## Query Implementation

**Query 1:**

***Description of logic:***

List the year-number of collisions per year. We use “group by” to group case by year (extracted from col\_date) and count the number of cases of each year.

***SQL statement***

SELECT EXTRACT (YEAR FROM col\_date) AS YEAR, count(\*) AS N\_collisions

FROM case

GROUP BY EXTRACT (YEAR FROM col\_date)

ORDER BY YEAR ASC

***Query result (if the result is big, just a snippet)***

|  |  |
| --- | --- |
| YEAR | N\_COLLISIONS |
| 2001 | 522562 |
| 2002 | 544741 |
| 2003 | 538954 |
| 2004 | 538295 |
| 2005 | 532725 |
| 2006 | 498850 |
| 2007 | 501908 |
| 2017 | 7 |
| 2018 | 21 |

**Query 2:**

***Description of logic:***

In the “take” table, group entries by “ve\_make” and count the number of parties of each “ve\_make”, then find the max count and the corresponding “ve\_make”. Before that we need to use “ve\_number” to know the “ve\_make”, so we first join table “vehicle” and “party\_involve”. To illustrate the whole row of the most popular, we fetch first 1 row only.

***SQL statement***

SELECT VE\_MAKE, COUNT(\*) AS N\_COLLISION

FROM (VEHICLE INNER JOIN PARTY\_INVOLVE ON VEHICLE.VE\_NUM = PARTY\_INVOLVE.VE\_NUM)

GROUP BY VE\_MAKE

ORDER BY N\_COLLISION DESC

FETCH FIRST 1 ROWS ONLY

***Query result (if the result is big, just a snippet)***

|  |  |
| --- | --- |
| VE\_MAKE | N\_VEHICLE |
| FORD | 1129701 |

**Query 3:**

***Description of logic:***

In the lighting attribute of condition, find the description that contains “dark”, and count the fraction of cases that occur in such condition, and and keep 2 significant digits..

***SQL statement***

SELECT ROUND(NOM/(SELECT COUNT(\*) FROM CASE),2)

FROM

(SELECT COUNT(\*) AS NOM

FROM CASE

WHERE CASE.LIGHTING LIKE '%dark%')

***Query result (if the result is big, just a snippet)***

|  |
| --- |
| FRACTION |
| 0.28 |

**Query 4:**

***Description of logic:***

Find the number of collisions that have occurred under snowy weather. We count the number of entries that have weather\_con = ‘snowing’ in the table “weather\_en”

***SQL statement***

SELECT COUNT(\*)

FROM WEATHER\_EN

WHERE WEATHER LIKE '%snowing%'

***Query result (if the result is big, just a snippet)***

|  |
| --- |
| N\_COLLISIONS |
| 8530 |

**Query 5:**

***Description of logic:***

Group by collisions by which day they are during a week, and count the total number of collisions of that day, then find the row of highest number of cases. We use TO\_CHAR (COL\_DATE, 'D') to extract the day of the week.

***SQL statement***

SELECT TO\_CHAR(COL\_DATE, 'D') AS WEEK\_DAY, COUNT(\*) AS N\_COLLISONS

FROM CASE

GROUP BY TO\_CHAR(COL\_DATE, 'D')

ORDER BY N\_COLLISONS DESC

FETCH FIRST 1 ROWS ONLY

***Query result (if the result is big, just a snippet)***

|  |  |
| --- | --- |
| WEEK\_DAY | N\_COLLISIONS |
| 6 | 614853 |

**Query 6:**

***Description of logic:***

List all weather types and their corresponding number of collisions in ascending order of the collisions.We group cases by “weather” and list “weather” and the count number.

***SQL statement***

SELECT WEATHER, COUNT(\*) AS N\_COLLISION

FROM WEATHER\_EN

GROUP BY WEATHER

ORDER BY N\_COLLISION

***Query result (if the result is big, just a snippet)***

|  |  |
| --- | --- |
| WEATHER | N\_COLLISION |
| other | 6960 |
| snowing | 8530 |
| wind | 13952 |
| fog | 21259 |
| raining | 223752 |
| cloudy | 548250 |
| clear | 2941042 |

**Query 7:**

***Description of logic:***

Count the number of parties that are at-fault, with financial responsibility and loose material. We first extract the “road\_num” of “road\_loose”, and find which parties are associated with such road condition. We filter the “party\_id” table who is at fault and with financial responsibility. Finally we count the number of the selected parties.

***SQL statement***

SELECT COUNT(\*) AS N\_PARTIES

FROM PARTY\_INVOLVE P, ROAD\_EN R

WHERE P.CASE\_ID = R.CASE\_ID AND P.AT\_FAULT = 1 AND P.FIN\_RESP = 'Y' AND R.ROAD\_CON LIKE '%loose material%'

***Query result (if the result is big, just a snippet)***

|  |
| --- |
| N\_PARTIES |
| 4803 |

**Query 8:**

***Description of logic:***

Find the median victim age: we directly use the “MEDIAN” function of SQL from the associate\_victim table.

Find the most common victim seating position. We group the victims with seating position, and count the number of victims of each vic\_seat, order them in the descending order of this number and find the max.

***SQL statement***

***8.a***

SELECT MEDIAN(vic\_age) AS MEDIAN\_VIC\_AGE

FROM ASSOCIATE\_VICTIM v2;

***8.b***

SELECT VIC\_SEAT AS MOST\_COMMON\_SEAT\_POSITION

FROM

(SELECT COUNT(vic\_seat) AS count, vic\_seat

FROM associate\_victim v2

GROUP BY vic\_seat

ORDER BY count DESC)

FETCH FIRST 1 ROWS ONLY;

***Query result (if the result is big, just a snippet)***

***8.a***

|  |
| --- |
| MEDIAN\_VIC\_AGE |
| 25 |

***8.b***

|  |
| --- |
| MOST\_COMMON\_SEAT\_POSITION |
| 3 |

**Query 9:**

***Description of logic:***

Fraction of all participants (victims + parties) that have been victims using a belt. All participants refer to both parties and victims, so our denominator is the sum of number of all victims and parties. We first extract the ‘vic\_id’s who use belt using table have\_vs and safety\_equip\_en. Then we count the unique ‘vic\_id’s and use this number as the numerator. Finally we get the fraction and keep 2 significant digits.

***SQL statement***

SELECT ROUND(A.FRACTION,3) AS fraction

FROM(SELECT DISTINCT

(SELECT COUNT(vic\_id) AS count

FROM

(SELECT h1.vic\_id as vic\_id

FROM SAFETY\_V H1

WHERE H1.SAFETY\_EQUIP like '%C%') v\_belt)/

((SELECT COUNT(party\_id) FROM party\_involve)

+(SELECT COUNT(vic\_id) FROM associate\_victim)) as fraction

FROM party\_involve) a

***Query result (if the result is big, just a snippet)***

|  |
| --- |
| FRACTION |
| 0.011 |

**Query 10:**

***Description of logic:***

Compute the fraction of collisions happening for each hour of the day, and display as ratio as fraction for all the hours of the day. We first use cast(col\_time as timestamp) to extract the hour in which the case occurred. Then we group the cases by the specific hour and count the number of the cases, then order them by the number. We also calculate the total number of the cases. Then we divide the count number of each hour by the total number to get each fraction.

***SQL statement***

SELECT DISTINCT

EXTRACT(hour from cast(col\_time as timestamp)) as hour, ROUND((COUNT(\*)/(SELECT COUNT(\*) FROM CASE)),3) as FRACTION

FROM CASE

GROUP BY EXTRACT(hour from cast(col\_time as timestamp))

ORDER BY hour ASC

***Query result (if the result is big, just a snippet)***

