# Data Analysis Project
# P3: Data Analysis and Visualization
# Impact of Weather on Traffic Intensity, Speed Violations and Accidents in Basel
# (Group 8)

Daniel Madoery and Mark Starzynski

University of Basel
Databases (CS244) course
Autumn Semester 2022

## 1 Introduction

It is generally assumed that weather effects can have an impact on traffic density, which in turn is a critical factor for traffic safety. This project aims to compare weather data with traffic density, speed limit violations and the amount of accidents over a period of at least one year, specifically in Basel.

Keeping in mind that this is a learning project, the initial goal was to get many different datasets in order for integration to become more meaningful and educational. 6 datasets were selected in the end. We have picked mostly high temporal data sets with the intention to augment and evaluate at hourly intervals where possible. Wherever spatial data is available, it would be useful to be able to display this information in form of an interactive map.

Our initial expectations were that good weather (higher temperatures and sunshine duration, less precipitation and snowfall) should correlate with more traffic, more speed limit violations and more accidents.

The GitLab repository for this project can be found here and the datadump is accessible with this link. The following subsections will go over the analysis and visualization goals of the project more specifically and outline the structure of this report.

### 1.1 Analysis Goals

The goal of this project is to analyze the correlations between weather effects and traffic density, speed limit violations and road accidents in Basel to see if and how much the weather might impact traffic safety. Specifically we want to inspect and accomplish the following goals and relations:

- **Goal 1: Analyze impact of weather on traffic.** Does weather impact traffic density and/or the amount of speed violations?
- **Goal 2: Analyze speed violations and accidents.** Can we draw correlations between speed violations and accidents data?
- **Goal 3: Visualization of spatial data.** Display of speed violations and accidents on a map.

### 1.2 Outline

The rest of the report is organized as follows:

The next section - **Section 2** - contains the descriptions of datasets used for the project and their respective ER diagrams.

In **Section 3**, the tools and methods used are summarized.

In **section 4**, the data integration is outlined, the according SQL queries for it are listed as well as the final ER diagram and relational schema of the database. Furthermore, a short subsection references to the manual for replicating this database.

In **section 5, 6 and 7** then the three above mentioned analysis goals are discussed and illustrated, including result plots and visualization gadget.

**Section 8** finishes the report with a conclusion and lessons learned chapter.

# 2 Datasets

This section goes through every data source used and describes its use, properties and ER diagram.

## 2.1 Speed-Monitoring Basel

The biggest dataset consisting of two separate files. This dataset lists any registered speed violations in Basel.

- Open Data Basel, `https://data.bs.ch/explore/dataset/100097/information/`
- temporal: 2017-2022, minutely; spatial: high-res, Basel
- 2'535'190 KB + 2'978'718 KB (5.25 GB)
- .csv

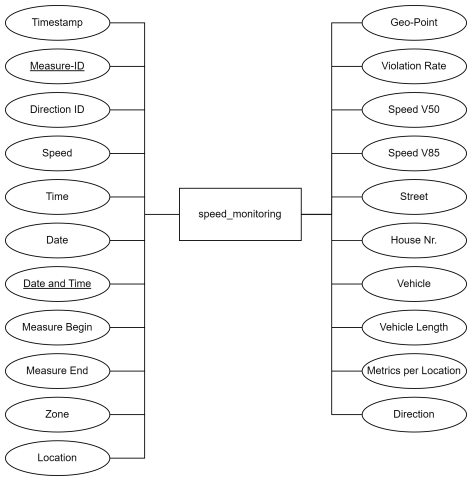


Fig. 1: ER diagram Speed-Monitoring Basel

## 2.2 MeteoBlue Weather Basel

This dataset consists of relevant weather information like temperature, precipitation, snowfall in Basel on an hourly basis for the years from 2008-2022.

- MeteoBlue, `https://www.meteoblue.com/de/wetter/archive/export/basel_schweiz_2661604`
- temporal: 2008-2022, hourly; spatial: Basel
- 1'417 KB
- .csv

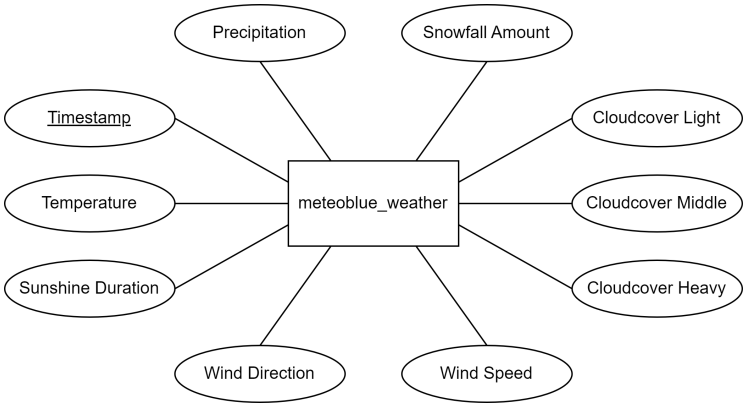


Fig. 2: ER diagram MeteoBlue Weather Basel

### 2.3 Vehicle Transit

This dataset tracks each vehicle that has passed at a certain point in Basel, more specifically Gundelingerstrasse. The data was collected as a result of a test project and is not particularly representative of the motorized vehicle traffic in all of Basel. Since its the best source we could find to accompany the cycle and pedestrian transit dataset accordingly we have decided to include it regardless. Also it poses an interesting integration exercise since the data is quite raw, representing one passed vehicle as one entry.

- Open Data Basel, `https://data.bs.ch/explore/dataset/100172/information/`
- temporal: 2022, minutely; spatial: Basel
- 77'688 KB
- .csv

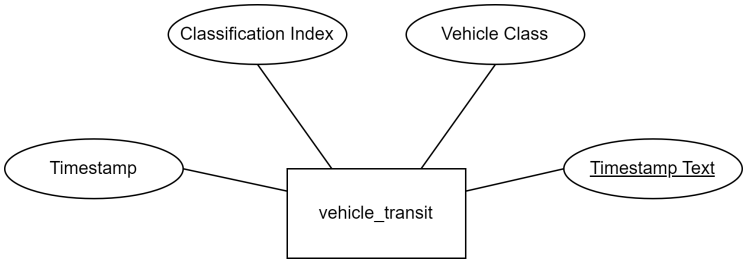


Fig. 3: ER diagram Vehicle Transit

### 2.4 Cycle and Pedestrian Transit

This dataset consists of various specific count locations, where bikes and pedestrians were counted on an hourly basis. It will be helpful in establishing a possible relationship between traffic density and weather.

- Open Data Basel, `https://data.bs.ch/explore/dataset/100013/information/`
- temporal: 2020-2022, hourly; spatial: low-res, Basel
- 391'404 KB
- .csv

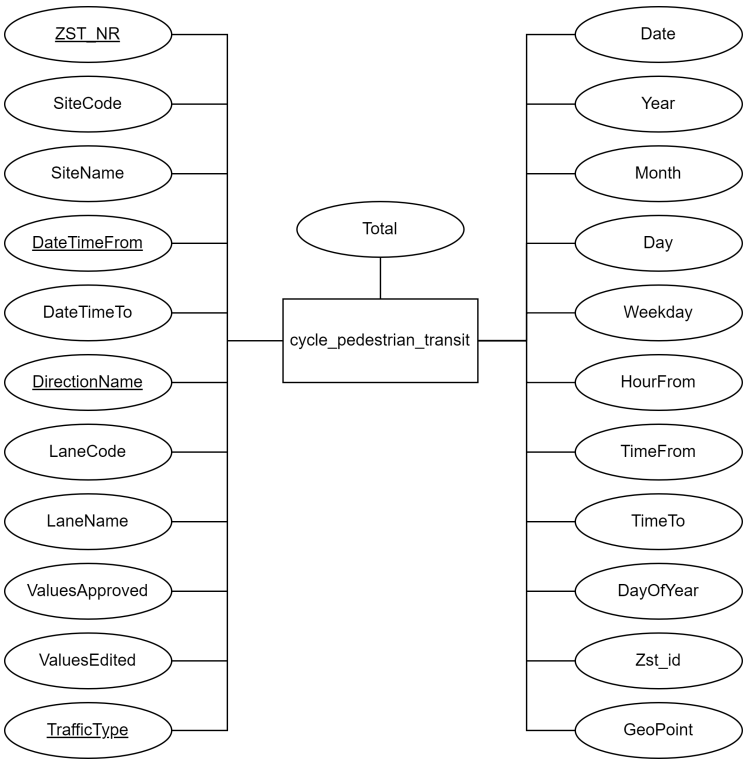


Fig. 4: ER diagram Cycle and Pedestrian Transit

### 2.5 Road Accidents Basel

This dataset consists of road accidents in Basel from 2011-2021. The dataset is sorted after weekdays and doesn't have a specific date included.

- Open Data Basel, `https://data.bs.ch/explore/dataset/100120/information/`
- temporal: 2011-2021, monthly; spatial: high-res, Basel
- 4'556 KB
- .csv

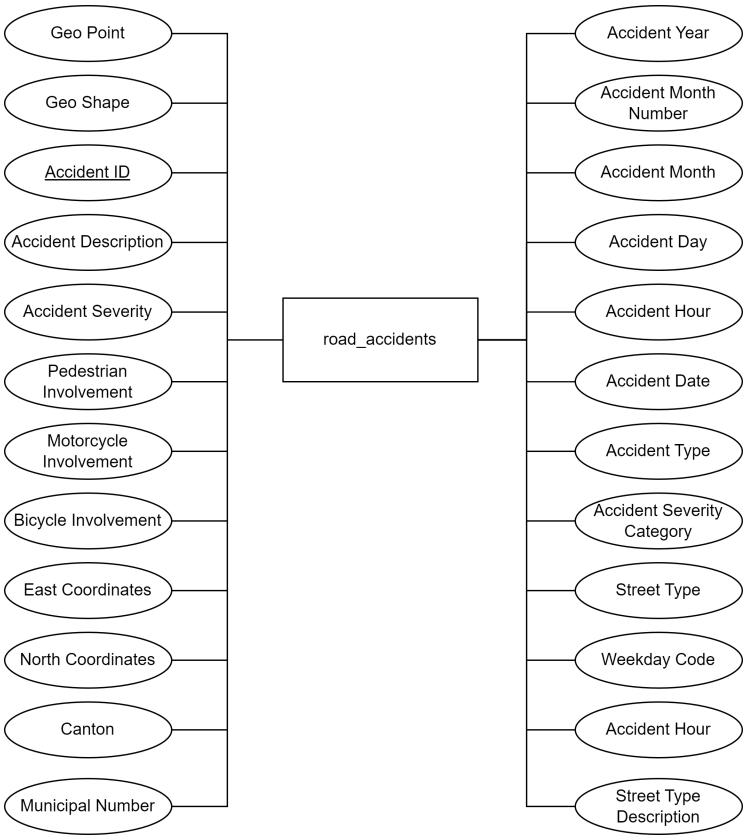


Fig. 5: ER diagram Road Accidents Basel

# 3 Tools and Methods

## 3.1 Tools

We have used the following tools for our project:

- Work Environment:
  - GitLab (Version Control)
  - Visual Studio Code
  - Large Text File Viewer
  - Discord (Communication)
- Database Management System:
  - MySQL Workbench 8
- Data Cleaning:
  - Python: Pandas library
- Data Analysis:
  - Python: MySQL Connector library
  - Python: Pandas library
  - Python: Matplotlib library
- Data Visualization:
  - Power BI

## 3.2 Methods

First, the datasets where roughly cleaned up with the Pandas library. This included for example renaming attributes, splitting geopoint attributes into langitude and longitude, adding unique identifiers and unifying attributes like the date across all datasets.

The cleaned up .csv files then were imported via infile commands into the MySQL database, where the independent sources could be integrated into one coherent database.

For the analysis the MySQL connector Python library was used to access the database via queries in order to further process it then in Python. Pandas Dataframes as well as the matplotlib library were used to create graphs with the data. For the visualization the program Power BI from Windows was used to generate two interactive maps.

# 4 Data Integration

The following subsections go through the integration methods, the resulting ER diagram and the according logical schema of the final database.

## 4.1 Data Integration Methods

After the initial rough cleaning in Python with the Pandas package, we've used the MySQL Workbench and SQL queries to transform and normalize the data. For this we've created tables, added ID's as primary keys or created views. The SQL queries used to establish the integrated schema are the following in the same order as presented. Dots at the end of a listing denote that more code was used but not neccessarily benefits the reader because of similarity/redundancy. The complete SQL queries can be viewed in the dedicated repository under the folder `/sql-code`.

With the view of the road_accident and the speed_violation_total table we've transformed those two datasets into a new table local_traffic_data which more comfortably lets us search through all data in regards to locality (coordinates).

Listing 1.1: SQL Query to get the views on the road_accident

```
use group8;
create view road_acciedents_view as
Select aid, accidentDay, accidentMonth, accidentYear,
    latitude, longitude
from road_accidents;

create view road_accidents_month_view as
select accidentMonth, accidentYear, count(accidentMonth) as
    amount  from road_accidents
group by accidentMonth, accidentYear;

create view road_accidents_onWeekdaysPerYear_view as
select accidentDay, accidentYear, count(accidentDay) as
    amount  from road_accidents
group by accidentDay, accidentYear;
```

Listing 1.2: SQL Query to get the table speed_violations_total

```
use group8;

create view speed_monitoringWithLevel_view as
select svid, date, hour, violationDegree,
case when violationDegree <= 0 then 0 when violationDegree >
    0 and violationDegree <= 20 then 1
when violationDegree > 20 and violationDegree <= 50 then 2
    else 3 end as violationLevel,
```

```
 latitude, longitude from speed_monitoring;

create view speed_violations_view as
select svid, date, hour, violationDegree, violationLevel,
    latitude, longitude from speed_monitoringWithLevel_view
where violationLevel > 0;


create table speed_violations select * from
    speed_violations_view;
alter table speed_violations add primary key (svid);

create view speed_violations_total_view as
select svid as svtid, date, hour, count(hour) as amount, sum(
    violationDegree)/count(hour) as violationDegreeAverage
    from speed_violations
group by date, hour;

create table speed_violations_total select * from
    speed_violations_total_view;
alter table speed_violations_total add primary key (svtid);
```

Listing 1.3: SQL Query to get the table local_traffic_data

```
use group8;

create table local_traffic_data (
    aid int,
    accidentDay varchar(50),
    accidentMonth int,
    accidentYear int,
    latitude varchar(50),
    longitude varchar(50),
    svid int,
    date varchar(50),
    hour int,
    violationLevel int,
    violationDegree double
);

insert into local_traffic_data (svid,date,hour,violationLevel
    ,violationDegree, latitude, longitude)
select svid,date,hour,violationLevel,violationDegree,
    latitude, longitude from speed_violations;

insert into local_traffic_data (aid,accidentDay,
    accidentMonth, accidentYear, latitude, longitude)
select aid, accidentDay, accidentMonth, accidentYear,
    latitude, longitude from road_accidents;
```

To include the amount of pedestrians and cycles in one row we've transformed the total of both traffic types into separate tables and combined those two tables with the view of the vehicle_density into the table traffic_density.

Listing 1.4: SQL Query to get the tables ped_sum, cycle_sum and finaly traffic_density

```
use group8;
create view ped_sum_view AS
Select cptid as psid, date, hour, trafficType, SUM(total) as
    total From cycle_pedestrian_transit
where trafficType = 'Fussg nger'
group by date,hour, trafficType;

create table ped_sum select * from ped_sum_view;

create view cycle_sum_view as
select cptid as csid, date, hour, trafficType, sum(total) as
    total from cycle_pedestrian_transit
where trafficType = 'Velo'
group by date, hour, trafficType;

create table cycle_sum select * from cycle_sum_view;

create view cycle_ped_sum_view as
select ps.psid as cpsid, ps.date, ps.hour, ps.trafficType as
    trafficType0, ps.total as total0, cs.trafficType as
    trafficType1, cs.total as total1 from ped_sum ps ,
    cycle_sum cs
where ps.date = cs.date and ps.hour = cs.hour
group by ps.date, ps.hour;

create view motorVehicles_sum_view AS
Select date, hour, SUM(total) as total from vehicle_density
group by date, hour;

create view traffic_density_view AS
Select cp.cpsid as tdid, cp.date, cp.hour, cp.trafficType0,
    cp.total0, cp.trafficType1, cp.total1, v.total as
    totalMotorVehicle
from cycle_ped_sum_view cp, motorVehicles_sum_view v
where cp.date = v.date and cp.hour = v.hour;

create table traffic_density select * from
    traffic_density_view;
alter table traffic_density add primary key (tdid);
```

Listing 1.5: SQL Query to get the table weather_indicator

```
use group8;

create view weather_condition_view as
select wcid, date, hour, weatherIndicator from
    meteoblue_weather;

create table weather_condition select * from
    weather_condition_view;
alter table weather_condition add primary key (wcid);
```

With the traffic_density, speed_violations_total and the weather_condition tables we then created the table hourly_traffic_data in which we had transformed 4 datesets into one table where all 4 datasets had an overlap. This table lets us search through all entries in regards to date and hour.

Listing 1.6: SQL Query to get the hourly_traffic_data table

```
use group8;

create view hourly_traffic_data_view as
select sv.svtid as htdid, sv.date, sv.hour, sv.amount as
    speedViolations, sv.violationDegreeAverage, td.
    trafficType0, td.total0, td.trafficType1,
td.total1, td.totalMotorVehicle, wc.weatherIndicator
from weather_condition_view wc, speed_violations_total sv,
    traffic_density td
where sv.date = wc.date and sv.date = td.date and sv.hour =
    wc.hour and sv.hour = td.hour;

create table hourly_traffic_data select * from
    hourly_traffic_data_view;
alter table hourly_traffic_data add primary key (htdid);
```

Furthermore we've created tables to access specific data easier for our analysis. In the following the SQL queries for those tables are listed.

Listing 1.7: SQL Query to get the table avg_traffic_data (weekday 0 until 6)

```
create table avg_traffic_data select h.date as date, weekday(
    h.date) as weekday, avg(total0) as avg_ped, avg(total1)
    as avg_cyc, avg(totalMotorVehicle) as avg_moto, avg(
    speedViolations) as avg_speedViolations, avg(m.
    temperature) as avg_temp, avg(m.precipitation) as
    avg_rain, avg(m.snowfall) as avg_snowfall
        from meteoblue_weather m, hourly_traffic_data h where
     m.date = h.date and h.hour = m.hour and weekday(h.date)
    = 0 group by h.date;

insert into avg_traffic_data select h.date, weekday(h.date)
    as weekday, avg(total0) as avg_ped, avg(total1) as
```

```
    avg_cyc, avg(totalMotorVehicle) as avg_moto, avg(
    speedViolations) as avg_speedViolations, avg(m.
    temperature) as avg_temp, avg(m.precipitation) as
    avg_rain, avg(m.snowfall) as avg_snowfall
        from meteoblue_weather m, hourly_traffic_data h where
     m.date = h.date and h.hour = m.hour and weekday(h.date)
    = 1 group by h.date;

...
```

Listing 1.8: SQL Query to get the table accident_peryear_monthly (from 2017 until 2021)

```
create Table accident_perYear_monthly select count(aid) as
    total, accidentYear as year, accidentMonth as month from
    local_traffic_data where accidentYear = 2017 group by
    accidentMonth order by accidentMonth;
create Table violations_perYear_monthly select count(svid) as
     total, year(date) as year, month(date) as month from
    local_traffic_data where year(date) = 2017 group by month
    (date) order by month(date);

Insert into accident_perYear_monthly select count(aid) as
    total, accidentYear as year, accidentMonth as month from
    local_traffic_data where accidentYear = 2018 group by
    accidentMonth order by accidentMonth;
Insert into violations_perYear_monthly select count(svid) as
    total, year(date) as year, month(date) as month from
    local_traffic_data where year(date) = 2018 group by month
    (date) order by month(date);

...
```

Listing 1.9: SQL Query to get the table percentage_violation_year from (2017 until 2021)

```
create table percentage_violation_year select t1.year, vio,
    vioDegree, noVio, noVioDegree from (
((select year, vio, vioDegree from (select year(date) as year
    , count(svid) as vio, avg(violationDegree) as vioDegree
    from speed_monitoring where year(date) = 2017 group by
    violationDegree > 0) as test where vioDegree > 0) t1
Inner Join
(select year, noVio, noVioDegree from (select year(date) as
    year, count(svid) as noVio, avg(violationDegree) as
    noVioDegree from speed_monitoring where year(date) = 2017
     group by violationDegree <= 0) as test2 where
    novioDegree <= 0) t2
on (t1.year = t2.year)));
```

```
insert into percentage_violation_year select t1.year, vio,
    vioDegree, noVio, noVioDegree from (
((select year, vio, vioDegree from (select year(date) as year
    , count(svid) as vio, avg(violationDegree) as vioDegree
    from speed_monitoring where year(date) = 2018 group by
    violationDegree > 0) as test where vioDegree > 0) t1
Inner Join
(select year, noVio, noVioDegree from (select year(date) as
    year, count(svid) as noVio, avg(violationDegree) as
    noVioDegree from speed_monitoring where year(date) = 2018
     group by violationDegree <= 0) as test2 where
    novioDegree <= 0) t2
on (t1.year = t2.year)));

...
```

Listing 1.10: SQL Query to add id's to the last four created tables

```
ALTER table violations_peryear_monthly
ADD vpymid int NOT NULL auto_increment unique first;

ALTER table percentage_violation_year
ADD pvyid int NOT NULL auto_increment unique first;

ALTER table accident_peryear_monthly
ADD apymid int NOT NULL auto_increment unique first;

ALTER table avg_traffic_data
ADD atdid int NOT NULL auto_increment unique first;
```

Adds the primary keys into the tables created above.

Listing 1.11: SQL Query to add id's to the last four created tables

```
use group8;

alter table accident_peryear_monthly
add primary key (apymid);

alter table avg_traffic_data
add primary key (atdid);

alter table cycle_sum
add primary key (csid);

alter table ped_sum
add primary key (psid);

alter table percentage_violation_year
```

```
add primary key (pvyid);

alter table violations_peryear_monthly
add primary key (vpymid);
```

### 4.2 Integrated ER Diagram

The final version of our entity relationship diagram looks as follows:

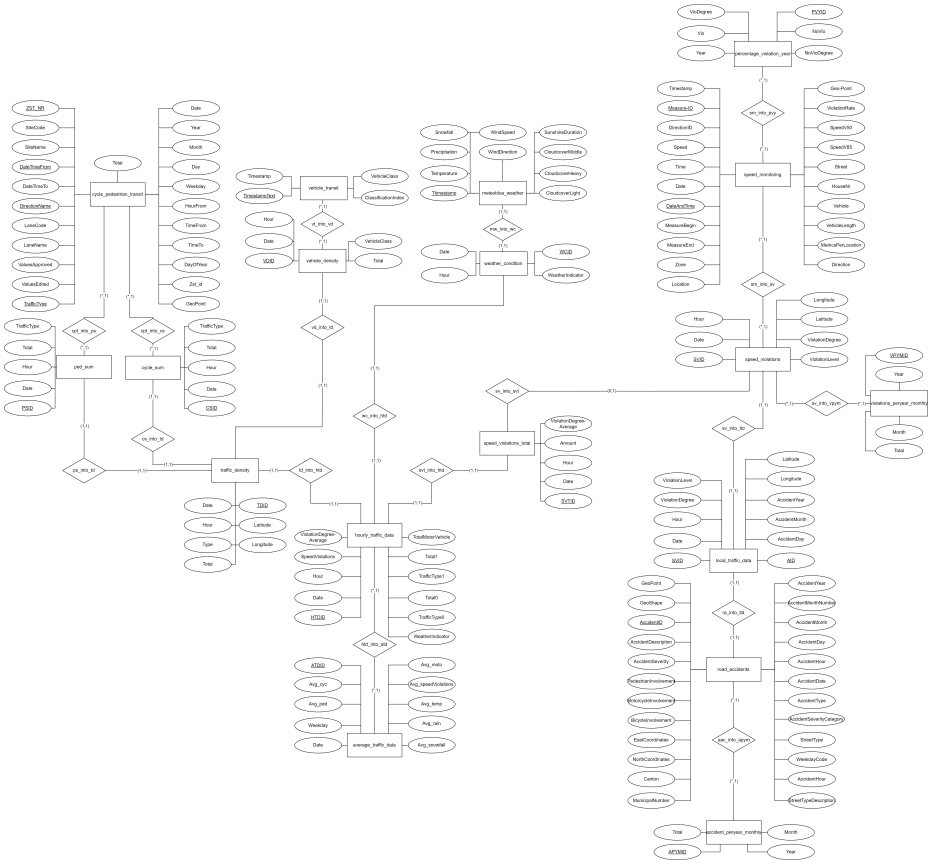

Fig. 6: Final entity relationship diagram

### 4.3 Logical Schema

The final version of our logical schema looks as follows:

local_traffic_data(ltdid, date, hour, violationLevel, violationDegree, latitude, longitude, aid, accidentYear, accidentMonth, accidentDay)
violations_peryear_monthly(vpymid, year, month, total)
percentage_violation_year(pvyid, year, vio, vioDegree, noVio, noVioDegree)
ped_sum(psid, date, hour, trafficType, total)
cycle_sum(csid, date, hour, trafficType, total)
average_traffic_data(atdid, date, weekday, avg_ped, avg_cyc, avg_snowfall, avg_rain, avg_temp, avg_speedViolations, avg_moto)
accident_peryear_monthly(apymid, year, month, total)
hourly_traffic_data(htdid, date, hour, speedViolations, violationDegreeAverage, weatherIndicator, trafficType0, total0, trafficType1, total1, totalMotorVehicle)
vehicle_density(vdid, date, hour, vehicleClass, total)
speed_violations_total(svtid, date, hour, violationDegreeAverage, amount)
speed_violations(svid, date, hour, latitude, longitude, violationDegree, violationLevel)
traffic_density(tdid, date, hour, latitude, longitude, type, total)
weather_condition(wcid, date, hour, weatherIndicator)
speed_monitoring(timestamp, measureID, directionID, speed, time, date, dateAndTime, measureBegin, measureEnd, zone, location, geoPoint, violationRate, speedV50, speedV85, street, houseNr, vehicle, vehicleLength, metricsPerLocation, direction)
meteoblue_weather(timestamp, temperature, sunshineDuration, windDirection, windSpeed, cloudcoverHeavy, cloudcoverMiddle, cloudcoverLight, snowfallAmount, precipitation)
vehicle_transit(timestampText, timestamp, classificationIndex, vehicleClass)
cycle_peestrian_transit(zst_nr, siteCode, siteName, dateTimeFrom, dateTimeTo, directionName, laneCode, laneName, valuesApproved, valuesEdited, trafficType, date, year, month, day weekday, hourFrom, timeFrom, timeTo, dayOfYear, zst_id, geoPoint)
road_accidents(geoPoint, geoShape, accidentID, accidentDescription, accidentSeverity, pedestrianInvolvement, motorcycleInvolvement, bicycleInvolvement, eastCoordinates, northCoordinates, canton, municipalNumber, accidentYear, accidentMonthNumber, accidentMonth, accidentDay, accidentHour, accidentDate, accidentType, accidentSeverityCategory, streetType, weekdayCode, accidentHour, streetTypeDescription)

### 4.4 Database Setup

To set up the database follow the introductions in the readme file from the dedicated GitLab repository.

## 5 Analysis Goal 1: Impact of weather on traffic

For this analysis goal we took a few metrics and compared those to the weather. We have split various traffic transit data into three parts, namely pedestrians, cycles and motorized vehicles, giving us samples of the average traffic density in Basel at an hourly rate. The limiting factor here was the vehicle transit dataset, which only contained data for 2022. Further we used the amount of speed violations per hour as another metric.
With most of the heavy work done in the integration process, a sample for typical SQL queries which would access this kind of information would be:

Listing 1.12: SQL Query to get pedestrian count for weekdays given a specific weather condition, in this case rain

```
Select avg(avg_ped) as ped, weekday, avg(avg_temp) as
    temperature
    from avg_traffic_data
    where avg_rain > 0
    group by weekday;
```

Listing 1.13: SQL Query to get hourly pedestrian count for a specific day

```
Select speedViolations as sv, m.temperature, m.precipitation,
     h.date, h.hour, m.snowfall from meteoblue_weather m,
    hourly_traffic_data h
where m.date = h.date and h.date = '2022-04-01' and h.hour =
    m.hour
order by h.date, h.hour;
```

We also wanted to look at the correlation between accidents and speed violations. As the accident dataset doesn't include a specific date for the accidents, we analyzed it over the different months and weekdays in general. Here we used two different queries from two different tables.
SQL queries for this look like so:

Listing 1.14: SQL Query to get montly accidents and speed violations for a specific year

```
select total as total_v, year, month from
    violations_perYear_monthly where year = yyyy;
select total as total_a, year, month from
    accident_perYear_monthly where year = yyyy;
```

Listing 1.15: SQL Query to get the accidents and speed violations depending on the weekday

```
Select count(svid) as total_v, weekday(date) as weekday from
    local_traffic_data where year(date) <= 2021 group by
    weekday(date) order by weekday(date);
```

```
Select count(aid) as total_a, accidentDay from
    local_traffic_data where accidentYear >= 2017 group by
    accidentDay order by accidentDay;
```

Listing 1.16: SQL Query to get the total amount of accidents and speed violations from 2017-2021

```
Select *, count(svid) as total_v from local_traffic_data
    where year(date) <= 2021;
Select count(aid) as total_a from local_traffic_data where
    accidentYear >= 2017;
```

For our analysis we have sampled several individual days with various weather conditions over workdays or weekends. In the following we will go over all the results we came up with and provide some example graphs.

### 5.1 Pedestrians

The pedestrian traffic in Basel follows a general schema. First of all the number of pedestrians follows the temperature curve. We always have two peaks on workdays, in between 12 and 13 pm and at around 17 to 19 pm which are almost always visible and do not follow any weather influence. In between those two peaks there is a low in the afternoon. When rain falls, then temperature drops as well and the number of pedestrians gets lower. For the few days where it snows in Basel the snow brings lot more pedestrians out into the streets of Basel.
In the following graphs exemplify the analyzed behaviour. The first graph from Figure 7 shows the correlation between temperature and number of pedestrians very well, the graph is a randomly sampled Saturday where we won't have those specific workday peaks. Also in this graph it is visible that there was snowfall in the morning and the average number of pedestrians per hour is 2548 which is not far from the average for most other Saturdays.

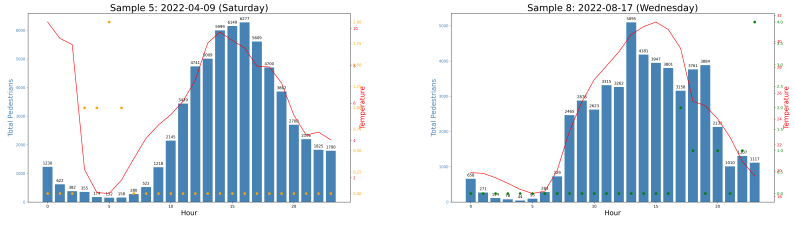


Fig. 7: Pedestrian monitoring for 09-04-22 and 17-08-22

In the second graph from Figure 7 it is visible that the rain has a impact on the number of pedestrians on the street. In the hour it started to rain the number of pedestrians dropped a lot. The peak around 19 pm still exists since the people go home after work. The average of pedestrians per hour in this graph is 2071. Which is about 300 under the average on a Wednesday.
To check whether those specific days are not just any random events we can compare the average of pedestrians per hour on the weekdays and the weather. Therefore we can look at the following three graphs where the first one consist of all clear days the second of the days rain and the last one of all days combined.

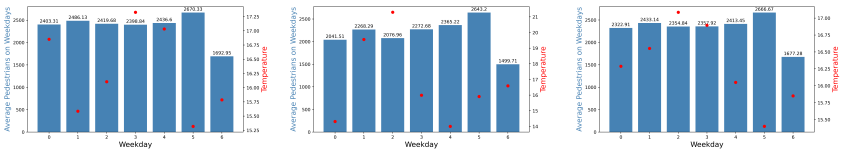


Fig. 8: Average pedestrians for clear days, rainy days and all days combined

It is visible that on all days except on the Saturday the rainy weather has an impact of the amount of pedestrians per hour in Basel.

### 5.2 Bicycles

For the Bicycles in Basel there are some usual workday peaks at around 8 am and 18 pm and often a little one at 13 pm, those peaks are visible no matter whether it rains or not or if its cold or warm. In general there are more bicycles on the streets if the weather is warmer. And on the weekends the number of cycles follows the temperature curve. The rain has a big impact on the number of bicycles.

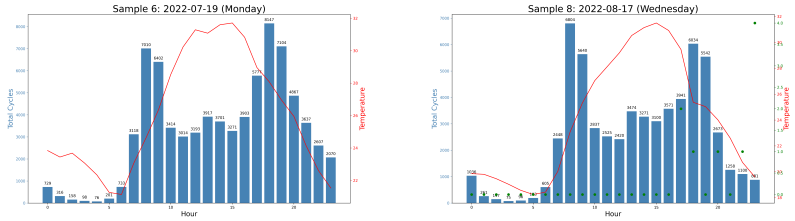


Fig. 9: Bicycles monitoring for 20-06-22 and 17-08-22

We can see in those two diagrams the difference between a warm summer day,

with an average of 3227 cycles per hour and a warm day with rain, with an average of 2493 cycles per hour. As we can tell from the averages the influence of rain has the result that less bicycles are on the streets.

To illustrate this we show the plots of the average bicycles on weekdays per hour. As you can see the assumption that the rain has an impact is in general true for all days, although the Tuesday tells otherwise.

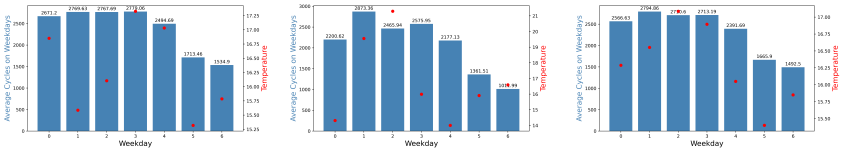


Fig. 10: Average bicycles for clear days, rainy days and all days combined

### 5.3 Motor Vehicles

For the analysis of the motorized vehicles we used a data set which measured just one road in Basel as it was the only source of car transit information we could get ahold of. The results of this limited data is more or less in accordance to previous observations, where weekday and specific times have a bigger impact on traffic transit observations than the weather.

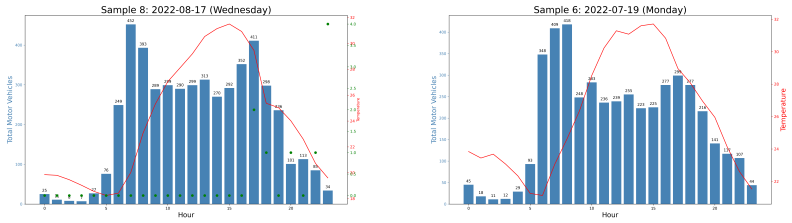


Fig. 11: Motor vehicles monitoring for 17-08-22 and 20-06-22

In those two diagrams it is visible that the rain has no real impact on the amount of motorized vehicles on the roads.

In the next three diagrams we can see that the difference between rainy days and clear days is not big. And since this data set just consists of one road it is not meaningful enough.

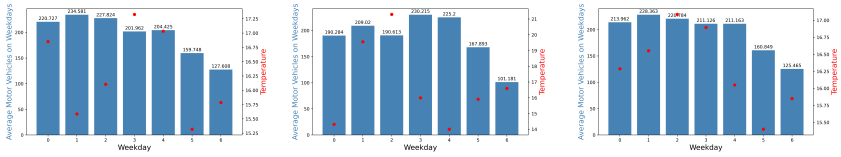


Fig. 12: Average motor vehicles for clear days, rainy days and all days combined

### 5.4 Speed Violations

The amount of speed violations increase during the rush-hours as the motorized vehicles did. The average amount of violations per hour of a day is higher on warm days, the impact of rain does increase the amount of speed violations. Those things can be seen in the three following diagrams, where in the first two we can see the difference between the temperature and in comparison to the first and the third diagram it occurs that the rain increases the amount of speed violations.

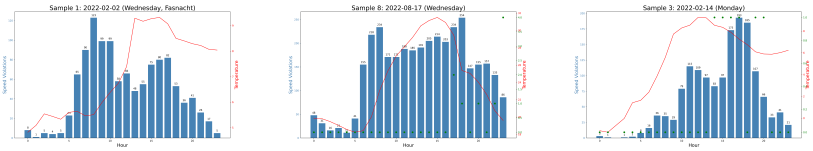


Fig. 13: Speed violations monitoring for 02-02-22, 17-08-22 and 14-02-22

The average speed violations on weekdays per hour shows that the rain increases the violations in the average of the workdays (143 rain and 133 clear). On the weekend it is the opposite. The rain decreases the amount of violations on average per hour (69 rain and 104 clear). This could be explained when taking human behaviour into account, where people in general would go out to work regardless of the weather, while on weekends they would stay home which in turn would lead to less violations.

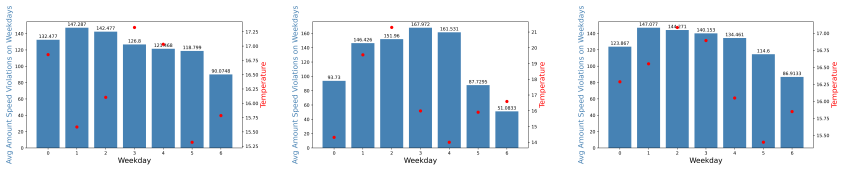


Fig. 14: Average speed violations for clear days, rainy days and all days combined

# 6 Analysis Goal 2: Correlation between speed violations and accidents

## 6.1 Accidents and Speed Violations

Over the time span from 2017 to 2021, where the two datasets road accidents and speed violations have an overlap, 3538 accidents and around 4 million speed violations happened .

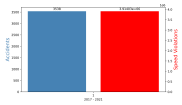

Fig. 15: Accidents and speed violations total 2017-2021

We can see in the next diagram that the amount of accidents decreases from year to year. But the amount of violations increases rapidly, this is the case because the amount of measure stations increases over the year.
Therefore we've used a relative value which uses the percentage of speed violations divided by total measurements next. With this we can see that in general there is a relation between accidents and speed violations. The year 2017 does not fit in this assumption, but this might be the case because there were not as many measure locations as in the other years.

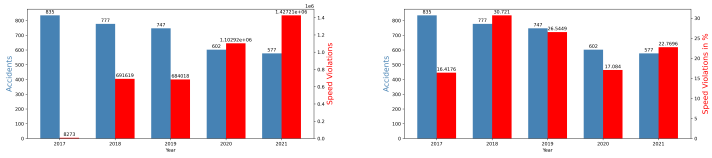


Fig. 16: Accidents and speed violations total per year (left); Accidents and speed violations relative to total measurements per year(right)

Furthermore we can see that the amount of violations in 2019 do not really correspond to the accidents, but in the year 2020 and 2021 there is a relation.

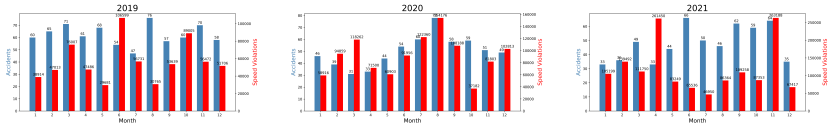


Fig. 17: Years 2019 - 2021 monthly resolution of accidents and speed violations

If we compare the months where the accidents and the speed violations happened in general, the amount of accidents and the amount of speed violations relate with each other but it is visible that in the summer months 5 - 7 (usually holidays months) there are less speed violations. But the best visible relation is when we map the accidents and speed violations onto the weekdays. There we can see that the amount of violations really relates with the amount of accidents.

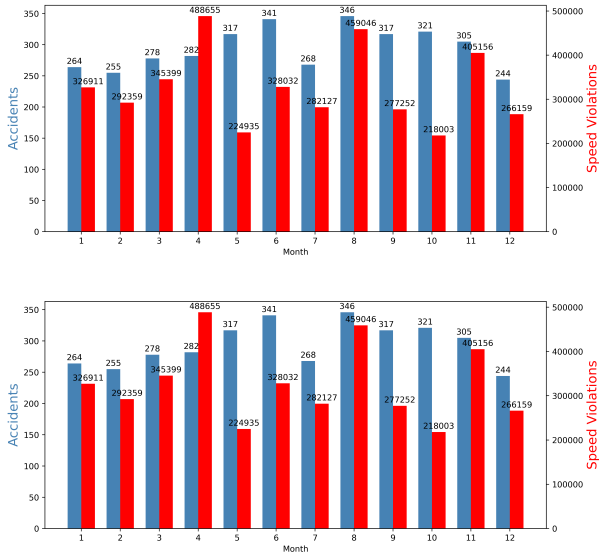


Fig. 18: Accidents speed violations total per weekday(upper) Accidents speed violations total per month (lower)

# 7 Analysis Goal 3: Visualization of spatial data

Here we visualised the datasets road_accidents and speed_violations into two maps. For the speed violations we used a heat map since the locations where the speed is monitored are often the same while the map for accidents still depicts each individual accident.

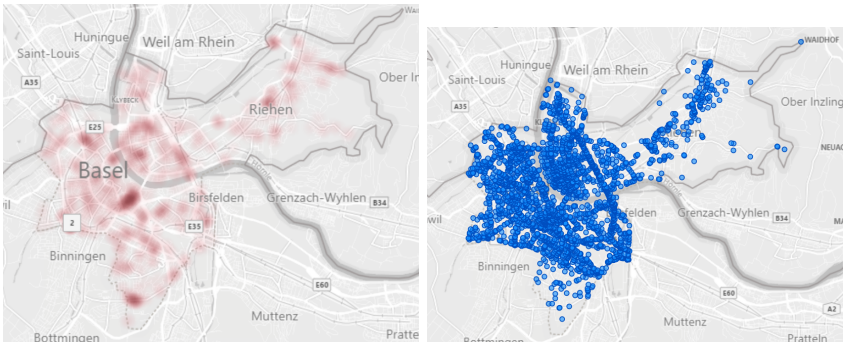


Fig. 19: Speed violations heat map and accidents map

The two maps are generated with Power BI and are interactive, it is selectable in which year and month the accidents or the speed violations are shown. The file can be found in the repository in the folder map. To open this file Windows Power BI is required, it can be downloaded for free here.

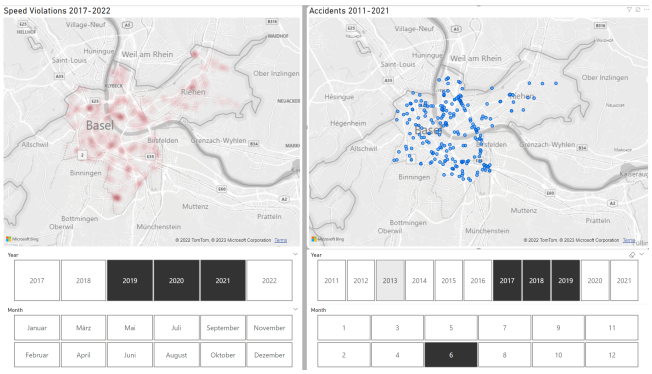


Fig. 20: Interactive visualization with PowerBI

## 8 Conclusions and Lessons Learned

Our initial expectations that weather has an influence on traffic, speed violations and accidents have been met, although we initially have not fully considered how big of an impact things like rush hours or work days would have on the data. Regardless we can say the data indicates that more people go outside when there is no rain and it is warmer. More people also lead to more traffic which in turn leads to more accidents.

In addition to those observations we have made a gadget which can display both speed violations and accidents on a map for a given timeframe. We would have liked to make a complete web app which accesses a live database and performs its own queries via sliders and buttons, but due to illness sadly only a rudimentary prototype using Dash and Plotly came about within the given time constraint, so we have chosen to not include it.

Regardless, we have learned a lot from the project. We've learned that choosing your datasets is probably the most important thing you can do for a successful analysis. A lot of interesting topic ideas didn't make it into the project because there was not enough data to be found to reach the 5 GB limit. We've learned how to create a database from beginning to end on a practical level and benefited on more than one occasion from the theoretical lecture. Given the abundance of different datasets we have chosen to integrate we got very familiar with SQL queries in general. We would have liked to conclude our deep dive into databases with a proper, useful application, but this will be something for future endeavours.