Pattern_analysis_for_ride_sharing_company_part

STEP 1: Parsing data from website

```
import requests
from bs4 import BeautifulSoup
import pandas as pd


URL = 'https://code.s3.yandex.net/data-analyst-eng/chicago_weather_2017.html'
req = requests.get(URL) # saving response object as req variable

soup = BeautifulSoup(req.text, 'lxml')

table = soup.find("table", attrs={"id": "weather_records"})

heading_table = [] # List where the names of the columns will be stored
# The names of the columns are inside <th> elements, so we'll find all <th> elements in the table and run them through in a loop. Then add

for row in table.find_all('th'):
        heading_table.append(row.text)

# Create an empty list where the table data will be stored
content=[]

# Each row is wrapped in a <tr> tag, we need to loop through all the rows. Within each row the cell content is wrapped in <td> </td> tags.

for row in table.find_all('tr'):
    if not row.find_all('th'):
# We need this condition to ignore the first row of the table, with headings
            content.append([element.text for element in row.find_all('td')])

# pass two-dimensional content list as data and heading_table as headings
weather_records = pd.DataFrame(content, columns=heading_table)
print(weather_records)
```

Step 2: Exploratory data analysis

1. Find the number of taxi rides for each taxi company for November 15-16, 2017. Name the resulting field *trips_amount* and print it along with the *company_name* field. Sort the results by the *trips_amount* field in descending order.

```
SELECT
    company_name,
    COUNT(DISTINCT trip_id :: real) AS trips_amount
FROM
    cabs
INNER JOIN
trips ON cabs.cab_id = trips.cab_id
WHERE
    trips.start_ts :: date IN ('2017-11-15','2017-11-16')
GROUP BY
    company_name
ORDER BY
    trips_amount DESC;
```

2. Find the number of rides for every taxi company whose name contains the words "Yellow" or "Blue" for November 1-7, 2017. Name the resulting variable *trips_amount*. Group the results by the *company_name* field.

```
SELECT
    company_name,
    trips_amount
FROM
(SELECT
```

```
    cabs.company_name,
    COUNT(DISTINCT trip_id :: real) AS trips_amount
FROM
    cabs
INNER JOIN
trips ON cabs.cab_id = trips.cab_id
WHERE
    trips.start_ts :: date BETWEEN '2017-11-1' AND '2017-11-7' AND
    cabs.company_name LIKE '%Yellow%'
GROUP BY
    company_name
UNION
SELECT
    cabs.company_name,
    COUNT(DISTINCT trip_id :: real) AS trips_amount
FROM
    cabs
INNER JOIN
trips ON cabs.cab_id = trips.cab_id
WHERE
    trips.start_ts :: date BETWEEN '2017-11-1' AND '2017-11-7' AND
    cabs.company_name LIKE '%Blue%'
GROUP BY
    company_name
) AS SUBQ

ORDER BY
    trips_amount DESC;
```

3. In November 2017, the most popular taxi companies were Flash Cab and Taxi Affiliation Services. Find the number of rides for these two companies and name the resulting variable *trips_amount*. Join the rides for all other companies in the group "Other." Group the data by taxi company names. Name the field with taxi company names *company*. Sort the result in descending order by *trips_amount*.

```
SELECT
    CASE
        WHEN cabs.company_name = 'Flash Cab' THEN 'Flash Cab'
        WHEN cabs.company_name = 'Taxi Affiliation Services'  THEN 'Taxi Affiliation Services'
        ELSE 'Other'
    END AS company,
    COUNT(DISTINCT trips.trip_id) AS trips_amount

FROM
    cabs
INNER JOIN trips ON cabs.cab_id = trips.cab_id

WHERE
    start_ts::date BETWEEN '2017-11-01'AND '2017-11-7'

GROUP BY
    company
ORDER BY
    trips_amount DESC;
```

Step 3. Test the hypothesis that the duration of rides from the the Loop to O'Hare International Airport changes on rainy Saturdays.

1. Retrieve the identifiers of the O'Hare and Loop neighborhoods from the *neighborhoods* table.

```
SELECT
    neighborhood_id,
    name
FROM
    neighborhoods
WHERE
    name LIKE '%Hare' OR name LIKE 'Loop';
```

2. For each hour, retrieve the weather condition records from the *weather_records* table. Using the CASE operator, break all hours into two groups: "Bad" if the *description* field contains the words '"rain" or "storm," and "Good" for others. Name the resulting field *weather_conditions*. The final table must include two fields: date and hour (*ts*) and *weather_conditions*.

```
SELECT
    ts,
    CASE
        WHEN description LIKE '%rain%' OR description LIKE '%storm%' THEN 'Bad'
        ELSE 'Good'
    END AS WEATHER_CONDITIONS
FROM
    weather_records;
```

3. For each hour, retrieve the weather condition records from the *weather_records* table. Using the CASE operator, break all hours into two groups: "Bad" if the *description* field contains the words "rain" or "storm," and "Good" for others. Name the resulting field *weather_conditions*. The final table must include two fields: date and hour (*ts*) and *weather_conditions*.

```
SELECT
    start_ts,
    CASE
        WHEN description LIKE '%rain%' OR description LIKE '%storm%' THEN 'Bad'
        ELSE 'Good'
    END AS weather_conditions,
    duration_seconds
FROM
    trips
INNER JOIN weather_records ON weather_records.ts = trips.start_ts

WHERE
    pickup_location_id = 50 AND dropoff_location_id = 63 AND
    EXTRACT(dow FROM start_ts) = 6

ORDER BY
    trip_id;
```