HOW DOES LONDON WEATHER AFFECT THE LONDON BIKE HIRE PATTERN?

1.Student Details :

- **Khushi Natraj Bijkal - A00049400**
  khushi.natrajbijkal2@mail.dcu.ie

- **Mrinal Vattiprolu - A00049723**
  mrinal.vattiprolu2@mail.dcu.ie

2. Dataset Link:

- London Weather(1979 to 2023) :
  https://www.kaggle.com/code/zongaobian/london-weather-trends-and-forecasts-1979-2023

- TFI - daily cycle hire(2010 to 2025):
  https://www.kaggle.com/datasets/belayethossainds/tfl-daily-cycle-hires-data

- Source Code :
  https://gitlab.computing.dcu.ie/khushi.natrajbijkal2/spark-bike-hire-weather-pipeline

3. Video Demonstration: Final Cloud video.mp4

https://drive.google.com/file/d/1FUcIdMAVOovSxzSONlMmMpqXIqdRkGt5/view

4. Motivation:

In major metropolitan cities like London and other urban areas, transportation has transformed into a data - driven source. Understanding the civilian transportation patterns helps in better development of the city's infrastructure, reduces congestion and also encourages sustainable travel. Transport for London (TFL) provides a sustainable way for people to travel around London, and the cycle hire data gives useful information about public travel behaviour. However, the number of people using the bikes is strongly affected by environmental factors, especially weather conditions.

This Project aims to build an end-to-end data processing pipeline to analyse how **Weather affects the TFL cycle hire** in the city of London. We integrated both the datasets and performed structured analysis using **Apache Spark**.

About the Pipeline:-

The pipeline that we developed covers skills such as extracting data, cleaning and transforming the data, joining the extracted data, and visualizing it. It is designed as a step-by-step workflow in Spark, where output of one stage becomes the input for the next stage. After joining, we also create extra features like year, month, weekend flag, rain flag, and temperature bands to make the analysis easier.

Objectives:

The primary objectives of this study is:

- To consider & visualise how Temperature and other weather factors such as sunshine, humidity, wind and rain influence public bike usage patterns.
- To perform Data ingestion, cleansing, transformation, and joining.
- To discover behavioural patterns and relationships between the different features in the datasets.
- To show how Spark can handle large real-world datasets efficiently through distributed processing.
- To calculate simple statistical measures (like correlation) to support the visual results.
- Creating a visualization to know more and analyze it further.

5. Related work

1: Weather effects on London's cycle hire usage

Van Lieshout and Strijkstra studied the London Barclays Cycle Hire / Santander Cycles system to measure how temperature, rainfall, humidity, and wind speed will influence the bikeshare demand. Using London hire records combined with meteorological data, they found that higher temperatures increase hires, while rainfall, high humidity, and stronger winds reduce usage, showing that weather-aware forecasting helps operators rebalance bikes and plan capacity during adverse conditions. [1]

2: London bike-share demand and weather with time-series modeling

Morton et al. examined long-term demand trends in the London Bicycle Sharing Scheme using autoregressive distributed lag (time-series) models with weather and temporal controls. Their results show that rainfall and colder periods cause significant demand drops, while warmer and clearer conditions raise hire levels, even after controlling for day type and system growth. The study also highlights strong seasonality and different weekday-versus-weekend responses to weather. [2]

3: Multi - city nonlinear meteorology impacts on bike sharing

Villarrasa-Sapiña et al. extended weather–bikeshare analysis to 13 European and North American cities, applying nonlinear methods (self-organizing maps and decision trees) rather than only linear correlation. They found consistent patterns across cities: warm, dry conditions drive higher demand, while precipitation and uncomfortable extremes suppress trips. They also show threshold effects, where demand rises with temperature up to a comfort range and then levels off. [3]

How our solution differ from these works?

- The related works mainly focus on Statistical or ML modeling of the datasets. In our project, we included an **end to end pipeline in Apache Spark** covering ingestion -> cleaning -> transformation -> joining -> feature engineering -> visualization, showing the whole data-engineering workflow, and not just analysis.

- **Our pipeline is reproducible** through a structured code pipeline. It can be rerun on updated data or moved to platforms like Databricks / AWS. All steps are implemented in Spark and can scale to larger datasets.

- We use **daily integrated data across a long time window**. Some related studies (like the work of van Lieshout & Strijkstra) [1] use only shorter windows or station-level subsets.

- We focus on **simple, explainable behavioural insights**, rather than using complex nonlinear models like decision trees / SOMs [3]. We prioritise clear correlations and visual trends (e.g., temperature bands, rainy vs non-rainy hire differences). This makes results easier to interpret for real transport planning.

- We used **extra feature engineering for better interpretation**, by creating useful features like is_weekend, dayofweek, rain_flag, temp_band, feels_like, which helps highlight behaviour patterns beyond raw weather variables.

6. Dataset Description:

We selected two datasets to perform this study, one is **“TFL Cycle Hire Data”** and the other is **“London Weather Data”**. Both are open-source datasets that were hosted on Kaggle [4][5].

1. **Source of the Datasets:**

- **TFL Cycle Hire Data:** The ownership of this dataset is held by the Transport for London (TFL) platform and was mirrored on Kaggle for easier access. This dataset provides the number of cycles hired all over London daily. It contains **4081 rows** and **2 columns** (date and number of hires), and helps in capturing the usage of the city's bike sharing scheme over a period of time from 2010 to 2025. [4]

- **London Weather Data:** This dataset gives the underlying data, which includes features such as temperature, precipitation, wind, and humidity. While the original ownership of this dataset is unknown, the Kaggle notebook serves as the source reference for this dataset. It holds the weather data recorded over multiple decades, starting from 1979 to 2023. It consists of **16436 rows** and **21 columns.** [5]

Since the weather dataset ends in 2023 while cycle hires extend to 2025, this study analyses only the overlapping period 2010 - 2023. Dates outside this overlap were excluded during joining.

a) Data Extraction process:

- **Weather Dataset :**

  This dataset contains weather data over multiple decades from **1979 to 2023**, and it was downloaded in CSV format.

```
In [4]:
# Load raw weather CSV into Spark
weather_raw = (
    spark.read
    .option("header", True)
    .option("inferSchema", True)
    .csv(weather_path)
)

print("Weather rows:", weather_raw.count())
weather_raw.printSchema()
weather_raw.show(5)
```

```
Weather rows: 16436
root
 |-- DATE: integer (nullable = true)
 |-- TX: double (nullable = true)
 |-- Q_TX: integer (nullable = true)
```

  Using **Spark's CSV data ingestion**, we loaded this Weather data into the environment using the above commands.

- **TFL Cycle Dataset:**

This TFL Cycle dataset was originally hosted by the Transport for London website, but it was also mirrored by a Kaggle notebook and was downloaded in CSV format

It gives the data on the number of cycles hired daily from 2010 to 2025. The dataset was loaded into the Apache Spark environment using the following command:

```
In [8]:
# Load TfL daily cycle hires using Spark only
bikes_raw = (
    spark.read
    .option("header", True)
    .option("inferSchema", True)
    .csv(bikes_path)
)

print("Raw TfL schema:")
bikes_raw.printSchema()
bikes_raw.show(5)
```

There were **4081 Rows** of data and **2 columns,** i.e, **date** and **number** of cycle hires.

7. Data Processing:

In the Above steps, we have seen that both the data sets have been loaded into the Spark environment. Now that our datasets have been loaded, we will next identify null values, and handle them for each dataset.

- The command below was used to find the null values for the weather dataset: (link to the code file - https://gitlab.computing.dcu.ie/khushi.natrajbijkal2/spark-bike-hire-weather-pipeline)

```
In [5]:
from pyspark.sql.functions import col, sum as spark_sum

weather_nulls = weather_raw.select([
    spark_sum(col(c).isNull().cast("int")).alias(c + "_nulls")
    for c in weather_raw.columns
])

weather_nulls.show()
```

```
+----------+--------+----------+--------+----------+--------+----------+--------+----------+--------+----------+--------+---
-------+--------+----------+--------+----------+--------+----------+--------+----------+
|DATE_nulls|TX_nulls|Q_TX_nulls|TN_nulls|Q_TN_nulls|TG_nulls|Q_TG_nulls|SS_nulls|Q_SS_nulls|SD_nulls|Q_SD_nulls|RR_nulls|Q_R
R_nulls|QQ_nulls|Q_QQ_nulls|PP_nulls|Q_PP_nulls|HU_nulls|Q_HU_nulls|CC_nulls|Q_CC_nulls|
+----------+--------+----------+--------+----------+--------+----------+--------+----------+--------+----------+--------+---
-------+--------+----------+--------+----------+--------+----------+--------+----------+
|         0|       0|         0|       0|         0|      29|         0|       0|         0|    1075|         0|       0|
0|      25|         0|       4|         0|      57|         0|      18|         0|
+----------+--------+----------+--------+----------+--------+----------+--------+----------+--------+----------+--------+---
-------+--------+----------+--------+----------+--------+----------+--------+----------+
```

We found that there are **1204 null values** in the weather dataset. So, in the next step, we will handle these null / missing values.

- **Handling Null Values:** The command below was used for basic cleaning and unit conversion, such as converting the Date column into a proper date format, converting weather units, and assigning clear column names to the selected features needed for our study.

In [6]:
```
# Basic cleaning & unit conversion
weather_clean = (
    weather_raw
    .withColumn("date", F.to_date(F.col("DATE").cast("string"), "yyyyMMdd"))
    .withColumn("tmax_c",  F.col("TX") / 10.0)
    .withColumn("tmin_c",  F.col("TN") / 10.0)
    .withColumn("tmean_c", F.col("TG") / 10.0)
    .withColumn("rain_mm", F.col("RR") / 10.0)
    .withColumn("sun_hrs", F.col("SS") / 10.0)
    .select("date", "tmax_c", "tmin_c", "tmean_c", "rain_mm", "sun_hrs", "HU")
    .filter("date IS NOT NULL")
)
```

- Now, we impute the missing values in rainfall and sunshine with 0, and the humidity with the **mean**. This lets us handle missing data without dropping rows, which could reduce the dataset size and affect the analysis.

```
# Impute missing rainfall & sunshine with 0, and humidity with mean
avg_hu = weather_clean.select(F.mean("HU")).first()[0]

weather_clean = weather_clean.fillna({
    "rain_mm": 0.0,
    "sun_hrs": 0.0,
    "HU": avg_hu
})
weather_clean = weather_clean.filter("tmean_c IS NOT NULL")

# Advanced feature: feels-like temperature (simple approximation)
weather_clean = weather_clean.withColumn(
    "feels_like",
    (F.col("tmax_c") + F.col("tmin_c")) / 2 + 0.1 * F.col("HU")
)

weather_clean.printSchema()
weather_clean.show(5)
```

- After removing outliers and imputing the missing values (rainfall and sunshine with 0, and humidity with the mean), there are **no null values left** in the weather dataset. We used the below command to verify and show that there are 0 null values left.

In [7]:
```
weather_nulls_after = weather_clean.select([
    F.sum(F.col(c).isNull().cast("int")).alias(c + "_nulls")
    for c in weather_clean.columns
])

weather_nulls_after.show()
```

```
+----------+-----------+-----------+------------+------------+------------+--------+---------------+
|date_nulls|tmax_c_nulls|tmin_c_nulls|tmean_c_nulls|rain_mm_nulls|sun_hrs_nulls|HU_nulls|feels_like_nulls|
+----------+-----------+-----------+------------+------------+------------+--------+---------------+
|         0|          0|          0|           0|           0|           0|       0|              0|
+----------+-----------+-----------+------------+------------+------------+--------+---------------+
```

- We repeated the similar process for the **Cycle dataset.** First we standardised the date column to the same format as the weather dataset so both datasets could be joined correctly.

- Then, we cleaned the cycle hire data by handling missing values and removing outliers using a **4 x standard deviation filtering rule** (keeping values within mean ± 4 x std)

```
# Clean and prepare
bikes_clean = (
    bikes_raw
    .withColumn("date", F.to_date("Day", "dd/MM/yyyy"))
    .withColumn("hires", F.col("Number of Bicycle Hires"))
    .select("date", "hires")
    .filter("date IS NOT NULL AND hires IS NOT NULL AND hires > 0")
)

print("Bike rows before outlier removal:", bikes_clean.count())

# Outlier removal using 4*std rule
stats = bikes_clean.select(mean("hires").alias("mean"), stddev("hires").alias("std")).first()
mean_val, std_val = stats["mean"], stats["std"]

bikes_clean = bikes_clean.filter(
    (F.col("hires") > mean_val - 4*std_val) &
    (F.col("hires") < mean_val + 4*std_val)
)

print("Bike rows after outlier removal:", bikes_clean.count())
bikes_clean.show(5)
```

- After this cleaning step, the dataset was ready for joining and analysis.

```
Raw TfL schema:
root
 |-- Day: string (nullable = true)
 |-- Number of Bicycle Hires: double (nullable = true)
 |-- _c2: string (nullable = true)

+----------+----------------------+----+
|       Day|Number of Bicycle Hires| _c2|
+----------+----------------------+----+
|30/07/2010|                6897.0|NULL|
|31/07/2010|                5564.0|NULL|
|01/08/2010|                4303.0|NULL|
|02/08/2010|                6642.0|NULL|
|03/08/2010|                7966.0|NULL|
+----------+----------------------+----+
only showing top 5 rows

Bike rows before outlier removal: 4081
Bike rows after outlier removal: 4078
+----------+------+
|      date| hires|
+----------+------+
|2010-07-30|6897.0|
|2010-07-31|5564.0|
|2010-08-01|4303.0|
|2010-08-02|6642.0|
|2010-08-03|7966.0|
+----------+------+
only showing top 5 rows
```

- Now that we have cleaned the datasets separately, we have joined the datasets into one single dataset.

- After joining the two cleaned datasets and selecting the required features for the study, the **total rows joined was 4078.**

8. Development of Pipeline:

The pipeline was designed with a modular and scalable architecture, consisting of 3 phases:

1. Data Extraction
2. Data processing
3. Data visualization

Data Extraction Tools:

The first step in loading the datasets into the Spark Dataframe is using the **.csv()** reader with header recognition and automatic schema inference. It consists of key features such as **.option("header", "true")**, which recognises column names automatically, **.option("inferschema"," true")** helps in detecting data types automatically, and these functions help in the process of loading the dataset. The second step in the processing is finding the null values and handling them, which is carried out using the **.na.drop() / .fill()**. The **to_date()** helps in converting a string date to the proper date type.

The extraction tool Spark CSV reader allows quick ingestion of large CSV files, which makes it scalable for future automation and API ingestion.

Data Processing Tool:

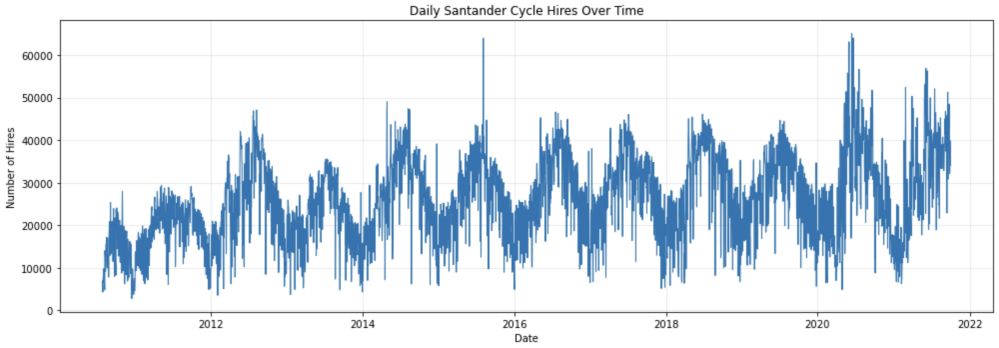
The major step involved in the data processing is data transformation. All the data transformation tasks were carried out by using PySpark SQL function and feature engineering tools . The first step in the transformation is data cleaning and formatting which involves finding null values in both the datasets separately and handling them. In our case we have removed the outliers and null values by imputing them with 0 and mean. The second step is joining both the datasets into one single dataset using the **.join()** function. The third step is feature engineering, we have selected the features from the combined dataset which are needed for our study.

The features that we selected were: -
date, tmax_c, tmin_c, tmean_c, rain_mm, sun_hrs, HU, feels_like, hires, year, month, dayofweek, is_weekend, rain_flag, temp_band.

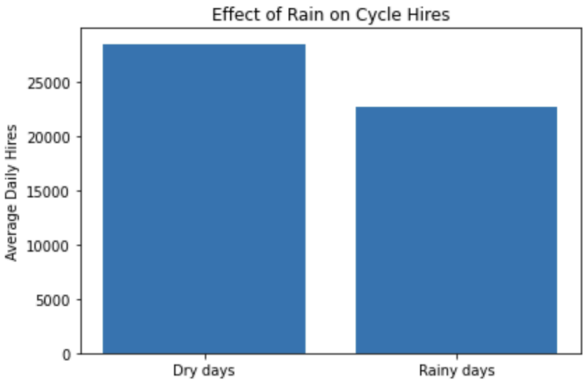
Data Visualization tools:

In the final step we involved development of the pipeline, and to visualize the insights we had gathered. We generated visual trends or insights using **Matplotlib** and **Spark SQL outputs**. These insights aid us in studying if the weather had an influence on the cycle hire in London.

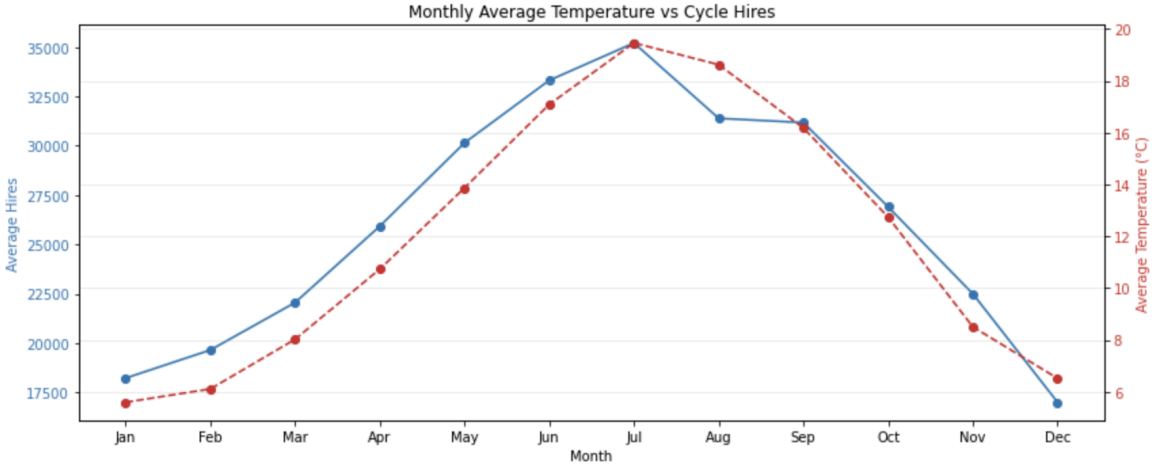
- The line graph below shows the number of cycles hired over the years from 2010 to 2022



- The line chart below shows how the rain affected the number of cycles hired



- The chart below shows the temperature and number of cycles hired over the period of 12 months simultaneously which can help us understand that higher the temperature higher the number of cycles hired.



- We can clearly see that in the month of Dec - Feb, the temperature is low and the number of cycles hired is also low, and as the temp increased gradually, the cycle hires also increased along with it, marking July as the highest temperature and highest cycles hired, and both temperature and cycle hires gradually decreased. This tells us that we can see that the weather has an influence on the cycle hire in the city of London.

- Along with the visual trends, we calculated correlations to measure the strength of the relationship between weather and daily cycle hires.

    ➔ Mean temperature shows a strong positive correlation with hires (**r = 0.648**), meaning warmer days generally lead to more bike usage.

    ➔ Sunshine hours also have a positive relationship (**r = 0.532**), supporting the idea that clearer days increase cycling.

    ➔ In contrast, rainfall is negatively correlated (**r = −0.246**), showing fewer hires on rainy days.

    ➔ Humidity also shows a moderate negative correlation (**r = −0.534**), while the “feels like” temperature is positively correlated with hires (**r = 0.568**).

These results support what we observed in the graphs: pleasant and dry conditions increase cycle hire demand, while wet or uncomfortable weather reduces usage.

9. Challenges & Lessons learned:

In the entire process of development of the pipeline, we have faced several technical challenges. Overcoming these issues helped me gain experience in deploying data-processing workflows and understand how Spark behaves with real-world datasets. The challenges we faced are listed below:

- Different Date formats: Both the datasets have different date formats, so we used the to_date() and date_format() functions to standardise date columns before joining

- Missing values and outliers: The weather dataset had null values in temperature and outliers in rainfall values. We used **.na.drop()** and **.na.fill()** to handle null values and impute them with the mean and value 0.

- Spark is column-oriented, so all operations should be written as a transformation on columns, so thinking in terms of SQL helps a lot.

- Feature engineering improves analysis by adding columns, such as is_weekend and feels_like, which makes it easier to understand the behaviour analysis.

- The current pipeline is made in such a way that it can be easily adapted by other cloud platforms like Databricks and AWS, which makes it scalable.

Overall, the pipeline helped us demonstrate how a high volume of real-world data can be cleaned, processed, and analyzed to gain insights efficiently using Spark, while preparing the foundation for Machine Learning techniques.

10. References:

[1] R. van Lieshout en J. Strijkstra. (2015). *The influence of weather conditions on the usage of the Barclays Cycle Hire: Contributing towards balancing London's bicycle sharing system*. Available:https://www.eur.nl/sites/corporate/files/400376_RolfvanLieshout_385196_JelmerStrijkstra_finalpaper.pdf

[2] C. Morton, "The demand for cycle sharing: Examining the links between weather conditions, air quality levels, and cycling demand for regular and casual users," *Journal of Transport Geography*, vol. 88, 102854, 2020, doi: 10.1016/j.jtrangeo.2020.102854. Available: https://www.sciencedirect.com/science/article/pii/S0966692319306167

[3] I. Villarrasa-Sapiña, J.-L. Toca-Herrera, M. Pellicer-Chenoll, K. Taczanowska, P. Rueda, and J. Devís-Devís, "Effects of meteorology on bike-sharing: Cases of 13 cities using non-linear analyses," *Cities*, 2024, doi: 10.1016/j.cities.2024.105457. Available: https://www.sciencedirect.com/science/article/pii/S0264275124006711

[4] B. Hossain, "TFL Daily Cycle Hires Data (2010 - 2025)," Kaggle dataset, 2025. Available: https://www.kaggle.com/datasets/belayethossainds/tfl-daily-cycle-hires-data

[5] Z. Bian, "London Weather Data From 1979 to 2023," Kaggle dataset, 2023. Available: https://www.kaggle.com/code/zongaobian/london-weather-trends-and-forecasts-1979-2023

[6] Apache Spark, "CSV Files - Spark SQL and DataFrames," *Spark 4.0.1 Documentation*, 2025. Available: https://spark.apache.org/docs/latest/sql-data-sources-csv.html

[7] Matplotlib Development Team, "Matplotlib Documentation," Matplotlib, 2025. Available: https://matplotlib.org/stable/contents.html