Pawan Chaudhary

1. Working with pandas

In [1]:

```
import pandas as pd
print(f"Panda's library version: {pd.__version__}")
```

```
Panda's library version: 2.2.3
```

2. Using Series Data Structures

In [8]:

```
# creating a list with 6 population numbers representing different city discticts
# to be precise, i have used the data from 6 metropolitan cities with highest popul
# reference: https://en.wikipedia.org/wiki/List_of_cities_in_Nepal

population = [862400, 513504, 369268, 294098, 272382, 243927]
print(f"Population numbers: {population}")
```

```
Population numbers: [862400, 513504, 369268, 294098, 272382, 243927]
```

In [5]:

```
# using the same reference as above, i have a list with respective district names

district = ["Kathmandu", "Pokhara", "Bharatpur", "Lalitpur", "Birgunj", "Biratnagar
print(f"District names: {district}")
```

```
District names: ['Kathmandu', 'Pokhara', 'Bharatpur', 'Lalitpur', 'Birgunj', 'Biratn
agar']
```

In [ ]:

```
# creating a pandas sereis and assigning the index

district_series = pd.Series(data = population, index = district, name = "Population
print(f"{district_series}")
```

```
Kathmandu     862400
Pokhara       513504
Bharatpur     369268
Lalitpur      294098
Birgunj       272382
Biratnagar    243927
Name: Population Series, dtype: int64
```

In [16]:

```
# calculating mean population across all districts
mean_population = district_series.mean()
print(f"Mean population across all districts: {mean_population}")
```

```
Mean population across all districts: 425929.8333333333
```

In [18]:

```
# identifying maximum populations
max_population = district_series.idxmax()
print(f"District with maximum population: {max_population}")
```

```
District with maximum population: Kathmandu
```

In [20]:

```
# identifying minimum populations
min_population = district_series.idxmin()
print(f"District with minimum population: {min_population}")
```

```
District with minimum population: Biratnagar
```

3. Using DataFrame Data Structures

In [35]:

```
# creating a dictionary with five nepali city parks, including their area in acres
parks_data = {
    "park_name": ["Garden of Dreams", "Rani Pokhari", "Ratna Park", "Godavari Botan
    "area_acres": [3.0, 1.5, 5.0, 82.0, 2.0],
    "annual_visitors": [250000, 200000, 500000, 150000, 100000]
}

print(f"City Parks: {parks_data}")
```

```
City Parks: {'park_name': ['Garden of Dreams', 'Rani Pokhari', 'Ratna Park', 'Godava
ri Botanical Garden', 'Sahid Gate Park'], 'area_acres': [3.0, 1.5, 5.0, 82.0, 2.0],
'annual_visitors': [250000, 200000, 500000, 150000, 100000]}
```

In [43]:

```
# converting dictionary to pandas dataframe
parks_df = pd.DataFrame(parks_data)
```

In [44]:

```
# set park name as the index
parks_df = parks_df.set_index("park_name")
```

In [48]:

```
print(f"Park DataFrame: \n{parks_df}")
```

```
Park DataFrame: 
                           area_acres  annual_visitors
park_name                                             
Garden of Dreams                  3.0           250000
Rani Pokhari                      1.5           200000
Ratna Park                        5.0           500000
Godavari Botanical Garden        82.0           150000
Sahid Gate Park                   2.0           100000
```

In [51]:

```
# the first three entries of the dataframe
print(f"First three parks: \n {parks_df.head(3)}")
```

```
First three parks: 
                   area_acres  annual_visitors
park_name                                    
Garden of Dreams         3.0           250000
Rani Pokhari             1.5           200000
Ratna Park               5.0           500000
```

In [ ]:

```
# total number of visitors of the park
total_visitors = parks_df["annual_visitors"].sum()
print(f"Total visitors of the park: {total_visitors}")
```

```
Total visitors of the park: 1200000
```

In [56]:

```
# average number of visitors of the park
average_visitors = parks_df["annual_visitors"].mean()
```

```
print(f"Mean visitors of the park: {average_visitors}")
```

```
Mean visitors of the park: 240000.0
```

In [58]:

```
# largest area and its details
largest_park = parks_df["area_acres"].idxmax()
print(f"Largest park in terms of area: {largest_park}")
print(parks_df.loc[largest_park])
```

```
Largest park in terms of area: Godavari Botanical Garden
area_acres              82.0
annual_visitors     150000.0
Name: Godavari Botanical Garden, dtype: float64
```

In [59]:

```
# parks with more 1,000,000 visitors
popular_parks = parks_df[parks_df["annual_visitors"] > 1_000_000]
print("\nParks with more than 1,000,000 visitors:\n", popular_parks)
```

```
Parks with more than 1,000,000 visitors:
 Empty DataFrame
Columns: [area_acres, annual_visitors]
Index: []
```

In [ ]:

```
# increasing the size of one of the parks by 10 areas and updating dataframe

parks_df.loc["Garden of Dreams", "area_acres"] += 10
print(f"Updated 'Garden of Dreams' area: {parks_df.loc["Garden of Dreams", "area_ac
```

```
Updated 'Garden of Dreams' area: 23.0
```

4. Combining Datasets

In [63]:

```
facilities_data = {
    "park_name": ["Garden of Dreams", "Rani Pokhari", "Ratna Park", "Godavari Botan
    "playgrounds": [2, 5, 1, 3, 8],
    "sports_facilities": [4, 10, 2, 6, 12]
}

print(f"Updated Dataframe with no. of school playground and sports facilities \n {f
```

```
Updated Dataframe with no. of school playground and sports facilities
 {'park_name': ['Garden of Dreams', 'Rani Pokhari', 'Ratna Park', 'Godavari Botanica
l Garden', 'Sahid Gate Park'], 'playgrounds': [2, 5, 1, 3, 8], 'sports_facilities':
[4, 10, 2, 6, 12]}
```

In [65]:

```
facilities_df = pd.DataFrame(facilities_data).set_index("park_name")
```

In [71]:

```
# merging using index (join)

parks_merged = parks_df.join(facilities_df)
print(f"Merged parks DataFrame with facilities:\n {parks_merged}")
```

```
Merged parks DataFrame with facilities:
                            area_acres  annual_visitors  playgrounds  \
park_name                                                              
Garden of Dreams                  23.0           250000            2   
Rani Pokhari                       1.5           200000            5   
Ratna Park                         5.0           500000            1   
Godavari Botanical Garden         82.0           150000            3   
Sahid Gate Park                    2.0           100000            8   

                           sports_facilities  
park_name                                     
Garden of Dreams                           4  
Rani Pokhari                              10  
Ratna Park                                 2  
Godavari Botanical Garden                  6  
Sahid Gate Park                           12  
```

5. Data Retrieval

In [74]:

```
column_using_indexing = parks_merged["area_acres"]
print(f"Area column (using indexing operator): {column_using_indexing}")
```

```
Area column (using indexing operator): park_name
Garden of Dreams             23.0
Rani Pokhari                  1.5
Ratna Park                    5.0
Godavari Botanical Garden    82.0
Sahid Gate Park               2.0
Name: area_acres, dtype: float64
```

In [77]:

```
column_using_iloc = parks_merged["annual_visitors"]
print(f"Annual visitors column (using iloc): {column_using_iloc}")
```

```
Annual visitors column (using iloc): park_name
Garden of Dreams             250000
Rani Pokhari                 200000
Ratna Park                   500000
Godavari Botanical Garden    150000
Sahid Gate Park              100000
Name: annual_visitors, dtype: int64
```

In [78]:

```
column_using_loc = parks_merged["sports_facilities"]
print(f"Sports facilities column (using loc): {column_using_loc}")
```

```
Sports facilities column (using loc): park_name
Garden of Dreams              4
Rani Pokhari                 10
Ratna Park                    2
Godavari Botanical Garden     6
Sahid Gate Park              12
Name: sports_facilities, dtype: int64
```

In [80]:

```
# iloc uses integer positions

rows_iloc = parks_merged.iloc[[2, 4]]
print(f"Rows 2 and 4 via iloc:\n {rows_iloc}")
```

```
Rows 2 and 4 via iloc:
                  area_acres  annual_visitors  playgrounds  sports_facilities
park_name
Ratna Park               5.0           500000            1                  2
Sahid Gate Park          2.0           100000            8                 12
```

In [81]:
```
# loc uses index labels

rows_loc = parks_merged.loc[[parks_merged.index[0], parks_merged.index[1]]]
print(f"Rows 0 and 1 via loc:\n {rows_loc}")
```

```
Rows 0 and 1 via loc:
                   area_acres  annual_visitors  playgrounds  sports_facilities
park_name
Garden of Dreams         23.0           250000            2                  4
Rani Pokhari              1.5           200000            5                 10
```

6. Understanding your Data (Bonus)

In [83]:
```
# random seed for reproducibility
import numpy as np
np.random.seed(60)
```

In [85]:
```
customer_ID = np.arange(1001, 1011)
print(f"Customer ID: {customer_ID}")
```

```
Customer ID: [1001 1002 1003 1004 1005 1006 1007 1008 1009 1010]
```

In [86]:
```
names = ["Aarav", "Priya", "Rahul", "Anika", "Kabir", "Sanya", "Rohan", "Isha", "Sa
print(f"Customer Names: {names}")
```

```
Customer Names: ['Aarav', 'Priya', 'Rahul', 'Anika', 'Kabir', 'Sanya', 'Rohan', 'Ish
a', 'Sameer', 'Naina']
```

In [87]:
```
ages = np.random.randint(18, 66, size=10)
print(f"Customer ages: {ages}")
```

```
Customer ages: [31 19 24 28 33 53 26 35 52 51]
```

In [88]:
```
total_spending = np.round(np.random.uniform(100, 5000, size=10), 2)
print(f"Total spending: {total_spending}")
```

```
Total spending: [2698.95 4662.01 3551.47 4484.18 4328.43  997.96 2691.32 4876.66 207
6.42
 2263.59]
```

In [90]:
```
# creating a dataa frame for these fictional customers

customers_df = pd.DataFrame({
    "customer_ID" : customer_ID,
    "name" : names,
    "age" : ages,
    "total_spending" : total_spending
})

print(f"Customer DataFrame: \n{customers_df}")
```

```
Customer DataFrame:
   customer_ID    name  age  total_spending
0         1001   Aarav   31         2698.95
1         1002   Priya   19         4662.01
2         1003   Rahul   24         3551.47
3         1004   Anika   28         4484.18
4         1005   Kabir   33         4328.43
5         1006   Sanya   53          997.96
6         1007   Rohan   26         2691.32
7         1008    Isha   35         4876.66
8         1009  Sameer   52         2076.42
9         1010   Naina   51         2263.59
```

In [91]:

```
# calculating correlation in the dataframe

customers_corr = customers_df[["age", "total_spending"]].corr()
print(f"Correlation (Age vs Total Spending):\n", customers_corr)
```

```
Correlation (Age vs Total Spending):
                      age  total_spending
age             1.000000       -0.720228
total_spending -0.720228        1.000000
```

The correlation analysis between age and total spending shows a strong negative relationship, meaning that as age increases, customers tend to spend less overall. This suggests younger customers are likely to spend more compared to older ones.