

✓ Data Acquisition Case Study

- ✓ Q1. Write Python code to create a new file named "sample_data.txt" in your documents folder and write the following content to it

ICTAK

Thejaswini,

Technopark Rd,

Technopark Campus,

Thiruvananthapuram,

Kerala 695581

```
file_path = "sample_data.txt"
```

```
content = """ICTAK
```

```
Thejaswini,
```

```
Technopark Rd,
```

```
Technopark Campus,
```

```
Thiruvananthapuram,
```

```
Kerala 695581"""
```

```
with open(file_path, "w") as file:  
    file.write(content)
```

```
print(f"File created at: {file_path}")
```

🔄 File created at: sample_data.txt

- ✓ Q2. Write Python code to read and print the contents in "sample_data.txt"

```
with open("sample_data.txt", "r") as file:  
    content = file.read()
```

```
print("Contents of sample_data.txt:\n")  
print(content)
```

🔄 Contents of sample_data.txt:

ICTAK

Thejaswini,

Technopark Rd,

Technopark Campus,

Thiruvananthapuram,

Kerala 695581


- ✓ Q3. Write Python code to check if "sample_data.txt" exists in documents folder

```
if os.path.exists("sample_data.txt"):
```

```

    print("sample_data.txt exists in the current directory.")
else:
    print("sample_data.txt does NOT exist in the current directory.")

```

 sample_data.txt exists in the current directory.

✓ Q4: Save the following dataframe content to a CSV file (data.csv) in your downloads folder

```

import pandas as pd

data = {"Id": [1, 2, 3],
        "Name": ["Alice", "Bob", "Charlie"],
        "Subject": ["Science", "Maths", "History"]}
df = pd.DataFrame(data)

import pandas as pd
import os

data = {
    "Id": [1, 2, 3],
    "Name": ["Alice", "Bob", "Charlie"],
    "Subject": ["Science", "Maths", "History"]
}

df = pd.DataFrame(data)

downloads_folder = os.path.join(os.path.expanduser("~"), "Downloads")
os.makedirs(downloads_folder, exist_ok=True)

file_path = os.path.join(downloads_folder, "data.csv")

df.to_csv(file_path, index=False)

print(f"data.csv successfully saved to: {file_path}")

```

 data.csv successfully saved to: /root/Downloads/data.csv

✓ Q5: Save the above dataframe content to an Excel (data.xlsx, sheet name: Sheet1) file in your documents folder

```

import pandas as pd
import os

data = {
    "Id": [1, 2, 3],
    "Name": ["Alice", "Bob", "Charlie"],
    "Subject": ["Science", "Maths", "History"]
}
df = pd.DataFrame(data)

r
documents_folder = os.path.join(os.path.expanduser("~"), "Documents")
os.makedirs(documents_folder, exist_ok=True)

file_path = os.path.join(documents_folder, "data.xlsx")

df.to_excel(file_path, index=False, sheet_name="Sheet1")

```

```
print(f"Excel file successfully saved to: {file_path}")
```

→ Excel file successfully saved to: /root/Documents/data.xlsx

Q6. Write code to get the list of files in your Downloads folder and save it to a CSV file name "download_list.csv"

```
import os
import pandas as pd
```

```
downloads_folder = os.path.join(os.path.expanduser("~"), "Downloads")
```

```
file_list = [f for f in os.listdir(downloads_folder) if os.path.isfile(os.path.join(downloads_folder, f))]
```

```
df = pd.DataFrame(file_list, columns=["File Name"])
```

```
csv_path = os.path.join(downloads_folder, "download_list.csv")
df.to_csv(csv_path, index=False)
```

```
print(f"List of files saved to: {csv_path}")
```

→ List of files saved to: /root/Downloads/download_list.csv

Q7. Write Python code to save the contents of the given random_array variable as a numpy file

```
import numpy as np
random_array = np.random.rand(10, 10)
```

```
import numpy as np
```

```
random_array = np.random.rand(10, 10)
```

```
np.save("random_array.npy", random_array)
```

```
print("random_array.npy file has been saved successfully.")
```

→ random_array.npy file has been saved successfully.

Q8. Write python code to save the contents of the above numpy file as text file named "random.txt" with a delimiter of ";" to Documents folder

```
import numpy as np
import os
```

```
random_array = np.load("random_array.npy")
```

```
documents_folder = os.path.join(os.path.expanduser("~"), "Documents")
os.makedirs(documents_folder, exist_ok=True)
```

```
text_file_path = os.path.join(documents_folder, "random.txt")
```

```
np.savetxt(text_file_path, random_array, delimiter=";", fmt="%.4f")
```

```
print(f"Array successfully saved to: {text_file_path}")
```

→ Array successfully saved to: /root/Documents/random.txt

Download and analyze Bike Sharing Dataset (hour.csv) for UCI Irvin Repository

- ✓ (<https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset>) and answer the following questions

```
import urllib.request
import zipfile

url = "http://archive.ics.uci.edu/ml/machine-learning-databases/00275/Bike-Sharing-Dataset.zip"
filename = "Bike-Sharing-Dataset.zip"

urllib.request.urlretrieve(url, filename)

with zipfile.ZipFile(filename, 'r') as zip_ref:
    zip_ref.extractall()

print("Dataset downloaded and extracted.")
```

↗ Dataset downloaded and extracted.

```
import os

os.listdir()
```

↗

```
['.config',
 'random_array.npy',
 'day.csv',
 'sample_data.txt',
 'Bike-Sharing-Dataset.zip',
 'Readme.txt',
 'hour.csv',
 'sample_data']
```

- ✓ Q9. What is the size of the dataset? (Number of rows and columns)

```
import pandas as pd

df = pd.read_csv("hour.csv")

rows, columns = df.shape
print(f"Number of rows: {rows}")
print(f"Number of columns: {columns}")

df.head()
```

↗

```
Number of rows: 17379
Number of columns: 17
```

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	16
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32

Next steps:

[Generate code with df](#)

[View recommended plots](#)

[New interactive sheet](#)

- ✓ Q10. What are the data types of each column?

df.dtypes

```

instant    int64
dteday     object
season     int64
yr         int64
mnth       int64
hr         int64
holiday    int64
weekday    int64
workingday int64
weathersit  int64
temp       float64
atemp      float64
hum        float64
windspeed  float64
casual     int64
registered int64
cnt        int64

```

df.dtypes

Q11. Are there any missing values in the dataset? If so, which columns have missing values and how many?

```

missing_values = df.isnull().sum()

missing_values = missing_values[missing_values > 0]

if missing_values.empty:
    print(" There are no missing values in the dataset.")
else:
    print(" Missing values found:\n")
    print(missing_values)

```

✓ There are no missing values in the dataset.

Q.12. For the windspeed column, calculate the mean, median, and standard deviation.

```

# Mean
mean_wind = df['windspeed'].mean()

# Median
median_wind = df['windspeed'].median()

# Standard Deviation
std_wind = df['windspeed'].std()

# Display results
print(f"Mean windspeed      : {mean_wind:.4f}")
print(f"Median windspeed     : {median_wind:.4f}")
print(f"Standard deviation      : {std_wind:.4f}")

```

↗ Mean windspeed : 0.1901
Median windspeed : 0.1940
Standard deviation : 0.1223

Q13. Identify any potential outliers in a numerical column of your choice. Explain your approach.

```
# Step 1: Select the column
col = 'windspeed'

# Step 2-4: Calculate Q1, Q3, IQR, and bounds
Q1 = df[col].quantile(0.25)
Q3 = df[col].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Step 5: Identify outliers
outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]

print(f"Total outliers in '{col}': {len(outliers)}")
print(f"Lower bound: {lower_bound:.4f}, Upper bound: {upper_bound:.4f}")
```

↗ Total outliers in 'windspeed': 342
Lower bound: -0.1193, Upper bound: 0.4775

Q.14 Find the correlation between numerical columns and discuss any interesting relationships.

```
correlation_matrix = df.corr(numeric_only=True)
```

correlation_matrix


	instant	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum
instant	1.000000	0.404046	0.866014	0.489164	-0.004775	0.014723	0.001357	-0.003416	-0.014198	0.136178	0.137615	0.009577
season	0.404046	1.000000	-0.010742	0.830386	-0.006117	-0.009585	-0.002335	0.013743	-0.014524	0.312025	0.319380	0.150625
yr	0.866014	-0.010742	1.000000	-0.010473	-0.003867	0.006692	-0.004485	-0.002196	-0.019157	0.040913	0.039222	-0.083546
mnth	0.489164	0.830386	-0.010473	1.000000	-0.005772	0.018430	0.010400	-0.003477	0.005400	0.201691	0.208096	0.164411
hr	-0.004775	-0.006117	-0.003867	-0.005772	1.000000	0.000479	-0.003498	0.002285	-0.020203	0.137603	0.133750	-0.276498
holiday	0.014723	-0.009585	0.006692	0.018430	0.000479	1.000000	-0.102088	-0.252471	-0.017036	-0.027340	-0.030973	-0.010588
weekday	0.001357	-0.002335	-0.004485	0.010400	-0.003498	-0.102088	1.000000	0.035955	0.003311	-0.001795	-0.008821	-0.037158
workingday	-0.003416	0.013743	-0.002196	-0.003477	0.002285	-0.252471	0.035955	1.000000	0.044672	0.055390	0.054667	0.015688
weathersit	-0.014198	-0.014524	-0.019157	0.005400	-0.020203	-0.017036	0.003311	0.044672	1.000000	-0.102640	-0.105563	0.418130
temp	0.136178	0.312025	0.040913	0.201691	0.137603	-0.027340	-0.001795	0.055390	-0.102640	1.000000	0.987672	-0.069881
atemp	0.137615	0.319380	0.039222	0.208096	0.133750	-0.030973	-0.008821	0.054667	-0.105563	0.987672	1.000000	-0.051918
hum	0.009577	0.150625	-0.083546	0.164411	-0.276498	-0.010588	-0.037158	0.015688	0.418130	-0.069881	-0.051918	1.000000
windspeed	-0.074505	-0.149773	-0.008740	-0.135386	0.137252	0.003988	0.011502	-0.011830	0.026226	-0.023125	-0.062336	-0.290105
casual	0.158295	0.120206	0.142779	0.068457	0.301202	0.031564	0.032721	-0.300942	-0.152628	0.459616	0.454080	-0.347028
registered	0.282046	0.174226	0.253684	0.122273	0.374141	-0.047345	0.021578	0.134326	-0.120966	0.335361	0.332559	-0.273933
cnt	0.278379	0.178056	0.250495	0.120638	0.394071	-0.030927	0.026900	0.030284	-0.142426	0.404772	0.400929	-0.322911

Next steps: [Generate code with correlation_matrix](#) [View recommended plots](#) [New interactive sheet](#)

✓ Q.15 Based on your analysis, provide a brief summary of any insights or patterns you discovered in the dataset.

1. Dataset Overview Total records: 17,379

Columns: 17

No missing values were found in the dataset —  data is clean.

2. Weather & Environment Effects Temperature (temp/atemp) has a moderate positive correlation with total bike usage (cnt). Warmer weather → more rentals.

Humidity (hum) and windspeed both show a slight negative correlation — likely because high humidity and wind make biking less comfortable.

3. User Type Influence Registered users (registered) have a very strong positive correlation (~0.97) with total usage — they are the main contributors to rental count.

Casual users correlate moderately (~0.53) — meaning weekend or occasional users contribute, but far less than subscribers.

4. Hourly and Temporal Trends (from earlier rows) Rentals are lower at night (e.g., hour = 0–5), start increasing in the morning, and peak around commuting hours (8–9 AM and 5–6 PM).

You can further visualize this with a line plot grouped by hour to confirm patterns.

5. Outliers Detected outliers in columns like windspeed using the IQR method — they should be investigated or cleaned for modeling tasks.

Final Thoughts: Time (hour/month/weekday) and user type are critical dimensions for understanding and forecasting bike usage.

Weather features affect behavior but to a lesser extent.


This dataset is well-suited for building predictive models for bike demand using regression or time-series methods.

✓ Q.16 In which season (Spring, Summer, Fall, Winter) people rented bikes the most?

```
# Map numeric season to names
season_map = {1: 'Spring', 2: 'Summer', 3: 'Fall', 4: 'Winter'}
df['season_name'] = df['season'].map(season_map)

# Group by season and sum the total bike rentals
season_rentals = df.groupby('season_name')['cnt'].sum().sort_values(ascending=False)

# Display results
print("Total bike rentals by season:\n")
print(season_rentals)
```

 Total bike rentals by season:


```
season_name
Fall      1061129
Summer    918589
Winter    841613
Spring    471348
Name: cnt, dtype: int64
```

✓ Q.17 What is the peak hour in which bike rents the most?

```
# Group by hour and sum total bike rentals
hourly_rentals = df.groupby('hr')['cnt'].sum()

# Find the hour with the maximum rentals
peak_hour = hourly_rentals.idxmax()
peak_rentals = hourly_rentals.max()

print(f"Peak hour: {peak_hour}:00 with {peak_rentals} rentals")
```

 Peak hour: 17:00 with 336860 rentals

✓ Q.18 In which day of a week bikes rents out most?

```
# Map weekday numbers to names
weekday_map = {0: 'Sunday', 1: 'Monday', 2: 'Tuesday', 3: 'Wednesday',
               4: 'Thursday', 5: 'Friday', 6: 'Saturday'}

df['weekday_name'] = df['weekday'].map(weekday_map)

# Sum bike rentals by weekday
weekday_rentals = df.groupby('weekday_name')['cnt'].sum()

# Sort by rental counts descending to find the day with most rentals
weekday_rentals = weekday_rentals.sort_values(ascending=False)

print("Total bike rentals by day of the week:\n")
print(weekday_rentals)
print(f"\nDay with most bike rentals: {weekday_rentals.idxmax()}")
```

➦ Total bike rentals by day of the week:

```
weekday_name
Friday      487790
Thursday    485395
Saturday    477807
Wednesday   473048
Tuesday     469109
Monday      455503
Sunday      444027
Name: cnt, dtype: int64
```

Day with most bike rentals: Friday

✓ Q.19 In which hour Casual users rents bikes the most?

```
# Group by hour and sum casual user rentals
casual_hourly = df.groupby('hr')['casual'].sum()

# Find the hour with maximum casual rentals
peak_casual_hour = casual_hourly.idxmax()
peak_casual_rentals = casual_hourly.max()

print(f"Peak hour for casual users: {peak_casual_hour}:00 with {peak_casual_rentals} rentals")
```

➦ Peak hour for casual users: 14:00 with 55089 rentals

✓ Q.20 What is the maximum temperature observed in each of the seasons?

```
# Using the 'season_name' column we created earlier

# Find max temperature for each season
max_temp_by_season = df.groupby('season_name')['temp'].max()

print("Maximum normalized temperature observed in each season:")
print(max_temp_by_season)
```

➦ Maximum normalized temperature observed in each season:

```
season_name
Fall      1.00
Spring    0.72
Summer    0.94
Winter    0.76
Name: temp, dtype: float64
```