A - Traveling history

Train.csv - the heart of the dataset contains travels from 01/01/2015 until 31/05/2015. Data from every second day of April and May is withheld for testing the 1st and 2nd challenge tasks. For the remaining days train.csv contains the following information for each travel:

1. bicycle_id - unique ID of the bicycle
2. start_time - start time of the trip
3. end_time - end time of the trip
4. start_location - source station ID
5. end_location - target station ID

Note that:

- During service travels, operators reallocate the bikes to make sure bikes are available at all stations all the time. Service travels were removed from the data.
- Invalid travels, when the duration is 0 were removed from the dataset.
- Station 0101 (Batthyány tér) was removed from the dataset, as there is no data from this station between January 16 and April 16.
- We always consider entire travels that start by taking a bike from the source station and returning it to a target station. For a travel spanning through more days (e.g. starting Saturday 11:50 PM and ending Sunday 0:10 AM), we always consider the start time for training or evaluation, i.e. if Saturday is a training day, then both the source and the target events are in the training set, even if Sunday is an evaluation day, and vice versa.

B - Additional datasets

Docking stations

station_data.csv contains information about the docking stations. The corresponding columns are:

1. place_id - ID of the station (string, not integer!)
2. place_name - name of the station
3. lat - GPS latitude
4. lon - GPS longitude
5. num_of_rack - capacity (the number of bikes that the station can officially accept)
6. datetime_start - starting date of the period when the station was at this location
7. datetime_end - end date of the period when the station was at this location

Note that:

- As some docking stations were re-located during 2015 by the provider, this file may contain more records for a given station.

- Even if the station is full, users may return additional bikes to the extra stands located near the stations.

C. Weather information

For the entire period, weather_data.csv contains the following information for every thirty minutes:

1. time - date and time
2. tempm - temperature in C
3. hum - humidity in %
4. wspdm - wind speed in kph
5. wdird - wind direction in degrees
6. wdire - wind direction description (ie. SW, NNE)
7. pressurem - pressure in mBar
8. vism - visibility in Km
9. windchillm - wind chill in C
10. fog - 1 in case of fog, 0 otherwise
11. rain - 1 in case of rain, 0 otherwise
12. snow - 1 in case of snow, 0 otherwise
13. hail - 1 in case of hail, 0 otherwise
14. thunder - 1 in case of thunder, 0 otherwise

TASKS

The task is to answer the following two questions based on the analysis of the station and rental data published by the MOL Bubi public bike sharing system in the first half of 2015. The description of the files can be found above.

Based on available data, answer the following questions:

1. Explore how the weather affects lending. Is there a route, perhaps a time of day, or a period that is less relevant to the weather. What could be the reason for this?

2. An advertiser would like to place an ad at three rental points for a period of two months. On the basis of the available data, make suggestions on which points it would be appropriate to set up ads so that they are seen by many users as possible.

SUPPLEMENTARY DATA

PostgreSQL was used to get exploratory information on the datasets.

Since the data available in the train table was skewed to show only information on every second day in April and May, two different approaches/models were used to perform the analysis on the dataset to observe the possible differences/similarities in results.

Model A – data analysis using all the dates as presented in the dataset.
Model B – data analysis with every second day withheld in January – March, as in April and May.

Based on results from the initial exploratory analysis using model A and B, subsequent analysis was performed using the latter method.

```
CREATE MATERIALIZED VIEW train_model_B
AS    SELECT    EXTRACT(YEAR FROM start_time) AS rental_year,
                  DATE_PART ('month', start_time) AS month_number,
                  DATE_PART ('dow', start_time) AS day_of_week,
                  DATE_PART ('day', start_time) AS day_number,
                   DATE_PART ('hour', start_time) AS time_of_day_in_hours,
                        CONCAT (DATE_PART ('hour', start_time), ':',
                CASE WHEN
                DATE_PART ('minute', start_time) < 30 THEN 0 ELSE 30 END) AS HalfHour,
                   bicycle_id AS id,
                   start_time AS start_of_rent,
                  end_time AS end_of_rent,
                    CASE
                        WHEN DATE_PART ('day', start_time)
                        IN ('1', '3', '5', '7', '9', '11', '13', '15', '17', '19', '21', '23', '25', '27', '29', '31')
                        THEN 1 ELSE 0 END AS shown_days,
                    CASE
                        WHEN DATE_PART ('day', start_time)
                        IN ('1', '3', '5', '7', '9', '11', '13', '15', '17', '19', '21', '23', '25', '27', '29', '31')
                        THEN 0 ELSE 1 END AS withheld_days,
                    start_location AS start_station,
                    end_location AS end_station,
                   CONCAT (start_location, ' - ', end_location) AS route,
                    COUNT (*) AS number_of_rental
        FROM train
        GROUP BY 1,2,3,4,5,6,7,8,9,10,11,12,13
        ORDER BY 2;
```

Monthly trend

Model B

```
WITH mth AS (SELECT * FROM train_model_b)
SELECT CASE
        WHEN month_number = 1 THEN 'January'
        WHEN month_number = 2 THEN 'February'
        WHEN month_number = 3 THEN 'March'
        WHEN month_number = 4 THEN 'April'
        ELSE 'May'
        END AS months,
        SUM (number_of_rental) as rental_on_alternating_days
FROM mth
WHERE shown_days = 1
GROUP BY 1
ORDER BY 2
```

Information about the various timeframes (monthly, day of the week, hour, 30-minute interval)

Model B

```
WITH t AS (SELECT * FROM train_model_b)
SELECT  CASE
        WHEN month_number = 1 THEN 'January'
        WHEN month_number = 2 THEN 'February'
        WHEN month_number = 3 THEN 'March'
        WHEN month_number = 4 THEN 'April'
        ELSE 'May'
        END AS months,
            CASE
        WHEN day_of_week = 0 THEN 'Sunday'
        WHEN day_of_week = 1 THEN 'Monday'
        WHEN day_of_week = 2 THEN 'Tuesday'
        WHEN day_of_week = 3 THEN 'Wednesday'
        WHEN day_of_week = 4 THEN 'Thursday'
        WHEN day_of_week = 5 THEN 'Friday'
        ELSE 'Saturday'
        END AS days_of_the_week,
```

```
            time_of_day_in_hours,
            HalfHour,
        SUM (number_of_rental) as rental_on_alternating_days
FROM t
WHERE shown_days = 1
GROUP BY 1,2,3,4
ORDER BY 1
```

Maximum and average rental time

Model B

```
WITH t_train AS (WITH mth AS (SELECT * FROM train_model_b)
SELECT id,
        start_of_rent,
        end_of_rent,
        start_station,
        end_station
FROM mth
WHERE shown_days = 1)

SELECT MAX (DATE_PART ('minute', AGE (end_of_rent, start_of_rent))) AS maximum_rental_time_in_min,
        ROUND (AVG (DATE_PART ('minute', AGE (end_of_rent, start_of_rent))) :: NUMERIC, 2) AS average_rental_time_in_min
FROM t_train
```

Station information – top 3 starting points (stations) for bike rental

Model B

```
WITH t2 AS (WITH CTE as
                (SELECT * FROM train_model_b)
SELECT DISTINCT start_station,
        SUM (number_of_rental) AS count_of_bicycle_rented_at_station
FROM CTE
WHERE shown_days = 1
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5)
SELECT DISTINCT
    place_name AS docking_station_name,
        count_of_bicycle_rented_at_station
```

```
FROM t2
JOIN station s
ON t2.start_station = s.place_id
ORDER BY 2 DESC
LIMIT 5
```

Station information – top 3 end points (stations) following a bike rental

Model B

```
WITH t2 AS (WITH CTE as (SELECT * FROM train_model_b)
SELECT DISTINCT end_station,
        SUM (number_of_rental) AS count_of_bicycle_rented_at_station
FROM CTE
WHERE shown_days = 1
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5)
SELECT DISTINCT
    place_name AS docking_station_name,
        count_of_bicycle_rented_at_station
FROM t2
JOIN station s
ON t2.end_station = s.place_id
ORDER BY 2 DESC
LIMIT 5
```

FURTHER EXPLORATORY ANALYSIS USING MODEL B (WITHHOLDING EVERY SECOND DAY)

Trend from days of the week

```
WITH dow AS
            (SELECT * FROM train_model_b)SELECT CASE
        WHEN day_of_week = 0 THEN 'Sunday'
        WHEN day_of_week = 1 THEN 'Monday'
        WHEN day_of_week = 2 THEN 'Tuesday'
        WHEN day_of_week = 3 THEN 'Wednesday'
        WHEN day_of_week = 4 THEN 'Thursday'
        WHEN day_of_week = 5 THEN 'Friday'
        ELSE 'Saturday'
        END AS days_of_the_week,
```

```
        SUM (number_of_rental) as rental_on_alternating_days
FROM dow
WHERE shown_days = 1
GROUP BY 1
ORDER BY 2 DESC
```

Hourly and 30 -min trend

```
WITH t AS (SELECT * FROM train_model_b)
SELECT  time_of_day_in_hours,
            HalfHour,
        SUM (number_of_rental) as rental_on_alternating_days
FROM t
WHERE shown_days = 1
GROUP BY 1,2
ORDER BY 3 DESC
```

Top 10 rental bike routes, from start station to end station

```
WITH CTE as (SELECT * FROM train_model_b)
SELECT DISTINCT route AS bike_route,
        SUM (number_of_rental) AS count_of_bicycle_trips_in_route
FROM CTE
WHERE shown_days = 1
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10
```

SQL query used to transfer the required data from the RDBMS to Power BI for visualization and report.

```
WITH mth AS (SELECT bicycle_id AS id,
                    start_time AS start_of_rent,
                    end_time AS end_of_rent,
                    CASE
                    WHEN DATE_PART ('day', start_time)
                    IN ('1', '3', '5', '7', '9', '11', '13', '15', '17', '19', '21', '23', '25', '27', '29', '31')
                    THEN 1 ELSE 0 END AS shown_days,
                    CASE
                    WHEN DATE_PART ('day', start_time)
                    IN ('1', '3', '5', '7', '9', '11', '13', '15', '17', '19', '21', '23', '25', '27', '29', '31')
```

```
                    THEN 0 ELSE 1 END AS withheld_days,
                start_location AS start_station,
                    end_location AS end_station
                    FROM train)
SELECT id,
                start_of_rent,
                end_of_rent,
                start_station,
                end_station
FROM mth
WHERE shown_days = 1
```

This query was executed on PostgreSQL to remove every second day from January – March, to ensure datasets were similar with April and May. The query output was saved in CSV format, and transformed using INDEX & MATCH function in MS Excel to integrate the docking station names into the train dataset from the station dataset. This combined table was extracted into Power BI for further transformations with Power Query before final analysis and visualization.

WEATHER AND BIKE RENTAL COUNT

Monthly relationship

The rental pattern across each month was analyzed in relation to several weather factors.

```
WITH r AS (WITH mth AS
                        (SELECT * FROM train_model_b)
SELECT rental_year,
month_number,
  CASE
  WHEN month_number = 1 THEN 'January'
  WHEN month_number = 2 THEN 'February'
  WHEN month_number = 3 THEN 'March'
  WHEN month_number = 4 THEN 'April'
  ELSE 'May'
  END AS months,
  SUM (number_of_rental) as bike_rental
FROM mth
```

```
WHERE shown_days = 1
GROUP BY 1,2,3
ORDER BY 2)  ,

w AS (SELECT
      EXTRACT(MONTH FROM time) AS weather_month,
      EXTRACT(YEAR FROM time) AS weather_year,
      ROUND (AVG(tempm):: NUMERIC, 2) AS average_temperature,
      ROUND (AVG(hum) :: NUMERIC,2 ) AS average_humidity,
      ROUND (AVG(wspdm) :: NUMERIC, 2) AS average_windspeed,
   ROUND (AVG(pressurem) :: NUMERIC, 2) AS average_pressure,
   COUNT (*) FILTER (WHERE fog) AS foggy,
  COUNT (*) FILTER (WHERE rain) AS rainy,
  COUNT (*) FILTER (WHERE snow) AS snowfall,
  COUNT (*) FILTER (WHERE hail) AS hail,
  COUNT (*) FILTER (WHERE thunder) AS thunderstorm
    FROM weather
  WHERE wspdm >= 0
    GROUP BY 1,2)
SELECT month_number, bike_rental,
     average_temperature, average_humidity, average_pressure, average_windspeed,
  foggy, rainy, snowfall, hail, thunderstorm
FROM r
JOIN w
ON r.month_number = w.weather_month AND r.rental_year = w.weather_year
ORDER BY r.month_number, r.month_number;
```

Hourly relationship

```
WITH r AS (WITH mth AS
(SELECT * FROM train_model_b)
SELECT rental_year,  rental_hour,
            SUM (number_of_rental) as bike_rental
FROM mth
WHERE shown_days = 1
GROUP BY 1,2 ORDER BY 2),
w AS (SELECT
      EXTRACT(hour FROM time) AS weather_hour,
      EXTRACT(YEAR FROM time) AS weather_year,
```

```
    ROUND (AVG(tempm) :: NUMERIC,2) AS average_temperature,
    ROUND (AVG(hum) :: NUMERIC,2) AS average_humidity,
        ROUND (AVG (pressurem) :: NUMERIC, 2) AS average_pressure,
    ROUND (AVG(wspdm) :: NUMERIC, 2) AS average_windspeed,
        COUNT (*) FILTER (WHERE fog) AS foggy,
        COUNT (*) FILTER (WHERE rain) AS rainy,
        COUNT (*) FILTER (WHERE snow) AS snowfall,
        COUNT (*) FILTER (WHERE hail) AS hail,
        COUNT (*) FILTER (WHERE thunder) AS thunderstorm
  FROM weather
      WHERE wspdm  >= 0
  GROUP BY 1,2)
SELECT rental_hour, bike_rental,
    average_temperature, average_humidity, average_pressure, average_windspeed,
        foggy, rainy, snowfall, hail, thunderstorm
FROM r
JOIN w
ON r.rental_hour = w.weather_hour AND r.rental_year = w.weather_year
ORDER BY r.rental_hour, r.rental_year;
```

The results from the above query for the monthly and hourly bike rental activity and weather factors were saved in CSV format. The file was subsequently extracted in MS Excel, and used to create correlation matrix to see the effect of weather factors on monthly rental information (See month_weather_correlation.xlsx and hour_weather_correlation.xlsx).

MONTHLY SPECIFIC INFORMATION

After determining the weather's effect on a monthly level, a more in-depth examination was conducted for each month, providing detailed insights into the influence of weather on bike rentals. Firstly, the rental count was estimated based on the day of the week, the day, and the hour. A comprehensive study was then undertaken to evaluate the impact of weather on renting patterns throughout the specified timeframe.

In the SQL query under this section, replacing the **WHERE monthly = 1** with **(2,3,4,5)** provided information for each month.

Day-to-day rental information in each month

```
WITH sub as (SELECT DATE_PART ('month', start_time) AS monthly,
            DATE_PART ('day', start_time) AS day_number,
            COUNT (*) AS number_of_rental
FROM train
GROUP BY 1,2
ORDER BY 2)

SELECT day_number,
           number_of_rental
FROM sub
WHERE monthly = 1  --(OR 2,3,4,5)
ORDER BY 2 DESC
```

Day-to-day information on several weather parameters in each month

A. Average of weather parameters

```
WITH sub as (SELECT DATE_PART ('month', time) AS monthly,
            DATE_PART ('day', time) AS day_number,
              tempm,  hum,   wspdm, pressurem
FROM weather)
SELECT day_number,
                  ROUND (AVG (tempm)::NUMERIC, 2) AS average_daily_temp_in_January,
                  ROUND (AVG (hum) :: NUMERIC, 2) AS average_daily_hum_in_January,
          ROUND (AVG (wspdm)::NUMERIC, 2) AS average_daily_windspeed_in_January,
          ROUND (AVG (pressurem)::NUMERIC, 2) AS average_daily_pressure_in_January
FROM sub
WHERE monthly = 1 --(OR 2,3,4,5)
GROUP BY 1
ORDER BY 2 DESC
```

B. Occurrence of weather parameters.

```
SELECT EXTRACT(DAY FROM time) AS day_number,
            CASE
            WHEN EXTRACT(MONTH FROM time) = 1 THEN 'January'
            WHEN EXTRACT(MONTH FROM time) = 2 THEN 'February'
            WHEN EXTRACT(MONTH FROM time) = 3 THEN 'March'
            WHEN EXTRACT(MONTH FROM time) = 4 THEN 'April'
            ELSE 'May' END AS month_names,
            COUNT (*) FILTER (WHERE fog) AS foggy,
            COUNT (*) FILTER (WHERE rain) AS rainy,
```

```
        COUNT (*) FILTER (WHERE snow) AS snowfall,
        COUNT (*) FILTER (WHERE hail) AS hail,
        COUNT (*) FILTER (WHERE thunder) AS thunderstorm
FROM weather
WHERE EXTRACT(MONTH FROM time) = 1 --(OR 2,3,4,5)
GROUP BY 1,2
ORDER BY 1
```

Days of the week rental information in each month

```
WITH sub as (SELECT DATE_PART ('month', start_time) AS monthly,
                    CASE
        WHEN DATE_PART ('dow', start_time) = 0 THEN 'Sunday'
        WHEN DATE_PART ('dow', start_time) = 1 THEN 'Monday'
        WHEN DATE_PART ('dow', start_time) = 2 THEN 'Tuesday'
        WHEN DATE_PART ('dow', start_time) = 3 THEN 'Wednesday'
        WHEN DATE_PART ('dow', start_time) = 4 THEN 'Thursday'
        WHEN DATE_PART ('dow', start_time) = 5 THEN 'Friday'
        ELSE 'Saturday'
        END AS days_of_the_week,
        COUNT (*) AS number_of_rental
FROM train
GROUP BY 1,2
ORDER BY 2)
SELECT days_of_the_week,
       number_of_rental
FROM sub
WHERE monthly = 1 --(OR 2,3,4,5)
ORDER BY 2 DESC
```

Days of the week information on several weather parameters in each month

A. Average of weather parameters

```
WITH sub as (SELECT DATE_PART ('month', time) AS monthly,
                    CASE
        WHEN DATE_PART ('dow', time) = 0 THEN 'Sunday'
        WHEN DATE_PART ('dow', time) = 1 THEN 'Monday'
        WHEN DATE_PART ('dow', time) = 2 THEN 'Tuesday'
        WHEN DATE_PART ('dow', time) = 3 THEN 'Wednesday'
        WHEN DATE_PART ('dow', time) = 4 THEN 'Thursday'
        WHEN DATE_PART ('dow', time) = 5 THEN 'Friday'
        ELSE 'Saturday'
```

```
        END AS days_of_the_week,
         tempm,  hum,  wspdm,  pressurem
FROM weather)
SELECT days_of_the_week,
            ROUND (AVG (tempm)::NUMERIC, 2) AS average_daily_temp_in_January,
            ROUND (AVG (hum) :: NUMERIC, 2) AS average_daily_hum_in_January,
      ROUND (AVG (wspdm)::NUMERIC, 2) AS average_daily_windspeed_in_January,
      ROUND (AVG (pressurem)::NUMERIC, 2) AS average_daily_pressure_in_January
FROM sub
WHERE monthly = 1 --(OR 2,3,4,5)
GROUP BY 1
ORDER BY 2 DESC
```

B. Occurrence of weather parameters.

```
SELECT  CASE
        WHEN DATE_PART ('dow', time) = 0 THEN 'Sunday'
        WHEN DATE_PART ('dow', time) = 1 THEN 'Monday'
        WHEN DATE_PART ('dow', time) = 2 THEN 'Tuesday'
        WHEN DATE_PART ('dow', time) = 3 THEN 'Wednesday'
        WHEN DATE_PART ('dow', time) = 4 THEN 'Thursday'
        WHEN DATE_PART ('dow', time) = 5 THEN 'Friday'
        ELSE 'Saturday'
        END AS days_of_the_week,
        COUNT (*) FILTER (WHERE fog) AS foggy,
        COUNT (*) FILTER (WHERE rain) AS rainy,
        COUNT (*) FILTER (WHERE snow) AS snowfall,
        COUNT (*) FILTER (WHERE hail) AS hail,
        COUNT (*) FILTER (WHERE thunder) AS thunderstorm
FROM weather
WHERE EXTRACT(MONTH FROM time) = 1 --(OR 2,3,4,5)
GROUP BY 1
ORDER BY 3 DESC
```

Hourly information on rental patterns in each month

```
WITH sub as (SELECT DATE_PART ('month', start_time) AS monthly,
                    DATE_PART ('hour', start_time) AS time_of_the_day,
                    COUNT (*) AS number_of_rental
FROM train
GROUP BY 1,2
```

```
ORDER BY 2)

SELECT time_of_the_day,
        number_of_rental
FROM sub
WHERE monthly = 1 --(OR 2,3,4,5)
ORDER BY 2 DESC
```

Hourly information on several weather parameters in each month

A. Average of weather parameters

```
WITH sub as (SELECT DATE_PART ('month', time) AS monthly,
                    DATE_PART ('hour', time) AS time_of_the_day,
          tempm,  hum,   wspdm,  pressurem
FROM weather)
SELECT time_of_the_day,
            ROUND (AVG (tempm)::NUMERIC, 2) AS average_daily_temp_in_January,
            ROUND (AVG (hum) :: NUMERIC, 2) AS average_daily_hum_in_January,
       ROUND (AVG (wspdm)::NUMERIC, 2) AS average_daily_windspeed_in_January,
       ROUND (AVG (pressurem)::NUMERIC, 2) AS average_daily_pressure_in_January
FROM sub
WHERE monthly = 1 --(OR 2,3,4,5)
GROUP BY 1
ORDER BY 2 DESC
```

B. Occurrence of weather parameters.

```
SELECT DATE_PART ('hour', time) AS time_of_the_day,
          COUNT (*) FILTER (WHERE fog) AS foggy,
        COUNT (*) FILTER (WHERE rain) AS rainy,
        COUNT (*) FILTER (WHERE snow) AS snowfall,
        COUNT (*) FILTER (WHERE hail) AS hail,
        COUNT (*) FILTER (WHERE thunder) AS thunderstorm
FROM weather
WHERE EXTRACT(MONTH FROM time) = 1 --(OR 2,3,4,5)
GROUP BY 1
ORDER BY 4 DESC
```