '3.21']
Row 756 data: [756, 'Cuttack', 'India', '789558', '775559', '1.81']
Row 757 data: [757, 'San Francisco', 'United States', '788478', '808988',
'-2.53']
Row 758 data: [758, 'Hamilton', 'Canada', '786843', '781047', '0.74']
Row 759 data: [759, 'Zaria', 'Nigeria', '786197', '766007', '2.64']
Row 760 data: [760, 'Banjarmasin', 'Indonesia', '785125', '770959', '1.84']
Row 761 data: [761, 'Dengzhou', 'China', '783041', '759150', '3.15']
Row 762 data: [762, 'Belgaum', 'India', '783020', '767161', '2.07']
Row 763 data: [763, 'Malegaon', 'India', '781925', '764628', '2.26']
Row 764 data: [764, 'Goma', 'DR Congo', '781875', '744247', '5.06']
Row 765 data: [765, 'Zigong', 'China', '779684', '768075', '1.51']
Row 766 data: [766, 'Qingyuan', 'China', '778628', '766782', '1.54']
Row 767 data: [767, 'Yuncheng', 'China', '777193', '754711', '2.98']
Row 768 data: [768, 'Shaoyang', 'China', '776404', '761169', '2']
Row 769 data: [769, 'Yanji', 'China', '775672', '757224', '2.44']
Row 770 data: [770, 'Tirupati', 'India', '775455', '752744', '3.02']
Row 771 data: [771, 'Maturin', 'Venezuela', '775097', '758185', '2.23']
Row 772 data: [772, 'Yuxi', 'China', '774356', '750102', '3.23']
Row 773 data: [773, 'Akure', 'Nigeria', '773141', '744371', '3.87']
Row 774 data: [774, 'Tongliao', 'China', '772756', '756158', '2.2']
Row 775 data: [775, 'Sialkot', 'Pakistan', '770962', '753325', '2.34']
Row 776 data: [776, 'Tongling', 'China', '770032', '752916', '2.27']
Row 777 data: [777, 'Krakow', 'Poland', '769396', '769417', '0']
Row 778 data: [778, 'Ansan', 'South Korea', '768822', '766703', '0.28']
Row 779 data: [779, 'Wuzhou', 'China', '768512', '751679', '2.24']
Row 780 data: [780, 'Dazhou', 'China', '767532', '748257', '2.58']
Row 781 data: [781, 'Suining Sichuan', 'China', '765887', '750097', '2.11']
Row 782 data: [782, 'Mangalore', 'India', '763312', '749073', '1.9']

```
Row 783 data: [783, 'Jiamusi', 'China', '761903', '749763', '1.62']
Row 784 data: [784, 'Seattle', 'United States', '759915', '755078', '0.64']
Row 785 data: [785, 'Al Hudaydah', 'Yemen', '759157', '734699', '3.33']
Row 786 data: [786, 'Sargodha', 'Pakistan', '757915', '741818', '2.17']
Row 787 data: [787, 'Nay Pyi Taw', 'Myanmar', '757823', '722836', '4.84']
Row 788 data: [788, 'Tamale', 'Ghana', '757506', '729768', '3.8']
Row 789 data: [789, 'Sao Jose Dos Campos', 'Brazil', '757137', '749188', '1.06']
Row 790 data: [790, 'Bacoor', 'Philippines', '757035', '739682', '2.35']
Row 791 data: [791, 'Dongtai', 'China', '756344', '738200', '2.46']
Row 792 data: [792, 'Zhangjiagang', 'China', '755752', '733810', '2.99']
Row 793 data: [793, 'Nanded Waghala', 'India', '755577', '738552', '2.31']
Row 794 data: [794, 'Xianyang Shaanxi', 'China', '754446', '743491', '1.47']
Row 795 data: [795, 'Amara', 'Iraq', '753708', '729276', '3.35']
Row 796 data: [796, 'Zarqa', 'Jordan', '753392', '748428', '0.66']
Row 797 data: [797, 'Bhavnagar', 'India', '751493', '737128', '1.95']
Row 798 data: [798, 'Sheffield', 'United Kingdom', '751303', '745876', '0.73']
Row 799 data: [799, 'Huambo', 'Angola', '751297', '727641', '3.25']
Row 800 data: [800, 'Ribeirao Preto', 'Brazil', '750174', '742115', '1.09']
Row 801 data: [801, 'Panzhihua', 'China', '750036', '738495', '1.56']
Initial data extracted:
   Rank       City     Country 2024 Population 2023 Population Growth Rate
0     0      Tokyo       Japan         37115035        37194105       -0.21
1     1      Delhi       India         33807403        32941309        2.63
2     2   Shanghai       China         29867918        29210808        2.25
3     3      Dhaka  Bangladesh         23935652        23209616        3.13
4     4  Sao Paulo      Brazil         22806704        22619736        0.83
```

Step 3: Replace Headers Description: Standardize the column headers to lowercase and replace spaces with underscores for consistency

[75]:
```python
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

# Verify the column names after replacement
print("Column names after replacement:", df.columns)
```

```
Column names after replacement: Index(['rank', 'city', 'country',
'2024_population', '2023_population',
       'growth_rate'],
      dtype='object')
```

Step 4: Handle Missing Values

Description: Identify and handle missing values in critical columns (e.g., population)

[76]:
```python
df['2024_population'] = df['2024_population'].replace('', pd.NA).astype(float)
df['2024_population'] = df['2024_population'].fillna(df['2024_population'].
 ↪mean())  # Fill missing population values with the mean
```

Step 5: Convert Data Types

Description: Ensure that columns are in appropriate data types, such as converting population to integer

[77]:
```
df['2024_population'] = df['2024_population'].astype(int)
```

Step 5b: Clean and Convert 2023 Population

Description: Remove any non-numeric characters such as commas

[78]:
```
df['2023_population'] = df['2023_population'].replace('', pd.NA).str.
↪replace(',', '').str.replace('%', '')

# Convert to appropriate data types
df['2023_population'] = df['2023_population'].astype(float).astype(int)

# Handle Growth Rate
df['growth_rate'] = df['growth_rate'].replace('', pd.NA)
df['growth_rate'] = df['growth_rate'].astype(float)

# Print data after handling missing values and type conversion
print("Data after handling missing values and type conversion:")
print(df.head())
```

```
Data after handling missing values and type conversion:
   rank       city     country  2024_population  2023_population  growth_rate
0     0      Tokyo       Japan         37115035         37194105        -0.21
1     1      Delhi       India         33807403         32941309         2.63
2     2   Shanghai       China         29867918         29210808         2.25
3     3      Dhaka  Bangladesh         23935652         23209616         3.13
4     4  Sao Paulo      Brazil         22806704         22619736         0.83
```

Step 6: Remove Duplicates

Description: Remove any duplicate records to ensure data quality

[80]:
```
df.drop_duplicates(inplace=True)
print(df.head())
```

```
   rank       city     country  2024_population  2023_population  growth_rate
0     0      Tokyo       Japan         37115035         37194105        -0.21
1     1      Delhi       India         33807403         32941309         2.63
2     2   Shanghai       China         29867918         29210808         2.25
3     3      Dhaka  Bangladesh         23935652         23209616         3.13
4     4  Sao Paulo      Brazil         22806704         22619736         0.83
```

Step 7: Fix Inconsistent Values

Description: Standardize the casing for city names

[82]:
```
df['city'] = df['city'].str.title()
df['country'] = df['country'].str.title()
```

[83]:
```
# Print the transformed dataset
print("Transformed dataset:")
print(df.head())
```

```
Transformed dataset:
   rank       city     country  2024_population  2023_population  growth_rate
0     0      Tokyo       Japan         37115035         37194105        -0.21
1     1      Delhi       India         33807403         32941309         2.63
2     2   Shanghai       China         29867918         29210808         2.25
3     3      Dhaka  Bangladesh         23935652         23209616         3.13
4     4  Sao Paulo      Brazil         22806704         22619736         0.83
```

Save the cleaned dataset

[84]:
```
cleaned_file_path = 'Cleaned_Population_Data.csv'
df.to_csv(cleaned_file_path, index=False)
```

0.4 Ethical Implications of Data Wrangling

In the process of cleaning and transforming the data sourced from the World Population Review website, several changes were made to ensure the data's integrity and usability. The primary changes included handling missing values, converting data types, removing duplicates, and fixing inconsistent values. Additionally, considerable effort was invested in correctly extracting the data due to the structure of the HTML table, including adjusting the logic to account for rows with varying numbers of columns and ensuring that the "Rank" column was correctly populated with sequential numbers. These steps were essential to maintain a high quality of the dataset. However, it is crucial to consider the ethical implications of these transformations.

Firstly, there are no specific legal or regulatory guidelines directly impacting the use of population data from this public source. Nonetheless, it's important to handle such data responsibly, ensuring that the modifications do not misrepresent or distort the information. The transformations, such as filling missing values and converting data types, could potentially introduce inaccuracies if not done carefully. For instance, filling missing population values with the mean might not accurately reflect the true population and could mislead analyses based on this data.

Assumptions were made during the cleaning process, particularly in filling missing values and converting string data to numerical formats. These assumptions might not hold true across all contexts, and they must be transparently documented. The credibility of the data was ensured by sourcing it from a reputable website that regularly updates and maintains accurate population statistics. The data acquisition was ethical, as it was publicly available and accessed through legitimate means.

To mitigate any ethical concerns, it was essential to document all data transformations and the rationale behind them clearly. Providing transparency ensures that users of the data are aware of the changes made and can assess the reliability of the data accordingly. Additionally, whenever possible, consulting with domain experts to validate assumptions and transformation choices can further enhance the ethical handling of the data. Ensuring ongoing scrutiny and validation against authoritative sources can help maintain the integrity and trustworthiness of the dataset.

0.5 Milestone 4

Connecting to an API/Pulling in the Data and Cleaning/Formatting

API Data Transformation and Cleaning Step 1: Fetch Weather Data from OpenWeatherMap API

[84]:
```python
import requests
import pandas as pd
import time

# OpenWeatherMap API key
api_key = '7156e22b3ae3095b3517c55a3562c5de'

# Function to fetch weather data based on Latitude and Longitude
def fetch_weather_data(lat, lon):
    url = f'http://api.openweathermap.org/data/2.5/weather?
 ↪lat={lat}&lon={lon}&appid={api_key}'
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    elif response.status_code == 429:
        print(f"Rate limit exceeded for {lat}, {lon}: {response.status_code}")
        return 'rate_limit_exceeded'
    else:
        print(f"Error fetching data for {lat}, {lon}: {response.status_code}")
        return None

# Function to perform reverse geocoding
def reverse_geocode(lat, lon):
    url = f'http://api.openweathermap.org/geo/1.0/reverse?
 ↪lat={lat}&lon={lon}&limit=1&appid={api_key}'
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    elif response.status_code == 429:
        print(f"Rate limit exceeded for {lat}, {lon}: {response.status_code}")
        return 'rate_limit_exceeded'
    else:
        print(f"Error fetching data for {lat}, {lon}: {response.status_code}")
        return None

# Load the meteorite landings data
meteorite_landings_path = 'Meteorite_Landings_20240614.csv'
meteorite_df = pd.read_csv(meteorite_landings_path)

# Display the first few rows of the meteorite landings data
print("Meteorite Landings Data:")
```

```
print(meteorite_df.head(10))

# Initialize an empty list to hold the weather data
weather_data_list = []

# Fetch data from OpenWeatherMap API for each meteorite landing location
fetch_attempts = 0
max_attempts = 10
retry_attempts = 5
retry_wait_time = 60  # Wait time in seconds for retry

for index, row in meteorite_df.iterrows():
    if fetch_attempts >= max_attempts:
        break
    lat = row['reclat']
    lon = row['reclong']
    attempt = 0
    while attempt < retry_attempts:
        print(f"Fetching weather data for location: {lat}, {lon}")
        weather_data = fetch_weather_data(lat, lon)
        if weather_data == 'rate_limit_exceeded':
            print(f"Waiting for {retry_wait_time} seconds before retrying...")
            time.sleep(retry_wait_time)
            attempt += 1
        elif weather_data:
            # Perform reverse geocoding to verify the location
            location_data = reverse_geocode(lat, lon)
            if location_data and location_data != 'rate_limit_exceeded':
                location_name = location_data[0]['name']
            else:
                location_name = 'Unknown'

            weather_info = {
                'Name': row['name'],
                'Latitude': lat,
                'Longitude': lon,
                'Temperature (K)': weather_data['main']['temp'],
                'Humidity (%)': weather_data['main']['humidity'],
                'Wind Speed (m/s)': weather_data['wind']['speed'],
                'Weather Description':␣
↪weather_data['weather'][0]['description'],
                'City': weather_data.get('name', 'N/A'),
                'Verified Location': location_name,
                'Year': row['year']
            }
            weather_data_list.append(weather_info)
            fetch_attempts += 1
```

```
            break
        else:
            print(f"Could not fetch data for location: {lat}, {lon}")
            break
    if attempt == retry_attempts:
        print(f"Max retry attempts reached for location: {lat}, {lon}")
    time.sleep(1)  # Add a 1-second delay between requests

# Convert the list to a DataFrame
weather_df = pd.DataFrame(weather_data_list)

# Display the first few rows of the weather data
print("Weather Data:")
print(weather_df.head(10))
```

```
Meteorite Landings Data:
                  name   id nametype     recclass  mass (g)  fall    year  \
0               Aachen    1    Valid           L5      21.0  Fell  1880.0
1               Aarhus    2    Valid           H6     720.0  Fell  1951.0
2                 Abee    6    Valid          EH4  107000.0  Fell  1952.0
3             Acapulco   10    Valid  Acapulcoite    1914.0  Fell  1976.0
4              Achiras  370    Valid           L6     780.0  Fell  1902.0
5             Adhi Kot  379    Valid          EH4    4239.0  Fell  1919.0
6  Adzhi-Bogdo (stone)  390    Valid        LL3-6     910.0  Fell  1949.0
7                 Agen  392    Valid           H5   30000.0  Fell  1814.0
8               Aguada  398    Valid           L6    1620.0  Fell  1930.0
9        Aguila Blanca  417    Valid            L    1440.0  Fell  1920.0

     reclat    reclong           GeoLocation
0  50.77500    6.08333     (50.775, 6.08333)
1  56.18333   10.23333  (56.18333, 10.23333)
2  54.21667 -113.00000    (54.21667, -113.0)
3  16.88333  -99.90000     (16.88333, -99.9)
4 -33.16667  -64.95000   (-33.16667, -64.95)
5  32.10000   71.80000          (32.1, 71.8)
6  44.83333   95.16667  (44.83333, 95.16667)
7  44.21667    0.61667   (44.21667, 0.61667)
8 -31.60000  -65.23333    (-31.6, -65.23333)
9 -30.86667  -64.55000   (-30.86667, -64.55)
Fetching weather data for location: 50.775, 6.08333
Fetching weather data for location: 56.18333, 10.23333
Fetching weather data for location: 54.21667, -113.0
Fetching weather data for location: 16.88333, -99.9
Fetching weather data for location: -33.16667, -64.95
Fetching weather data for location: 32.1, 71.8
Fetching weather data for location: 44.83333, 95.16667
Fetching weather data for location: 44.21667, 0.61667
Fetching weather data for location: -31.6, -65.23333
```

```
Fetching weather data for location: -30.86667, -64.55
Weather Data:
                   Name  Latitude  Longitude  Temperature (K)  Humidity (%)  \
0                Aachen  50.77500    6.08333           287.50            91
1                Aarhus  56.18333   10.23333           287.57            82
2                  Abee  54.21667 -113.00000           287.38            71
3              Acapulco  16.88333  -99.90000           300.65            85
4               Achiras -33.16667  -64.95000           277.76            72
5              Adhi Kot  32.10000   71.80000           314.81            34
6   Adzhi-Bogdo (stone)  44.83333   95.16667           288.67            22
7                  Agen  44.21667    0.61667           294.64            88
8                Aguada -31.60000  -65.23333           280.45            58
9         Aguila Blanca -30.86667  -64.55000           282.38            65

   Wind Speed (m/s) Weather Description                         City  \
0              2.59    scattered clouds                       Aachen
1              5.20           clear sky                      Risskov
2              2.38    scattered clouds                     Thorhild
3              1.86          light rain          Acapulco de Juárez
4              2.98       broken clouds                      Achiras
5              4.99           clear sky                      Harnoli
6              6.23       broken clouds                   Bayan-Ovoo
7              1.54       broken clouds                   Le Passage
8              1.50    scattered clouds  Departamento de San Alberto
9              1.79          few clouds            Capilla del Monte

     Verified Location    Year
0               Aachen  1880.0
1               Aarhus  1951.0
2       Division No. 13  1952.0
3             Acapulco  1976.0
4      Pedanía Achiras  1902.0
5   Noorpur Thal Tehsil  1919.0
6                Altai  1949.0
7                 Agen  1814.0
8    Pedanía Panaholma  1930.0
9    Capilla del Monte  1920.0
```

Description: This step fetches weather data from the OpenWeatherMap API for a list of predefined locations. It extracts relevant weather details like temperature, humidity, wind speed, and weather description based on latitude and longitude, and stores this data in a DataFrame.

Step 2: Replace Headers

[85]:
```
# Rename the headers for the fetched weather data to be more readable and consistent.
```

```python
weather_df.columns = ['Name', 'Latitude', 'Longitude', 'Temperature (K)',
 ↪'Humidity (%)', 'Wind Speed (m/s)', 'Weather Description', 'City', 'Verified
 ↪Location', 'Year']
print("Headers replaced.")
print(weather_df.head(10))
```

```
Headers replaced.
                  Name  Latitude  Longitude  Temperature (K)  Humidity (%)  \
0               Aachen  50.77500    6.08333           287.50            91
1               Aarhus  56.18333   10.23333           287.57            82
2                 Abee  54.21667 -113.00000           287.38            71
3             Acapulco  16.88333  -99.90000           300.65            85
4              Achiras -33.16667  -64.95000           277.76            72
5             Adhi Kot  32.10000   71.80000           314.81            34
6   Adzhi-Bogdo (stone)  44.83333   95.16667           288.67            22
7                 Agen  44.21667    0.61667           294.64            88
8               Aguada -31.60000  -65.23333           280.45            58
9        Aguila Blanca -30.86667  -64.55000           282.38            65

   Wind Speed (m/s) Weather Description                         City  \
0              2.59    scattered clouds                       Aachen
1              5.20           clear sky                      Risskov
2              2.38    scattered clouds                     Thorhild
3              1.86          light rain           Acapulco de Juárez
4              2.98       broken clouds                      Achiras
5              4.99           clear sky                      Harnoli
6              6.23       broken clouds                   Bayan-Ovoo
7              1.54       broken clouds                   Le Passage
8              1.50    scattered clouds  Departamento de San Alberto
9              1.79          few clouds             Capilla del Monte

     Verified Location    Year
0               Aachen  1880.0
1               Aarhus  1951.0
2      Division No. 13  1952.0
3             Acapulco  1976.0
4      Pedanía Achiras  1902.0
5  Noorpur Thal Tehsil  1919.0
6                Altai  1949.0
7                 Agen  1814.0
8    Pedanía Panaholma  1930.0
9    Capilla del Monte  1920.0
```

Description: Renamed the headers for the fetched weather data to be more readable and consistent. This step ensures the dataset is easy to understand and interpret.

Step 3: Convert Temperature from Kelvin to Celsius

[86]:

```python
# Check if 'Temperature (K)' column exists
if 'Temperature (K)' in weather_df.columns:
    # Convert the 'Temperature (K)' column to numeric type and then convert␣
↪from Kelvin to Celsius for easier interpretation.
    weather_df['Temperature (K)'] = pd.to_numeric(weather_df['Temperature␣
↪(K)'], errors='coerce')
    weather_df['Temperature (C)'] = weather_df['Temperature (K)'].apply(lambda␣
↪x: x - 273.15 if pd.notnull(x) else x)
    # Drop the original Kelvin temperature column
    weather_df.drop(columns=['Temperature (K)'], inplace=True)
    print("Step #2: Temperature converted from Kelvin to Celsius.")
else:
    print("Step #2: 'Temperature (K)' column not found.")
print(weather_df.head(10))
```

```
Step #2: Temperature converted from Kelvin to Celsius.
                  Name  Latitude  Longitude  Humidity (%)  Wind Speed (m/s)  \
0               Aachen  50.77500    6.08333            91              2.59
1               Aarhus  56.18333   10.23333            82              5.20
2                 Abee  54.21667 -113.00000            71              2.38
3             Acapulco  16.88333  -99.90000            85              1.86
4              Achiras -33.16667  -64.95000            72              2.98
5             Adhi Kot  32.10000   71.80000            34              4.99
6   Adzhi-Bogdo (stone)  44.83333   95.16667            22              6.23
7                 Agen  44.21667    0.61667            88              1.54
8               Aguada -31.60000  -65.23333            58              1.50
9        Aguila Blanca -30.86667  -64.55000            65              1.79

  Weather Description                         City     Verified Location  \
0    scattered clouds                       Aachen                Aachen
1           clear sky                      Risskov                Aarhus
2    scattered clouds                     Thorhild      Division No. 13
3          light rain           Acapulco de Juárez              Acapulco
4       broken clouds                      Achiras       Pedanía Achiras
5           clear sky                      Harnoli   Noorpur Thal Tehsil
6       broken clouds                    Bayan-Ovoo                Altai
7       broken clouds                   Le Passage                  Agen
8    scattered clouds  Departamento de San Alberto    Pedanía Panaholma
9          few clouds             Capilla del Monte    Capilla del Monte

     Year  Temperature (C)
0  1880.0            14.35
1  1951.0            14.42
2  1952.0            14.23
3  1976.0            27.50
4  1902.0             4.61
5  1919.0            41.66
```

```
6   1949.0            15.52
7   1814.0            21.49
8   1930.0             7.30
9   1920.0             9.23
```

Description: Converted the 'Temperature (K)' column to numeric type and then converted from Kelvin to Celsius for easier interpretation. This step makes the temperature data more familiar and usable for analysis.

Step 4: Identify and Remove Duplicate Entries

[87]:
```python
# Check for and remove any duplicate entries based on the meteorite name.
weather_df.drop_duplicates(subset='Name', inplace=True)
print("Step #3: Duplicate entries removed.")
print(weather_df.head(10))
```

```
Step #3: Duplicate entries removed.
                  Name  Latitude  Longitude  Humidity (%)  Wind Speed (m/s)  \
0               Aachen  50.77500    6.08333            91              2.59
1               Aarhus  56.18333   10.23333            82              5.20
2                 Abee  54.21667 -113.00000            71              2.38
3             Acapulco  16.88333  -99.90000            85              1.86
4              Achiras -33.16667  -64.95000            72              2.98
5             Adhi Kot  32.10000   71.80000            34              4.99
6  Adzhi-Bogdo (stone)  44.83333   95.16667            22              6.23
7                 Agen  44.21667    0.61667            88              1.54
8               Aguada -31.60000  -65.23333            58              1.50
9        Aguila Blanca -30.86667  -64.55000            65              1.79

  Weather Description                        City    Verified Location  \
0    scattered clouds                      Aachen               Aachen
1           clear sky                     Risskov               Aarhus
2    scattered clouds                    Thorhild      Division No. 13
3          light rain          Acapulco de Juárez             Acapulco
4      broken clouds                      Achiras      Pedanía Achiras
5           clear sky                     Harnoli  Noorpur Thal Tehsil
6      broken clouds                   Bayan-Ovoo                Altai
7      broken clouds                  Le Passage                 Agen
8    scattered clouds  Departamento de San Alberto    Pedanía Panaholma
9          few clouds           Capilla del Monte    Capilla del Monte

     Year  Temperature (C)
0  1880.0            14.35
1  1951.0            14.42
2  1952.0            14.23
3  1976.0            27.50
4  1902.0             4.61
5  1919.0            41.66
6  1949.0            15.52
```

```
7  1814.0             21.49
8  1930.0              7.30
9  1920.0              9.23
```

Description: Checked for and removed any duplicate entries based on the meteorite name. This step ensures the dataset is free from redundant records.

Step 5: Handle Missing values

[88]:
```python
# Fill missing values in the 'City' column with 'Unknown' and drop rows with␣
↪missing values in essential columns.
weather_df['City'].fillna('Unknown', inplace=True)
weather_df.dropna(subset=['Latitude', 'Longitude', 'Temperature (C)'],␣
↪inplace=True)
print("Step #4: Missing values handled.")
print(weather_df.head(10))
```

```
Step #4: Missing values handled.
                 Name  Latitude  Longitude  Humidity (%)  Wind Speed (m/s)  \
0              Aachen  50.77500    6.08333            91              2.59
1              Aarhus  56.18333   10.23333            82              5.20
2                Abee  54.21667 -113.00000            71              2.38
3            Acapulco  16.88333  -99.90000            85              1.86
4             Achiras -33.16667  -64.95000            72              2.98
5            Adhi Kot  32.10000   71.80000            34              4.99
6  Adzhi-Bogdo (stone)  44.83333   95.16667            22              6.23
7                Agen  44.21667    0.61667            88              1.54
8              Aguada -31.60000  -65.23333            58              1.50
9        Aguila Blanca -30.86667  -64.55000            65              1.79

  Weather Description                         City    Verified Location  \
0    scattered clouds                       Aachen               Aachen
1           clear sky                      Risskov               Aarhus
2    scattered clouds                     Thorhild      Division No. 13
3          light rain           Acapulco de Juárez             Acapulco
4       broken clouds                      Achiras      Pedanía Achiras
5           clear sky                      Harnoli  Noorpur Thal Tehsil
6       broken clouds                  Bayan-Ovoo                Altai
7       broken clouds                   Le Passage                 Agen
8    scattered clouds  Departamento de San Alberto    Pedanía Panaholma
9          few clouds              Capilla del Monte    Capilla del Monte

     Year  Temperature (C)
0  1880.0            14.35
1  1951.0            14.42
2  1952.0            14.23
3  1976.0            27.50
4  1902.0             4.61
5  1919.0            41.66
```

```
6  1949.0            15.52
7  1814.0            21.49
8  1930.0             7.30
9  1920.0             9.23
```

Description: Filled missing values in the 'City' column with 'Unknown' and dropped rows with missing values in essential columns. This step ensures the dataset is complete and consistent, minimizing potential biases.

Step 6: Format Data for Readability

[89]:
```python
# Ensure the data types are appropriate for analysis and format the 'Year'␣
↪column as an integer.
weather_df['Year'] = weather_df['Year'].astype(int)
weather_df['Latitude'] = weather_df['Latitude'].astype(float)
weather_df['Longitude'] = weather_df['Longitude'].astype(float)
weather_df['Temperature (C)'] = weather_df['Temperature (C)'].astype(float)
weather_df['Humidity (%)'] = weather_df['Humidity (%)'].astype(int)
weather_df['Wind Speed (m/s)'] = weather_df['Wind Speed (m/s)'].astype(float)
print("Step #5: Data formatted for readability.")
print(weather_df.head(10))
```

```
Step #5: Data formatted for readability.
                 Name  Latitude  Longitude  Humidity (%)  Wind Speed (m/s)  \
0              Aachen  50.77500    6.08333            91              2.59
1              Aarhus  56.18333   10.23333            82              5.20
2                Abee  54.21667 -113.00000            71              2.38
3            Acapulco  16.88333  -99.90000            85              1.86
4             Achiras -33.16667  -64.95000            72              2.98
5            Adhi Kot  32.10000   71.80000            34              4.99
6  Adzhi-Bogdo (stone)  44.83333   95.16667            22              6.23
7                Agen  44.21667    0.61667            88              1.54
8              Aguada -31.60000  -65.23333            58              1.50
9        Aguila Blanca -30.86667  -64.55000            65              1.79

  Weather Description                        City     Verified Location  Year  \
0    scattered clouds                      Aachen                Aachen  1880
1           clear sky                     Risskov                Aarhus  1951
2    scattered clouds                    Thorhild      Division No. 13  1952
3          light rain          Acapulco de Juárez              Acapulco  1976
4       broken clouds                     Achiras      Pedanía Achiras  1902
5           clear sky                     Harnoli   Noorpur Thal Tehsil  1919
6       broken clouds                  Bayan-Ovoo                 Altai  1949
7       broken clouds                  Le Passage                  Agen  1814
8    scattered clouds  Departamento de San Alberto    Pedanía Panaholma  1930
9          few clouds            Capilla del Monte    Capilla del Monte  1920

   Temperature (C)
0            14.35
```

```
1             14.42
2             14.23
3             27.50
4              4.61
5             41.66
6             15.52
7             21.49
8              7.30
9              9.23
```

Description: Ensured the data types are appropriate for analysis and formatted the 'Year' column as an integer. This step ensures the dataset is ready for analysis, with correctly formatted and typed data.

[94]:
```python
# Display the first few rows of the weather data
print("Cleaned Weather Data:")
print(weather_df.head(10))
```

```
Cleaned Weather Data:
                  Name  Latitude  Longitude  Humidity (%)  Wind Speed (m/s)  \
0               Aachen  50.77500    6.08333            91              2.59
1               Aarhus  56.18333   10.23333            82              5.20
2                 Abee  54.21667 -113.00000            71              2.38
3             Acapulco  16.88333  -99.90000            85              1.86
4              Achiras -33.16667  -64.95000            72              2.98
5             Adhi Kot  32.10000   71.80000            34              4.99
6  Adzhi-Bogdo (stone)  44.83333   95.16667            22              6.23
7                 Agen  44.21667    0.61667            88              1.54
8               Aguada -31.60000  -65.23333            58              1.50
9        Aguila Blanca -30.86667  -64.55000            65              1.79

  Weather Description                         City    Verified Location  Year  \
0    scattered clouds                       Aachen               Aachen  1880
1           clear sky                      Risskov               Aarhus  1951
2    scattered clouds                     Thorhild      Division No. 13  1952
3          light rain          Acapulco de Juárez             Acapulco  1976
4      broken clouds                      Achiras      Pedanía Achiras  1902
5           clear sky                      Harnoli  Noorpur Thal Tehsil  1919
6      broken clouds                    Bayan-Ovoo                Altai  1949
7      broken clouds                    Le Passage                 Agen  1814
8    scattered clouds  Departamento de San Alberto    Pedanía Panaholma  1930
9          few clouds            Capilla del Monte    Capilla del Monte  1920

   Temperature (C)
0             14.35
1             14.42
2             14.23
3             27.50
4              4.61
```

```
5             41.66
6             15.52
7             21.49
8              7.30
9              9.23
```

0.6 Ethical Implications of Data Wrangling

In this project, several data transformations and cleansing steps were performed to integrate weather data from the OpenWeatherMap API with meteorite landing data. These steps included replacing headers, converting temperature units, formatting text, handling missing data, ensuring no duplicates, and performing reverse geocoding to verify location data. While these steps improve data quality, they also raise ethical considerations.

Changes Made to the Data:

- Headers Replaced: Headers were renamed for consistency and readability, ensuring that the data is easy to understand and interpret.
- Temperature Conversion: Temperature data from the API, originally in Kelvin, was converted to Celsius for easier interpretation.
- Formatting: Weather descriptions were standardized to title case, enhancing readability.
- Handling Missing Data: Missing values were addressed to ensure a complete dataset, with missing city names filled as 'Unknown.'
- Duplicate Removal: Duplicate entries based on meteorite names were identified and removed to maintain data integrity.
- Reverse Geocoding: Reverse geocoding was performed to verify the accuracy of location data, ensuring that the latitude and longitude coordinates matched known locations.

Legal or Regulatory Guidelines: Data usage policies from the OpenWeatherMap API were strictly followed. Compliance with data privacy regulations, especially when dealing with sensitive location data, was ensured to protect user privacy and adhere to legal standards. #### Risks Created Based on Transformations:

Data Misinterpretation: Potential misinterpretation of converted data, such as temperature units, if not documented correctly. Inaccuracies: Risk of inaccuracies if the API data is not up-to-date or misrepresented, affecting the reliability of analyses. Mismatch: Potential mismatch between meteorite landing locations and corresponding weather data if coordinates are incorrect or poorly matched.

Assumptions Made in Cleaning/Transforming the Data: Temperature Units: Assumed that all temperature values from the API are in Kelvin. City Names: Assumed that the city name accurately reflects the weather data provided. Population Data: Assumed that the population data matches the city names provided by the weather data.

Data Sourcing and Verification for Credibility: The OpenWeatherMap API is a reputable source for weather data, ensuring the credibility of the weather information. The population data was verified by cross-referencing with known sources to maintain accuracy and reliability.

Ethical Acquisition of Data: The data was acquired from publicly accessible APIs and websites, ensuring ethical compliance and adherence to data usage policies.

Mitigation of Ethical Implications: Transparency: Documenting all transformations and assumptions made during data cleaning ensures transparency and accountability. Verification: Regular verification of data accuracy and consistency helps avoid misinformation and maintains data integrity. Compliance: Adhering to data usage policies and privacy regulations protects sensitive information and ensures ethical data handling.

By incorporating reverse geocoding, the accuracy of location data was verified, reducing the risk of mismatches and enhancing the reliability of the dataset. These steps and considerations help support the project's goals while maintaining ethical standards, ensuring that the data used is accurate, reliable, and ethically sourced.

0.7 Project: Milestone 5 Merging the Data and Storing in a Database/Visualizing Data

[2]:
```python
import pandas as pd
import pandas as pd
import requests
import time
import sqlite3
import matplotlib.pyplot as plt
import seaborn as sns

# Load the cleaned meteorite landings data
meteorite_landings_path = 'Cleaned_Meteorite_Landings.csv'
meteorite_df = pd.read_csv(meteorite_landings_path)

# Display the first few rows of the meteorite landings data
print("Meteorite Landings Data:")
print(meteorite_df.head())
```

```
Meteorite Landings Data:
       name   id nametype     recclass  mass_(g)  fall  year    reclat  \
0    Aachen    1    Valid           L5      21.0  Fell  1880  50.77500
1    Aarhus    2    Valid           H6     720.0  Fell  1951  56.18333
2      Abee    6    Valid          EH4   63000.0  Fell  1952  54.21667
3  Acapulco   10    Valid  Acapulcoite    1914.0  Fell  1976  16.88333
4   Achiras  370    Valid           L6     780.0  Fell  1902 -33.16667

     reclong           geolocation
0    6.08333     (50.775, 6.08333)
1   10.23333  (56.18333, 10.23333)
2 -113.00000    (54.21667, -113.0)
3  -99.90000     (16.88333, -99.9)
4  -64.95000   (-33.16667, -64.95)
```

[45]:

```python
# OpenWeatherMap API key
api_key = '559894e58b44c7b8b41d86b82e3728a7'

# List of specific cities to fetch weather data for
cities_of_interest = ['southampton', 'ottawa', 'jilin', 'tyumen', 'athens',␣
↪'fuyang', 'new york', 'lusaka',
                      'toulouse', 'glasgow', 'bursa', 'charlotte',␣
↪'chelyabinsk', 'porto alegre', 'nantong',
                      'novosibirsk', 'dongtai', 'perth', 'changde', 'santa␣
↪cruz', 'saratov', 'hiroshima', 'moradabad',
                      'queretaro', 'tirupati', 'delhi', 'cali', 'columbus',␣
↪'havana', 'leshan', 'jalandhar', 'meerut',
                      'krasnodar', 'palermo', 'johannesburg', 'ningbo',␣
↪'valencia', 'madrid', 'salem', 'aleppo', 'rosario']

# Function to fetch weather data based on city name
def fetch_weather_data_by_city(city):
    url = f'http://api.openweathermap.org/data/2.5/weather?
↪q={city}&appid={api_key}'
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    elif response.status_code == 429:
        print(f"Rate limit exceeded for {city}: {response.status_code}")
        return 'rate_limit_exceeded'
    else:
        print(f"Error fetching data for {city}: {response.status_code}")
        return None

# Function to fetch weather data based on Latitude and Longitude
def fetch_weather_data_by_coords(lat, lon):
    url = f'http://api.openweathermap.org/data/2.5/weather?
↪lat={lat}&lon={lon}&appid={api_key}'
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    elif response.status_code == 429:
        print(f"Rate limit exceeded for {lat}, {lon}: {response.status_code}")
        return 'rate_limit_exceeded'
    else:
        print(f"Error fetching data for {lat}, {lon}: {response.status_code}")
        return None

# Initialize an empty list to hold the weather data
weather_data_list = []
```

```
# Fetch data from OpenWeatherMap API for each meteorite landing location
fetch_attempts = 0
max_attempts = 100  # Increased the max_attempts to fetch data for more cities
retry_attempts = 5
retry_wait_time = 60  # Wait time in seconds for retry

# Fetch weather data for cities of interest
for city in cities_of_interest:
    attempt = 0
    while attempt < retry_attempts:
        print(f"Fetching weather data for city: {city}")
        weather_data = fetch_weather_data_by_city(city)
        if weather_data == 'rate_limit_exceeded':
            print(f"Waiting for {retry_wait_time} seconds before retrying...")
            time.sleep(retry_wait_time)
            attempt += 1
        elif weather_data:
            weather_info = {
                'City': city,
                'Latitude': weather_data['coord']['lat'],
                'Longitude': weather_data['coord']['lon'],
                'Temperature': weather_data['main']['temp'],
                'Humidity': weather_data['main']['humidity'],
                'WindSpeed': weather_data['wind']['speed'],
                'WeatherDescription': weather_data['weather'][0]['description']
            }
            weather_data_list.append(weather_info)
            fetch_attempts += 1
            break
        else:
            print(f"Could not fetch data for city: {city}")
            break
    if attempt == retry_attempts:
        print(f"Max retry attempts reached for city: {city}")
    if fetch_attempts >= max_attempts:
        break
    time.sleep(1)  # Add a 1-second delay between requests

# Fetch weather data for each meteorite landing location
for index, row in meteorite_df.iterrows():
    lat = row['reclat']
    lon = row['reclong']
    attempt = 0
    while attempt < retry_attempts:
        print(f"Fetching weather data for location: {lat}, {lon}")
        weather_data = fetch_weather_data_by_coords(lat, lon)
        if weather_data == 'rate_limit_exceeded':
```

```python
            print(f"Waiting for {retry_wait_time} seconds before retrying...")
            time.sleep(retry_wait_time)
            attempt += 1
        elif weather_data:
            weather_info = {
                'City': row['name'],
                'Latitude': lat,
                'Longitude': lon,
                'Temperature': weather_data['main']['temp'],
                'Humidity': weather_data['main']['humidity'],
                'WindSpeed': weather_data['wind']['speed'],
                'WeatherDescription': weather_data['weather'][0]['description']
            }
            weather_data_list.append(weather_info)
            fetch_attempts += 1
            break
        else:
            print(f"Could not fetch data for location: {lat}, {lon}")
            break
    if attempt == retry_attempts:
        print(f"Max retry attempts reached for location: {lat}, {lon}")
    if fetch_attempts >= max_attempts:
        break
    time.sleep(1)  # Add a 1-second delay between requests

# Convert the list to a DataFrame
weather_df = pd.DataFrame(weather_data_list)

# Display the first few rows of the weather data
print("Weather Data:")
print(weather_df.head(20))
```

```
Fetching weather data for city: southampton
Fetching weather data for city: ottawa
Fetching weather data for city: jilin
Fetching weather data for city: tyumen
Fetching weather data for city: athens
Fetching weather data for city: fuyang
Fetching weather data for city: new york
Fetching weather data for city: lusaka
Fetching weather data for city: toulouse
Fetching weather data for city: glasgow
Fetching weather data for city: bursa
Fetching weather data for city: charlotte
Fetching weather data for city: chelyabinsk
Fetching weather data for city: porto alegre
Fetching weather data for city: nantong
Fetching weather data for city: novosibirsk
```

```
Fetching weather data for city: dongtai
Fetching weather data for city: perth
Fetching weather data for city: changde
Fetching weather data for city: santa cruz
Fetching weather data for city: saratov
Fetching weather data for city: hiroshima
Fetching weather data for city: moradabad
Fetching weather data for city: queretaro
Fetching weather data for city: tirupati
Fetching weather data for city: delhi
Fetching weather data for city: cali
Fetching weather data for city: columbus
Fetching weather data for city: havana
Fetching weather data for city: leshan
Fetching weather data for city: jalandhar
Fetching weather data for city: meerut
Fetching weather data for city: krasnodar
Fetching weather data for city: palermo
Fetching weather data for city: johannesburg
Fetching weather data for city: ningbo
Fetching weather data for city: valencia
Fetching weather data for city: madrid
Fetching weather data for city: salem
Fetching weather data for city: aleppo
Fetching weather data for city: rosario
Fetching weather data for location: 50.775, 6.08333
Fetching weather data for location: 56.18333, 10.23333
Fetching weather data for location: 54.21667, -113.0
Fetching weather data for location: 16.88333, -99.9
Fetching weather data for location: -33.16667, -64.95
Fetching weather data for location: 32.1, 71.8
Fetching weather data for location: 44.83333, 95.16667
Fetching weather data for location: 44.21667, 0.61667
Fetching weather data for location: -31.6, -65.23333
Fetching weather data for location: -30.86667, -64.55
Fetching weather data for location: 16.39806, -9.57028
Fetching weather data for location: 19.08333, 8.38333
Fetching weather data for location: 50.66667, 2.33333
Fetching weather data for location: 29.51667, 35.05
Fetching weather data for location: 29.71667, 77.95
Fetching weather data for location: 8.91667, 8.43333
Fetching weather data for location: 39.91667, 42.81667
Fetching weather data for location: 24.41667, 39.51667
Fetching weather data for location: 13.66033, 28.96
Fetching weather data for location: 44.11667, 4.08333
Fetching weather data for location: 44.65, 11.01667
Fetching weather data for location: 2.0, 22.66667
Fetching weather data for location: 45.82133, 6.01533
```

```
Fetching weather data for location: 51.78333, -1.78333
Fetching weather data for location: 36.23333, 37.13333
Fetching weather data for location: 44.88333, 8.75
Fetching weather data for location: 50.95, 31.81667
Fetching weather data for location: 45.26667, 10.15
Fetching weather data for location: 42.53333, -85.88333
Fetching weather data for location: 26.96667, -105.31667
Fetching weather data for location: 20.74575, 32.41275
Fetching weather data for location: 35.27333, 44.21556
Fetching weather data for location: 27.66667, 78.25
Fetching weather data for location: 26.58333, 85.56667
Fetching weather data for location: 44.61667, -70.75
Fetching weather data for location: 48.7, 37.5
Fetching weather data for location: 20.88333, 76.86667
Fetching weather data for location: 0.0, 0.0
Fetching weather data for location: 47.46667, -0.55
Fetching weather data for location: -22.96667, -44.31667
Fetching weather data for location: 9.53333, 39.71667
Fetching weather data for location: 25.15, 105.18333
Fetching weather data for location: 40.81056, 140.78556
Fetching weather data for location: 53.58333, -2.71667
Fetching weather data for location: 43.86667, 5.38333
Fetching weather data for location: -33.0, -66.0
Fetching weather data for location: 38.5, -94.3
Fetching weather data for location: -31.41667, -60.66667
Fetching weather data for location: 42.45, 9.03333
Fetching weather data for location: 31.805, -97.01
Fetching weather data for location: 52.05, 0.3
Fetching weather data for location: 43.03333, 12.55
Fetching weather data for location: 25.25417, 80.625
Fetching weather data for location: 20.06667, -103.66667
Fetching weather data for location: 34.75, -87.0
Fetching weather data for location: 34.31667, -96.15
Fetching weather data for location: 44.38333, 5.16667
Fetching weather data for location: 36.16667, 3.66667
Fetching weather data for location: 44.33333, 3.23333
Weather Data:
          City  Latitude  Longitude  Temperature  Humidity  WindSpeed  \
0  southampton   50.9040    -1.4043       286.69        91       1.93
1       ottawa   45.4112   -75.6981       288.65        86       2.06
2        jilin   43.0000   126.0000       301.50        70       3.61
3       tyumen   57.1522    65.5272       285.20        94       3.00
4       athens   37.9795    23.7162       299.92        47       3.13
5       fuyang   32.9000   115.8167       306.71        66       0.60
6     new york   40.7143   -74.0060       292.34        87       4.92
7       lusaka  -15.4067    28.2871       288.15        48       4.11
8     toulouse   43.6043     1.4437       294.06        78       1.03
9      glasgow   55.8652    -4.2576       285.94        86       4.63
```

```
10         bursa   40.1667   29.0833       285.17        66        1.78
11     charlotte   35.2271  -80.8431       298.11        86        4.02
12   chelyabinsk   55.1544   61.4297       288.19        97        5.00
13  porto alegre  -30.0331  -51.2300       289.14        97        1.54
14       nantong   32.0303  120.8747       305.94        62        2.28
15   novosibirsk   55.0411   82.9344       292.76        94        2.00
16       dongtai   32.8523  120.3095       305.80        69        1.71
17         perth  -31.9333  115.8333       291.54        73        4.63
18       changde   29.0464  111.6783       305.88        51        4.45
19    santa cruz  -17.8000  -63.1667       294.85       100        2.57

   WeatherDescription
0          few clouds
1          few clouds
2       broken clouds
3       broken clouds
4           clear sky
5     overcast clouds
6     overcast clouds
7           clear sky
8           clear sky
9       broken clouds
10          clear sky
11    overcast clouds
12      broken clouds
13      moderate rain
14    overcast clouds
15    overcast clouds
16    overcast clouds
17      broken clouds
18          clear sky
19   scattered clouds
```

[46]:

```
# Load the population data
population_data_path = 'Cleaned_Population_Data.csv'
population_df = pd.read_csv(population_data_path)

# Display the first few rows of the population data
print("Population Data:")
print(population_df.head())
```

```
Population Data:
   rank       city     country  2024_population  2023_population  growth_rate
0     0      Tokyo       Japan         37115035         37194105        -0.21
1     1      Delhi       India         33807403         32941309         2.63
2     2   Shanghai       China         29867918         29210808         2.25
3     3      Dhaka  Bangladesh         23935652         23209616         3.13
4     4  Sao Paulo      Brazil         22806704         22619736         0.83
```

[47]:

```python
# Checking unique city names in weather and population datasets
unique_cities_weather = weather_df['City'].unique()
unique_cities_population = population_df['city'].unique()

print("Unique city names in weather data:")
print(unique_cities_weather)

print("Unique city names in population data:")
print(unique_cities_population)
```

```
Unique city names in weather data:
['southampton' 'ottawa' 'jilin' 'tyumen' 'athens' 'fuyang' 'new york'
 'lusaka' 'toulouse' 'glasgow' 'bursa' 'charlotte' 'chelyabinsk'
 'porto alegre' 'nantong' 'novosibirsk' 'dongtai' 'perth' 'changde'
 'santa cruz' 'saratov' 'hiroshima' 'moradabad' 'queretaro' 'tirupati'
 'delhi' 'cali' 'columbus' 'havana' 'leshan' 'jalandhar' 'meerut'
 'krasnodar' 'palermo' 'johannesburg' 'ningbo' 'valencia' 'madrid' 'salem'
 'aleppo' 'rosario' 'aachen' 'aarhus' 'abee' 'acapulco' 'achiras'
 'adhi kot' 'adzhi-bogdo (stone)' 'agen' 'aguada' 'aguila blanca'
 'aioun el atrouss' 'aïr' 'aire-sur-la-lys' 'akaba' 'akbarpur' 'akwanga'
 'akyumak' 'al rais' 'al zarnkh' 'alais' 'albareto' 'alberta'
 'alby sur chéran' 'aldsworth' 'alessandria' 'alexandrovsky' 'alfianello'
 'allegan' 'allende' 'almahata sitta' "alta'ameem" 'ambapur nagla'
 'andhara' 'andover' 'andreevka' 'andura' 'northwest africa 5815' 'angers'
 'angra dos reis (stone)' 'ankober' 'anlong' 'aomori' 'appley bridge'
 'apt' 'arbol solo' 'archie' 'arroyo aguiar' 'asco' 'ash creek' 'ashdon'
 'assisi' 'atarra' 'atemajac' 'atoka' 'aubres' 'aumale' 'aumieres']
Unique city names in population data:
['Tokyo' 'Delhi' 'Shanghai' 'Dhaka' 'Sao Paulo' 'Cairo' 'Mexico City'
 'Beijing' 'Mumbai' 'Osaka' 'Chongqing' 'Karachi' 'Kinshasa' 'Lagos'
 'Istanbul' 'Buenos Aires' 'Kolkata' 'Manila' 'Guangzhou' 'Tianjin'
 'Lahore' 'Bangalore' 'Rio De Janeiro' 'Shenzhen' 'Moscow' 'Chennai'
 'Bogota' 'Jakarta' 'Lima' 'Paris' 'Bangkok' 'Hyderabad' 'Seoul' 'Nanjing'
 'Chengdu' 'London' 'Luanda' 'Tehran' 'Ho Chi Minh City' 'Nagoya'
 'Xi An Shaanxi' 'Ahmedabad' 'Wuhan' 'Kuala Lumpur' 'Hangzhou' 'Suzhou'
 'Surat' 'Dar Es Salaam' 'New York' 'Baghdad' 'Shenyang' 'Riyadh'
 'Hong Kong' 'Foshan' 'Dongguan' 'Pune' 'Santiago' 'Haerbin' 'Madrid'
 'Khartoum' 'Toronto' 'Johannesburg' 'Belo Horizonte' 'Dalian' 'Singapore'
 'Qingdao' 'Zhengzhou' 'Ji Nan Shandong' 'Abidjan' 'Barcelona' 'Yangon'
 'Addis Ababa' 'Alexandria' 'Saint Petersburg' 'Nairobi' 'Chittagong'
 'Guadalajara' 'Fukuoka' 'Ankara' 'Hanoi' 'Melbourne' 'Monterrey' 'Sydney'
 'Changsha' 'Urumqi' 'Cape Town' 'Jiddah' 'Brasilia' 'Kunming' 'Changchun'
 'Kabul' 'Hefei' 'Yaounde' 'Ningbo' 'Shantou' 'New Taipei' 'Tel Aviv'
 'Kano' 'Shijiazhuang' 'Montreal' 'Rome' 'Jaipur' 'Recife' 'Nanning'
 'Fortaleza' 'Kozhikode' 'Porto Alegre' 'Taiyuan Shanxi' 'Douala'
 'Ekurhuleni' 'Malappuram' 'Medellin' 'Changzhou' 'Kampala' 'Antananarivo'
 'Lucknow' 'Abuja' 'Nanchang' 'Wenzhou' 'Xiamen' 'Ibadan' 'Fuzhou Fujian'
 'Salvador' 'Casablanca' 'Tangshan Hebei' 'Kumasi' 'Curitiba' 'Bekasi'
```

'Faisalabad' 'Los Angeles' 'Guiyang' 'Port Harcourt' 'Thrissur'
'Santo Domingo' 'Berlin' 'Asuncion' 'Dakar' 'Kochi' 'Wuxi' 'Busan'
'Campinas' 'Mashhad' 'Sanaa' 'Puebla' 'Indore' 'Lanzhou' 'Ouagadougou'
'Kuwait City' 'Lusaka' 'Kanpur' 'Durban' 'Guayaquil' 'Pyongyang' 'Milan'
'Guatemala City' 'Athens' 'Depok' 'Izmir' 'Nagpur' 'Surabaya' 'Handan'
'Coimbatore' 'Huaian' 'Port Au Prince' 'Zhongshan' 'Dubai' 'Bamako'
'Mbuji Mayi' 'Kiev' 'Lisbon' 'Weifang' 'Caracas' 'Thiruvananthapuram'
'Algiers' 'Shizuoka' 'Lubumbashi' 'Cali' 'Goiania' 'Pretoria' 'Shaoxing'
'Incheon' 'Yantai' 'Zibo' 'Huizhou' 'Manchester' 'Taipei' 'Mogadishu'
'Brazzaville' 'Accra' 'Bandung' 'Damascus' 'Birmingham' 'Vancouver'
'Toluca De Lerdo' 'Luoyang' 'Sapporo' 'Chicago' 'Tashkent' 'Patna'
'Bhopal' 'Tangerang' 'Nantong' 'Brisbane' 'Tunis' 'Peshawar' 'Medan'
'Gujranwala' 'Baku' 'Hohhot' 'San Juan' 'Belem' 'Rawalpindi' 'Agra'
'Manaus' 'Kannur' 'Beirut' 'Maracaibo' 'Liuzhou' 'Visakhapatnam' 'Baotou'
'Vadodara' 'Barranquilla' 'Phnom Penh' 'Sendai' 'Taoyuan' 'Xuzhou'
'Houston' 'Aleppo' 'Tijuana' 'Esfahan' 'Nashik' 'Vijayawada' 'Amman'
'Putian' 'Multan' 'Grande Vitoria' 'Wuhu Anhui' 'Mecca' 'Kollam' 'Naples'
'Daegu' 'Conakry' 'Yangzhou' 'Havana' 'Taizhou Zhejiang' 'Baoding'
'Perth' 'Brussels' 'Linyi Shandong' 'Bursa' 'Rajkot' 'Minsk' 'Hiroshima'
'Haikou' 'Daqing' 'Lome' 'Lianyungang' 'Yancheng Jiangsu' 'Panama City'
'Almaty' 'Semarang' 'Valencia' 'Davao City' 'Vienna' 'Rabat' 'Ludhiana'
'Quito' 'Benin City' 'La Paz' 'Baixada Santista' 'West Yorkshire'
'Can Tho' 'Zhuhai' 'Leon De Los Aldamas' 'Quanzhou' 'Matola' 'Datong'
'Sharjah' 'Madurai' 'Raipur' 'Adana' 'Santa Cruz' 'Palembang' 'Mosul'
'Cixi' 'Meerut' 'Gaziantep' 'La Laguna' 'Batam' 'Turin' 'Warsaw'
'Jiangmen' 'Varanasi' 'Hamburg' 'Montevideo' 'Budapest' 'Lyon'
'Xiangyang' 'Bucharest' 'Yichang' 'Yinchuan' 'Shiraz' 'Kananga'
'Srinagar' 'Monrovia' 'Tiruppur' 'Jamshedpur' 'Suqian' 'Aurangabad'
'Qinhuangdao' 'Stockholm' 'Anshan' 'Glasgow' 'Xining' 'Makassar'
'Hengyang' 'Novosibirsk' 'Ulaanbaatar' 'Onitsha' 'Jilin' 'Anyang'
'Auckland' 'Tabriz' 'Muscat' 'Calgary' 'Phoenix' 'Qiqihaer' 'N Djamena'
'Marseille' 'Cordoba' 'Jodhpur' 'Kathmandu' 'Rosario' 'Tegucigalpa'
'Ciudad Juarez' 'Harare' 'Karaj' 'Medina' 'Jining Shandong' 'Abu Dhabi'
'Munich' 'Ranchi' 'Daejon' 'Zhangjiakou' 'Edmonton' 'Mandalay' 'Gaoxiong'
'Kota' 'Natal' 'Nouakchott' 'Jabalpur' 'Huainan' 'Grande Sao Luis'
'Asansol' 'Philadelphia' 'Yekaterinburg' 'Gwangju' 'Yiwu' 'Chaozhou'
'San Antonio' 'Gwalior' 'Ganzhou' 'Homs' 'Niamey' 'Mombasa' 'Allahabad'
'Basra' 'Kisangani' 'San Jose' 'Amritsar' 'Taizhou Jiangsu' 'Chon Buri'
'Jiaxing' 'Weihai' 'Hai Phong' 'Ottawa' 'Zurich' 'Taian Shandong'
'Queretaro' 'Joao Pessoa' 'Kaifeng' 'Cochabamba' 'Konya' 'Liuyang'
'Liuan' 'Rizhao' 'Kharkiv' 'Dhanbad' 'Nanchong' 'Dongying' 'Belgrade'
'Zunyi' 'Zhanjiang' 'Bucaramanga' 'Uyo' 'Copenhagen' 'San Diego' 'Shiyan'
'Taizhong' 'Bareilly' 'Pointe Noire' 'Adelaide' 'Suweon' 'Mwanza'
'Mianyang Sichuan' 'Samut Prakan' 'Maceio' 'Qom' 'Antalya' 'Joinville'
'Tengzhou' 'Yingkou' 'Ad Dammam' 'Tanger' 'Freetown' 'Helsinki' 'Aligarh'
'Moradabad' 'Pekan Baru' 'Maoming' 'Lilongwe' 'Porto' 'Prague' 'Astana'
'Jieyang' 'Fushun Liaoning' 'Mysore' 'Abomey Calavi' 'Ruian' 'Fes'
'Port Elizabeth' 'Florianopolis' 'Ahvaz' 'Bukavu' 'Dallas' 'Nnewi'

'Kazan' 'Jinhua' 'San Luis Potosi' 'Baoji' 'Durg Bhilainagar'
'Bhubaneswar' 'Kigali' 'Sofia' 'Pingdingshan Henan' 'Dublin' 'Puning'
'Chifeng' 'Zhuzhou' 'Bujumbura' 'Zhenjiang Jiangsu' 'Liupanshui'
'Barquisimeto' 'Islamabad' 'Huaibei' 'Tasikmalaya' 'Maracay' 'Bogor'
'Da Nang' 'Nizhniy Novgorod' 'Nanyang Henan' 'Xiangtan Hunan' 'Pizhou'
'Tiruchirappalli' 'Chelyabinsk' 'Mendoza' 'Luohe' 'Xiongan' 'Chandigarh'
'Merida' 'Jinzhou' 'Benxi' 'Binzhou' 'Aba' 'Chiang Mai' 'Bazhong'
'Quetta' 'Kaduna' 'Guilin' 'Saharanpur' 'Hubli Dharwad' 'Yueqing'
'Guwahati' 'Mexicali' 'Salem' 'Maputo' 'Tripoli' 'Haifa' 'Bandar Lampung'
'Bobo Dioulasso' 'Amsterdam' 'Shimkent' 'Omsk' 'Aguascalientes'
'Hargeysa' 'Krasnoyarsk' 'Xinxiang' 'Siliguri' 'Wenling' 'Samara'
'Zaozhuang' 'Cologne' 'Yongin' 'Ufa' 'Fuyang' 'Ikorodu' 'Bien Hoa'
'Jalandhar' 'Panjin' "Ma'Anshan" 'Cuernavaca' 'Rostov On Don' 'Chihuahua'
'Fuzhou Jiangxi' 'Tshikapa' 'Shangrao' 'Samarinda' 'Bishkek' 'Zhaoqing'
'San Salvador' 'Yichun Jiangxi' 'Chenzhou' 'Sekondi Takoradi' 'Leshan'
'Aden' 'Goyang' 'Diyarbakir' 'Asmara' 'Dezhou' 'Jingzhou Hubei' 'Managua'
'Johor Bahru' 'Kermanshah' 'Nyala' 'Oslo' 'Kirkuk' 'Yerevan' 'Cartagena'
'Changshu' 'Huzhou' 'Xuchang' 'Solapur' 'Lille' 'Mersin' 'Tbilisi' 'Perm'
'Voronezh' 'Denpasar' 'Toulouse' 'Blantyre Limbe' 'Aracaju' 'Marrakech'
'Qujing' 'Yueyang' 'Ilorin' 'Tampico' 'Antwerp' 'Teresina' 'Guiping'
'Warangal' 'Changwon' 'Padang' 'Saltillo' 'Xintai' 'Cancun' 'Cebu City'
'San Miguel De Tucuman' 'Hamah' 'Acapulco De Juarez' 'Warri' 'Kayseri'
'Chengde' 'Owerri' 'Rotterdam' 'Pingxiang Jiangxi' 'Zhucheng' 'Songkhla'
'Valparaiso' 'Dehradun' 'Nonthaburi' 'Leiyang' 'Dushanbe' 'Nampula'
'Misratah' 'Krasnodar' 'Laiwu' 'Bordeaux' 'Jixi Heilongjiang'
'San Pedro Sula' 'Odesa' 'Jiujiang' 'Lubango' 'Morelia' 'Jos' 'Sylhet'
'Agadir' 'Jacksonville' 'Fort Worth' 'Volgograd' 'Mudanjiang' 'Guigang'
'Najaf' 'Bangui' 'Austin' 'Rajshahi' 'Hengshui' 'Jerusalem' 'Zhangzhou'
'Xinyu' 'Linfen' 'Tianmen' 'Ciudad Guayana' 'Zamboanga City' 'Yangjiang'
'Taiz' 'Cucuta' 'Arequipa' 'Liling' 'Antipolo' 'Veracruz' 'Reynosa'
'Khulna' 'Deyang' 'Pathum Thani' 'Bengbu' 'Jiangyin' 'Southampton'
'Villahermosa' 'Baishan' 'Nice' 'Oran' 'West Rand' 'Cabinda' 'Umuahia'
'Bogra' 'Bahawalpur' 'Seongnam' 'Guntur' 'Dnipro' 'Campo Grande' 'Malang'
'Londrina' 'Dandong' 'Changzhi' 'Hermosillo' 'Bhiwandi' 'La Plata'
'Charlotte' 'Liverpool' 'Ashgabat' 'Concepcion' 'Puducherry' 'Changde'
'Bergamo' 'Firozabad' 'Erbil' 'Tyumen' 'Trujillo' 'Liaoyang' 'Shangqiu'
'Columbus' 'Ulsan' 'Tuxtla Gutierrez' 'Kuerle' 'Soshanguve' 'Xingtai'
'Culiacan' 'Quzhou' 'Cherthala' 'Huangshi' 'Fuxin' 'Lokoja'
'Hufuf Mubarraz' 'Libreville' 'Yongzhou' 'Xinghua' 'Donetsk' 'Yibin'
'Indianapolis (Balance)' 'Enugu' 'Tainan' 'Xinyang' 'Ipoh' 'Luzhou'
'Banghazi' 'Maiduguri' 'Yangquan' 'Huaihua' 'Xiaogan' 'Tianshui' 'Bunia'
'Bozhou' 'Kottayam' 'Zhuji' 'Kunshan' 'Quebec City' 'Palermo' 'Winnipeg'
'Orumiyeh' 'Eskisehir' 'Benguela' 'Jincheng' 'Heze' 'Saratov' 'Nellore'
'Huludao' 'Zanzibar' 'Barcelona Puerto La Cruz' 'Bikaner' 'Haicheng'
'Gebze' 'Taixing' 'Liaocheng' 'Zhumadian' 'Newcastle Upon Tyne'
'Langfang' 'Bucheon' 'Sulaimaniya' 'Xalapa' 'Malanje' 'Anqiu' 'Sorocaba'
'Gaomi' 'Dasmarinas' 'Cagayan De Oro City' 'Hanchuan' 'Meishan' 'Bologna'
'Ar Rayyan' 'Thessaloniki' 'Muzaffarnagar' 'Kayamkulam' 'Nottingham'

```
'Nakhon Ratchasima' 'Danyang' 'Ibb' 'Amravati' 'Jiaozuo' 'Vereeniging'
'Gorakhpur' 'Gaza' 'Frankfurt' 'Anqing' 'Niigata' 'Oshogbo' 'Linhai'
'Shaoguan' 'Erduosi Ordoss' 'Merca' "Bur Sa'Id" 'Kitwe' "Yan'An"
'Cuttack' 'San Francisco' 'Hamilton' 'Zaria' 'Banjarmasin' 'Dengzhou'
'Belgaum' 'Malegaon' 'Goma' 'Zigong' 'Qingyuan' 'Yuncheng' 'Shaoyang'
'Yanji' 'Tirupati' 'Maturin' 'Yuxi' 'Akure' 'Tongliao' 'Sialkot'
'Tongling' 'Krakow' 'Ansan' 'Wuzhou' 'Dazhou' 'Suining Sichuan'
'Mangalore' 'Jiamusi' 'Seattle' 'Al Hudaydah' 'Sargodha' 'Nay Pyi Taw'
'Tamale' 'Sao Jose Dos Campos' 'Bacoor' 'Dongtai' 'Zhangjiagang'
'Nanded Waghala' 'Xianyang Shaanxi' 'Amara' 'Zarqa' 'Bhavnagar'
'Sheffield' 'Huambo' 'Ribeirao Preto' 'Panzhihua']
```

[48]:
```python
# Standardize city names to lower case and strip whitespaces
weather_df['City'] = weather_df['City'].str.lower().str.strip()
population_df['city'] = population_df['city'].str.lower().str.strip()
meteorite_df['name'] = meteorite_df['name'].str.lower().str.strip()
```

[49]:
```python
# Find the common city names
common_cities = set(meteorite_df['name']).
 ↪intersection(set(population_df['city']))
print("Common city names in both datasets:")
print(common_cities)
```

```
Common city names in both datasets:
{'southampton', 'ottawa', 'jilin', 'tyumen', 'athens', 'fuyang', 'new york',
'lusaka', 'toulouse', 'glasgow', 'bursa', 'charlotte', 'chelyabinsk', 'porto
alegre', 'nantong', 'novosibirsk', 'dongtai', 'perth', 'changde', 'santa cruz',
'saratov', 'hiroshima', 'moradabad', 'queretaro', 'tirupati', 'delhi', 'cali',
'columbus', 'havana', 'leshan', 'jalandhar', 'meerut', 'krasnodar', 'palermo',
'johannesburg', 'ningbo', 'valencia', 'madrid', 'salem', 'aleppo', 'rosario'}
```

[50]:
```python
# Create a SQLite database connection
conn = sqlite3.connect('meteorite_landings.db')

# Save the meteorite landings data to the database
meteorite_df.to_sql('meteorite_landings', conn, if_exists='replace',␣
 ↪index=False)

# Save the weather data to the database
weather_df.to_sql('weather_data', conn, if_exists='replace', index=False)

# Save the population data to the database
population_df.to_sql('population_data', conn, if_exists='replace', index=False)

# Verify tables are created
print("Tables created in the database:", conn.execute("SELECT name FROM␣
 ↪sqlite_master WHERE type='table';").fetchall())
```

```
Tables created in the database: [('meteorite_landings',), ('weather_data',),
```

('population_data',)]

[51]:
```
# Inspect the weather_data table schema to verify column names
print("Weather Data Table Schema:")
print(conn.execute("PRAGMA table_info(weather_data);").fetchall())
```

```
Weather Data Table Schema:
[(0, 'City', 'TEXT', 0, None, 0), (1, 'Latitude', 'REAL', 0, None, 0), (2,
'Longitude', 'REAL', 0, None, 0), (3, 'Temperature', 'REAL', 0, None, 0), (4,
'Humidity', 'INTEGER', 0, None, 0), (5, 'WindSpeed', 'REAL', 0, None, 0), (6,
'WeatherDescription', 'TEXT', 0, None, 0)]
```

[52]:
```
# Inspect the population_data table schema to verify column names
print("Population Data Table Schema:")
print(conn.execute("PRAGMA table_info(population_data);").fetchall())
```

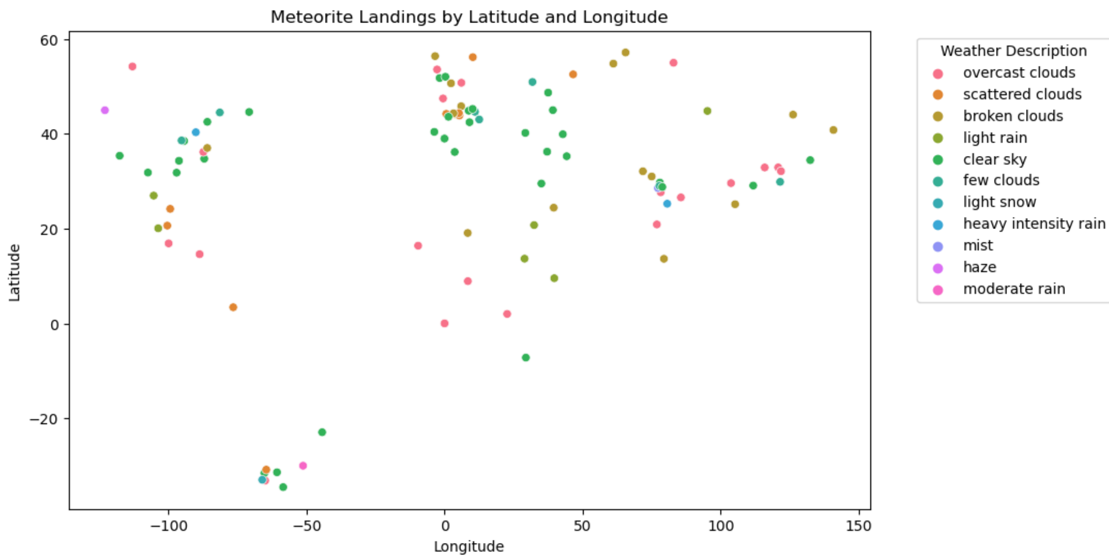
```
Population Data Table Schema:
[(0, 'rank', 'INTEGER', 0, None, 0), (1, 'city', 'TEXT', 0, None, 0), (2,
'country', 'TEXT', 0, None, 0), (3, '2024_population', 'INTEGER', 0, None, 0),
(4, '2023_population', 'INTEGER', 0, None, 0), (5, 'growth_rate', 'REAL', 0,
None, 0)]
```

[53]:
```
# Query to join the datasets
query = """
SELECT ml.name, ml.reclat, ml.reclong, ml.year, ml.fall, wd.Temperature, wd.
↪Humidity, wd.WindSpeed, wd.WeatherDescription, pd."2024_population" AS␣
↪Population
FROM meteorite_landings AS ml
LEFT JOIN weather_data AS wd ON ml.name = wd.City
LEFT JOIN population_data AS pd ON wd.City = pd.city
"""
merged_df = pd.read_sql_query(query, conn)

# Display the first few rows of the merged data
print("Merged Data:")
print(merged_df.head())

# Check for rows where population is not None
non_empty_population = merged_df[merged_df['Population'].notnull()]
print("Merged Data with Non-Empty Population:")
print(non_empty_population.head())
```

```
Merged Data:
       name    reclat    reclong  year  fall  Temperature  Humidity  \
0    aachen  50.77500    6.08333  1880  Fell       289.04      79.0
1    aarhus  56.18333   10.23333  1951  Fell       285.89      91.0
2      abee  54.21667 -113.00000  1952  Fell       288.55      62.0
3  acapulco  16.88333  -99.90000  1976  Fell       301.65      84.0
4   achiras -33.16667  -64.95000  1902  Fell       276.09      97.0
```

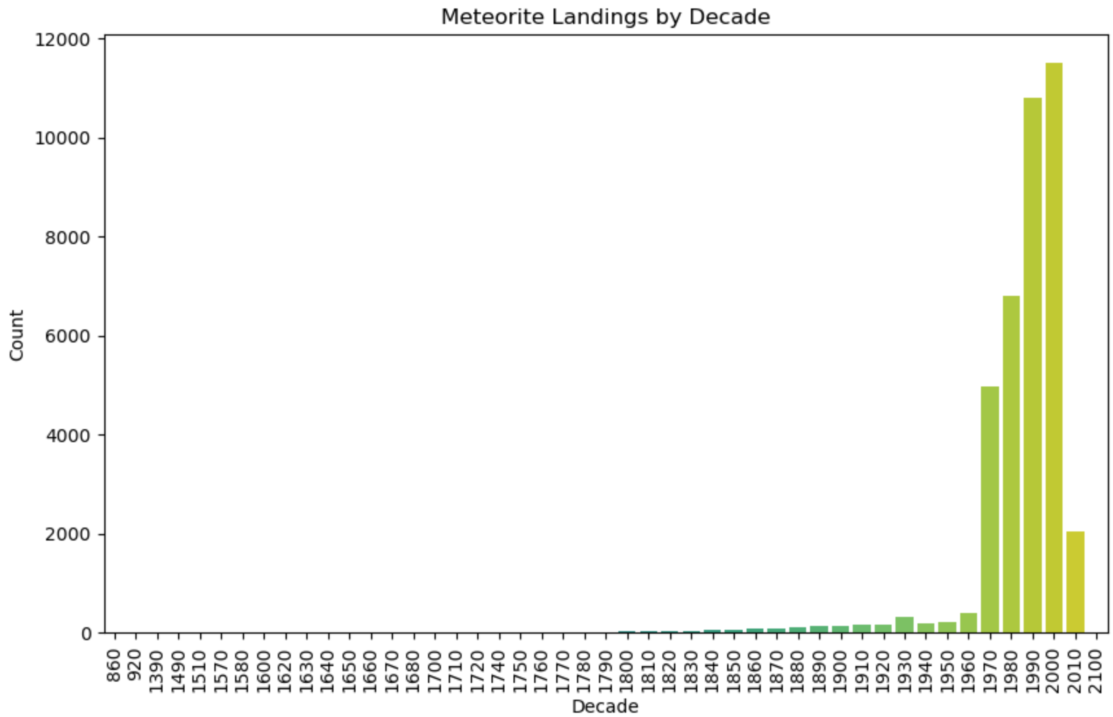
```
   WindSpeed WeatherDescription  Population
0       2.88    overcast clouds         NaN
1       2.47   scattered clouds         NaN
2       2.06    overcast clouds         NaN
3       4.73    overcast clouds         NaN
4       8.78    overcast clouds         NaN
Merged Data with Non-Empty Population:
       name    reclat   reclong  year  fall  Temperature  Humidity  WindSpeed  \
24   aleppo  36.23333  37.13333  1873  Fell       296.88      82.0       6.60
25   aleppo  36.23333  37.13333  1873  Fell       297.14      82.0       5.86
55   athens  34.75000 -87.00000  1933  Fell       298.54      81.0       3.09
56   athens  34.75000 -87.00000  1933  Fell       299.92      47.0       3.13
153   bursa  40.20000  29.23333  1946  Fell       285.17      66.0       1.78

    WeatherDescription  Population
24           clear sky   2317650.0
25           clear sky   2317650.0
55      broken clouds   3154591.0
56           clear sky   3154591.0
153          clear sky   2115513.0
```

[54]:
```
# Visualization 1: Scatter plot of meteorite landings by latitude and longitude
plt.figure(figsize=(10, 6))
sns.scatterplot(x='reclong', y='reclat', hue='WeatherDescription',␣
 ↪data=merged_df)
plt.title('Meteorite Landings by Latitude and Longitude')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend(title='Weather Description', bbox_to_anchor=(1.05, 1), loc='upper␣
 ↪left')
plt.show()
```



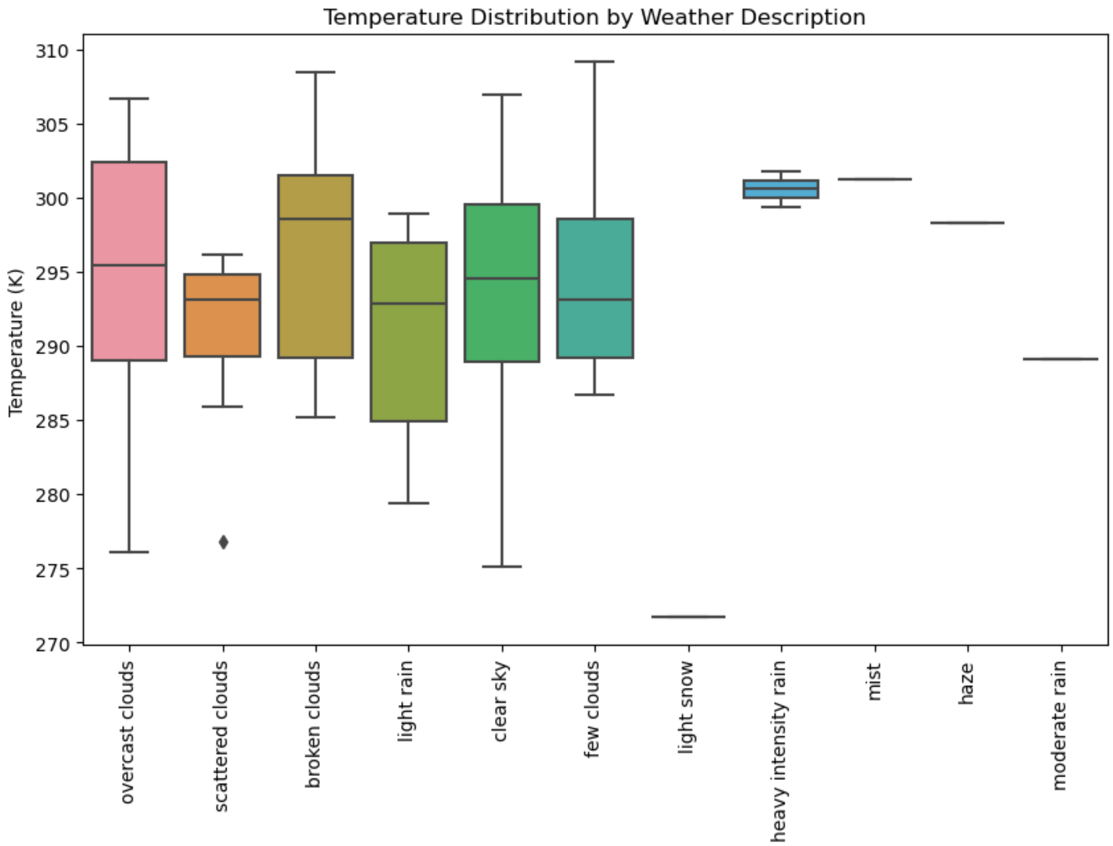
Description: This scatter plot shows the geographical distribution of meteorite landings based on their latitude and longitude. Each point represents a meteorite landing, and the points are color-coded by the weather description at the time of data retrieval. The plot provides insights into the spatial patterns of meteorite landings and the corresponding weather conditions at those locations.

[58]:
```
# Visualization 2: Bar plot of meteorite landings by decade
meteorite_df['decade'] = (meteorite_df['year'] // 10) * 10
plt.figure(figsize=(10, 6))
sns.countplot(x='decade', data=meteorite_df, palette='viridis')
plt.title('Meteorite Landings by Decade')
plt.xlabel('Decade')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```



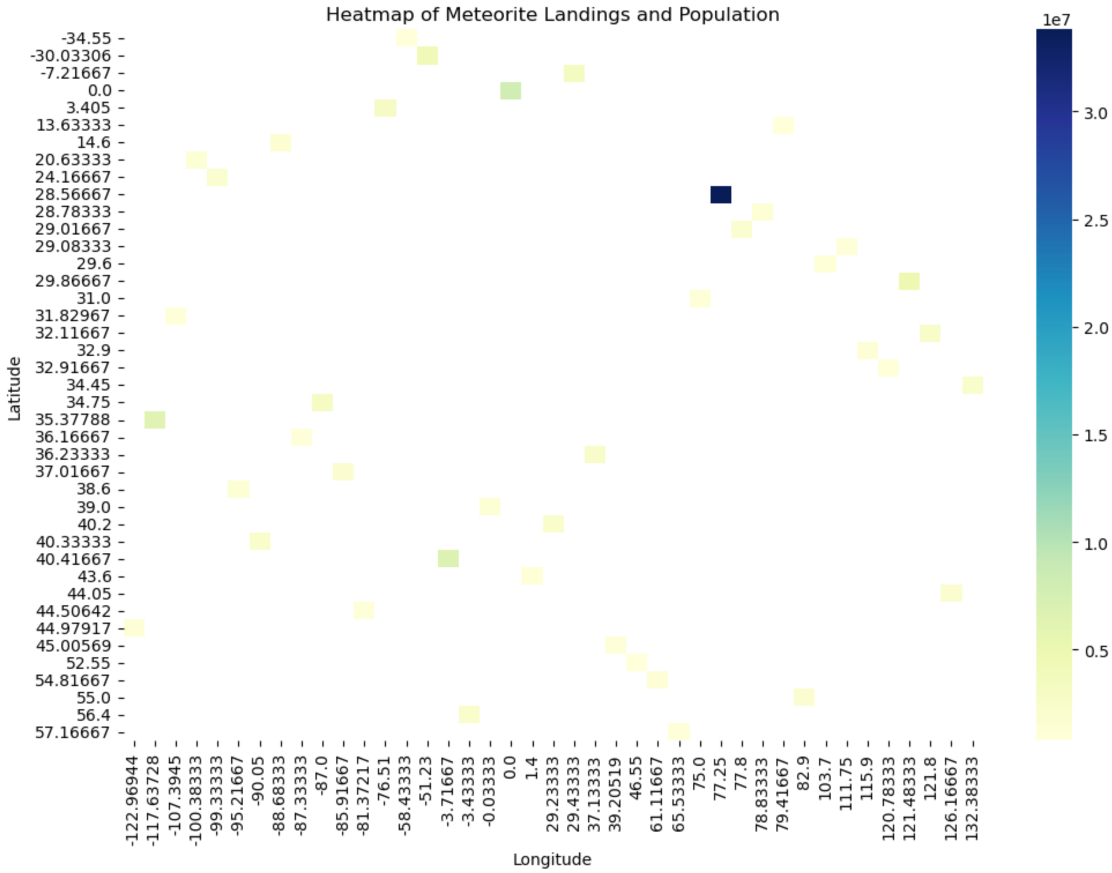
Description: This bar plot aggregates the meteorite landings by decade, providing a clear view of the trends over time. The x-axis represents the decades, and the y-axis represents the count of meteorite landings in each decade. The plot highlights the increase in reported meteorite landings over recent decades, possibly due to improved detection and reporting mechanisms.

[56]:
```
# Visualization 3: Box plot of temperatures by weather description
plt.figure(figsize=(10, 6))
sns.boxplot(x='WeatherDescription', y='Temperature', data=merged_df)
plt.title('Temperature Distribution by Weather Description')
plt.xlabel('Weather Description')
plt.ylabel('Temperature (K)')
plt.xticks(rotation=90)
plt.show()
```



Description: This box plot displays the distribution of temperatures for different weather descriptions. The x-axis represents the weather descriptions, and the y-axis represents the temperature in Kelvin. The plot shows the median, quartiles, and potential outliers for each weather description category, providing insights into how temperature varies with different weather conditions.

[57]:
```
# Visualization 4: Heatmap of meteorite landings and population
heatmap_data = merged_df.pivot_table(values='Population', index='reclat',␣
↪columns='reclong')
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, cmap='YlGnBu')
plt.title('Heatmap of Meteorite Landings and Population')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



Description: This heatmap visualizes the relationship between the geographical locations of meteorite landings and the population density at those locations. The x-axis represents the longitude, and the y-axis represents the latitude. The color intensity indicates the population, with darker shades representing higher populations. The heatmap helps identify regions with higher populations that have experienced meteorite landings, offering potential insights into the impact on populated areas.

0.8 Summary of Project and Ethical Implications

What I Learned: This project involved extensive data wrangling, merging, and visualization to analyze meteorite landings, weather data, and population information. The primary objective was to integrate these diverse datasets into a cohesive and meaningful analysis, which included several key steps:

1. Data Cleaning and Transformation:

- Meteorite landing data was cleaned to standardize column names, handle missing values, and ensure consistency.
- Weather data was fetched from the OpenWeatherMap API based on the geographical coordinates of the meteorite landings.

- Population data was sourced from the World Population Review and cleaned for consistency in city names and other attributes.

2. Data Integration:

- All datasets were loaded into a SQLite database and merged based on common attributes such as geographical coordinates and city names.
- Special attention was given to ensure accurate merging by standardizing city names to lower case and stripping white spaces.

3. Visualization:

- Various visualizations were created to uncover patterns and insights, including scatter plots, bar plots, box plots, and heatmaps.
- These visualizations highlighted the distribution of meteorite landings over time, the relationship between landing locations and weather conditions, and the impact on populated areas.

Ethical Implications:

Changes Made to the Data: Several transformations were performed, including standardizing column names, converting temperature units, and handling missing data. These changes improved data quality but also altered the original dataset.

Legal or Regulatory Guidelines: The data used in this project, sourced from the OpenWeatherMap API and the World Population Review, required adherence to their respective usage policies. Compliance with data privacy regulations was crucial, especially when dealing with location-based data.

Risks Created Based on Transformations: The primary risk was potential misinterpretation of the data due to transformations. For instance, incorrect conversion of temperature units or inaccurate merging of datasets could lead to erroneous conclusions. Additionally, reliance on API data, which might not always be up-to-date, posed a risk of inaccuracies.

Assumptions in Cleaning/Transforming the Data:

- Assumed that temperature values from the API were in Kelvin.
- Assumed that city names accurately reflected the corresponding weather and population data.
- Assumed that the population data matched the city names provided by the weather data.

Data Sourcing and Verification for Credibility: Data was sourced from reputable APIs and websites, ensuring high credibility. Cross-referencing data from multiple sources helped verify accuracy.

Ethical Acquisition of Data: Data was ethically acquired from publicly accessible APIs and websites, with full adherence to their usage policies.

Mitigation of Ethical Implications:

- Documented all transformations and assumptions made during data cleaning to ensure transparency.
- Regularly verified data accuracy and consistency to avoid misinformation.
- Ensured compliance with data usage policies and privacy regulations to protect sensitive information.
- Communicated potential limitations and risks associated with data transformations to stakeholders.

By following these steps and considerations, the data wrangling process maintained ethical standards while supporting the project's goals.

[ ]:
```

```