Showcase of database analytics on car accidents dataset

Matej Poljak
Faculty of Management Science and Informatics
University of Žilina
Žilina, Slovakia
poljak@stud.uniza.sk

***Abstract*— The number of cars has been rapidly growing in the recent years and so has the number of related accidents. Given the availability of a public resource providing this data, it would be interesting to analyze it from various aspects. During this work we quantify the impact of alcohol usage on car accidents, considering factors such as accident frequency, time of the day, regions and associated damage. The results may be distorted because gathered data does not seem to correspond to reality always. The main goal of this paper involves suggesting database analytic tools (specifically aggregate and analytic functions) available on Oracle platform which provide us with an effective and easy extraction of information from vast amounts of data. After that we show optimization of created queries using database index. Eventually, this study demonstrates the power of these tools, as well as the importance of proper data gathering. The results can motivate national institutions to focus more on precise data gathering for matters such as car accidents, as significant information can be gained from such data to implement smarter measures that contribute to a better world.**

***Keywords*— analytics, alcohol usage, car accidents, database**

I. Introduction

This work is trying to emphasize the importance of gathering and processing data while we live in era of data explosion. Spread of the Internet, social networks, Internet of Things, cloud-computing, etc. This all comes with the vast amount of data generated. But there are also areas that are not still monitored well or at all, but they should. The reason we should care about it is that we can extract from them some interesting and important information which can help us to understand some phenomenon, process and (ideally) based on that create better solutions if possible. As a good example of consistent data gathering is Czech police, which reveal car accidents data every month on their web [1] and are free for everyone. These data are going to be used as an example of data processing using Oracle database analytic tools. Specifically, this work focuses on alcohol impact of car accidents from various points of view. It is possible to observe this while information about alcohol usage is provided within stored data.

II. Dataset overview

A. Preparing dataset

First, we need to download data from [1] in zip format. We will focus on the year 2024 only – that means we must download twelve structured files. These files contain IDs from many lookup tables. Values of lookup tables are specified in file *Položky formuláře – data* also available on [1]. Then we can create database tables. Final structure is shown in Fig. 1.

There are many references to database tables that serve as lookup tables. For our work, we will use just these tables: (since the names of tables and their attributes are in the Slovak language, we will use aliases throughout this paper to enhance reader comprehension)

- Table *CR_NEHODY* – this is the main table that stores all information about car accidents and IDs to other lookup tables. We will refer to this table as *acc* (shortly for accident).
- Table *CR_KRAJE* – is a lookup table for regions in Czech Republic. We will refer to this table as *regions*.
- Table *CR_PRITOMNOST_ALKO* – is a lookup table of observations of alcohol level in the blood of person that was responsible for an accident. We will refer to this table as *alco*.

Downloaded zip file from [1] contains *xls* file with records. Records can be imported to the database by creating formulas that construct insert statements.

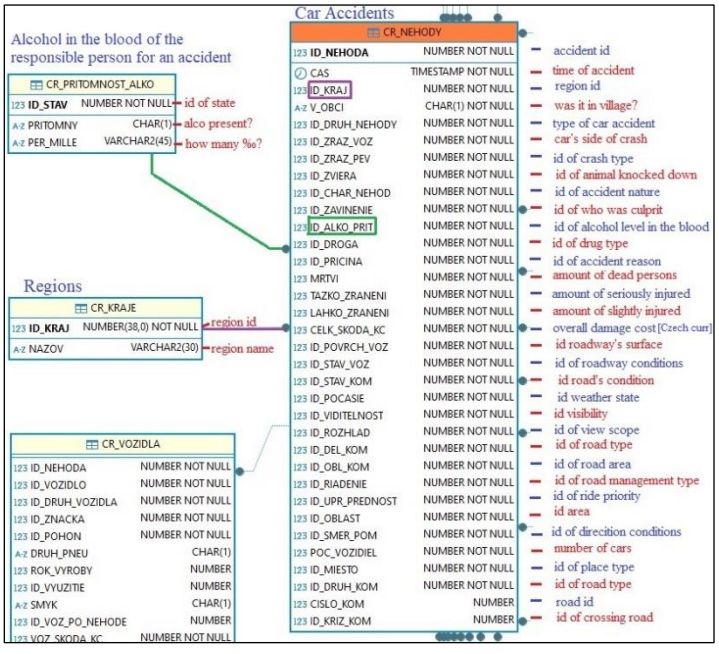

Figure 1 - Structure of database tables

B. Understanding possible cases of alcohol usage

We already know that *acc* table has a relation to *alco* table. Now, we could get insight of what alcohol cases are observed when an accident occurred. Results of select query are shown in Fig. 2. We see that recordings can be negative

on the presence of alcohol or positive with different amounts of per mills. There can also be records with refusion of alcohol observation for an unknown reasons. There are also cases when it was not observed at all.

```sql
SELECT  alco.id_stav AS state_id, -- presence OF alcohol - states
        alco.pritomny AS present,
        CASE alco.pritomny
        WHEN 'A' THEN 'Yes'
        WHEN 'N' THEN 'No'
        WHEN 'O' THEN 'Refused'
        WHEN 'X' THEN 'Not observed'
    END AS was_present,
    per_mille
  FROM CR_PRITOMNOST_ALKO alco
   ORDER BY was_present desc NULLS LAST, state_id
```

	STATE_ID	PRESENT	WAS_PRESENT	PER_MILLE
1	1	A	Yes	under 0.24 ‰
2	3	A	Yes	0.24 - 0.5 ‰
3	6	A	Yes	0.5 - 0.8 ‰
4	7	A	Yes	0.8 - 1.0 ‰
5	8	A	Yes	1.0 - 1.5 ‰
6	9	A	Yes	1.5 ‰ and more
7	4	O	Refused	[NULL]
8	0	X	Not observed	[NULL]
9	2	N	No	[NULL]
10	5	[NULL]	[NULL]	[NULL]

Figure 2 - Presence of alcohol during an accident (script)

III. Basic statistics

With knowledge from the previous paragraph, we can dive into analysis of our dataset.

A. All accidents and alcohol-related accidents

Before we start doing more in-depth analysis, we should get a general overview of accidents in 2024, as well as the number of accidents which were somehow related to alcohol usage. Script with results is shown in Fig. 3. In the first select of UNION statement we query for all accidents in 2024 with any per mill amount of alcohol found in the blood. We sum all cars that were related to those accidents also. In the second select of UNION statement we query for all accidents in 2024. Results of UNION statement are then grouped to one row using analytic function *LAG* (assigns each row its predecessor) and filtering proper row. We see that only 4,475 out of 91,211 accidents had relation to alcohol usage, what is approximately 5%. 6,808 crashed cars were found in accidents with alcohol involvement, what is also 5% of all crashed cars (149,208). We might have expected a higher number of accidents related to alcohol usage.

```sql
SELECT  'Alco-ratio (accidents): ' || round(100*alco_acc_cnt/accidents_count, 2) || '%'
        ||' ['||alco_acc_cnt||' / '|| accidents_count||']'AS ratio_accidents_count,
        'Alco-ratio (vehicles): '  || round(100*alco_veh_cnt/vehicles_count, 2) || '%'
        ||' ['||alco_veh_cnt||' / '|| vehicles_count||']' AS ratio_vehicles_count
 FROM (
    SELECT  lag(accidents_count) OVER (ORDER BY id) AS alco_acc_cnt,
            accidents_count,
            lag(vehicles_count) OVER (ORDER BY id) AS alco_veh_cnt,
            vehicles_count
     FROM (
        select 1 AS id, -- Accidents with alcohol (2024)
         count(*) as accidents_count,
         sum(acc2.poc_vozidiel) AS vehicles_count
         from cr_nehody acc2 JOIN cr_pritomnost_alko alco ON acc2.id_alko_prit = alco.id_stav
          WHERE EXTRACT(YEAR FROM acc2.cas) = 2024 AND
           alco.pritomny IS NOT NULL AND alco.pritomny LIKE 'A'
        UNION ALL
        select 2 AS id, -- All accidents (2024)
         count(*) as accidents_count,
         sum(acc1.poc_vozidiel) AS vehicles_count
         from cr_nehody acc1
          WHERE EXTRACT(YEAR FROM acc1.cas) = 2024)
 ) WHERE alco_acc_cnt IS NOT null
```

	RATIO_ACCIDENTS_COUNT	RATIO_VEHICLES_COUNT
1	Alco-ratio (accidents): 4.91% [4475 / 91211]	Alco-ratio (vehicles): 4.56% [6808 / 149208]

Figure 3 - Ratio of alcohol-accidents to all accidents (script)

B. Observations for all possible states of alco table

Let us observe the number of accidents for each case of *alco* table to get a better idea of what is going on. Results from Chart 1 reveal that 37,572 accidents, what is 41% of all accidents, were without any observation of alcohol involvement what is interesting, and we do not know the real reason for that. In most cases, which is 53% of accidents, the presence alcohol was not confirmed as shown in Chart 1. In the query from Fig. 4 we compute number of cases for each alcohol presence in the blood as well as an absence of all three types {N,X,O}. Relative proportions of each state are also computed to better understanding of dominant cases.

```sql
-- statistics of alcohol level in blood during an accident
select case when alco.pritomny='A' then 'Yes'
            when alco.pritomny='N' then 'No'
            when alco.pritomny='O' then 'Refused'
            when alco.pritomny='X' then 'Not observed'
            else 'Other' end as alco_present,
    alco.per_mille as alco_value_in_blood,
    count(*) as accidents_count,
    round(count(*)/
    (SELECT count(*) AS overall_cnt FROM cr_nehody WHERE EXTRACT(YEAR FROM cas)=2024)
    ,4)*100 || '%' AS ratio
 from cr_nehody n
 join cr_pritomnost_alko alco on n.id_alko_prit = alco.id_stav
  where extract(year from cas)=2024
   group by alco.id_stav, alco.pritomny, alco.per_mille
    order by accidents_count DESC
```

	ALCO_PRESENT	ALCO_VALUE_IN_BLOOD	ACCIDENTS_COUNT	RATIO
1	No	[NULL]	49,004	53.73%
2	Not observed	[NULL]	37,572	41.19%
3	Yes	1.5 ‰ and more	2,734	3%
4	Yes	1.0 - 1.5 ‰	751	.82%
5	Yes	0.24 - 0.5 ‰	315	.35%
6	Yes	0.5 - 0.8 ‰	282	.31%
7	Yes	0.8 - 1.0 ‰	210	.23%
8	Yes	under 0.24 ‰	183	.2%
9	Refused	[NULL]	160	.18%

Figure 4 -Accidents for each possible state from alco table (script)

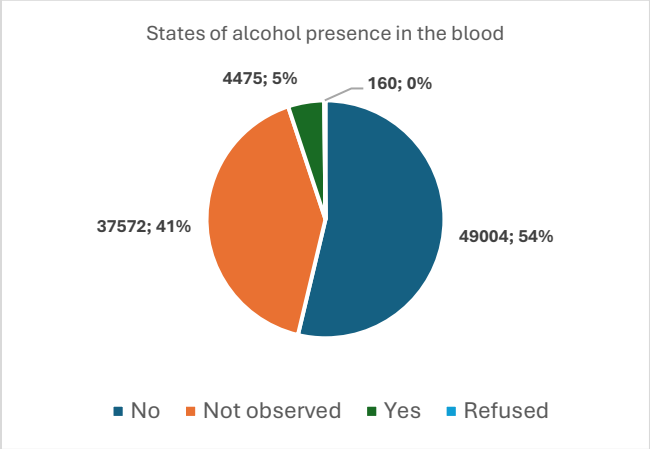

Chart 1 - States of alcohol presence in the blood during an accident

Another interesting fact is that if there was an accident with confirmed alcohol, then the probability of measuring at least 1.5 per mills in the blood is greater than 60%. (Chart 2) This assumption was taken by extracting all result rows (Fig. 4) with positive presence of alcohol in the blood.

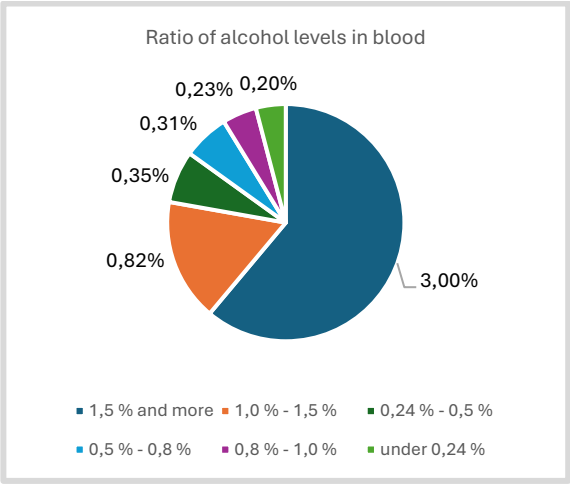

Chart 2 - Amount of alcohol per mill in the blood

IV. Analysis Based on Time

Now that we understand work with our database tables, we can conduct wider analysis. We will start with time analysis of car accidents for the year 2024.

A. Monthly alcohol in-accident involvement

In the previous chapter, we have found out that majority of accidents end up with any alcohol involvement or it was not observed. We should investigate whether this behavior was the same throughout the entire year or if it has been changing. We are going to focus just on four alcohol-type states: alcohol present, not present, not observed, refused. We start by grouping records by extracting the month of accident occurrence. While we want to pivot all these counts of wanted alcohol-types states for each month, we can create sum expressions with *CASE* clauses to count only cases when an accident contains desired alcohol-type state. Besides that, we compute overall counts of cases for each month to compute relative proportions of alcohol-types later. The whole script is shown in Fig. 5. Results are visualized from two perspectives in Chart 3 and 4. In Chart 3 we can see that proportions are similar throughout the entire year. Chart 4 suggests that most alcohol-present accidents were recorded in the summer months July and August. To confirm this observation and see more detailed numbers, we can look at Chart 5, which visualizes numbers of column *alco_yes* from script in Fig.5. We can assume that this could be related to the time of vacation when people are more prone to drink alcohol and forget to be responsible.

```sql
-- monthly ratio of accidents with confirmed alcohol usage among all accidents
select mon_name, ' Amounts: ' as desc1, alco_yes, alco_no, refused, not_found_out,
       ' Percentage: ' as desc2,
       round(100 * alco_yes       /month_sum,2) ||'%' as p_yes,
       round(100 * alco_no        /month_sum,2) ||'%' as p_no,
       round(100 * refused        /month_sum,2) ||'%' as p_refused,
       round(100 * not_found_out  /month_sum,2) ||'%' as p_not_found_out
from (
    select extract(month from cas) as month_id,
           TO_CHAR(cas, 'Month','NLS_DATE_LANGUAGE=English') as mon_name,
           sum(case alco.pritomny when 'A' then 1 else 0 end) as alco_yes,
           sum(case alco.pritomny when 'N' then 1 else 0 end) as alco_no,
           sum(case alco.pritomny when 'O' then 1 else 0 end) as refused,
           sum(case alco.pritomny when 'X' then 1 else 0 end) as not_found_out,
           count(*) as month_sum
    from cr_nehody acc
    join cr_pritomnost_alko alco on acc.id_alko_prit = alco.id_stav
     where extract(year from cas)=2024
      group by extract(month from cas), TO_CHAR(cas, 'Month','NLS_DATE_LANGUAGE=English')
)
 order by month_id
```

	MON_NAME	DESC1	ALCO_YES	ALCO_NO	REFUSED	NOT_FC	DESC2	P_YE	P_NC
1	January	Amounts:	282	4,041	11	3,095	Percentage:	3.8%	54.39%
2	February	Amounts:	258	3,989	11	2,693	Percentage:	4.34%	50.23%
3	March	Amounts:	330	3,473	11	2,982	Percentage:	4.86%	51.1%

Figure 5 - Monthly ratios of alcohol involvement in accidents (script)

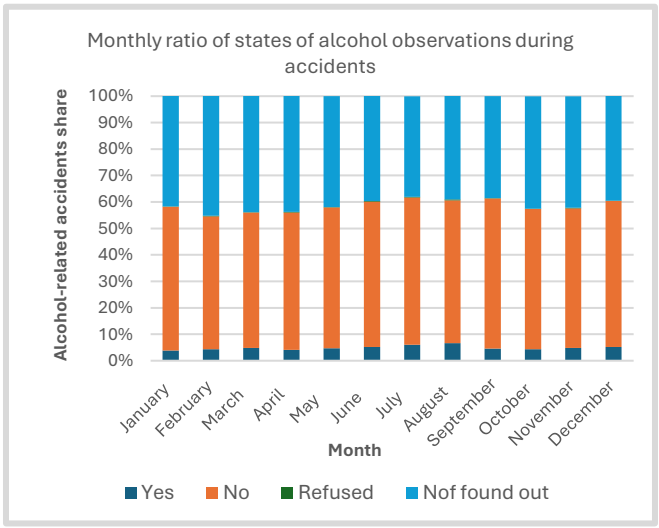

Chart 3 - Monthly ratio of states of alcohol observations during accidents (percentual)

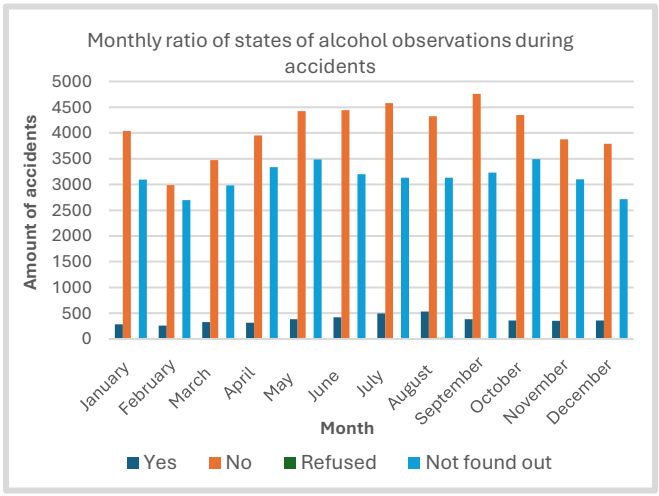

Chart 4 - Monthly ratio of states of alcohol observations during accidents (amounts)

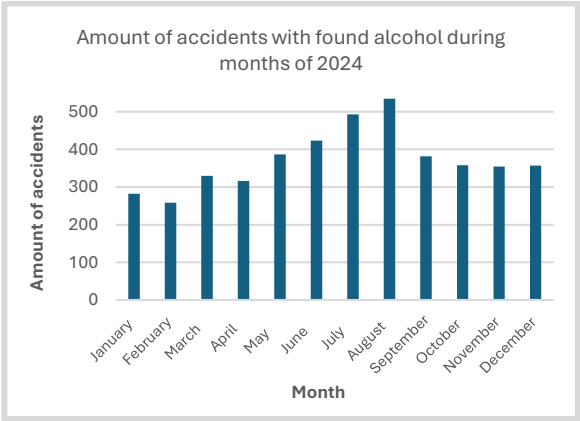

Chart 5 - Number of accidents with found alcohol per month

B. *Amount of accidents based on the hour of day*

Next, we can take a closer look at which hours of the day are most significant regarding accident frequency. We must group records by the hour of the day. This can be done using the *EXTRACT* function where we define that we want to get hour of the day (script is shown in Fig. 6). We can see in Chart 6 that most frequent hours are from 18 to 20 PM when people tend to meet or have more free time. We can also notice that there is a high number of accidents before midnight. This can be related to when people come home from some event where they have drunk.

```
-- in which hours has occured most accidents related to alcohol
SELECT  count(*) AS accidents,
        EXTRACT(HOUR FROM acc.cas) AS hr
 FROM cr_nehody acc
 JOIN cr_pritomnost_alko alco ON acc.id_alko_prit = alco.id_stav
  WHERE EXTRACT(YEAR FROM acc.cas)=2024 AND alco.pritomny LIKE 'A'
  GROUP BY EXTRACT(HOUR FROM acc.cas)
   ORDER BY hr
```

ACCIDENTS	HR
175	0
167	1
131	2

Figure 6 - Number of accidents for each hour of day (script)

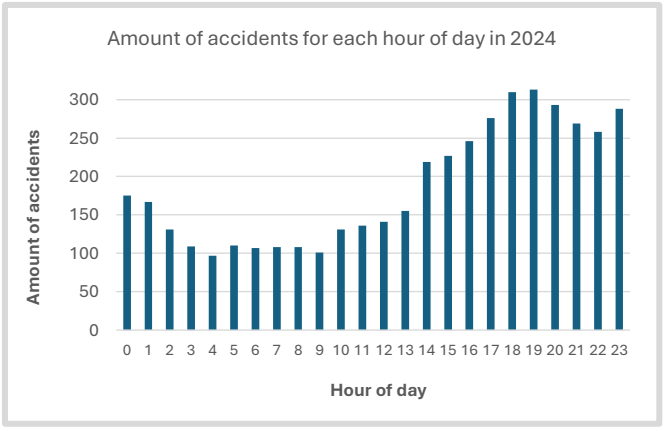

Chart 6 - Number of accidents for each hour of day

C. *Amount of accidents for each day in the week*

Aside from the typical hours associated with frequent accidents, it is also important to observe the development of accident frequency across the days of the week, based on data for the entire year 2024. Query (Fig. 7) is simple but can tell us very interesting information. That is the power of database analytic tools. We group data by converting timestamp to string in format of the day of week using *TO_CHAR* function with parameter 'Day'. Then, we can simply compute the number of cases that occur on their corresponding day of week. The number of accidents (Chart 7) during the weekend is the highest, but especially on Saturday. We can assume that more people are prone to drinking more on Saturday because they have one extra day to get fit before they go to work.

```
SELECT  count(*) AS cnt,
        to_char(acc.cas, 'Day') AS dd
 FROM cr_nehody acc
 JOIN cr_pritomnost_alko alco ON acc.id_alko_prit = alco.id_stav
  WHERE EXTRACT(YEAR FROM acc.cas)=2024 AND alco.pritomny LIKE 'A'
  GROUP BY to_char(acc.cas, 'Day')
   ORDER BY cnt desc
```

CNT	DD
1,048	Saturday
831	Sunday

Figure 7 – Number of accidents for each day of the week (script)

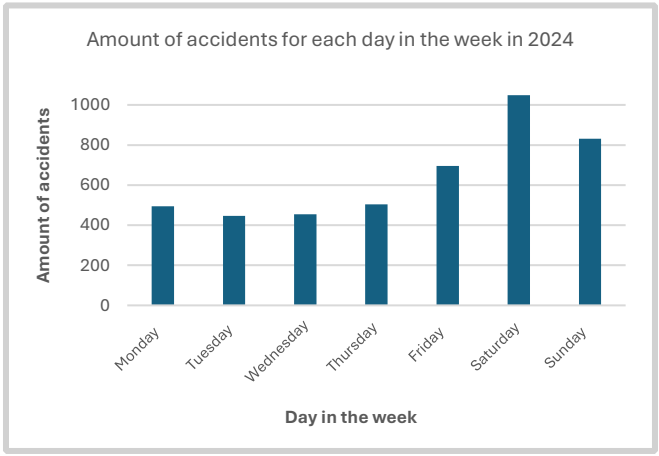

Chart 7 – Number of accidents for each day of the week

V. Analysis Based on Regions

We have already provided insights into the most accident-prone hours of the day, days of the week, and months, along with presenting their behavioral trends. Now moving on analysis on region level to find out which regions are related with the most accidents with alcohol involvement.

A. *Accidents on region level*

Firstly, we will compute the accident amounts for each region using the complete dataset from 2024. Script is shown in Fig. 8. We join accidents with corresponding regions from *regions* table and then group them by region identifier. After this, we calculate the number of accidents for each region.

Results shown in Chart 8 indicate that region Středočeský kraj experienced far more accidents than any other, more specifically over 650 accidents. Three following regions, Moravskoslezský kraj, Prague and Jihomoravský kraj, had over 400 accidents with alcohol in the blood in 2024.

We must bear in mind that each region has a different number of residents; therefore, these figures do not imply that regions in TOP positions have the highest ratios of accidents relative to their resident populations.

```sql
SELECT acc.id_kraj, regions.nazov AS region_name, count(*) AS accidents
 FROM CR_NEHODY acc
 JOIN CR_PRITOMNOST_ALKO alco ON acc.id_alko_prit = alco.id_stav
 JOIN CR_KRAJE regions ON acc.id_kraj = regions.id_kraj
  WHERE extract(YEAR FROM acc.cas)=2024 AND alco.pritomny LIKE 'A'
   GROUP BY acc.id_kraj, regions.nazov
    ORDER BY accidents desc
```

ID_KRAJ	REGION_NAME	ACCIDENTS
1	Středočeský kraj	656
7	Moravskoslezský kraj	439

Figure 8 - accidents of each region (script)

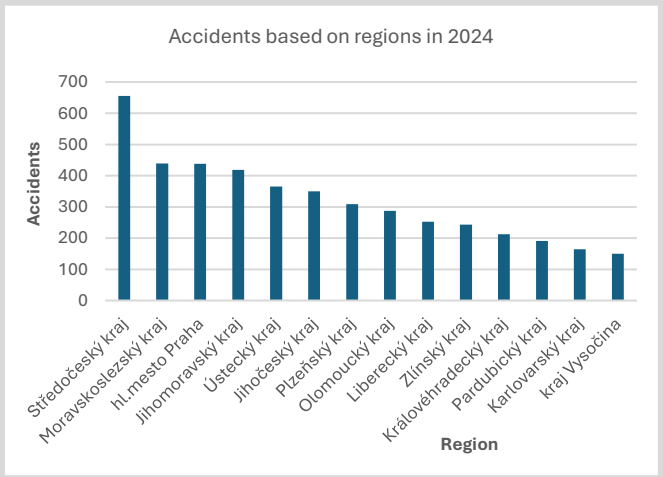

Chart 8 - Number of accidents per region

B. *Regions with consistently high accident occurrences in monthly rankings*

Previously, we have analyzed accidents per region for the whole year. Now we want to know which regions consistently appeared among TOP 3 regions with the highest numbers per month. We also want to know the exact numbers of region appearances in TOP 3.

Firstly, we compute the number of accidents for each group defined by region ID and month of the year. Then, we assign a ranking to every record based on the number of accidents. This ranking is performed independently for each month. Subsequently, we determine how many times each region appeared with ranking lower than or equal to 3 (meaning within the TOP 3). After that, we apply the analytic function *RANK* again, but this time on the number of occurrences in the TOP 3 across all months. The resulting *SELECT* query is shown in Fig. 9.

Now we see from looking at Chart 9 that the first three regions are different from the first three regions in Chart 8. Regions that have occurred most times in TOP 3 are Karlovarský kraj (11 times) and Vysočina (10 times). These results are absolute, not proportional.

It is important to note that the result of this query could be empty if different regions experienced the highest number of accidents each month. However, this was not our case. If it was, we would simply modify the constant against which the rank value is compared or remove WHERE condition entirely.

```sql
SELECT id_region, region_name, being_in_top_3,
       RANK() OVER (ORDER BY being_in_top_3 desc) AS ranking
 FROM
(
    SELECT id_region, region_name, count(*) AS being_in_top_3
     FROM
     (
        SELECT id_region, region_name,
               rank() OVER (PARTITION BY month_id ORDER BY alco_accidents) AS rnk,
               alco_accidents
         FROM
         (
            SELECT count(*) AS alco_accidents,
             regions.id_kraj AS id_region, regions.nazov AS region_name,
             extract(MONTH FROM acc.cas) AS month_id
             from cr_nehody acc
             JOIN CR_KRAJE regions ON acc.id_kraj = regions.id_kraj
             JOIN CR_PRITOMNOST_ALKO alco ON acc.id_alko_prit = alco.id_stav
              WHERE EXTRACT(YEAR FROM acc.cas) = 2024 AND alco.pritomny LIKE 'A'
               GROUP BY regions.id_kraj, regions.nazov, extract(MONTH FROM acc.cas)
         )
    ) WHERE rnk <= 3
     GROUP BY id_region, region_name
)
```

ID_REGION	REGION_NAME	BEING_IN_TOP_3	RANKING
19	Karlovarský kraj	11	1
16	kraj Vysočina	10	2

Figure 9 - Regions appearing most times in TOP 3 monthly accidents count (script)

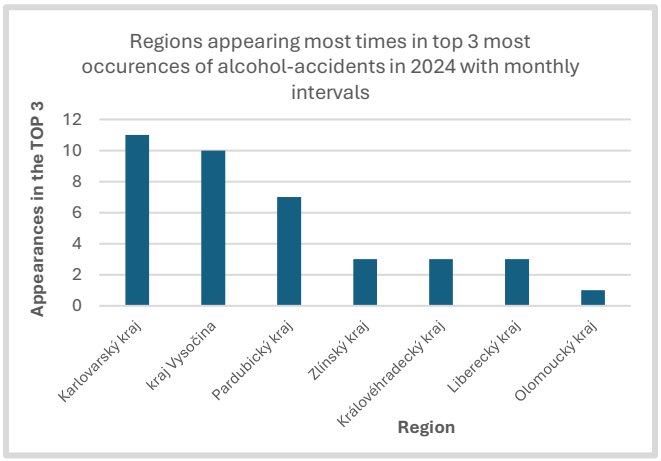

Chart 9 - Regions appearing most times in TOP 3 monthly accidents count

C. *Weighted average ratio of alcohol-involved accidents per region and month*

We move from absolute to relative numbers. As shown in Fig. 10, we compute the weighted average ratio of monthly accidents counts with alcohol involvement relative to all accidents in given month for each region. The number of days serve as the weight. Assigning IDs is done to accomplish having results of both united select results in one row. This technique was used in the third chapter also. Weights representing the number of days for specific month are computed as composing the last day of month which is then extracted from the resulting value. Finally, ranking is assigned to each result by ordering output based on weighted average ratio. Results revealed that the highest percentage ratio is 7% and these value reached three out of fourteen regions - Jihomoravský kraj, Jihočeský kraj, and Plzeňský kraj.

```sql
SELECT DENSE_RANK() OVER (ORDER BY wavg_ratio desc) AS ranking, region_name,
        round(wavg_ratio, 3) || '%' wavg_ratio, round(avg_ratio, 3) || '%' avg_ratio
FROM (SELECT id_region, region_name,
      avg((alco_amount/overall_amount)*100) AS avg_ratio,
      sum(days_in_month * (alco_amount/overall_amount)*100/365) AS wavg_ratio
   FROM
   (SELECT id_region, region_name,
        month_id,
        max(CASE WHEN amount_type_id = 1 THEN accidents ELSE NULL END) AS alco_amount,
        max(CASE WHEN amount_type_id = 2 THEN accidents ELSE NULL END) AS overall_amount,
        extract(DAY FROM last_day(to_date('2024-' || month_id || '-01', 'YYYY-MM-DD')))
        AS days_in_month
        FROM (
           (SELECT 1 AS amount_type_id, count(*) AS accidents, -- alco amounts
            regions.id_kraj AS id_region, regions.nazov AS region_name,
            extract(MONTH FROM acc.cas) month_id
            from cr_nehody acc
            JOIN CR_KRAJE regions ON acc.id_kraj = regions.id_kraj
            JOIN CR_PRITOMNOST_ALKO alco ON acc.id_alko_prit = alco.id_stav
             WHERE EXTRACT(YEAR FROM acc.cas) = 2024 AND alco.pritomny LIKE 'A'
              GROUP BY regions.id_kraj, regions.nazov, extract(MONTH FROM cas))
          UNION ALL
          (SELECT 2 AS amount_type_id, count(*) AS accidents, -- overall amounts
           regions.id_kraj AS id_region, regions.nazov AS region_name,
           extract(MONTH FROM acc.cas) month_id
           from cr_nehody acc
           JOIN CR_KRAJE regions ON acc.id_kraj = regions.id_kraj
            WHERE EXTRACT(YEAR FROM acc.cas) = 2024
             GROUP BY regions.id_kraj, regions.nazov, extract(MONTH FROM acc.cas))
       ) GROUP BY id_region, region_name, month_id
   ) GROUP BY id_region, region_name
     ORDER BY wavg_ratio)
```

RANKING	REGION_NAME	WAVG_RATIO	AVG_RATIO
1	Jihomoravský kraj	7,448%	7,427%

Figure 10 - Weighted averages of monthly alcohol-related accident ratios (script)

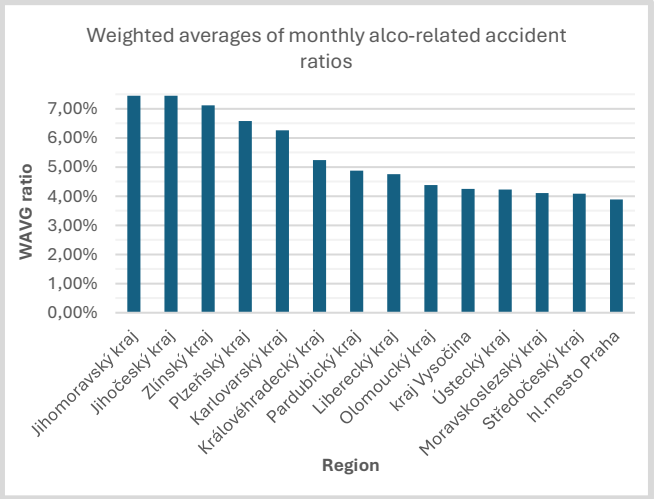

Chart 10 - Weighted averages of monthly alcohol-related accident ratios

D. TOP 3 hours with most alcohol-accidents for each region

In the previous chapter we created a query to find out the most frequent hours of the day regarding accidents with alcohol involvement. We found out that most frequent hours are 18, 19, 20 and 23. Let us now break this analysis into smaller units – regions. We will observe TOP 3 most frequent hours for each region of Czech Republic and see whether these hours are most frequent almost everywhere or not.

We create groups of accidents by hour of the day and corresponding region ID to find out the wanted number of accidents (Fig. 11). Then we use the analytic function *ROW_NUMBER* to assign rankings to hours (based on descending ordering for the number of accidents) within partition defined by region ID. We use *ROW_NUMBER* function instead of *RANK* function, because we will create exactly three series for TOP 3 most frequent hours. Lastly, we filter output to get just results with ranking better or equal to a constant three and then we order results by assigned rankings to group numbers for TOP 1, 2 and 3 hours. We also then order results by region ID to get numbers of corresponding region on the same position in every TOP time series.

To understand the visualization of Chart 11, we can take an example of Prague. We can see that in Prague, most accidents occur during 17-th hour, the second most frequent time was 23 PM and third was 01 AM. Judging by the results from Chart 11, we cannot say that TOP 1 most accidents occur at the same hour in each region. It depends on the region. On the other hand, what we can say is that generally, the most frequent accident hours are surely after 16-th hour.

```sql
SELECT  region_id, region_name, cnt, ihour, rnk
FROM
(
    SELECT  region_id, region_name, cnt, ihour,
            row_number() OVER (PARTITION BY region_id ORDER BY cnt desc) AS rnk
    FROM (
        SELECT  regions.id_kraj AS region_id, regions.nazov AS region_name,
                count(*) AS cnt, EXTRACT(HOUR FROM acc.cas) AS iHour
         FROM cr_nehody acc
         JOIN cr_pritomnost_alko alco ON acc.id_alko_prit = alco.id_stav
         JOIN cr_kraje regions ON acc.id_kraj = regions.id_kraj
          WHERE EXTRACT(YEAR FROM acc.cas)=2024 AND alco.pritomny LIKE 'A'
          GROUP BY EXTRACT(HOUR FROM acc.cas), regions.id_kraj, regions.nazov
           ORDER BY cnt DESC
        )
) WHERE rnk <= 3
  ORDER BY rnk, region id
```

REGION_ID	REGION_NAME	CNT	IHOUR	RNK
1	hl.mesto Praha	40	17	1

Figure 11 - TOP 3 hours with most alcohol-type accidents for each region (script)

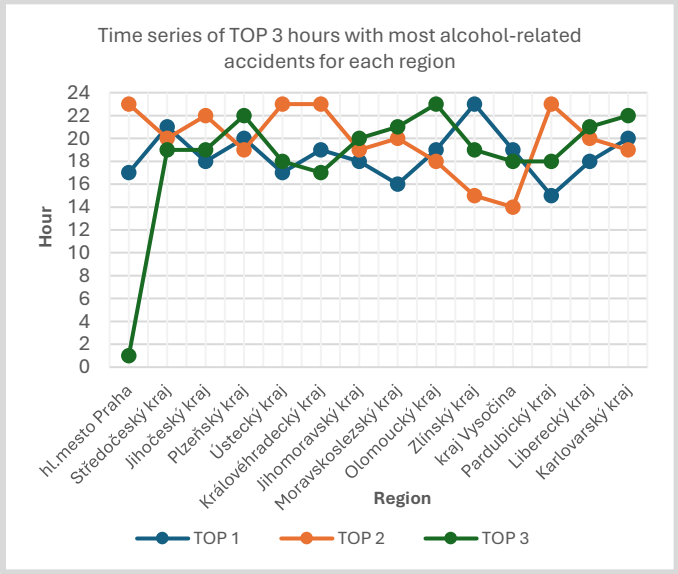

Chart 11 - TOP 3 hours with most alcohol-type accidents for each region

VI. Analysis of Damages

The final topic concerns the damage caused by accidents in which alcohol was present at any positive per mill level.

A. Average number of vehicles involved in an accident

To begin with, we can review the statistics on the average number of cars involved in accidents with the presence of alcohol.

But before that, we can get middle value of the number of vehicles per accident. Table of accidents provides us with number of cars involved in a single accident, so we can apply aggregate function *MEDIAN* over this table (Fig. 12) to get the middle value. From the dataset, we observed a median of two vehicles per accident. A single-vehicle accident is also a possible scenario, particularly in cases involving a collision

with a pedestrian, an animal (such as forest game) or fixed object.

```
SELECT median(acc.poc_vozidiel) AS MEDI4N
    FROM cr_nehody acc
```

MEDI4N
2

Figure 12 - Median of vehicles per accident (script)

Let us now compute the weighted average of cars involved. As Fig. 13 suggests, we use here a common table expression (CTE) named as *vehicles_amount* for better readability. In that *vehicles_amount* we group records by vehicles involved in an accident so we can compute the number of such cases. We also want to get the number of vehicles involved, because this value will be used as the weight for computing the average with weights in the next part. For our curiosity we add aggregate function *MAX* to find out maximum vehicles involved in a single accident.

Results from Fig. 13 indicate that, on average, 1.5 vehicles are involved in an accident. Besides that, there was at least one accident in which 9 cars crashed. This reveals how dangerous drinking alcohol can be before entering the road. For better understanding of how many accidents occurred for each possible case of the number of vehicles per accident, we plot histogram of these values. (Chart 12) We observe that most alcohol-related accidents involved one or two vehicles, with single-vehicle incidents being the most prevalent in over 2,500 cases.

```
WITH vehicles_amount AS (
    SELECT vehicles_per_acc, count(*) AS accidents
     FROM
    (
        SELECT acc.poc_vozidiel AS vehicles_per_acc
         FROM cr_nehody acc
         JOIN cr_pritomnost_alko alco ON acc.id_alko_prit = alco.id_stav
          WHERE EXTRACT(YEAR FROM acc.cas)=2024 AND alco.pritomny LIKE 'A'
    ) GROUP BY vehicles_per_acc
)
SELECT round(sum(vehicles_per_acc*accidents)/sum(accidents), 3) AS wavg,
       max(vehicles_per_acc) AS M4X
 FROM vehicles_amount
```

WAVG	M4X
1.521	9

Figure 13 - Weighted average number of vehicles in alcohol-related accidents (script)

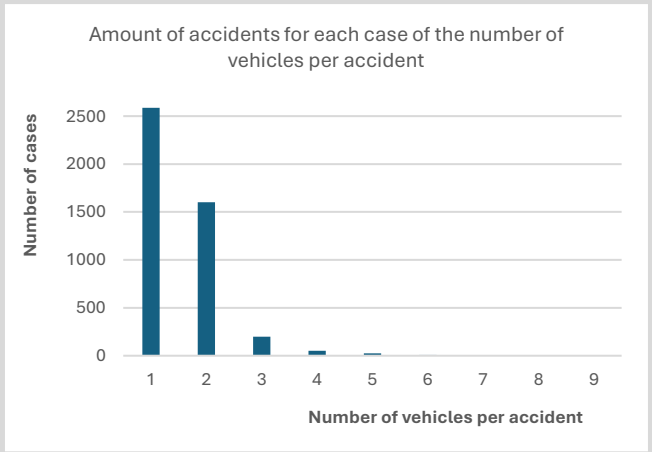

Chart 12 - Number of accidents for each possible number of vehicles per accident

B. 2.5-month moving median of injured per half-month

In the *acc* table, information regarding the number of slightly and seriously injured individuals is stored in two separate columns. For this analysis, we aim to determine the trend and seasonality of a time series composed of the total number of injured persons, which is derived by summing both slightly and seriously injured counts.

To make it more interesting we will compose time series with frequency equal to half-month. This means that we must sum the number of injuries into half-month intervals. We will divide the entire script into two parts. In the first (Fig. 14), we will focus on creating half-month data, which we will use then in the second part (Fig. 15).

In the first part of this process, we begin by creating records with numbers 1-12 to represent monthly numbers. These are used in CTE named *months*, where we compose dates of months beginnings using the *TO_DATE* function with the corresponding month number. As the next step, we create another CTE named *halfmonths* containing half-month dates. We take all dates from the *months* CTE and union them with the newly created half-month dates. These half-month dates are generated by determining half the number of days between the beginning and the end of a specific month (plus one day for the midpoint) and then adding this resulting half-month days count to the beginning of the month. In the last step of the first part, we create CTE named *halfmon_data* where (in the inner select) we join *acc* table with *halfmonths* on their corresponding beginning of the month. This way, we get two result rows for each accident: one with the date of the beginning of the month and the other with the date in the middle of the month. We want to retrieve only one of these two resulting rows. To retrieve only one of these two resulting rows, we introduce an *hms_priority* column into the inner select query. This column assigns priority by ranking both rows (partitioning is done for pairs). Specifically, the *ROW_NUMBER* function, utilizing a *CASE* statement in its ordering clause, assigns the highest priority to the half-month date that is chronologically closest to and immediately following the accident's occurrence time. After that, we can filter rows to get only those which have the highest priority (*hms_priority*=1) and then compute sums of injuries for all groups defined by their closest half-month.

In the second part of the script, we utilize the *halfmon_data* CTE. We note that the *halfmon_data* CTE contains a distinct row for each half-month, meaning the half-month value is unique within *halfmon_data*. We aim to union each half-month with its corresponding number of injuries and with the number of injuries from the four previous half-months. Retrieving these previous numbers is achieved by using the analytic function *LAG*, within which we define the constant for the N-th previous lag and specify the ordering of rows by half-month date value before picking the lags. We do this, because we compute 2.5-month moving median, that means we need five half-months (specific half-month + 4 previous).

The resulting time series is visualized in Chart 13. We can observe an ascending trend from the beginning of the year 2024 up to mid-September, after which the trend changes to descending. No seasonality is apparent in this single year's data; however, analyzing data over multiple years could reveal signs of cyclicality. An unambiguous observation is that most injuries during accidents with alcohol involvement occurred during the summer months.

```
with month_nr as (
    select 1 as month_id from dual
    union select 2 from dual
    union select 3 from dual
    union select 4 from dual
    union select 5 from dual
    union select 6 from dual
    union select 7 from dual
    union select 8 from dual
    union select 9 from dual
    union select 10 from dual
    union select 11 from dual
    union select 12 from dual
),
months as (
    select month_id,
    to_date('2024-'||month_id||'-01', 'yyyy-mm-dd') as month_bt
      from month_nr
),
halfmonths as (
    select month_bt as halfmon from months
     union
    select
    month_bt + floor((last_day(month_bt)-month_bt+1)/2) as halfmon
     from months
),
halfmon_data as (
select halfmon, sum(count_injured) as count_injured
from (
    select halfmon, count_injured
     from (
        select acc.id_nehoda, hms.halfmon,
         (acc.lahko_zraneni + acc.tazko_zraneni) as count_injured,
         row_number() over (partition by id_nehoda order by
          (case when trunc(acc.cas)-hms.halfmon>=0
           then trunc(acc.cas)-hms.halfmon else 10000 end)) as hms_priority
         from cr_nehody acc
          join halfmonths hms on trunc(acc.cas, 'MM')=trunc(hms.halfmon, 'MM')
          join cr_pritomnost_alko alco on acc.id_alko_prit = alco.id_stav
           where extract(year from acc.cas)=2024 AND alco.pritomny LIKE 'A'
     ) where hms_priority=1 -- each record is going to have assigned 1 halfmon
    ) group by halfmon -- development of injuries number with frequency equal t
)
```

Figure 14 - Composing half-month data for computing moving median of injured (script)

```
    ) group by halfmon -- development of injuries number with frequ
)
select halfmon, median(count_injured) median_injured
 from (
    select halfmon, count_injured from halfmon_data
     union all select halfmon,
     lag(count_injured, 1) over(order by halfmon) from halfmon_data
     union all select halfmon,
     lag(count_injured, 2) over(order by halfmon) from halfmon_data
     union all select halfmon,
     lag(count_injured, 3) over(order by halfmon) from halfmon_data
     union all select halfmon,
     lag(count_injured, 4) over(order by halfmon) from halfmon_data
) group by halfmon order by halfmon
```

HALFMON	MEDIAN_INJURED
2024-01-01 00:00:00.000	46
2024-01-16 00:00:00.000	49.5
2024-02-01 00:00:00.000	47
2024-02-15 00:00:00.000	50

Figure 15 - Computing windowed median on grouped half-month numbers of injuries (script)

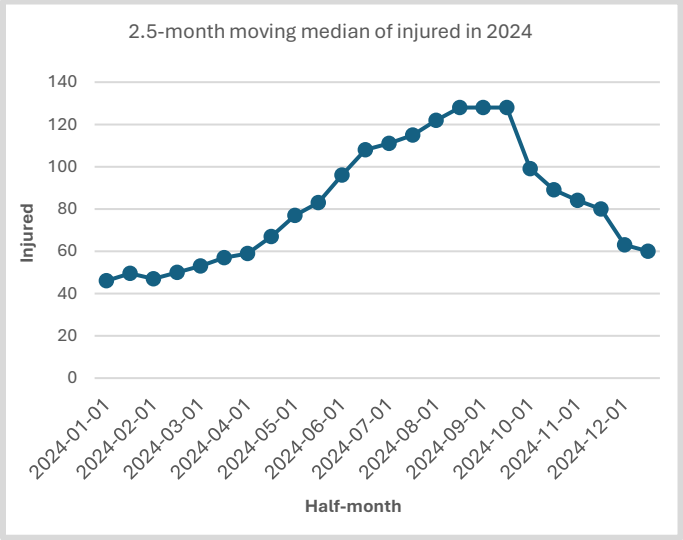

Chart 13 – 2.5-month moving median of injured per half-month

C. *Contribution of alcohol-related accidents to total damage in the TOP 1000 most costly accidents*

Lastly, we compute the percentage of total damage attributable to alcohol-related accidents among the top 1000 most costly accidents. We are curious whether alcohol contribution has great consequences or not in those 1000 cases. For better understanding of how we are going to construct select queries, we visualize it first using Fig. 16. We must create two select queries, *damage_alco* and *damage_overall*, from which we will then compute wanted ratio.

```
select damage_alco, damage_overall,
round(100 * damage_alco/damage_overall, 2) || '%' as perc_ratio
 from (
     select
     (select 1 from dual) as damage_alco,
     (select 2 from dual) as damage_overall
     from dual
)
```

DAMAGE_ALCO	DAMAGE_OVERALL	PERC_RATIO
1	2	50%

Figure 16 - Template for query of contribution of alcohol-related accident damages (script)

Both queries (Fig. 17) are straightforward. Starting with the *damage_overall* query, we assign a ranking to all accidents based on the damage caused by the accident (using the value of the *acc.celk_skoda_kc* column). From these results, we select all rows with a ranking lower than or equal to a constant value of 1000. The *damage_alco* query is constructed similarly, but when extracting the resulting rows, we add a filtering *WHERE* condition that specifies accidents must have some alcohol involvement. Finally, we compute the proportion of the results from these two queries and transform it into a percentage format.

As the results (Chart 14) suggest, alcohol-related accidents account for only a 4% share of the damage caused by the top 1000 most costly accidents.

```
select 'Share of alcohol-related accidents damage among TOP
       1000 most costly damages:' as description,
       round(100*(damage_alco/damage_overall),3) || '%' as perc,
       damage_alco, damage_overall
 from (select
    (select sum(damage)
      from (
            select alco.pritomny as alco_present,
                   acc.celk_skoda_kc as damage,
                   row_number() over(order by acc.celk_skoda_kc desc) as rn
          from cr_nehody acc
           join cr_pritomnost_alko alco on acc.id_alko_prit = alco.id_stav
            where extract(year from acc.cas)=2024
           ) where rn <= 1000 and alco_present='A'
    ) AS damage_alco,
    (select sum(damage)
      from (
        select celk_skoda_kc as damage,
            row_number() over(order by acc.celk_skoda_kc desc) as rn
          from cr_nehody acc
           where extract(year from acc.cas)=2024
           ) where rn <= 1000
     ) AS damage_overall
    from dual)
```

DESCRIPTION	PERC	DAMAGE_ALCO	DAMAGE_OVERALL
Share of alcohol-related	3.81%	53,773,000	1,411,356,000

Figure 17 - contribution of alcohol-related accident damages among TOP 1000 (script)

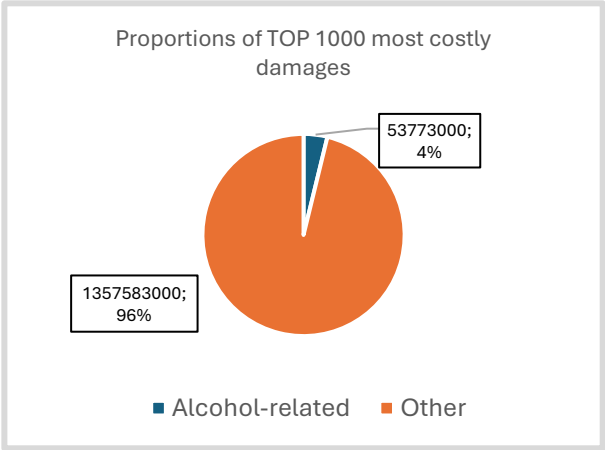

Chart 14 - Proportions of TOP 1000 most costly damage

VII. Optimization of Queries

Before we try any optimization of queries, we should regenerate statistics first. (Fig 18)

```
499 BEGIN
500     dbms_output.put_line('Running gather_schema_stats...');
501     dbms_stats.gather_schema_stats('YOUR_SCHEMA_NAME');
502     dbms_output.put_line('Done.');
503 END;
```

Results 1 | Results 2 | Results 3 | Statistics 4 | Execution plan - 1 | Output

```
Running gather_schema_stats...
Done.
```

Figure 18 - Regenerating database schema statistics

While our analysis frequently involves joining the *acc* table with the *alco* table, it is desirable to create an index on the foreign key, which by default does not exist. A bitmap index could be considered (Fig. 19) because the number of distinct values in the alco lookup table is finite. More specifically, there are only 10 distinct states, which is significantly less than 1% of the acc table's cardinality, while the number of records in the *acc* table exceeds 91,000 as observed in the third chapter. Before creating the index, we examine the execution plan to ascertain the query's costs. We use the query from Fig. 11 for testing purposes. The execution plan revealed that the costs of this query are 772 units. After creating the index and re-executing the *SELECT* statement, the same results were obtained (Fig. 19). Fig. 19 also illustrates that a hash join was employed instead of utilizing the newly created bitmap index. The reason for this could be the small number of records in the lookup table, or even in the main *acc* table itself.

```
530 create bitmap index idx_acc_alco_type_bitmap on cr_nehody(id_alko_prit);
```

Statistics 1 | Execution plan - 6

Operation	Object	Optimizer	Cost	Cardinality	Bytes
SELECT STATEMENT		ALL_ROWS	772	1,273	87,837
SORT (ORDER BY)			772	1,273	87,837
VIEW			771	1,273	87,837
WINDOW (SORT PUSHED RANK)			771	1,273	58,558
VIEW			770	1,273	58,558
SORT (ORDER BY)			770	1,273	54,739
HASH (GROUP BY)			770	1,273	54,739
HASH JOIN			768	1,273	54,739
TABLE ACCESS (F	CR_KRAJE	ANALYZED	3	14	294
HASH JOIN			765	1,273	28,006
TABLE ACCESS	CR_PRITOMNOST	ANALYZED	3	6	30
TABLE ACCESS	CR_NEHODY	ANALYZED	762	1,909	32,453

Figure 19 – Results of execution plan after creating bitmap index on foreign key referenced to alco table

Moving to another consideration, we could try optimizing queries that use the timestamp column *cas* from *acc* table. This column is used in almost all our queries in *WHERE* conditions when extracting records from the year 2024. We also have data from the year 2023 and some from 2025 in the *acc* table so creating a b-tree index could be reasonable optimization. We will use the query from Fig. 10 for testing purposes. The execution plan for this query initially revealed costs equal to 1538 units. It is important to create a functional index rather than a normal one, because we filter not directly on the *acc.cas* column but on the result of the *EXTRACT* function applied to it. Created functional b-tree index (Fig. 20) was successfully applied two times in the query. Total costs dropped from 1538 units to just 303 units as Fig. 20 suggests.

```
547 create index idx_acc_year_btree on cr_nehody(extract(year from cas));
```

esults 1 | Execution plan - 1 | Execution plan - 2

Operation	Object	Optimizer	Cost	Cardinality	Bytes
SELECT STATEMENT		ALL_ROWS	303	139	5,977
WINDOW (SORT)			303	139	5,977
VIEW			302	139	5,977
SORT (ORDER BY)			302	139	8,340
HASH (GROUP BY)			302	139	8,340
VIEW			301	139	8,340
HASH (GROUP BY)			301	139	6,950
VIEW			300	3,182	159,100
UNION-ALL			299	0	0
HASH (GROUP			151	1,273	71,288
HASH (GROUP			148	1,909	91,632

Figure 20 - Functional index on time column of acc table

VIII. Conclusion

We have carried out analysis of car accidents in 2024 dataset from several points of view: overall frequency, time, region and damage analysis. Regarding caused damage, we can say that the results are surprising, because we would say that alcohol has bigger impact on accident consequences regarding the number of accidents or total damage caused by them. This declaration is supported by our analytical objectives, which are predominantly focused on relative numbers over absolute measures in certain tasks. As observed in chapter 3, the number of accidents where the presence of alcohol was not observed was over 40%, which is a huge number. Results might have looked different if alcohol involvement had been measured in those cases. According to [2], Czech Republic had about 10.91 million residents at the

end of 2024. Considering the number of alcohol-related accidents (4,475 incidents) and the limited damage attributed to them (only 4% of costs within the top 1000 most costly accidents), it appears that residents of the Czech Republic generally exhibit relative caution and responsibility concerning alcohol consumption and road safety. From a time perspective, monthly numbers of alcohol-related accidents are highest during summer months. This goes with the biggest number of injuries during summer. The biggest number of alcohol-related accidents occur on Saturday. Most frequent hours of the day are in the evening, specifically the time from 18 to 20 PM and 23 PM from a general perspective. If we look at the most frequent hours from a region's perspective, we can observe that every region has different most frequent hours, but majority of them are between 16 and 23 PM. The highest number of accidents was recorded in the Středočeský kraj region (656 accidents), followed by the Moravskoslezský kraj region (439 accidents). Furthermore, the highest monthly ratio of alcohol-related accidents relative to all accidents within a specific month and region was observed in three regions (out of 14), each recording a ratio over 7%. These regions were Jihomoravský kraj, Jihočeský kraj, and Zlínsky kraj. At the end, we showed an example of successful optimization of some queries by implementing functional index. This paper suggests the application of database analytic tools (in this paper was used Oracle as a database platform) for robust data analysis. Once their application is mastered, they are both powerful and straightforward to use. Using this method assumes well-collected data in a structured way. This approach to data analysis can serve as a motivation for national institutions and other stakeholders to gain a deeper understanding of the processes they manage, thereby enabling the implementation of more effective strategies.

REFERENCES

[1] Policie České republiky, "Statistika nehodovosti," *Policie České republiky*, [Online]. Available: https://policie.gov.cz/clanek/statistika-nehodovosti-900835.aspx. [Accessed: July 8, 2025].

[2] Czech Statistical Office, "Population estimates, structure, and projection," *Czech Statistical Office*, May 16, 2025. [Online]. Available: https://csu.gov.cz/population-estimates-structure-and-projection. [Accessed: July 8, 2025]