MOTOR VEHICLE COLLISIONS NYC - 2018

PROJECT MEMBERS

GIULIA SARESINI (MAT. 864967)
G.SARESINI@CAMPUS.UNIMIB.IT

SARA NAVA (MAT. 870885)
S.NAVA38@CAMPUS.UNIMIB.IT

University of Milano-Bicocca
Department of Computer Science, Systems and Communication

Master's Degree in Data Science – Data Management Course
Academic Year 2023-2024

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Introduction

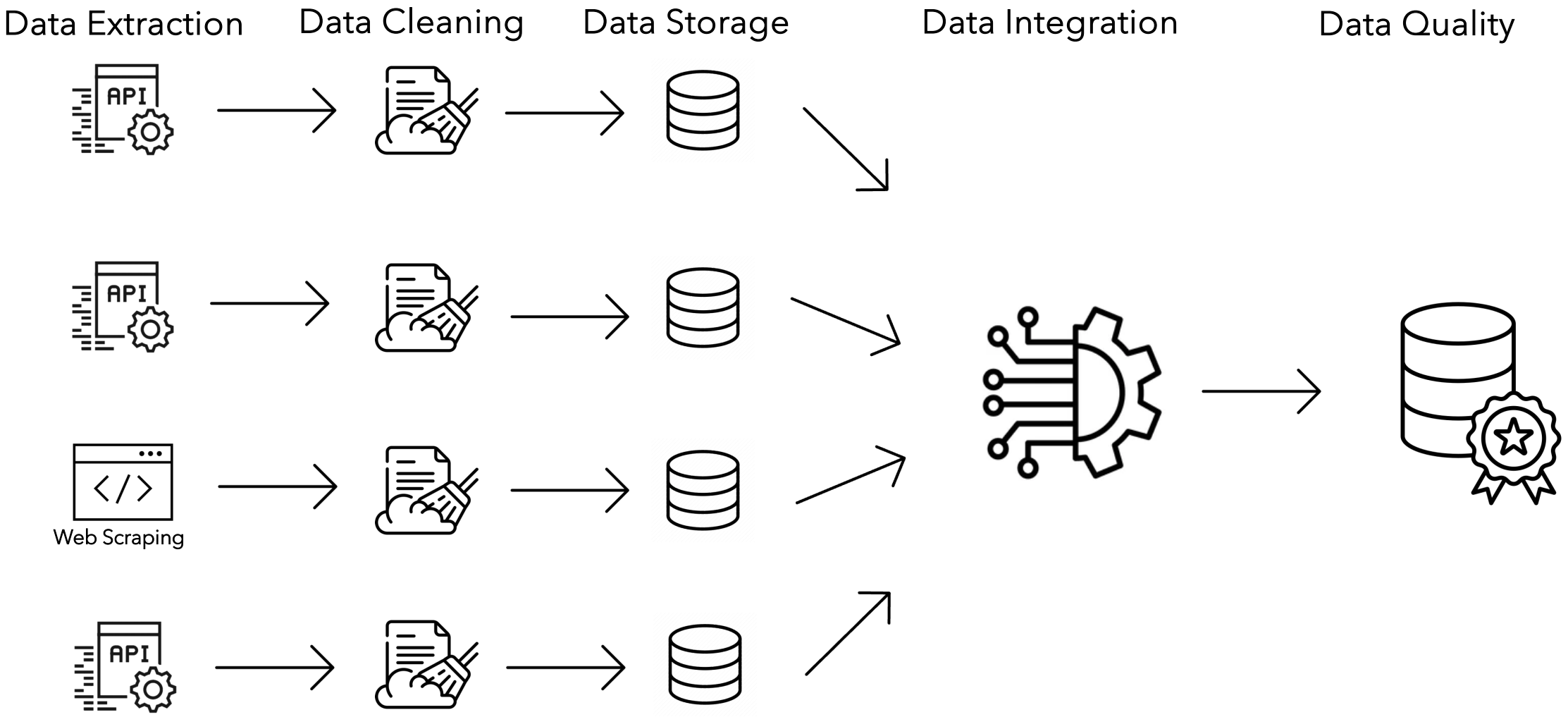
Our project aims to integrate data regarding the **incidents** that **occurred** in **New York City in 2018** with other data sources providing information about:

vehicles involved in the incidents

weather conditions present at the time of the incident

date on which they **occurred**

Project Structure



Used Tools

Data Extraction

Data Cleaning

Python

Data Storage

Data Integration

Data Quality

SQLite as DB

MacOS Terminal

VS Code's Extensions

Vehicles and Collisions Datasets

We retrieved these two information from **NYC Open Data**, a platform that hosts a vast collection of data related to the city of New York, through API requests, selecting only the information of interest.

Collisions API Docs for Collisions

collision_id
borough
zip_code
latitude
longitude
crash_date
crash_time
number_of_persons_injured
number_of_persons_killed

It contains details of **NYC's collisions in 2018**. Each row corresponds to a specific collision.

Vehicles API Docs for Vehicles

unique_id
collision_id
pre_crash
vehicle_occupants
contributing_factor_1
driver_license_status

It contains **details of vehicles involved** in NYC's collisions in 2018.

Vehicles and Collisions Datasets

We retrieved these two information from **NYC Open Data**, a platform that hosts a vast collection of data related to the city of New York, through API requests, selecting only the information of interest.

To make the request it was necessary to create a **Socrata Client** in which specify the API URL, the API token, the username and the related password.

```
client = Socrata("data.cityofnewyork.us",  token, username, password)
```

After saving the parameters to be inserted into the **GET** method as global variables, specifically a string containing the list of variables to select separated by commas, and two variables containing the start date and the end date, it was possible to make a 'filtered request' using **SoQL Queries** as well.

```
client.get(dataset_id, select=vars_to_select, where=f"crash_date >= '{start_date_formatted}'
AND crash_date <= '{end_date_formatted}'", limit=600000)
```

Calendar Dataset

We retrieved this dataset from **timeanddate** web site through **web scraping**.

Data was organized into a table, so we used **BeautifulSoup** and **requests** libraries in Python to extract information from the related HTML file. Then, we created a new table where **each date** was paired with its corresponding **day of the week** (based on the column index in the web site table) and **checked whether it was a federal holiday** based on the color coding (red for holidays, black for regular days).

We repeat the process for each months of 2018 and the concatenate the results.

Calendar for January 2018 (United States)

January

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Phases of the Moon: 1○ 8◑ 16● 24◐ 31○

Holidays: 1: New Year's Day, 15: Martin Luther King Jr. Day

	Date	Day	Flag_Holiday
0	2018-01-01	Monday	True
1	2018-01-02	Tuesday	False
2	2018-01-03	Wednesday	False
3	2018-01-04	Thursday	False
4	2018-01-05	Friday	False
...	...	...	...
26	2018-12-27	Thursday	False
27	2018-12-28	Friday	False
28	2018-12-29	Saturday	False
29	2018-12-30	Sunday	False
30	2018-12-31	Monday	False

365 rows × 3 columns

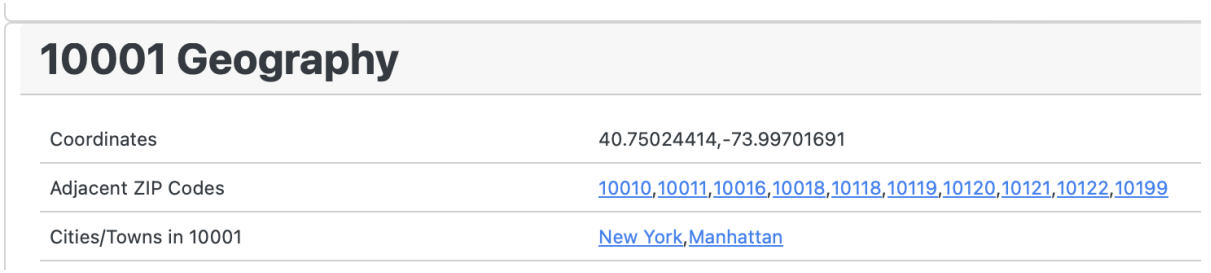
Weather Dataset

We retrieved this dataset from **Open-meteo** API.

To obtain the weather conditions for all zip codes in New York City, we first needed to **acquire the list of zip codes**, that can be retrieved from NYC Open Data.

Then, to get the **geographical coordinates associated to each zip code** we made a web scraping from ZipDataMaps web site.

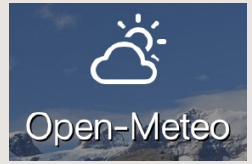
Also there, data was organized in a table so, to extract latitude and longitude, we identified the row in the table containing 'Coordinates' and extracted the remaining text from that cell to obtain the coordinates values.



10001 Geography

Coordinates	40.75024414,-73.99701691
Adjacent ZIP Codes	10010,10011,10016,10018,10118,10119,10120,10121,10122,10199
Cities/Towns in 10001	New York,Manhattan

Weather Dataset



After acquiring a dataset containing the zip codes of NYC along with their corresponding geographical coordinates, we proceeded to make **API requests to the Open-Meteo API**.

We specified the **input parameters required for the request**, including the start date, end date, lists of latitudes and longitudes, and a list of variables to retrieve from the API.

```
params = {
    "latitude": list of latitudes,
    "longitude": list of longitudes,
    "hourly": ["temperature_2m", "relative_humidity_2m", "rain", "snowfall", "cloud_cover"],
    "timezone": "auto",
    "start_date": start date,
    "end_date": stop date
 }
url = "https://archive-api.open-meteo.com/v1/archive"
response = openmeteo.weather_api(url, params=params)
```

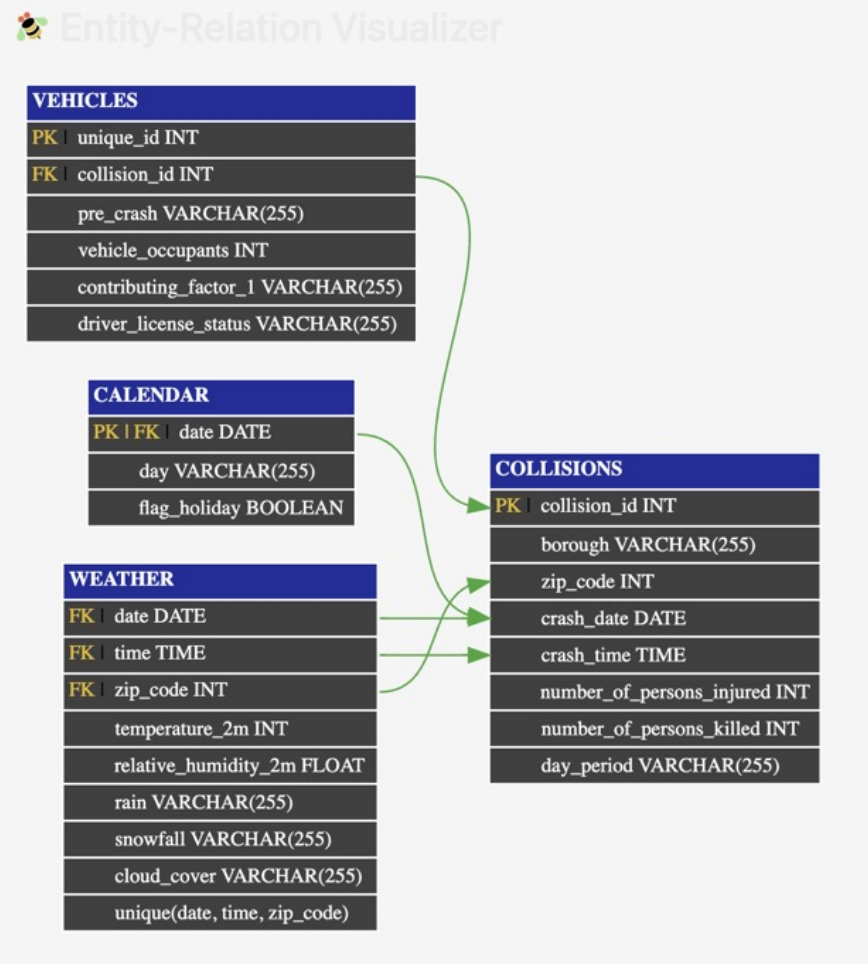
→ Hourly variables to retrieve

→ API Endpoint

→ Request

The final dataset contains **hourly weather data for all of 2018** for each zip code associated with NYC.

Data Storage



Having access to 4 data sources where there is an evident relationship between the Vehicles, Calendar, and Weather datasets with the Collision dataset, and considering the need to combine these data sources to answer our initial questions, we opted for a **relational database structure**.

To access the database, we utilized the **Database Client** extension in Visual Studio Code. For creating the Entity-Relationship (ER) schema, we opted for the **dBizzy** extension.

Database Client

dBizzy

Data Integration

After loading the data into database tables, we combined them using an SQL query with primary and foreign keys. We used an INNER JOIN from the main table (Collisions) instead of a LEFT JOIN. This decision was made because some data cleaning operations removed information from the Vehicles dataset, and some ZIP CODES in the Weather dataset were not recognized as valid. Therefore, to avoid missing fields in the resulting dataset, we focused on common observations among the datasets.

```
CREATE VIEW MVC_DATA AS
    SELECT A.*
        , B.pre_crash, B.vehicle_occupants,
        B.contributing_factor_1 as contributing_factor,
        B.driver_license_status
        , C.day, C.flag_holiday
        , D.cloud_cover, D.rain, D.relative_humidity_2m,
        D.snowfall, D.temperature_2m
    FROM "COLLISIONS" AS A
    INNER JOIN "VEHICLES" AS B
    ON A.collision_id = B.collision_id
    INNER JOIN "CALENDAR" as C
    ON A.crash_date = C."date"
    INNER JOIN "WEATHER" AS D
    ON A.crash_date = D."date"
        AND A.crash_time = D.time
        AND A.zip_code = D.zip_code;
```

Data Cleaning - Collisions

Before

collision_id INT	borough VARCHAR	zip_code INT	latitude FLOAT	longitude FLOAT	crash_date TIMESTAMP	crash_time VARCHAR(2	number_of_persons_injured INT	number_of_persons_killed INT
3820157	MANHATTAI	10025	40.8018	-73.96108	2018-01-01T00:00:00.0(	4:16	1	0
3818846	QUEENS	11373	40.743973	-73.8851	2018-01-01T00:00:00.0(	20:30	0	0
3820776			40.72143	-73.892746	2018-01-01T00:00:00.0(	23:41	0	0
3818947	MANHATTAI	10025	40.80174	-73.96477	2018-01-01T00:00:00.0(	15:30	0	0
3820540			40.666225	-73.80086	2018-01-01T00:00:00.0(	15:00	0	0
3820645	QUEENS	11354	40.763073	-73.816345	2018-01-01T00:00:00.0(	12:10	0	0
3819261	BRONX	10459	40.820305	-73.89083	2018-01-01T00:00:00.0(	18:35	0	0

After

collision_id INT	borough VARCHAR	zip_code INT	crash_date DATE	crash_time TIME	number_of_persons_inj INT	number_of_persons_kill INT	day_period VARCHAR(255)
3820157	MANHATTAI	10025	2018-01-01	04:00:00	1	0	Night
3818846	QUEENS	11373	2018-01-01	20:00:00	0	0	Evening
3818947	MANHATTAI	10025	2018-01-01	15:00:00	0	0	Afternoon
3820645	QUEENS	11354	2018-01-01	12:00:00	0	0	Night
3819261	BRONX	10459	2018-01-01	18:00:00	0	0	Evening
3820853	BROOKLYN	11207	2018-01-01	13:00:00	0	0	Afternoon

Data Cleaning - Vehicles

Before

unique_ INT	collision_id INT	pre_crash VARCHAR(255)	vehicle_occupants INT	contributing_factor_1 VARCHAR(255)	driver_license_status VARCHAR(255)
17675653	3824399	Going Straight Ahead			
17675094	3820775	Parked	0	Unspecified	
17674184	3819574	Going Straight Ahead	1	Unspecified	Licensed
17674943	3818864	Parked	0	Unspecified	
17674640	3819262	Going Straight Ahead	2	Unspecified	Licensed

After

unique_ INT	collision_id INT	pre_crash VARCHAR(255)	vehicle_occupants INT	contributing_factor_1 VARCHAR(255)	driver_license_status VARCHAR(255)
17675653	3824399	Going Straight Ahead			
17674184	3819574	Going Straight Ahead	1	Unspecified	Licensed
17674640	3819262	Going Straight Ahead	2	Unspecified	Licensed
18490595	4060558	Going Straight Ahead	1	Driver Inattention/Distract	
17674359	3821451	Starting from Parking	1	Unsafe Lane Changing	

Data Cleaning - Weather

Before

date TIMESTAMP	temperature_2m FLOAT	relative_humidity_2m FLOAT	rain FLOAT	snowfall FLOAT	cloud_cover FLOAT	zip_code INT
2018-01-01 00:00:00	-11.676	48.09236	0.0	0.0	0.0	10001
2018-01-01 01:00:00	-11.876	48.238712	0.0	0.0	0.0	10001
2018-01-01 02:00:00	-12.226	48.972702	0.0	0.0	0.0	10001
2018-01-01 03:00:00	-12.526	49.521194	0.0	0.0	0.0	10001

After

date DATE	time TIME	zip_code INT	temperature_2m INT	relative_humidity_2m FLOAT	rain VARCHAR	snowfall VARCHAR	cloud_cover VARCHAR(25
2018-01-01	17:00:00	10001	-9	0.3	No Rain	No Snow	No Cloud
2018-01-01	18:00:00	10001	-8	0.3	No Rain	No Snow	No Cloud
2018-01-01	19:00:00	10001	-7	0.2	No Rain	No Snow	Cloud
2018-01-01	20:00:00	10001	-7	0.2	No Rain	No Snow	Cloud

Data Quality - Completeness

Only the fields **vehicle_occupants**, **contributing_factor**, and **driver_license_status** contain missing values.

With a total of 390.589 observations in the dataset MVC_DATA, the **percentage of missing** values for these fields is **less than 50%.** Therefore, we opted not to remove these columns as they are relevant to our research questions. Directly removing instances with missing values would result in losing approximately 20% of the dataset, which could lead to significant data loss.

	VARIABLE	NULL_COUNT
1	unique_id	0
2	borough	0
3	zip_code	0
4	crash_date	0
5	crash_time	0
6	number_of_persons_injured	0
7	number_of_persons_killed	0
8	day_period	0
9	pre_crash	0
10	vehicle_occupants	12321
11	contributing_factor	576
12	driver_license_status	68701
13	day	0
14	flag_holiday	0
15	cloud_cover	0
16	rain	0
17	relative_humidity_2m	0
18	snowfall	0
19	temperature_2m	0

Data Quality – Syntattic Accuracy

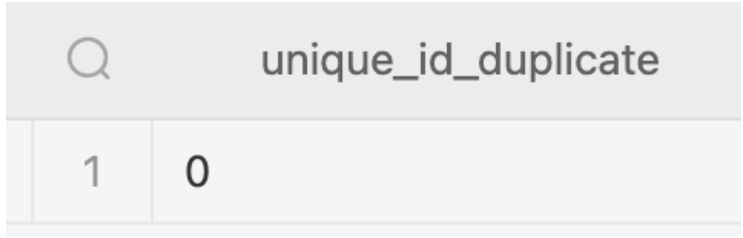
We then considered it useful to assess the syntactic accuracy of the qualitative variables in the table that had a finite domain. We then examined the values in the **borough** field, verifying that there were no data with values other than the five possible districts in the city of New York. Additionally, we conducted the same check on the **day** field, confirming the absence of values outside the relevant domain of the variable.

	borough VARCHAR(255)
1	BRONX
2	BROOKLYN
3	MANHATTAN
4	QUEENS
5	STATEN ISLAND

	day VARCHAR(255)
1	Friday
2	Monday
3	Saturday
4	Sunday
5	Thursday
6	Tuesday
7	Wednesday

Data Quality – Consistency

Since the unique_id field remained as the primary key of the table, which identifies the vehicle involved in a specific incident (and not the vehicle itself, as it can be involved in multiple incidents during the year and therefore cannot function as the primary key), **we verified if**, after the join, there **were any duplicates in the data**. The verification revealed that the data was integrated correctly.

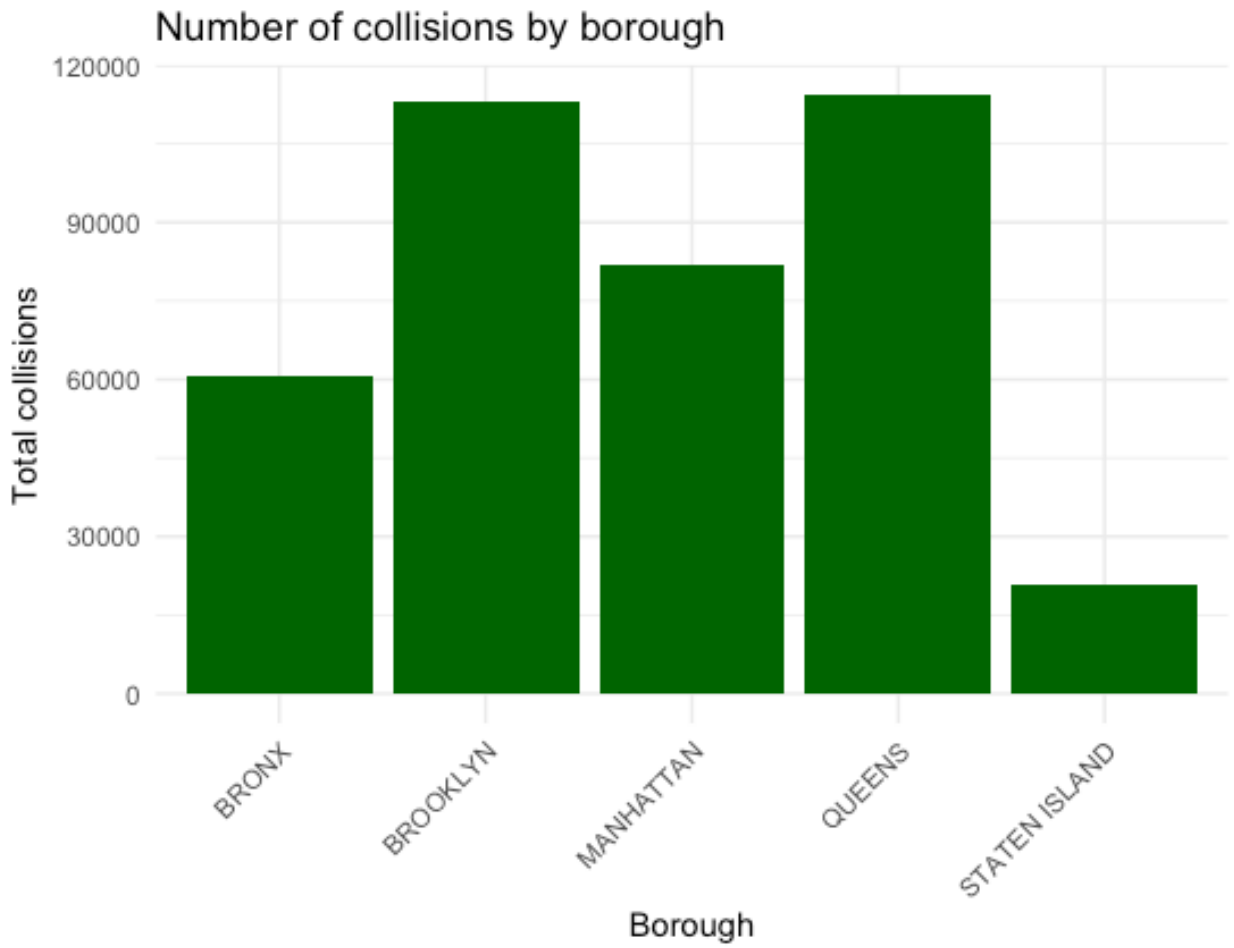


	unique_id_duplicate
1	0

Data Querying – Question A

GOAL: observing the distribution of collisions across the five boroughs of New York City (Brooklyn, Manhattan, Queens, Staten Island, Bronx).

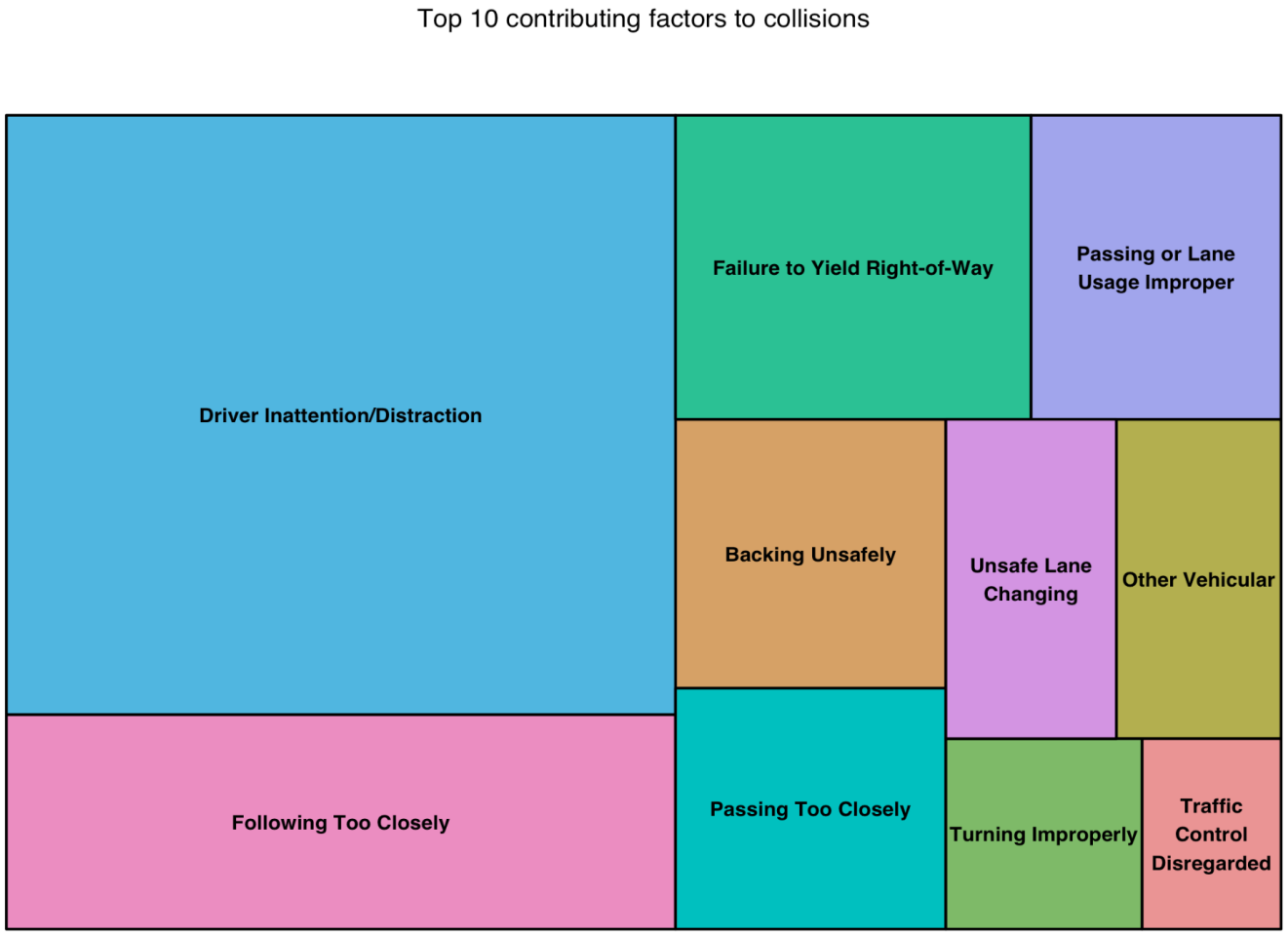
OBSERVATION: the districts that appear to have a significantly higher number of collisions for the year 2018 are Queens and Brooklyn.



Data Querying – Question B

GOAL: investigating the pre-collision dynamics to identify situations that may lead to collisions more frequently.

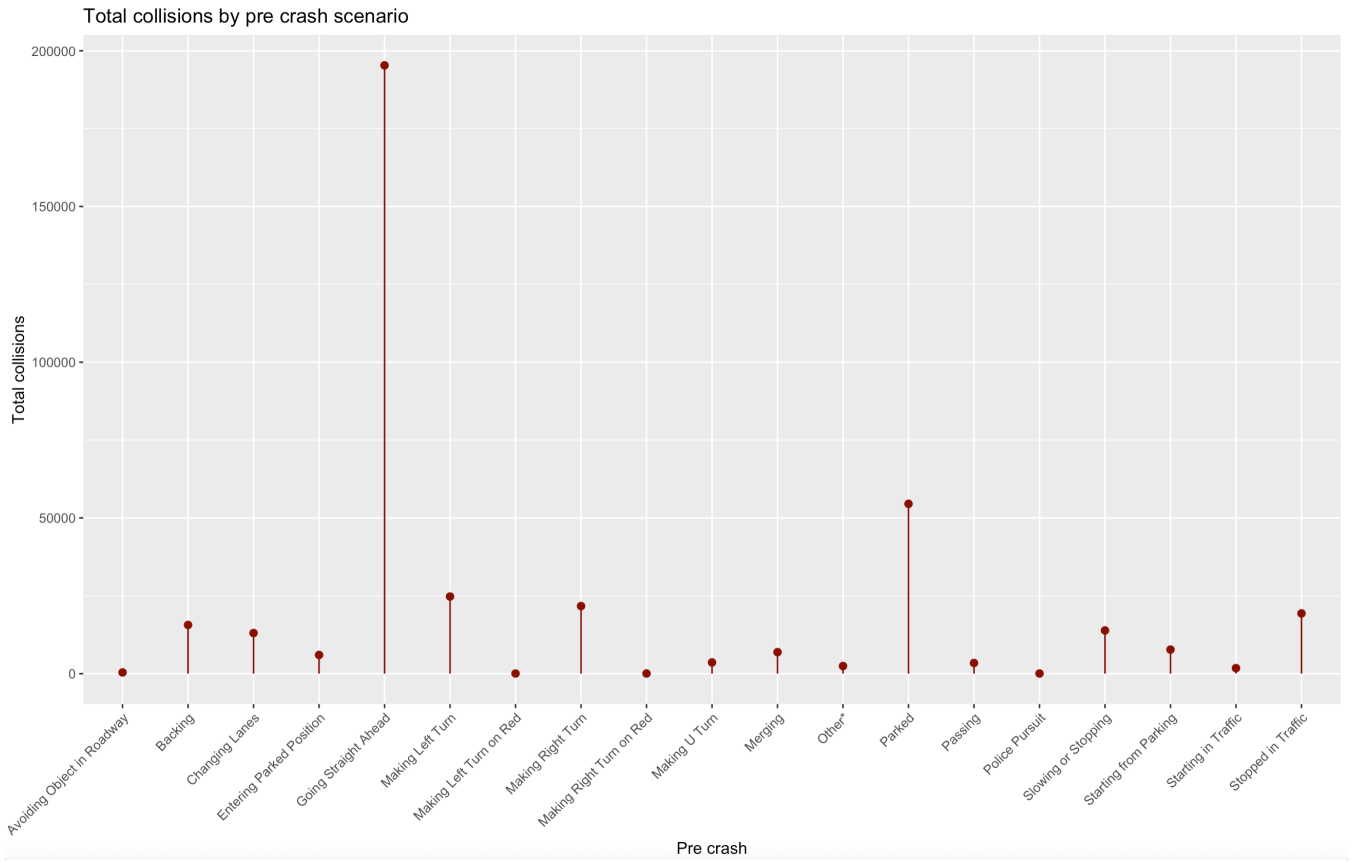
OBSERVATION: the most common cause of accidents is driver inattention, followed by 'following too closely' (which often leads to rear-end collisions).



Data Querying – Question C

GOAL: evaluating which driving behaviours may increase the probability of a collision.

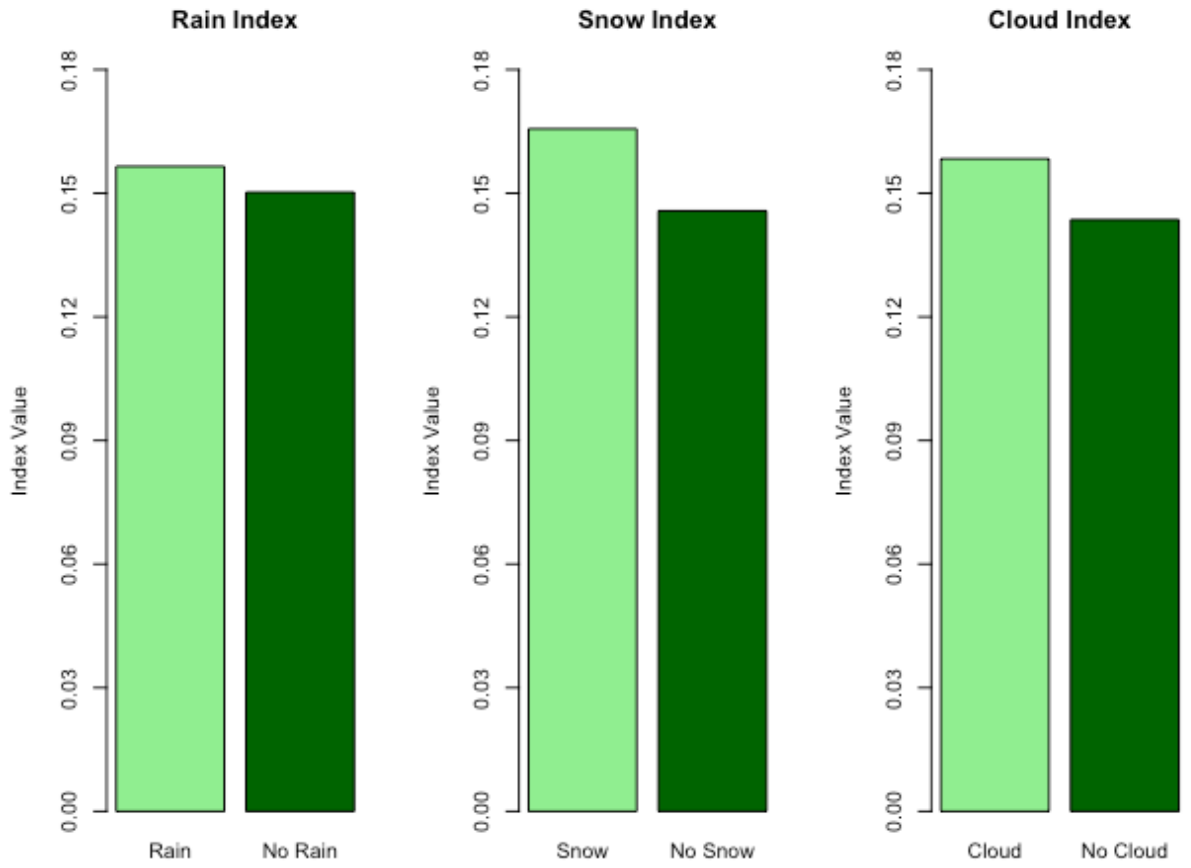
OBSERVATION: in the moments before the incident, most of the cars were traveling straight; a second factor that stands out more than others is the number of parked cars that were involved in the accidents.



Data Querying – Question D

GOAL: exploring the meteorological conditions in which collisions primarily occur.

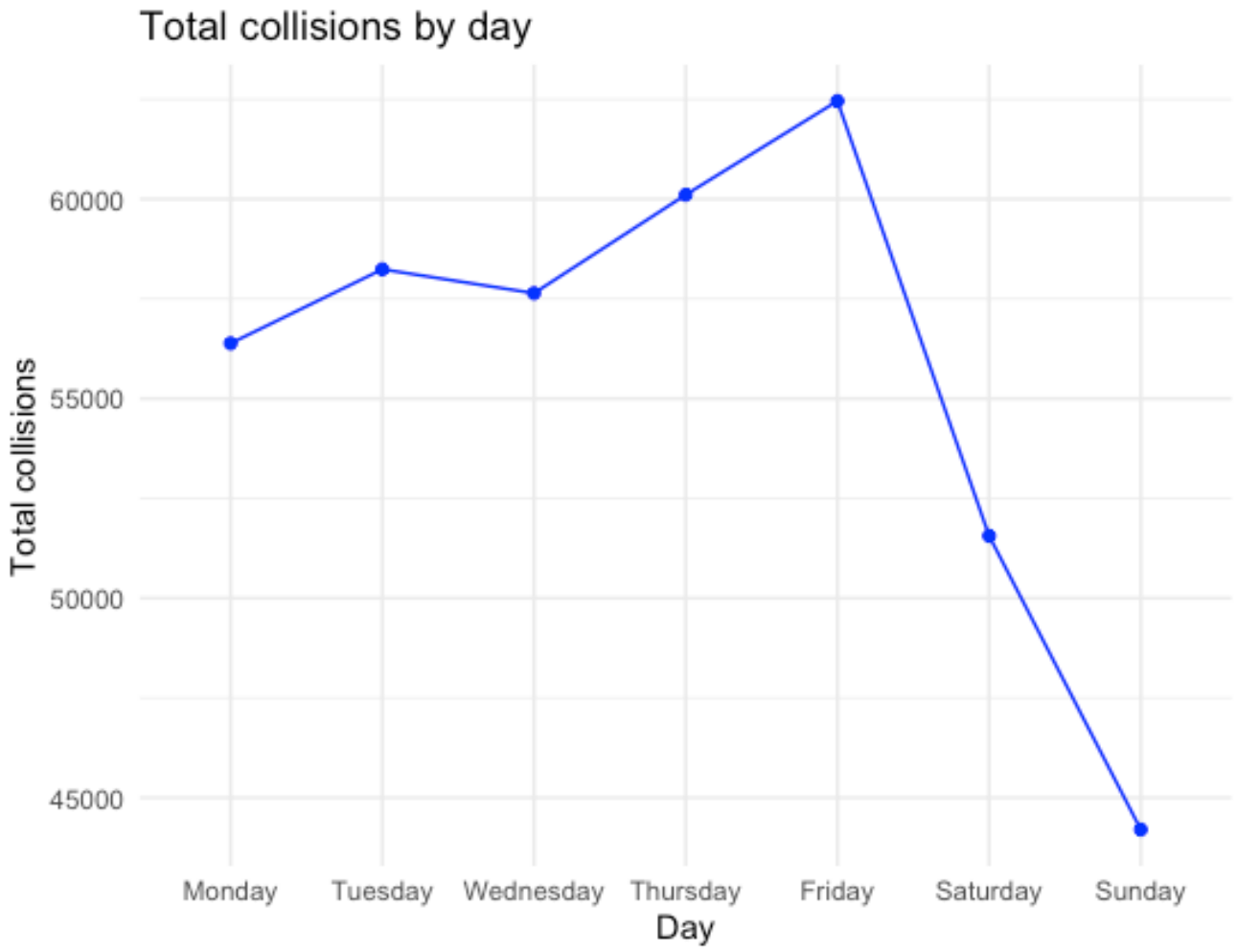
OBSERVATION: unlike what we expected, weather conditions do not seem to be relevant regarding the likelihood of having accidents. There is a slight difference regarding snow indices, but we cannot draw conclusions; it might be the case.



Data Querying – Question E

GOAL: verifying if there are certain days of the week or holidays when collisions occur more frequently, and if there is a variation compared to the daily average.

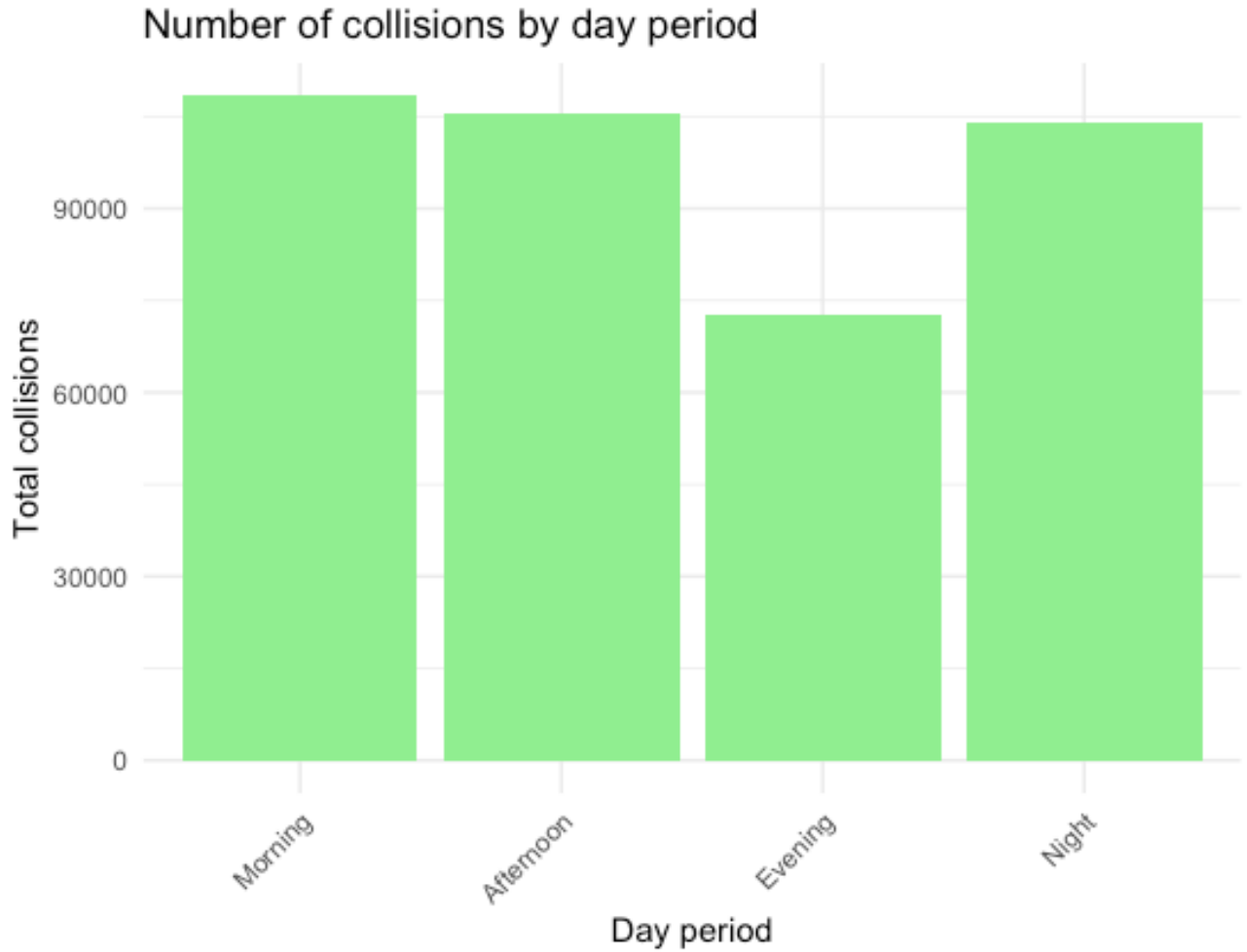
OBSERVATION: most accidents occur between Monday and Friday, decreasing on the weekends.



Data Querying – Question F

GOAL: analysing if there are specific moments of the day when there is a higher concentration of collisions.

OBSERVATION: there is no evidence that more accidents occur during certain time periods, except for the evening, which has about 30,000 fewer accidents in 2018 compared to morning, afternoon, and night.



THANK YOU FOR YOUR ATTENTION

Giulia Saresini, Sara Nava