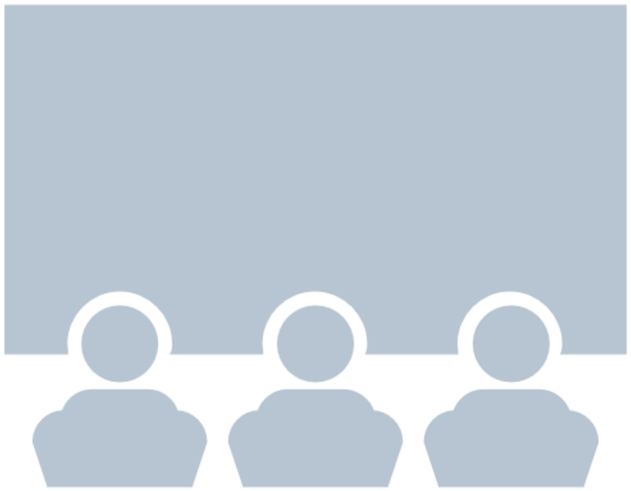
# Applied Data Science with R Capstone project

## (Bike-Rental Demand Prediction)

LEARNER'S NAME: MEHWISH YOUNUS SHAIKH

DATE OF SUBMISSION: MAY 4, 2023

# Outline



- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

# Executive Summary

In this project, I aimed to predict the demand for bike rentals based on weather conditions using the Seoul Bike Sharing Demand dataset. I built a linear regression model using various algorithms such as lm, stepwise, and glmnet. After comparing the results of these models, I found that the glmnet model performed the best, with a root mean squared error (RMSE) of under 300 and an R-squared value of around 80%. I achieved this by incorporating polynomial and interaction terms into the formula used for the model. My findings suggest that weather conditions can be a useful predictor for bike rental demand and the glmnet model can be used to accurately predict the number of bikes required each hour of the day.

# Introduction

- The availability and accessibility of rental bikes is essential for many cities worldwide, and it is vital to ensure a reliable supply of bikes while minimizing program costs.
- To achieve this, it is necessary to predict the number of bikes required each hour, based on current conditions, such as the weather.
- The Seoul Bike Sharing Demand Data Set was designed to aid in this process by providing hourly bike rental data and weather information (Temperature, Humidity, Wind speed, Visibility, Dew point, Solar radiation, Snowfall, Rainfall).
- In this project, we build a linear regression model to predict the number of bikes rented each hour, based on weather data, providing a valuable tool for bike rental companies and city planners to optimize the supply of rental bikes and reduce costs.

# Methodology

- Perform data collection
- Perform data wrangling
- Perform exploratory data analysis (EDA) using SQL and visualization
- Perform predictive analysis using regression models
  - How to build the baseline model
  - How to improve the baseline model
- Build a R Shiny dashboard app

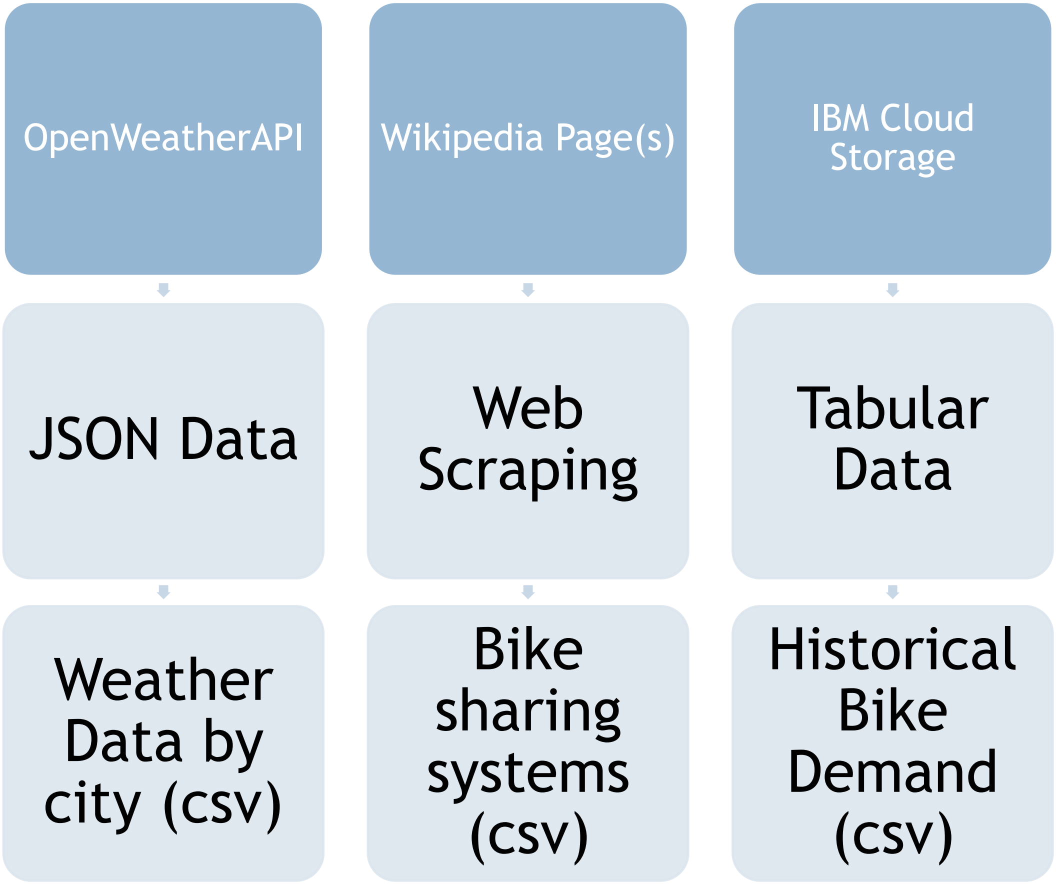
# 1 Methodology

This section of the presentation covers in details the methods applied to standardize, normalize and process data.

# Data collection

- To collect data for bike sharing predictions various sources were used:
- Firstly, weather and city related data is obtained from OpenWeatherAPI HTTP calls in JSON format.
- Another crucial source in data collection were Wikipedia pages. The bike rental system data was scraped from a Wikipedia page and saved in csv format.
- Finally, some relevant data was also extracted from cloud storage.

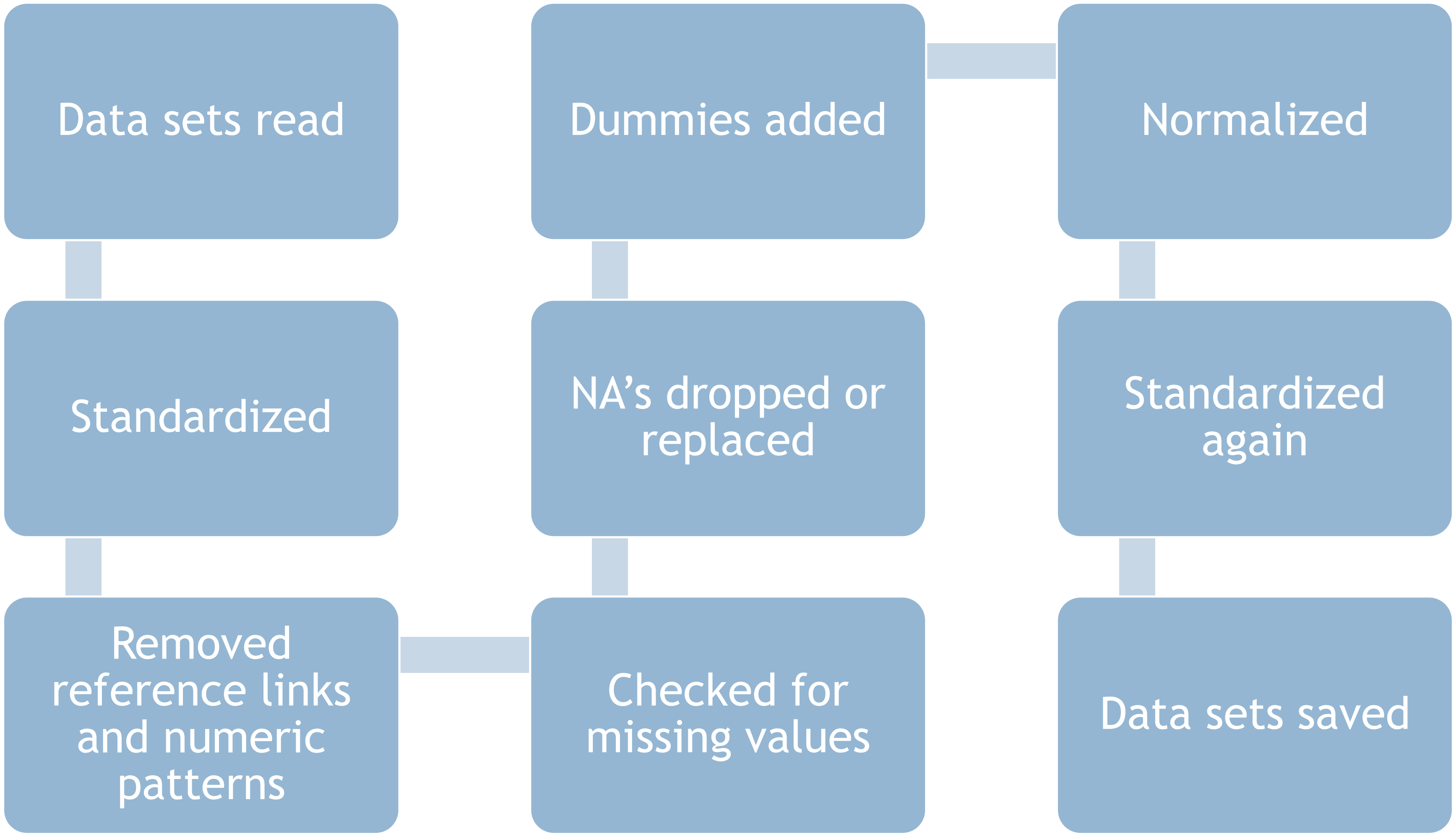
Data Collection Flowchart:



# Data wrangling

- To increase human readability data sets needed to be standardized therefore:
- All column names were converted to Uppercase.
- And text was separated using underscores.
- Regular expressions were used to remove unnecessary and redundant data (numeric patterns) from columns that was imported when scraping Wikipedia page(s).
- Columns containing missing values(NA's) were dropped and where required replaced with average value.
- Dummies were added to categorical variables for better processing.
- Data was normalized using min/max technique as columns with large values may adversely influence (bias) the predictive models and degrade model accuracy.
- Finally, column names were standardized again.

# Data wrangling



# EDA with SQL

Following queries were performed on datasets:

- Record Count: Determine how many records are in the Seoul Bike Sharing dataset.
- Operational Hours: Determine how many hours had non-zero rented bike count.
- Weather Outlook: Query the weather forecast for Seoul over the next 3 hours.
- Seasons: Find which seasons are included in the Seoul Bike Sharing dataset.
- Date Range: Find the first and last dates in the Seoul Bike Sharing dataset.
- Subquery - 'all-time high': Determine which date and hour had the most bike rentals.
- Hourly popularity and temperature by season: Determine the average hourly temperature and the average number of bike rentals per hour over each season. List the top ten results by average bike count.
- Rental Seasonality: Find the average hourly bike count during each season.

# EDA with SQL

- Weather Seasonality: Consider the weather over each season. On average, what were the temperature, humidity, wind speed, visibility, dew point temperature, solar radiation, rainfall, and snowfall per season?
- Total Bike Count and City Info for Seoul.
- Find all city names and coordinates with comparable bike scale to Seoul's bike sharing system.

Note: All query(s) result are listed in the Appendix section at the end of this presentation.

# EDA with data visualization

- Create a scatter plot of RENTED_BIKE_COUNT vs DATE.
- Create the same plot of the RENTED_BIKE_COUNT time series, but add HOURS as the color.
- Create a histogram overlaid with a kernel density curve to visualize the distribution of the data.
- Use a scatter plot to visualize the correlation between RENTED_BIKE_COUNT and TEMPERATURE by SEASONS.
- Create a display of four boxplots of RENTED_BIKE_COUNT vs. HOUR grouped by SEASONS.
- Use the summarize() function to calculate the daily total rainfall and snowfall and plot it as a grouped bar chart.

# Predictive analysis

- Model building:

1. Split data into training and testing sets
2. Trained and evaluated different regression models (e.g. multi-linear, lasso, polynomial, interaction)
3. Selected glmnet model as best performing model based on RMSE and R-squared metrics
4. Used polynomial and interaction terms in the glmnet model formula to improve performance
5. Model evaluation and improvement:

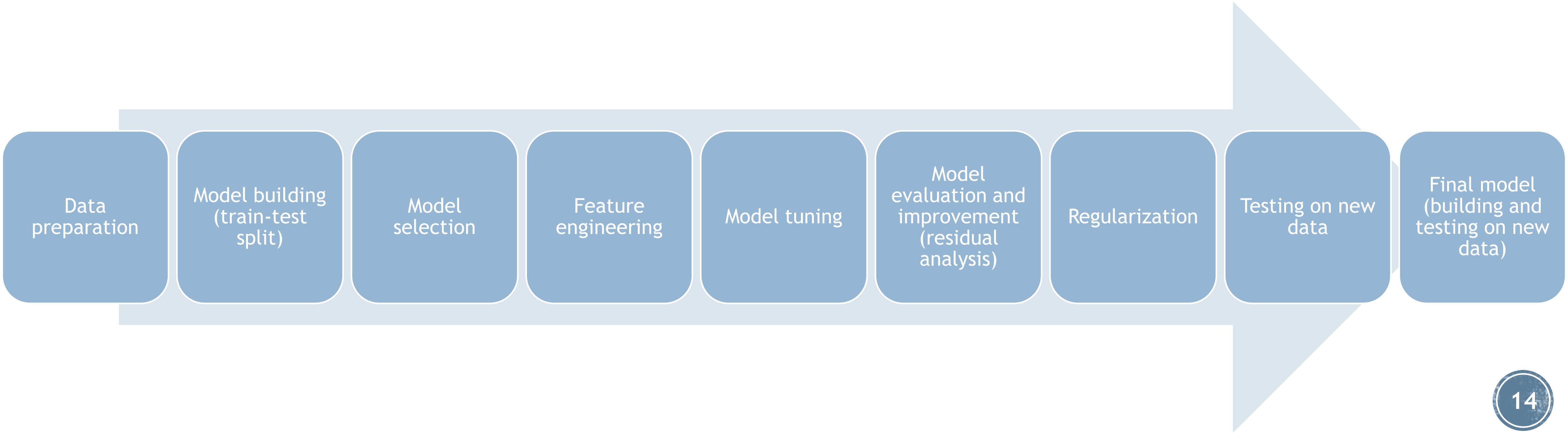
- Assessed model assumptions and residual plots

1. Made adjustments to the model formula to improve performance
2. Used regularization techniques to prevent overfitting
3. Tested model on new data to ensure generalizability

# Predictive analysis

- Final model:

1. Built final glmnet model using all available data
2. Tested final model on new data
3. Obtained RMSE under 300 and R-squared around 80%



# Build a R Shiny dashboard

- The ShinyApp shows bike rental demand for five cities: Paris, London, Suzhou, Seoul, and New York
- The cities can be selected from a dropdown menu in the side-panel
- A map is displayed in the main panel using Leaflet library, with yellow, green, and red circles representing higher, lower, and lowest demand predictions, respectively
- Clicking on a circle displays a popup showing the city's current weather
- Line charts are plotted to show the rental predictions according to hourly-temperature trend and rental bike count prediction according to date and time
- Exact bike rental count can be obtained by clicking on the points on the graph
- A graph is plotted to show whether humidity affects rental demand or not
- Clicking on the city's marker displays a detailed pop-up label.

# Results

- Exploratory data analysis results:

Exploratory data analysis showed that bike rental demand varies significantly across different cities, with Seoul having the highest demand and Suzhou having the lowest demand. Hourly bike rental demand also showed clear seasonal patterns, with demand being higher during the warmer months and during peak hours.

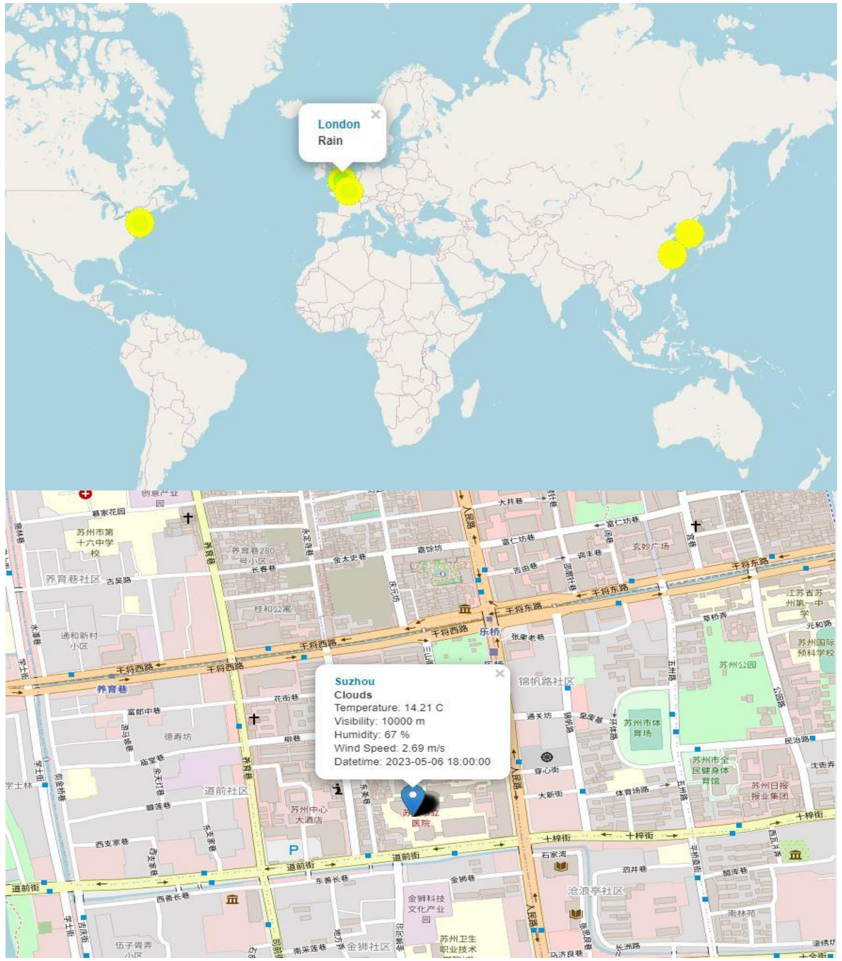
- Predictive analysis results:

Predictive analysis using machine learning models showed that the glmnet model with polynomial and interaction terms was the best performing model with an RMSE of under 300 and R-squared of around 80%.

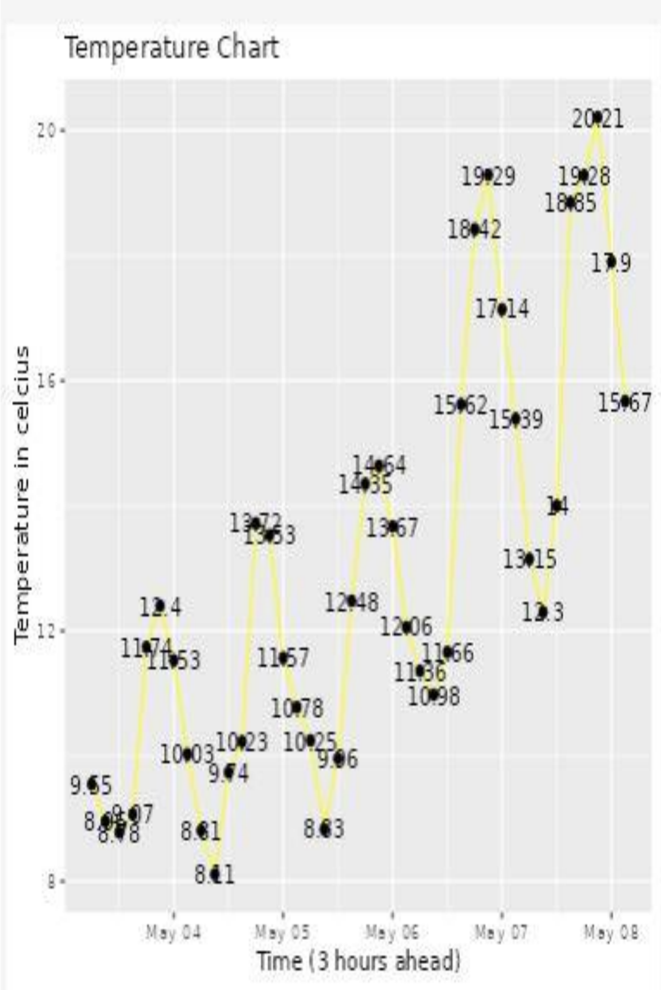
# Results

## A dashboard demo in screenshots

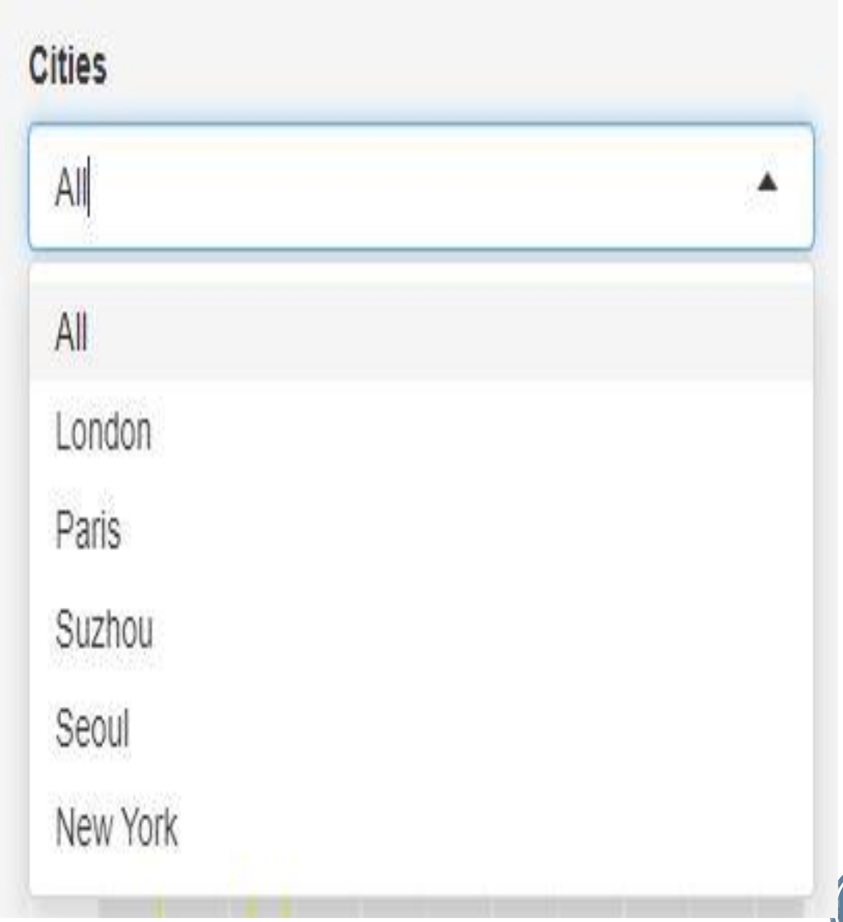
Main Panel




Side Panel




Dropdown



2

# EDA with SQL

This section of the presentation covers exploratory data analysis that was performed on structured data obtained from IBM cloud storage.

# Busiest bike rental times

```
view<-sqlQuery(conn, "SELECT DATE, HOUR, RENTED_BIKE_COUNT FROM
SEOUL_BIKE_SHARING WHERE RENTED_BIKE_COUNT=
(SELECT MAX(RENTED_BIKE_COUNT) FROM SEOUL_BIKE_SHARING)")
view
```

- The busiest bike rental hour was 18:00.
- Followed by 19:00, 20:00, 21:00 and lastly 17:00.

# Hourly popularity and temperature by seasons

```
view<-sqlQuery(conn, "SELECT SEASONS, HOUR, AVG(TEMPERATURE) AS AVG_HOURLY_TEMP, AVG(RENTED_BIKE_COUNT) AS AVG_BIKE_RENTALS FROM SEOUL_BIKE_SHARING GROUP BY SEASONS, HOUR ORDER BY AVG(RENTED_BIKE_COUNT) DESC LIMIT 10")
View
```

- The seasons that stood out to be most popular for bike rentals were Summer and Autumn.
- Summer had average bike rentals up to 2135 and the most popular hour for bike rentals was 18:00 followed by 19:00 and 20:00.
- What is to note here is that as the temperature rises, bike rentals increases and it is most likely that bike rentals occur in evening time.

# Rental Seasonality

- view<-sqlQuery(conn, "SELECT SEASONS, AVG(RENTED_BIKE_COUNT) AS HOURLY_BIKE_COUNT, MIN(RENTED_BIKE_COUNT) AS MINIMUM, MAX(RENTED_BIKE_COUNT) AS MAXIMUM, STDDEV(RENTED_BIKE_COUNT) AS STANDARD_DEV FROM SEOUL_BIKE_SHARING GROUP BY SEASONS, HOUR")
- view
- According to seasons, Summer had the highest bike rentals with hourly bike rental count around eight hundreds, closely followed by Autumn and then Spring.
- And Winter season had the lowest bike rentals around hundreds.

# Weather Seasonality

```
view<-sqlQuery(conn, "SELECT SEASONS,
                     AVG(TEMPERATURE) AS AVG_TEMP,
                     AVG(HUMIDITY) AS AVG_HUMIDITY,
                     AVG(WIND_SPEED) AS AVG_WIND_SPEED,
                     AVG(VISIBILITY) AS AVG_VISIBILITY,
                     AVG(DEW_POINT_TEMPERATURE) AS AVG_DEW_POINT_TEMP,
                     AVG(SOLAR_RADIATION) AS AVG_SOLAR_RADIATION,
                     AVG(RAINFALL) AS AVG_RAINFALL,
                     AVG(SNOWFALL) AS AVG_SNOWFALL,
                     AVG(RENTED_BIKE_COUNT) AS AVG_BIKE_COUNT
                     FROM SEOUL_BIKE_SHARING
                     GROUP BY SEASONS
                     ORDER BY AVG(RENTED_BIKE_COUNT) DESC")
```

- It turned out to be similar to Rental Seasonality.

# Bike-sharing info in Seoul

```
view <- sqlQuery(conn, "SELECT B.BICYCLES, C.CITY, C.COUNTRY, C.LAT, C.LNG, C.POPULATION
                        FROM WORLD_CITIES C, BIKE_SHARING_SYSTEM B
                        WHERE C.CITY = B.CITY AND C.CITY = 'Seoul'")
view
```

- Seoul city had total bicycles up to 20000 and a population of around 21794000.
- Other parameters that were recorded in this query were latitude and longitudes.

# Cities similar to Seoul

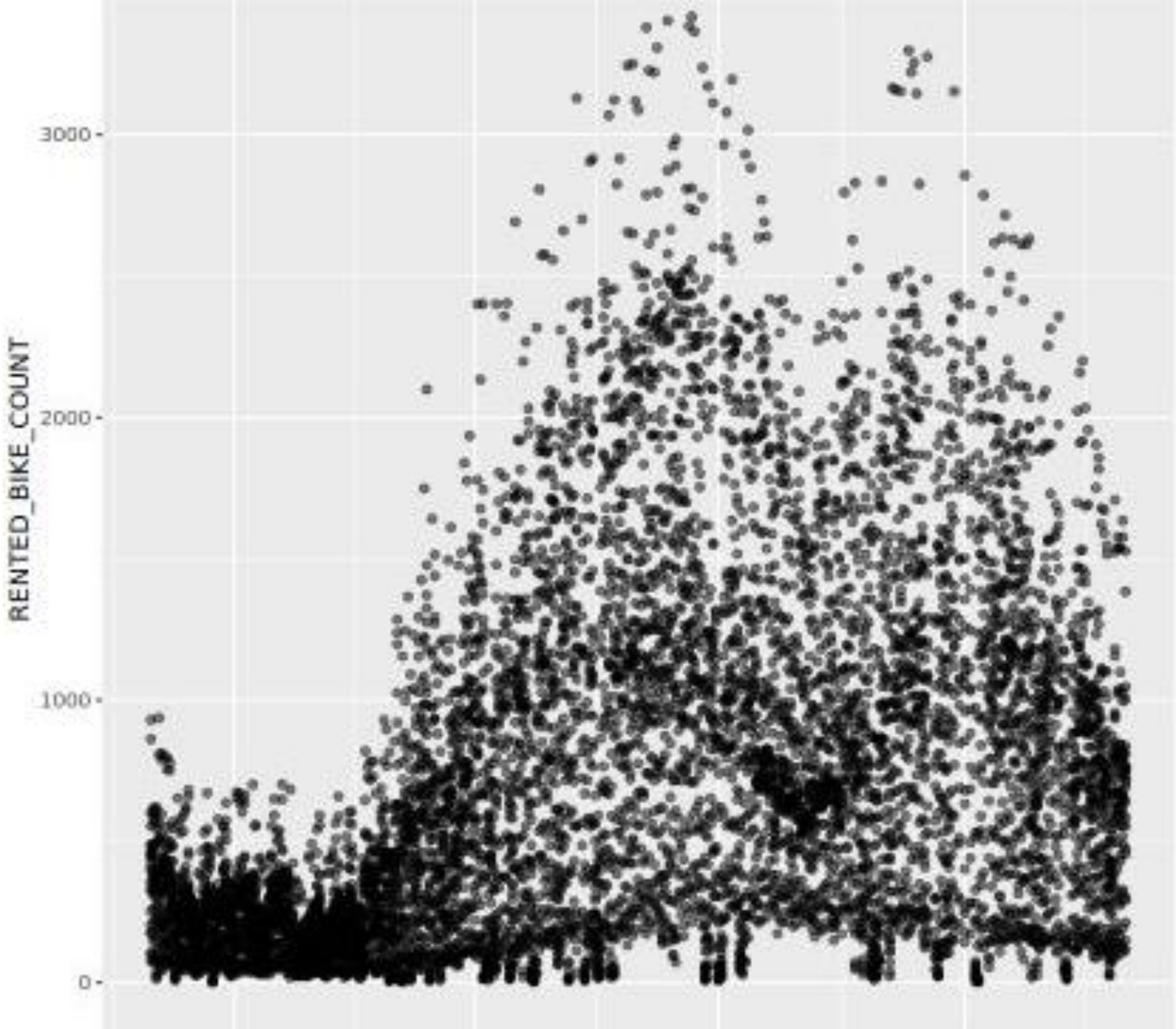
```
view <- sqlQuery(conn, "SELECT B.BICYCLES, C.CITY, C.COUNTRY, C.LAT, C.LNG,
C.POPULATION
                FROM BIKE_SHARING_SYSTEM B, WORLD_CITIES C
                WHERE C.CITY = B.CITY AND C.COUNTRY=B.COUNTRY AND
                B.BICYCLES BETWEEN 15000 AND 20000")

view
```

- The cities that had a comparable bike scale to Seoul were Zhuzhou, Ningbo, Weifang, Beijing and Shanghai with total bike counts between 15000 to 20000.

3

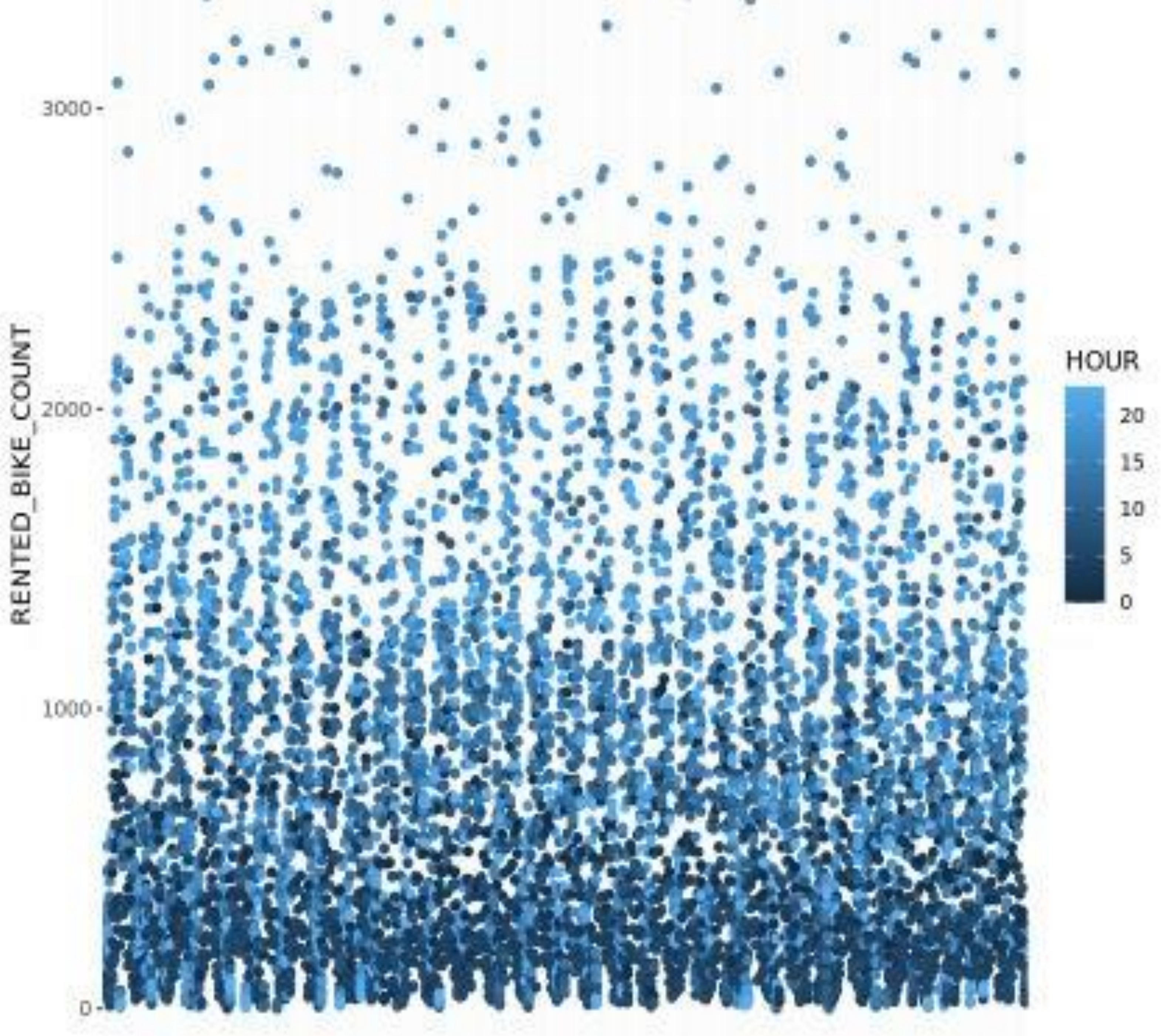
# EDA with Visualization

This section of the presentation covers visualization that was performed on structured data obtained from IBM cloud storage.



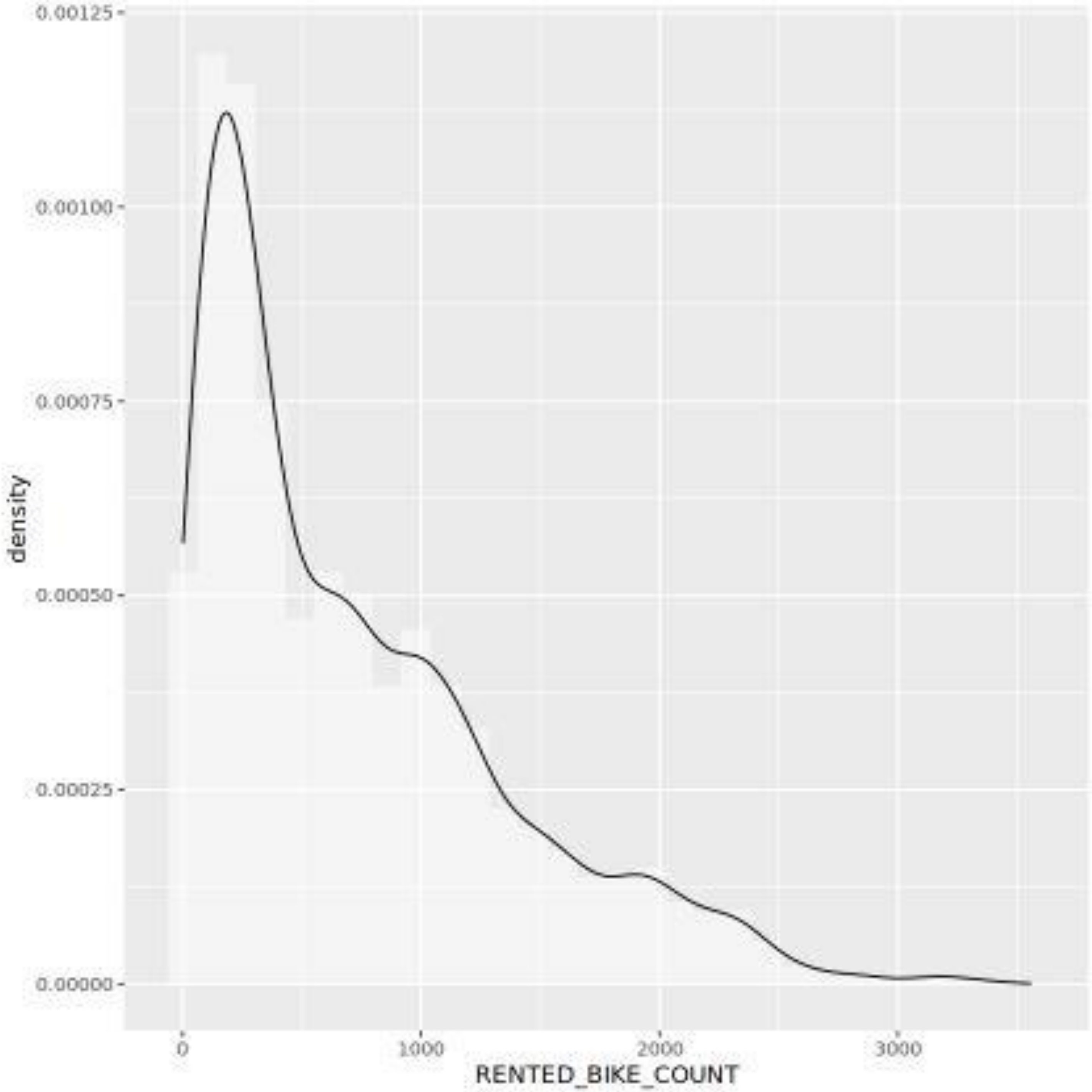

# Bike rental vs. Date

The RENTED_BIKE_COUNT is positively correlated with the DATE. This means that as the DATE increases, the RENTED_BIKE_COUNT increases as well.



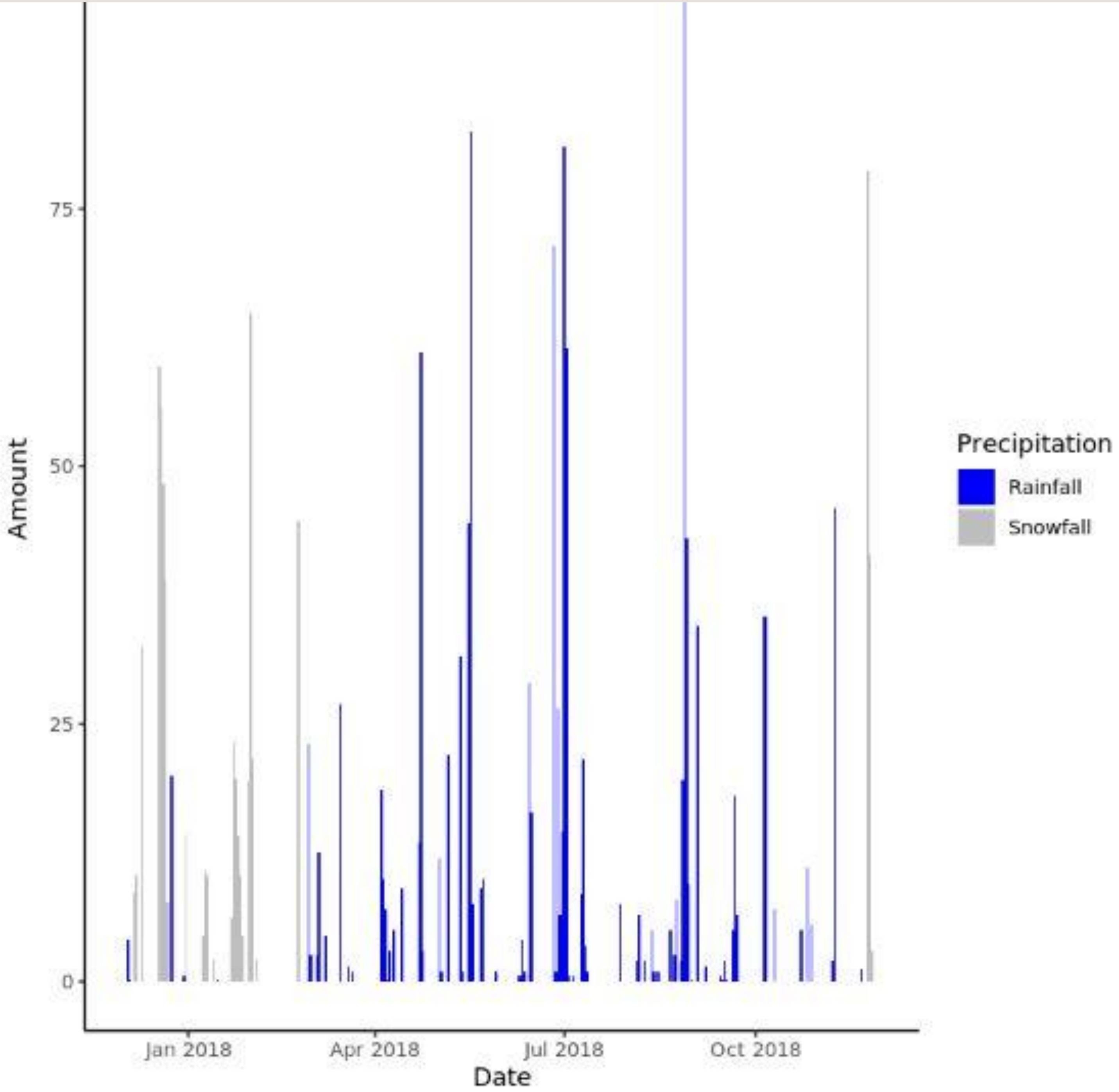

# Bike rental vs. Datetime

The RENTED_BIKE_COUNT and Hours are positively correlated. As Hours increase, the RENTED_BIKE_COUNT also increases.



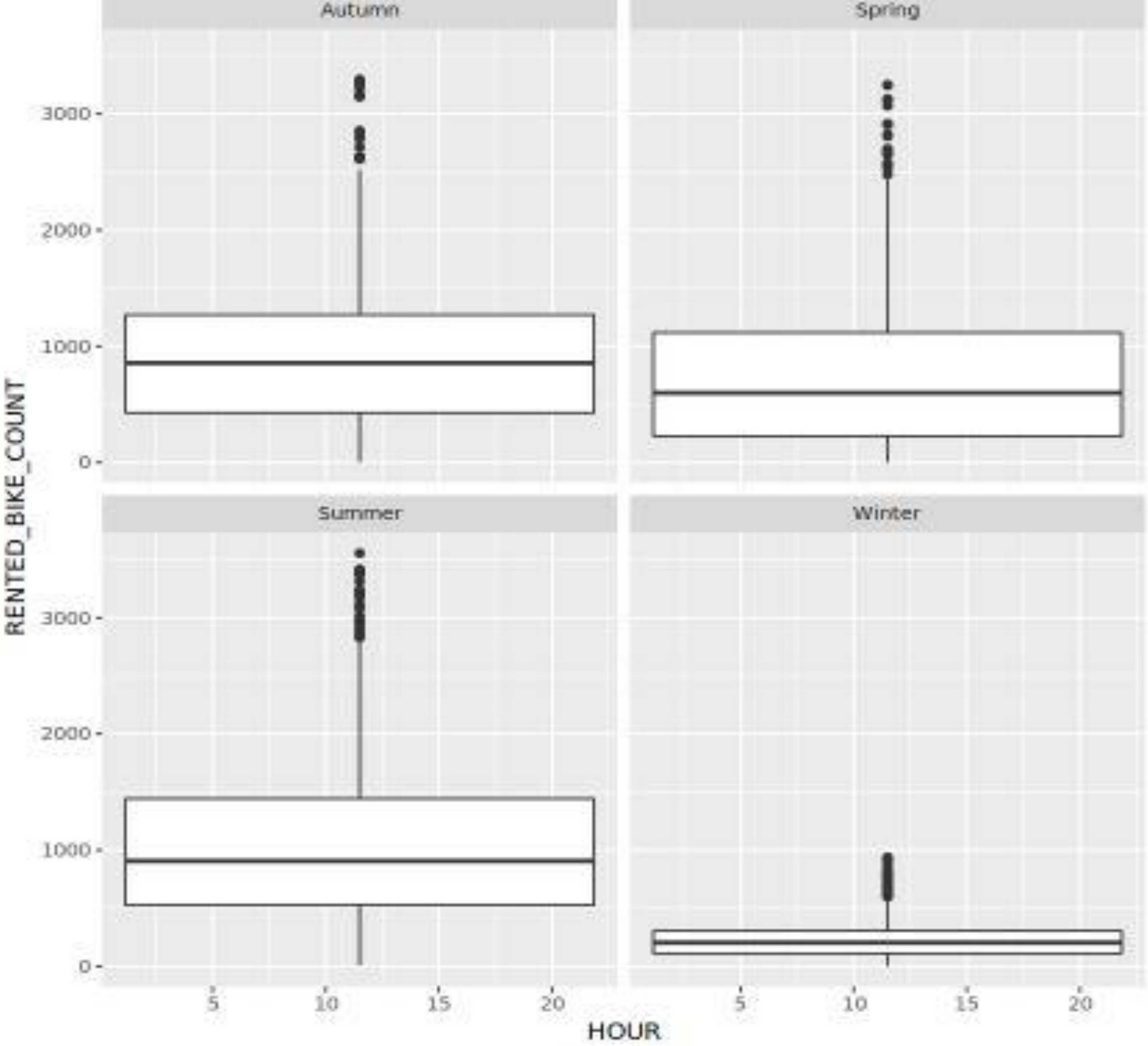

# Bike rental histogram

The kernel density curve is a non-parametric way to estimate the probability density function of a given random variable. It is used to estimate the probability of a given rental bike count in a given area. According to the kernel density curve, the rentals that have the highest probability are those with the highest bike count. It could be viewed on the spike in the graph.




# Daily total rainfall and snowfall

According to the graph, the months with the most daily rainfall were July and April, while the months with the most daily snowfall were December, January and February.



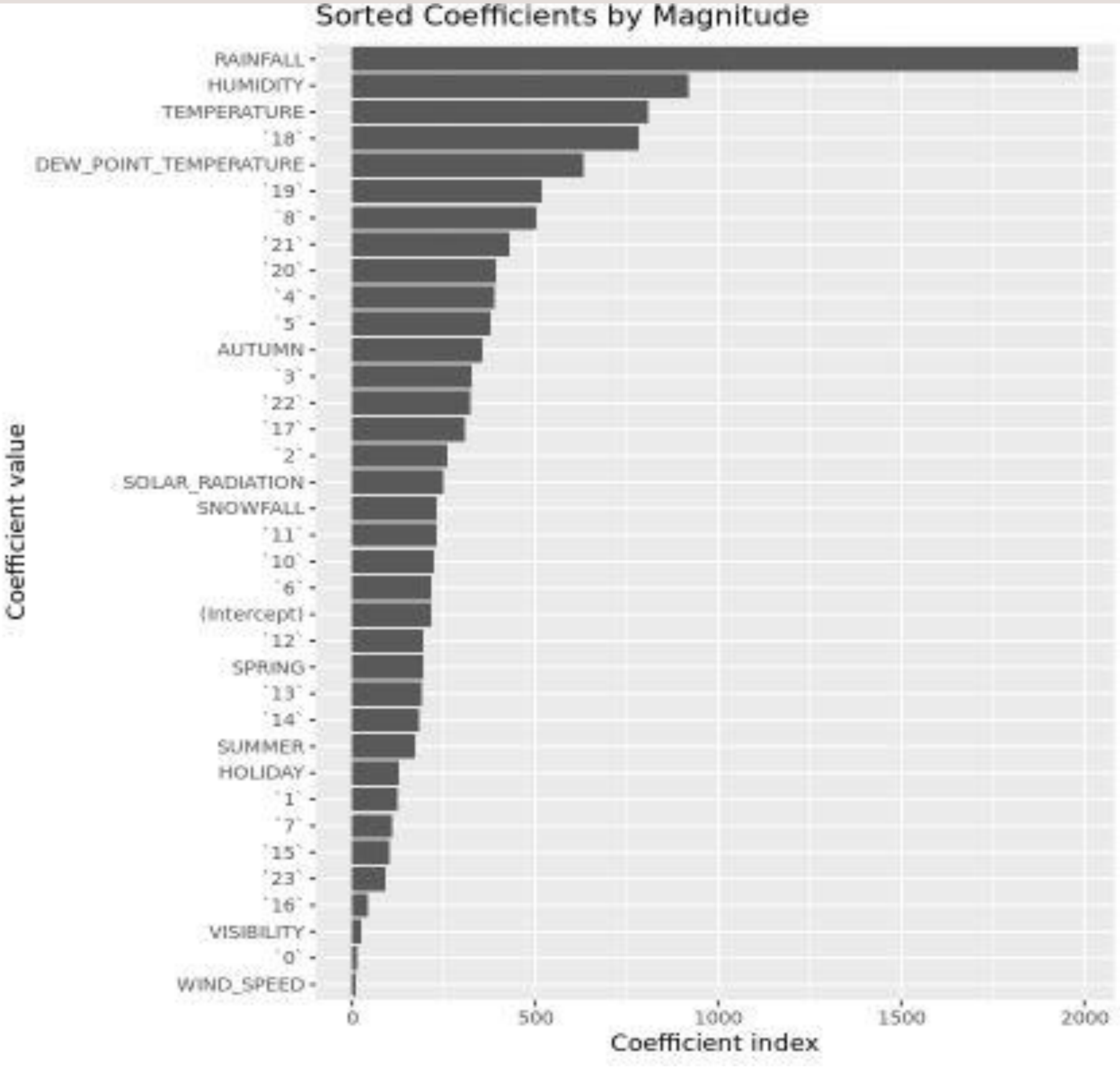

# Bike rental vs. Seasons

RENTED_BIKE_COUNT, SEASONS and HOUR are related in that the number of bicycles rented is likely to increase during the warmer seasons, and peak during specific hours of the day.

4

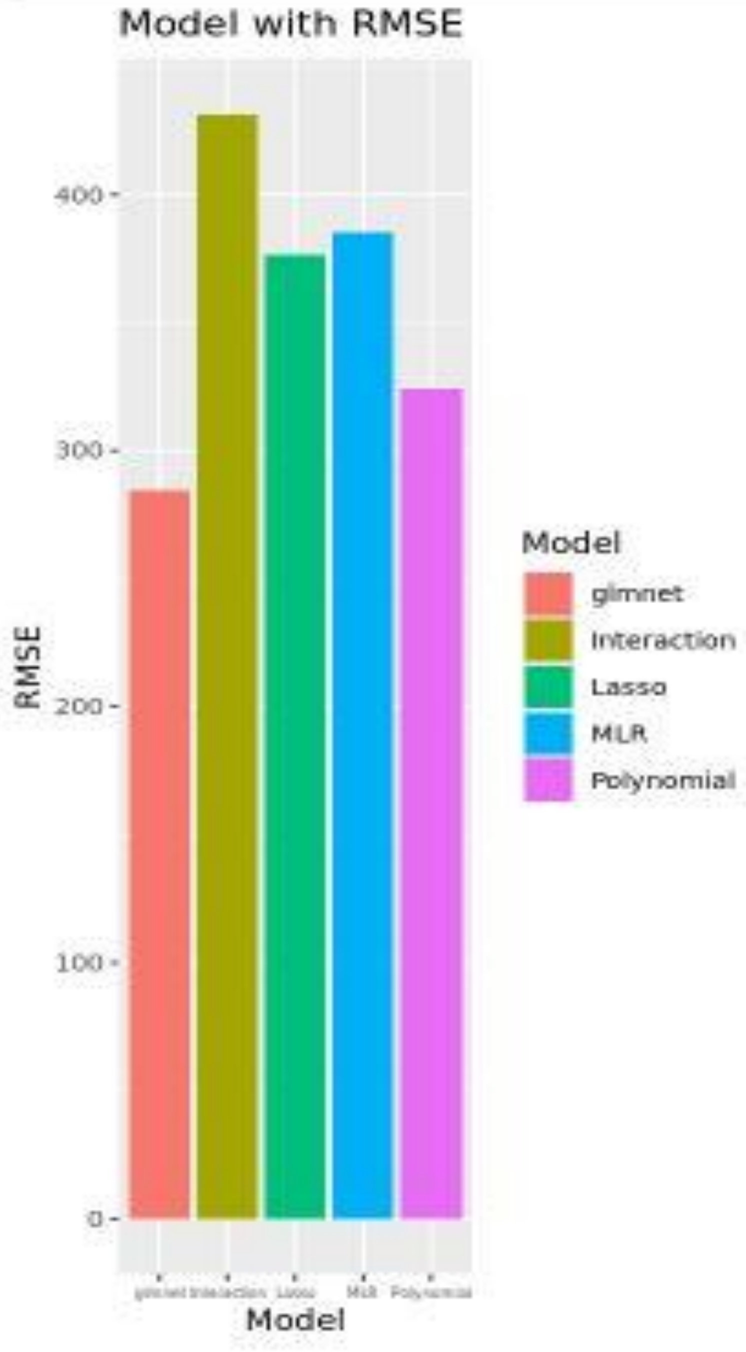
# Predictive analysis

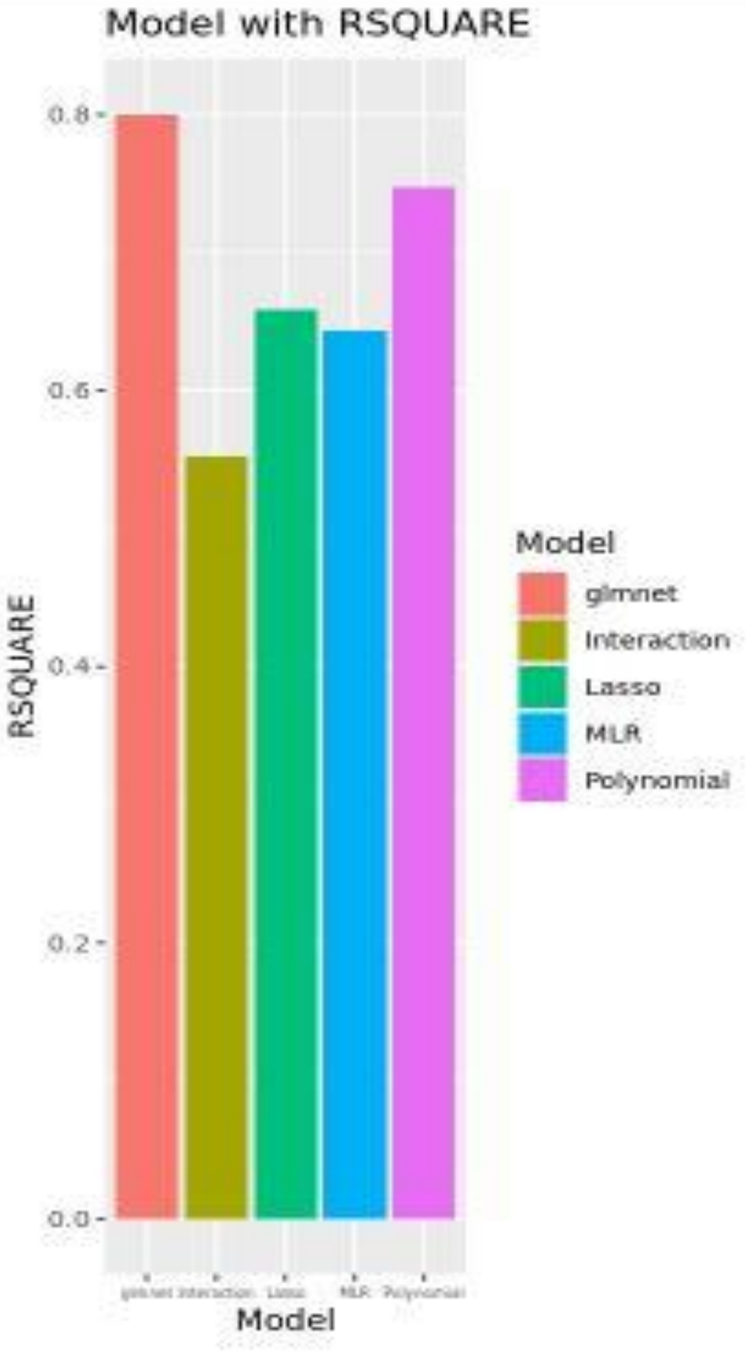
This section of the presentation covers all models that were developed to predict rental demand.




# Ranked coefficients

- The most important coefficients in a bike rental prediction model would be those corresponding to features that strongly influence bike rental demand, such as weather conditions, day of the week, time of day, holidays, and events in the area.
- However, the exact importance of each coefficient will depend on the specific dataset and model used, and it is typically evaluated through techniques such as feature selection.
- A high coefficient value indicates that the predictor variable has a strong association with the response variable, but it does not necessarily mean that it is a good predictor.
- Other factors such as collinearity, interaction effects, and model complexity can also impact the overall predictive power of the model.
- Additionally, the accuracy of a model's predictions depends on the quality and representativeness of the data used to train and test the model.




# Model evaluation

- To check which terms affect bike-rental demand, five different models were made.
- Namely, polynomial, multilinear regression, Lasso, Interaction and glmnet spec.
- Polynomial and Interaction models result in improved metrics but are prone to overfitting.
- Regularization like Lasso, Ridge and Elastic net reduce the affects of overfitting.
- All models were evaluated using RMSE and R-square metrics.
- The best performing model's RMSE is kept under 330 and R-square above 0.72.

# Find the best performing model

- Best model metrics:

A tibble: 1 × 3

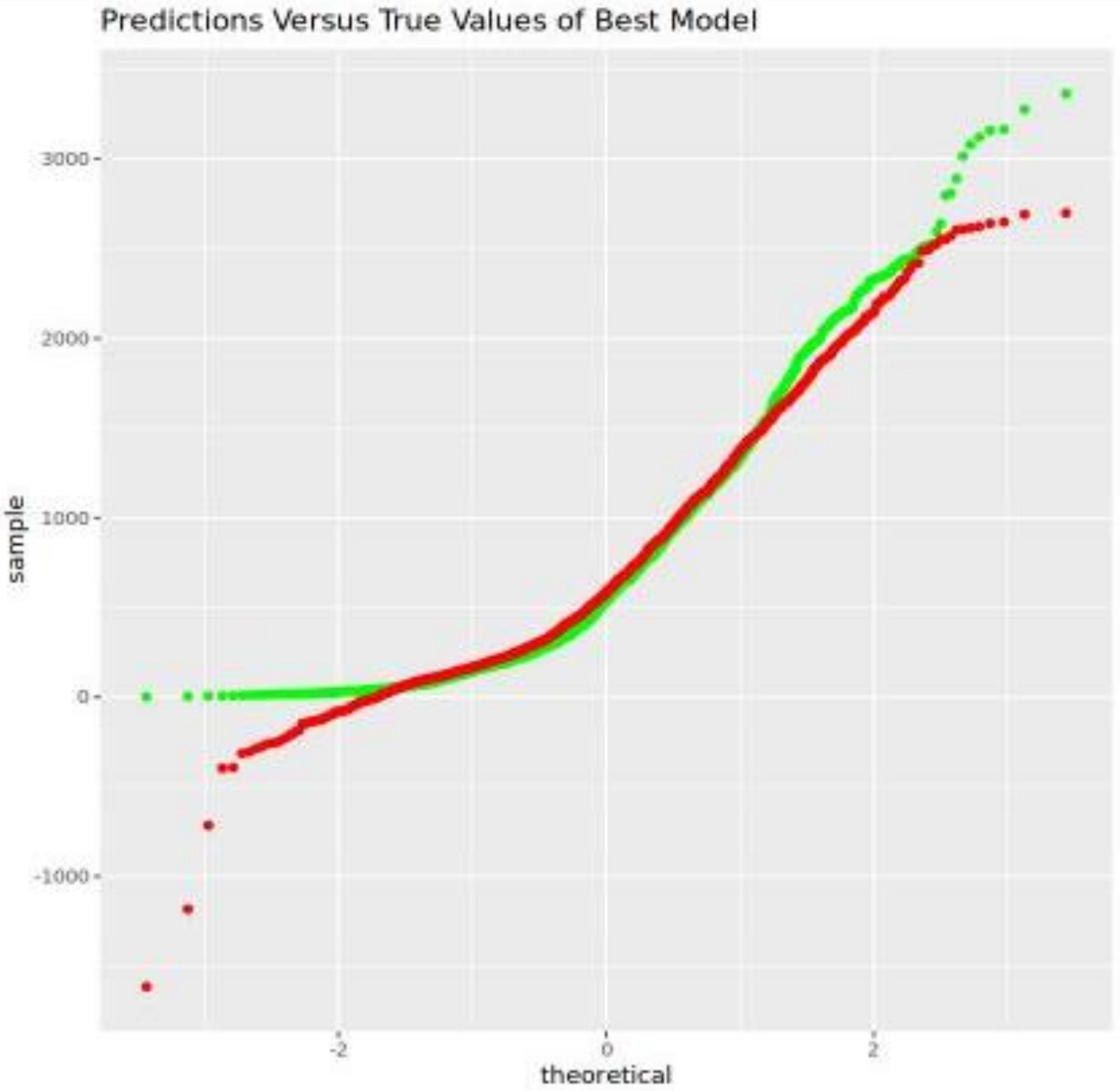
| .metric | .estimator | .estimate |
|---:|---:|---:|
| <chr> | <chr> | <dbl> |
| rmse | standard | 284.7232 |

A tibble: 1 × 3

| .metric | .estimator | .estimate |
|---:|---:|---:|
| <chr> | <chr> | <dbl> |
| rsq | standard | 0.8004955 |

- Formula:

```
lm_glmnet<-glmnet_spec%>%fit(RENTED_BIKE_COUNT ~  (poly(TEMPERATURE,8) + poly(HUMIDITY,7) + poly(SOLAR_RADIATION,6) + poly(WIND_SPEED,5) + poly(VISIBILITY,4) + poly(DEW_POINT_TEMPERATURE,3) + poly(SNOWFALL,2) + `0`+`1`+`2`+`3`+`4`+`5`+`6`+`7`+`8`+`9`+`10`+`11`+`12`+`13`+`14`+`15`+`16`+`17`+`18`+`19`+`20`+`21`+`22`+`23`) * (TEMPERATURE + HUMIDITY + SOLAR_RADIATION + WIND_SPEED + VISIBILITY + DEW_POINT_TEMPERATURE + SUMMER), data = train_data)
```




# Q-Q plot of the best model

The theoretical and sample values align such that the model fits perfectly for most of the data.

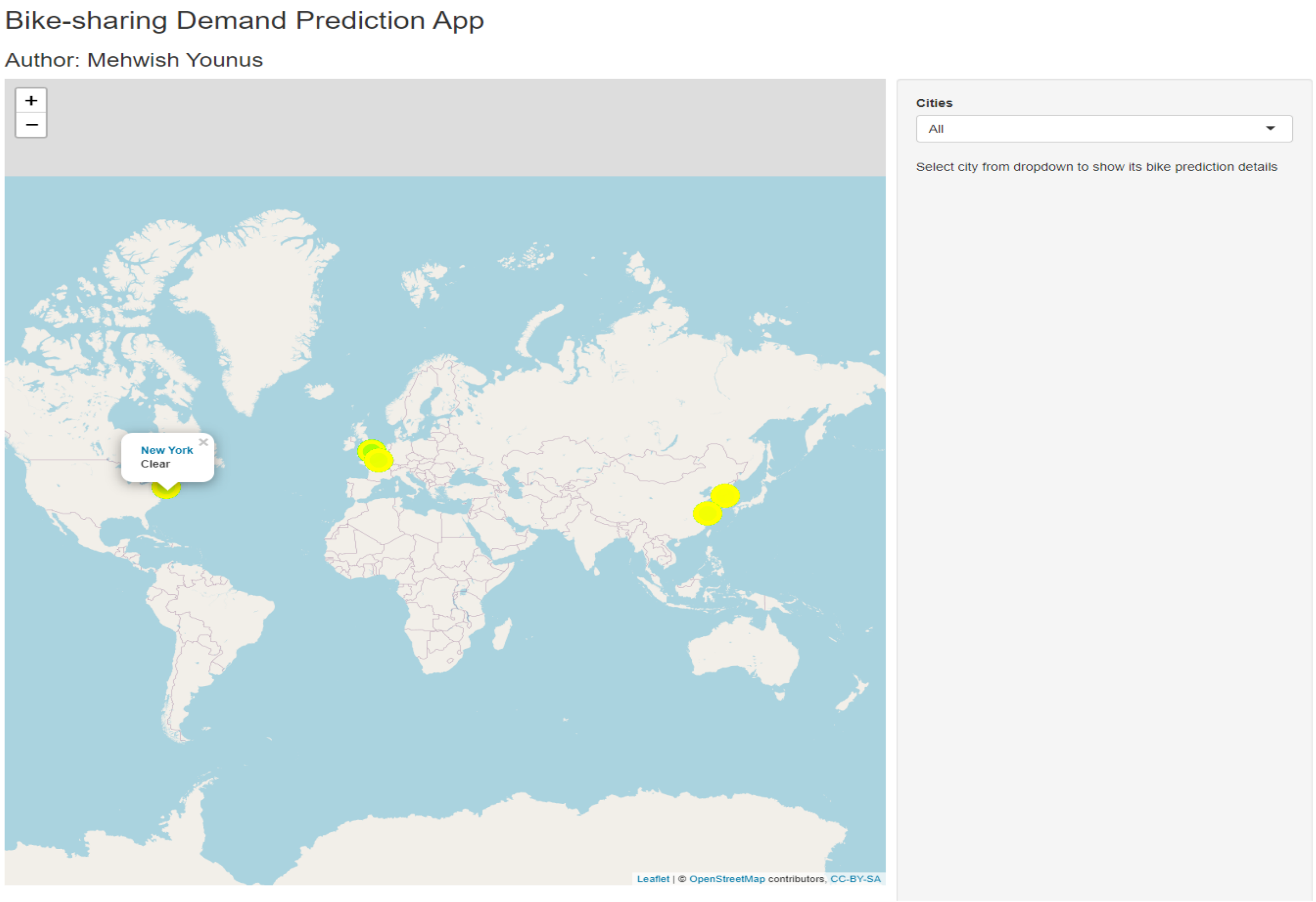
5

# Dashboard

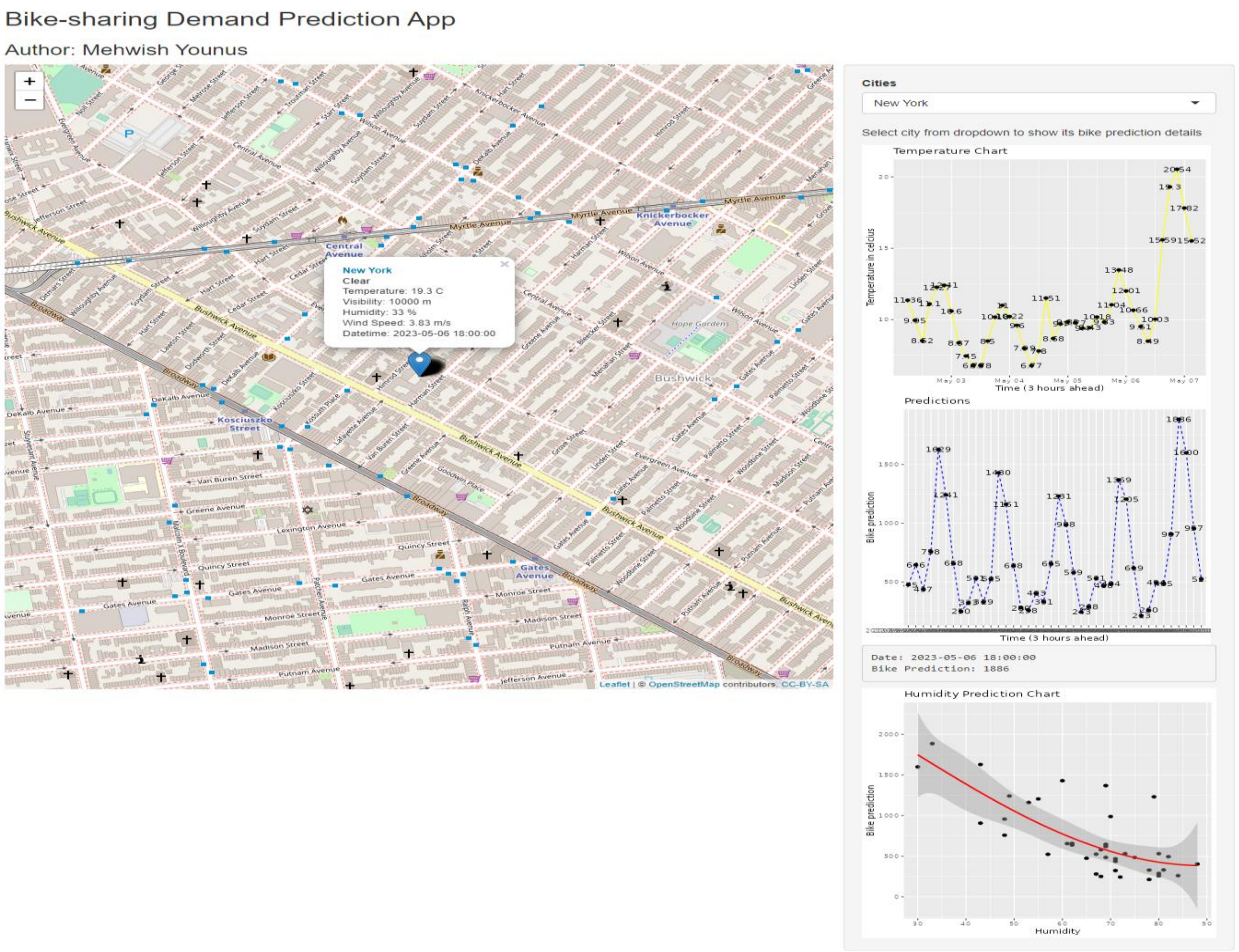
This section of presentation covers the shinyApp that was build using Leaflet and ggplot2 libraries to visualize the performance of best performing model, to better explore rental demand cities, rental demand season and rental times.

# Bike-rental Demand Prediction for all Cities

At this instant all cities have high bike rental demand predictions, as weather is warmer. This can be seen on circle markers that were colored yellow to represent peak demand.



# Bike-rental Demand Prediction for New York




In the side-panel the temperature chart shows the hourly temperature and the chart below shows the predicted bike count. Using these charts, one can predict what would be the rented bike count at a particular hour in New York.

At the moment, the weather is clear so rented-bike count is predicted to be high to medium.

# CONCLUSION

- In conclusion, the bike-rental demand Rshiny App provides a user-friendly interface for exploring and visualizing bike rental demand and weather data in major cities around the world.
- The Rshiny dashboard allows users to interact with the data and explore bike rental demand predictions based on hourly temperature trends and date and time. Users can also explore whether humidity affects rental demand and see the current weather for each city by clicking on the markers on the map.
- Overall, the bike-rental demand Rshiny App provides a powerful tool for analyzing and predicting bike rental demand, which can be useful for bike-sharing companies and urban planners in making informed decisions about bike-sharing infrastructure and policies.

# APPENDIX

All coding used in this project are listed in this section.

# Wikipedia Web Scraping:

```
url <- "https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems"
# Get the root HTML node by calling the `read_html()` method with URL
root_node <- read_html(url)
table_nodes <-html_nodes(root_node,"table")
table_nodes

# Print table_nodes using for loop
for (i in 1:length(table_nodes)) {
    print(table_nodes[[i]])
}
```

[29]:

```
# Convert the bike-sharing system table into a dataframe
raw_bike_sharing_systems<-html_table(table_nodes[[1]],fill="TRUE",header="TRUE")
head(raw_bike_sharing_systems)
```

A data.frame: 6 × 10

| | Country | City | Name | System | Operator | Launched | Discontinued | Stations | Bicycles | Daily ridership |
|---|---|---|---|---|---|---|---|---|---|---|
| | <chr> | <chr> | <chr> | <chr> | <chr> | <chr> | <chr> | <chr> | <chr> | <chr> |
| 1 | Albania | Tirana[5] | Ecovolis | | | March 2011 | | 8 | 200 | |
| 2 | Argentina | Buenos Aires[6][7] | Ecobici | Serttel Brasil[8] | Bike In Baires Consortium[9] | 2010 | | 400 | 4000 | 21917 |
| 3 | Argentina | Mendoza[10] | Metrobici | | | 2014 | | 2 | 40 | |
| 4 | Argentina | Rosario | Mi Bici Tu Bici[11] | | | 2 December 2015 | | 47 | 480 | |
| 5 | Argentina | San Lorenzo, Santa Fe | Biciudad | Biciudad | | 27 November 2016 | | 8 | 80 | |
| 6 | Australia | Melbourne[12] | Melbourne Bike Share | PBSC & 8D | Motivate | June 2010 | 30 November 2019[13] | 53 | 676 | |

Open Weather API HTTP calls:

```r
# Get forecast data for a given city list
get_weather_forecaset_by_cities <- function(city_names){
    df <- data.frame()
    for (city_name in city_names){
        # Forecast API URL
        forecast_url <- 'https://api.openweathermap.org/data/2.5/forecast'
        # Create query parameters
        forecast_query <- list(q = city_name, appid = "1968ac8ff90d782396fdb4f27d8365db", units="metric")
        # Make HTTP GET call for the given city
        response <- GET(forecast_url, query=forecast_query)
        # Note that the 5-day forecast JSON result is a list of lists. You can print the reponse to check the results
        json_result <- content(response, as="parsed")
```

```r
cities <- c("Seoul", "Washington, D.C.", "Paris", "Suzhou")
cities_weather_df <- get_weather_forecaset_by_cities(cities)
cities_weather_df
```

A data.frame: 160 × 12

| city | weather | visibility | temp | temp_min | temp_max | pressure | humidity | wind_speed | wind_deg | forecast_datetime | season |
|---|---|---|---|---|---|---|---|---|---|---|---|
| <fct> | <fct> | <int> | <dbl> | <dbl> | <dbl> | <int> | <int> | <dbl> | <int> | <fct> | <fct> |
| Seoul | Clouds | 10000 | 14.80 | 14.80 | 15.49 | 1011 | 79 | 1.16 | 88 | 2023-04-18 06:00:00 | spring |
| Seoul | Clouds | 10000 | 15.67 | 15.67 | 17.40 | 1011 | 71 | 1.01 | 51 | 2023-04-18 09:00:00 | spring |
| Seoul | Clouds | 10000 | 15.03 | 15.03 | 15.15 | 1012 | 72 | 0.82 | 125 | 2023-04-18 12:00:00 | spring |
| Seoul | Clear | 10000 | 13.61 | 13.61 | 13.61 | 1013 | 76 | 0.34 | 116 | 2023-04-18 15:00:00 | spring |
| Seoul | Clouds | 10000 | 12.43 | 12.43 | 12.43 | 1012 | 83 | 0.51 | 121 | 2023-04-18 18:00:00 | spring |
| Seoul | Clear | 10000 | 11.36 | 11.36 | 11.36 | 1011 | 88 | 1.23 | 66 | 2023-04-18 21:00:00 | spring |
| Seoul | Clear | 10000 | 15.52 | 15.52 | 15.52 | 1012 | 62 | 1.84 | 53 | 2023-04-19 00:00:00 | spring |
| Seoul | Clear | 10000 | 22.33 | 22.33 | 22.33 | 1010 | 32 | 0.85 | 130 | 2023-04-19 03:00:00 | spring |
| Seoul | Clear | 10000 | 25.91 | 25.91 | 25.91 | 1008 | 20 | 1.19 | 204 | 2023-04-19 06:00:00 | spring |

# Data Wrangling using Regular Expressions:

```
for (dataset_name in dataset_list){
    # Read dataset
    dataset <- read_csv(dataset_name)
    # Standardized its columns:

    # Convert all column names to uppercase
    colnames(dataset)<-toupper(colnames(dataset))
    # Replace any white space separators by underscores, using the str_replace_all function
    colnames(dataset)<-str_replace_all(colnames(dataset)," ","_")
    # Save the dataset
    write.csv(dataset, dataset_name, row.names=FALSE)
}
```

```
sub_bike_sharing_df %>%
    select(BICYCLES) %>%
    filter(find_character(BICYCLES)) %>%
    slice(0:10)
```

A spec_tbl_df: 10 × 1

| BICYCLES |
|---|
| <chr> |
| 4115[22] |
| 310[59] |
| 500[72] |
| [75] |
| 180[76] |
| 600[77] |
| [78] |
| initially 800 (later 2500) |
| 100 (220) |
| 370[114] |

```
# Check whether the CITY column has any reference links
sub_bike_sharing_df %>%
    select(CITY) %>%
    filter(find_reference_pattern(CITY)) %>%
    slice(0:10)
```

A spec_tbl_df: 10 × 1

| CITY |
|---|
| <chr> |
| Melbourne[12] |
| Brisbane[14][15] |
| Lower Austria[18] |
| Namur[19] |
| Brussels[21] |
| Salvador[23] |
| Belo Horizonte[24] |
| João Pessoa[25] |
| (Pedro de) Toledo[26] |

```
# Check whether the System column has any reference links
sub_bike_sharing_df %>%
    select(SYSTEM) %>%
    filter(find_reference_pattern(SYSTEM)) %>%
    slice(0:10)
```

A spec_tbl_df: 7 × 1

| SYSTEM |
|---|
| <chr> |
| EasyBike[58] |
| 4 Gen.[61] |
| 3 Gen. SmooveKey[113] |
| 3 Gen. Smoove[141][142][143][139] |
| 3 Gen. Smoove[179] |
| 3 Gen. Smoove[181] |
| 3 Gen. Smoove[183] |

## Data Wrangling using Regular Expressions:

```
result<-sub_bike_sharing_df %>% mutate(SYSTEM=remove_ref(SYSTEM), CITY=remove_ref(CITY),BICYCLES=extract_num(BICYCLES))
result
```

| COUNTRY | CITY | SYSTEM | BICYCLES |
|---|---|---|---|
| <chr> | <chr> | <chr> | <dbl> |
| Albania | Tirana | NA | 200 |
| Argentina | Mendoza | NA | 40 |
| Argentina | San Lorenzo, Santa Fe | Biciudad | 80 |
| Argentina | Buenos Aires | Serttel Brasil | 4000 |
| Argentina | Rosario | NA | 480 |
| Australia | Melbourne | PBSC & 8D | 676 |
| Australia | Brisbane | 3 Gen. Cyclocity | 2000 |
| Australia | Melbourne | 4 Gen. oBike | 1250 |
| Australia | Sydney | 4 Gen. oBike | 1250 |
| Australia | Sydney | 4 Gen. Ofo | 600 |
| Australia | Sydney | Reddy Go | 2000 |
| Austria | Vienna | 3 Gen. Cyclocity | 1500 |
| Austria | Burgenland | 3 Gen. nextbike | NA |

Data Wrangling using dplyr:

```
# Drop rows with `RENTED_BIKE_COUNT` column == NA
bike_sharing_df<-bike_sharing_df%>%drop_na(RENTED_BIKE_COUNT)
bike_sharing_df
```

```
# Calculate the summer average temperature
library(dplyr)

# Compute the mean temperature

mean_temp <- mean(bike_sharing_df$TEMPERATURE, na.rm = TRUE)

mean_temp
```

12.7543222143364

```
# Impute missing values for TEMPERATURE column with summer average temperature
# Impute missing values with the mean

bike_sharing_df <- bike_sharing_df %>%

mutate(TEMPERATURE_IMPUTED = ifelse(is.na(TEMPERATURE), mean_temp, TEMPERATURE))

bike_sharing_df
```

```
# Convert SEASONS, HOLIDAY, FUNCTIONING_DAY, and HOUR columns into indicator columns.
bike_sharing_df<-bike_sharing_df%>%mutate(dummy=1)%>%spread(key=SEASONS, value=dummy, fill=0)
bike_sharing_df<-bike_sharing_df%>%mutate(dummy=1)%>%spread(key=HOLIDAY, value=dummy, fill=0)
bike_sharing_df<-bike_sharing_df%>%mutate(dummy=1)%>%spread(key=HOUR, value=dummy, fill=0)
bike_sharing_df
```

## Data Wrangling using dplyr:

```
# Use the `mutate()` function to apply min-max normalization on columns
# `RENTED_BIKE_COUNT`, `TEMPERATURE`, `HUMIDITY`, `WIND_SPEED`, `VISIBILITY`, `DEW_POINT_TEMPERATURE`, `SOLAR_RADIATION`, `RAINI

# define custom scaleminmax function
scaleminmax <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# use mutate to apply scaleminmax function to specified columns
bike_sharing_df_scaled <- bike_sharing_df %>%
  mutate(RENTED_BIKE_COUNT = scaleminmax(RENTED_BIKE_COUNT),
         TEMPERATURE = scaleminmax(TEMPERATURE_IMPUTED),
         HUMIDITY = scaleminmax(HUMIDITY),
         WIND_SPEED = scaleminmax(WIND_SPEED),
         VISIBILITY = scaleminmax(VISIBILITY),
         DEW_POINT_TEMPERATURE = scaleminmax(DEW_POINT_TEMPERATURE),
         SOLAR_RADIATION = scaleminmax(SOLAR_RADIATION),
         RAINFALL = scaleminmax(RAINFALL),
         SNOWFALL = scaleminmax(SNOWFALL))

# view the scaled data frame
bike_sharing_df_scaled
```

Data Wrangling using dplyr:

| DATE | RENTED_BIKE_COUNT | TEMPERATURE | HUMIDITY | WIND_SPEED | VISIBILITY | DEW_POINT_TEMPERATURE | SOLAR_RADIATION |
|---|---|---|---|---|---|---|---|
| <chr> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| 01/12/2017 | 0.07090602 | 0.2202797 | 0.3775510 | 0.29729730 | 1.0000000 | 0.2249135 | 0.000000000 |
| 01/12/2017 | 0.05683737 | 0.2150350 | 0.3877551 | 0.10810811 | 1.0000000 | 0.2249135 | 0.000000000 |
| 01/12/2017 | 0.04811480 | 0.2062937 | 0.3979592 | 0.13513514 | 1.0000000 | 0.2231834 | 0.000000000 |
| 01/12/2017 | 0.02954418 | 0.2027972 | 0.4081633 | 0.12162162 | 1.0000000 | 0.2249135 | 0.000000000 |
| 01/12/2017 | 0.02138436 | 0.2062937 | 0.3673469 | 0.31081081 | 1.0000000 | 0.2076125 | 0.000000000 |
| 01/12/2017 | 0.02757456 | 0.1993007 | 0.3775510 | 0.20270270 | 1.0000000 | 0.2058824 | 0.000000000 |
| 01/12/2017 | 0.05036579 | 0.1958042 | 0.3571429 | 0.17567568 | 1.0000000 | 0.1920415 | 0.000000000 |
| 01/12/2017 | 0.12886888 | 0.1818182 | 0.3877551 | 0.12162162 | 1.0000000 | 0.1955017 | 0.000000000 |
| 01/12/2017 | 0.26111424 | 0.1783217 | 0.3775510 | 0.14864865 | 1.0000000 | 0.1868512 | 0.002840909 |
| 01/12/2017 | 0.13731007 | 0.1975524 | 0.2755102 | 0.06756757 | 0.9635073 | 0.1418685 | 0.065340909 |
| 01/12/2017 | 0.09482273 | 0.2500000 | 0.2448980 | 0.16216216 | 0.9979726 | 0.1626298 | 0.184659091 |
| 01/12/2017 | 0.10073157 | 0.3024476 | 0.2142857 | 0.17567568 | 0.9675621 | 0.1799308 | 0.267045455 |

# EDA with SQL:

## Task 1 - Record Count

Determine how many records are in the seoul_bike_sharing dataset.

### Solution 1

```
# provide your solution here
view<-sqlQuery(conn,"SELECT COUNT(RENTED_BIKE_COUNT) AS NUMBER_OF_RECORDS FROM SEOUL_BIKE_SHARING")
view
```

A data.frame: 1 × 1

| | NUMBER_OF_RECORDS |
|---|---|
| | <int> |
| 1 | 8465 |

## Task 2 - Operational Hours

Determine how many hours had non-zero rented bike count.

### Solution 2

```
# provide your solution here
view<-sqlQuery(conn, "SELECT COUNT(HOUR) AS NUMBER_OF_HOURS FROM SEOUL_BIKE_SHARING WHERE RENTED_BIKE_COUNT!=0")
view
```

A data.frame: 1 × 1

| | NUMBER_OF_HOURS |
|---|---|
| | <int> |
| 1 | 8465 |

## Task 3 - Weather Outlook

Query the the weather forecast for Seoul over the next 3 hours.

Recall that the records in the CITIES_WEATHER_FORECAST dataset are 3 hours apart, so we just need the first record from the query.

### Solution 3

```
# provide your solution here
view<-sqlQuery(conn, "SELECT * FROM CITIES_WEATHER_FORECAST WHERE CITY='Seoul' LIMIT 1")
view
```

A data.frame: 1 × 12

| | CITY | WEATHER | VISIBILITY | TEMP | TEMP_MIN | TEMP_MAX | PRESSURE | HUMIDITY | WIND_SPEED | WIND_DEG | SEASON | FORECAST_DA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | <fct> | <fct> | <int> | <dbl> | <dbl> | <dbl> | <int> | <int> | <dbl> | <int> | <fct> | |
| 1 | Seoul | Clear | 10000 | 12.32 | 10.91 | 12.32 | 1015 | 50 | 2.18 | 248 | Spring | 2021-04-16 |

## Task 4 - Seasons

Find which seasons are included in the seoul bike sharing dataset.

### Solution 4

```
# provide your solution here
view<-sqlQuery(conn, "SELECT DISTINCT(SEASONS) FROM SEOUL_BIKE_SHARING")
view
```

A data.frame: 4 × 1

| | SEASONS |
|---|---|
| | <fct> |
| 1 | Autumn |
| 2 | Spring |
| 3 | Summer |
| 4 | Winter |

# EDA with SQL:

## Task 5 - Date Range

Find the first and last dates in the Seoul Bike Sharing dataset.

### Solution 5

```
# provide your solution here
view<-sqlQuery(conn, "SELECT MIN(DATE) AS FIRST_DATE, MAX(DATE) AS LAST_DATE FROM SEOUL_BIKE_SHARING")
view
```

A data.frame: 1 × 2

| | FIRST_DATE | LAST_DATE |
|---|---|---|
| | <date> | <date> |
| 1 | 2017-12-01 | 2018-11-30 |

## Task 6 - Subquery - 'all-time high'

determine which date and hour had the most bike rentals.

### Solution 6

```
# provide your solution here
view<-sqlQuery(conn, "SELECT DATE, HOUR, RENTED_BIKE_COUNT FROM SEOUL_BIKE_SHARING WHERE RENTED_BIKE_COUNT=
(SELECT MAX(RENTED_BIKE_COUNT) FROM SEOUL_BIKE_SHARING)")
view
```

A data.frame: 1 × 3

| | DATE | HOUR | RENTED_BIKE_COUNT |
|---|---|---|---|
| | <date> | <int> | <int> |
| 1 | 2018-06-19 | 18 | 3556 |

## Task 7 - Hourly popularity and temperature by season

Determine the average hourly temperature and the average number of bike rentals per hour over each season. List the top ten results by average bike count.

### Solution 7

```
# provide your solution here
view<-sqlQuery(conn, "SELECT SEASONS, HOUR, AVG(TEMPERATURE) AS AVG_HOURLY_TEMP, AVG(RENTED_BIKE_COUNT) AS AVG_BIKE_RENTALS FROM
view
```

A data.frame: 10 × 4

| | SEASONS | HOUR | AVG_HOURLY_TEMP | AVG_BIKE_RENTALS |
|---|---|---|---|---|
| | <fct> | <int> | <dbl> | <int> |
| 1 | Summer | 18 | 29.38696 | 2135 |
| 2 | Autumn | 18 | 16.03086 | 1983 |
| 3 | Summer | 19 | 28.27283 | 1889 |
| 4 | Summer | 20 | 27.06630 | 1801 |

## Task 8 - Rental Seasonality

Find the average hourly bike count during each season.

Also include the minimum, maximum, and standard deviation of the hourly bike count for each season.

### Solution 8

```
# provide your solution here
view<-sqlQuery(conn, "SELECT SEASONS, AVG(RENTED_BIKE_COUNT) AS HOURLY_BIKE_COUNT, MIN(RENTED_BIKE_COUNT) AS MINIMUM, MAX(RENTED
view
```

A data.frame: 96 × 5

| | SEASONS | HOURLY_BIKE_COUNT | MINIMUM | MAXIMUM | STANDARD_DEV |
|---|---|---|---|---|---|
| | <fct> | <int> | <int> | <int> | <dbl> |
| 1 | Autumn | 709 | 119 | 1336 | 219.14298 |
| 2 | Spring | 481 | 22 | 1089 | 253.38673 |
| 3 | Summer | 899 | 26 | 1394 | 285.31199 |
| 4 | Winter | 165 | 42 | 342 | 63.81163 |

# EDA with SQL:

## Task 9 - Weather Seasonality

Consider the weather over each season. On average, what were the TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, and SNOWFALL per season?

Include the average bike count as well , and rank the results by average bike count so you can see if it is correlated with the weather at all.

### Solution 9

```
view<-sqlQuery(conn, "SELECT SEASONS,
                    AVG(TEMPERATURE) AS AVG_TEMP,
                    AVG(HUMIDITY) AS AVG_HUMIDITY,
                    AVG(WIND_SPEED) AS AVG_WIND_SPEED,
                    AVG(VISIBILITY) AS AVG_VISIBILITY,
                    AVG(DEW_POINT_TEMPERATURE) AS AVG_DEW_POINT_TEMP,
                    AVG(SOLAR_RADIATION) AS AVG_SOLAR_RADIATION,
                    AVG(RAINFALL) AS AVG_RAINFALL,
                    AVG(SNOWFALL) AS AVG_SNOWFALL,
                    AVG(RENTED_BIKE_COUNT) AS AVG_BIKE_COUNT
               FROM SEOUL_BIKE_SHARING
               GROUP BY SEASONS
               ORDER BY AVG(RENTED_BIKE_COUNT) DESC")
view
```

A data.frame: 4 × 10

| | SEASONS | AVG_TEMP | AVG_HUMIDITY | AVG_WIND_SPEED | AVG_VISIBILITY | AVG_DEW_POINT_TEMP | AVG_SOLAR_RADIATION | AVG_RAINF |
|---|---|---|---|---|---|---|---|---|
| | <fct> | <dbl> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dl |
| 1 | Summer | 26.587274 | 64 | 1.609420 | 1501 | 18.750136 | 0.7612545 | 0.25348 |
| 2 | Autumn | 13.821167 | 59 | 1.492101 | 1558 | 5.150594 | 0.5227827 | 0.11765 |
| 3 | Spring | 13.021389 | 58 | 1.857778 | 1240 | 4.091389 | 0.6803009 | 0.18694 |
| 4 | Winter | -2.540463 | 49 | 1.922685 | 1445 | -12.416667 | 0.2981806 | 0.03282 |

## Task 10 - Total Bike Count and City Info for Seoul

Use an implicit join across the WORLD_CITIES and the BIKE_SHARING_SYSTEMS tables to determine the total number of bikes avaialble in Seoul, plus the following city information about Seoul: CITY, COUNTRY, LAT, LON, POPULATION, in a single view.

Notice that in this case, the CITY column will work for the WORLD_CITIES table, but in general you would have to use the CITY_ASCII column.

### Solution 10

```
# provide your solution here
view <- sqlQuery(conn, "SELECT B.BICYCLES, C.CITY, C.COUNTRY, C.LAT, C.LNG, C.POPULATION
                        FROM WORLD_CITIES C, BIKE_SHARING_SYSTEM B
                        WHERE C.CITY = B.CITY AND C.CITY = 'Seoul'")
view
```

A data.frame: 1 × 6

| | BICYCLES | CITY | COUNTRY | LAT | LNG | POPULATION |
|---|---|---|---|---|---|---|
| | <int> | <fct> | <fct> | <dbl> | <dbl> | <int> |
| 1 | 20000 | Seoul | Korea, South | 37.58 | 127 | 21794000 |

## Task 11 - Find all city names and coordinates with comparable bike scale to Seoul's bike sharing system

Find all cities with total bike counts between 15000 and 20000. Return the city and country names, plus the coordinates (LAT, LNG), population, and number of bicycles for each city.

Later we will ask you to visualize these similar cities on leaflet, with some weather data.

### Solution 11

```
# provide your solution here
view <- sqlQuery(conn, "SELECT B.BICYCLES, C.CITY, C.COUNTRY, C.LAT, C.LNG, C.POPULATION
                        FROM BIKE_SHARING_SYSTEM B, WORLD_CITIES C
                        WHERE C.CITY = B.CITY AND C.COUNTRY=B.COUNTRY AND
                        B.BICYCLES BETWEEN 15000 AND 20000")

view
```

A data.frame: 5 × 6

| | BICYCLES | CITY | COUNTRY | LAT | LNG | POPULATION |
|---|---|---|---|---|---|---|
| | <int> | <fct> | <fct> | <dbl> | <dbl> | <int> |
| 1 | 19165 | Shanghai | China | 31.16 | 121.46 | 22120000 |
| 2 | 16000 | Beijing | China | 39.90 | 116.39 | 19433000 |

EDA with Data Visualization:

## Task 10 - Create a scatter plot of `RENTED_BIKE_COUNT` vs `DATE` .

Tune the opacity using the `alpha` parameter such that the points don't obscure each other too much.

## Solution 10

```
# provide your solution here
df%>%ggplot(aes(DATE, RENTED_BIKE_COUNT))+
    geom_point(alpha=0.5)
```

## Task 11 - Create the same plot of the `RENTED_BIKE_COUNT` time series, but now add `HOURS` as the colour.

## Solution 11

```
# provide your solution here
df%>%ggplot(aes(DATE, RENTED_BIKE_COUNT,color=HOUR))+
    geom_point()
```

## Task 15 - Group the data by `DATE`, and use the summarize() function to calculate the daily total rainfall and snowfall. ¶

```
# create the grouped bar chart using ggplot2
ggplot(results, aes(x = DATE)) +
  geom_bar(aes(y = daily_rainfall, fill = "Rainfall"), position = "dodge", stat = "identity",width=1) +
  geom_bar(aes(y = daily_snowfall, fill = "Snowfall"), position = "dodge", stat = "identity", width=1) +
  labs(x = "Date", y = "Amount", fill = "") +
  scale_fill_manual(values = c("Rainfall" = "blue", "Snowfall" = "gray"), name = "Precipitation") +
  theme_classic()
```

EDA with Data Visualization:

# Outliers (boxplot)

## Task 14 - Create a display of four boxplots of `RENTED_BIKE_COUNT` vs. `HOUR` grouped by `SEASONS`.

Use `facet_wrap` to generate four plots corresponding to the seasons.

## Solution 14

```
# provide your solution here
df%>%ggplot(aes(HOUR,RENTED_BIKE_COUNT))+
    geom_boxplot(aes(group = SEASONS))+
facet_wrap(~SEASONS)
```

## Task 12 - Create a histogram overlaid with a kernel density curve

Normalize the histogram so the y axis represents 'density'. This can be done by setting `y=..density..` in the aesthetics of the histogram.

► Click here for a hint

► Click here for another hint

## Solution 12

```
# provide your solution here
df%>% ggplot(aes(x=RENTED_BIKE_COUNT)) +
  geom_histogram(aes(y=..density..), fill="white", alpha=0.5) +
  geom_density(color="black")
```