DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Databases Project – Spring 2021

Team No: G34

Members: Siran Li, Ke Wang, Ningwei Ma

Contents

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Deliverable 1

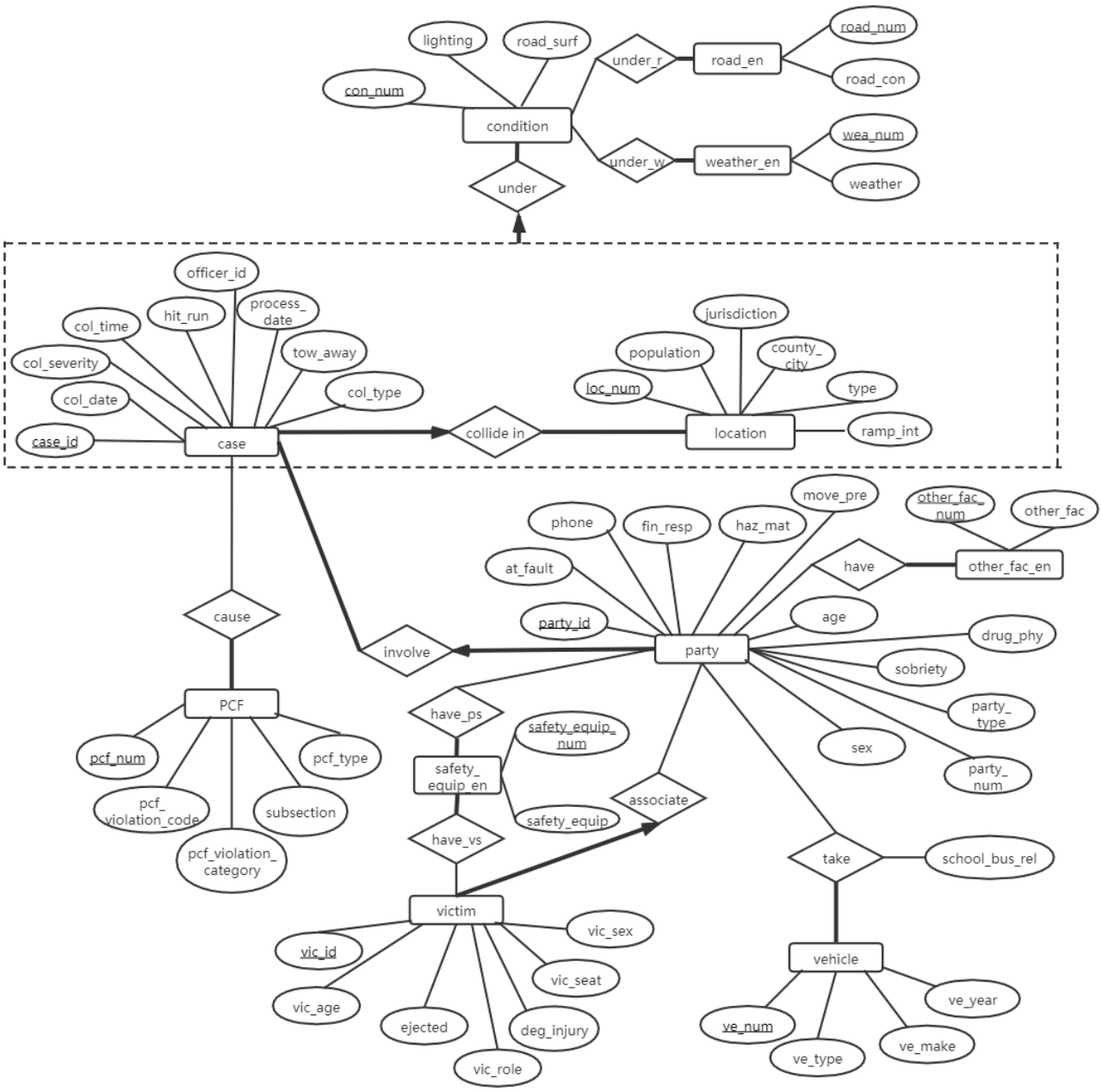
Assumptions

Each collision case has its unique case_id.
Each case has a primary collision factor (pcf); Different cases could have the same pcf.
Each case has several parties.
Each case collides in exactly one location; Different cases could collide in same location.
Each collision happened under exactly one condition, among which there could be several weather and road conditions; Different cases could collide under same condition.
Each party has a unique party_id.
Each party is involved in exactly one case; A case could involve several parties.
Each party may take a vehicle; Vehicles with same attributes are recognized as the same vehicle, and under this condition different parties can take the same vehicle.
“Party_number” refers to the specific party of a particular case, so “party_number + case_id” is unique for each party, playing the same role as party_id.
Each victim has a unique vic_id.
Each victim is associated with exactly a party in a case, by “party_number + case_id”. A party could be associated with 0 or several victims.
Each party of the case may have some other factors for the collision. Different parties may have same other factors.
Each party and victim may have their different safety_equipment. Different parties and victims may have same safety_equipment.

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Entity Relationship Schema

Schema



Description

<Describe all the choices you made for Entities and Relationships>

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Relational Schema

ER schema to Relational schema

<Describe the transition from ER schema to Relational schema>

DDL

<Provide the DDL>

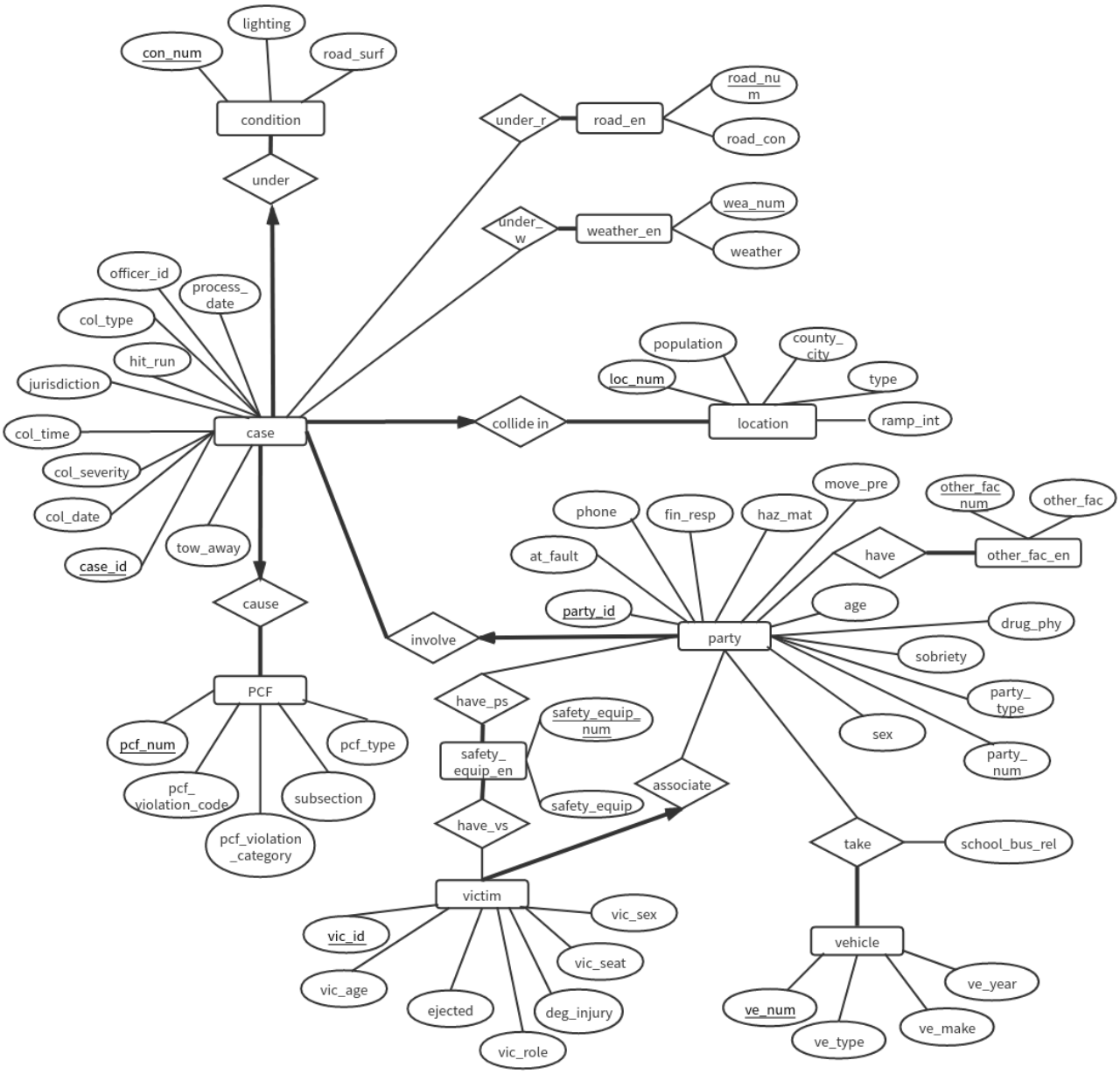
General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Deliverable 2

Entity Relationship Schema (updated)



DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Assumptions

Based on the feedback on deliverable 1, we add these assumptions and explanations, along with some modifications.

1. We found that there are duplicates of case_id in the collision2018.csv (6 of them), and since their total number is small, we recognize them as incorrectly registered data and removed them. After removing the duplicated terms in case, we have verified that in the parties2018.csv, each "party_num + case_id" corresponds to unique "party_id". So we consider party as the weak entity of case. In the victims2018.csv, each victim is associated with the specific party by unique "party_num + case_id", and we also consider victim as the weak entity of party. Therefore, we merge the "victim-associated_with-party" and "party-involved_in-case" relationships into the tables of victim and party entity respectively to illustrate the weak entity constraints. Meanwhile, we added "ON DELETE CASCADE" to the DDL (based on DDL of deliverable 1) of victim and party entity to implement weak entity relationship.

2. For entities which may appear multiple times (weather, road_condition...), two identical value may appear in both con_1 and con_2 columns, when we encounter this situation, we only record once in the "under_condition"-like tables (relationship between party/case/victim and various condition).

3. We found that PCF related attributes (pcf_violation, pcf_violation_category, pcf_violation_subsection, parimary_collision_factor) are quite similar and have close connection with each other, so we grouped them into one entity. Each case has exactly one corresponding PCF (entity) that caused it, which means each case participates exactly once in the relation. So, we changed the line between case and cause (in the ER model) into an thick arrow to illustrate the participation constraint. Meanwhile, in the DDL the cause relationship is merged into case, since case has total participation.

4. Each case happens under exactly one condition and location, so condition has 1-to-1 relationship with both case and the collide_in relationship (the relationship connecting case and location). Thus, there is no need to use the aggregation in the previous ER model, and we directly connect condition to case. Besides, we merged the "case-under-condition" and "case-collide_in-location" relationship into the case table, since case has total participation in these two relationships.

5. We connect the entity "road_en" and "weather_en" directly with entity "case" (instead of entity "condition"), because it is more simple, direct and efficient.

6. We move the attribute "jurisdiction" from the entity "location" to entity "case" because jurisdiction has little to do with the collision scene.

7. We didn't change the value of "null", just dropped them when needed. For example, we don't consider cases without weather conditions when querying weather and corresponding count of cases.

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Relational Schema

DDL (UPDATED)

```
ALTER SESSION SET nls_date_format = 'YYYY-MM-DD HH24:MI:SS'

CREATE TABLE Other_fac_en(
other_fac_num INTEGER,
other_fac VARCHAR2(3),
PRIMARY KEY (other_fac_num)
);

CREATE TABLE Safety_equip_en(
safety_equip_num INTEGER,
safety_equip VARCHAR2(3),
PRIMARY KEY (safety_equip_num)
);

CREATE TABLE Vehicle(
ve_num INTEGER,
ve_type VARCHAR2(50),
ve_make VARCHAR2(20),
ve_year INTEGER,
PRIMARY KEY (ve_num)
);

CREATE TABLE PCF(
pcf_num INTEGER,
pcf_violation_code INTEGER,
pcf_violation_category VARCHAR2(50),
subsection VARCHAR2(3),
pcf_type VARCHAR2(50),
PRIMARY KEY (pcf_num)
);

CREATE TABLE Location(
loc_num INTEGER,
population INTEGER,
county_city INTEGER,
loc_type VARCHAR2(20),
ramp_int VARCHAR2(10),
PRIMARY KEY (loc_num)
);


CREATE TABLE Condition(
con_num INTEGER,
lighting VARCHAR2(50),
```

```
road_surf VARCHAR2(10),
PRIMARY KEY (con_num)
);

CREATE TABLE Road_en(
road_num INTEGER,
road_con VARCHAR2(20),
PRIMARY KEY (road_num)
);

CREATE TABLE Weather_en(
wea_num INTEGER,
weather_con VARCHAR2(20),
PRIMARY KEY (wea_num)
);

CREATE TABLE Case(
case_id INTEGER,
loc_num INTEGER NOT NULL,
con_num INTEGER NOT NULL,
pcf_num INTEGER NOT NULL,
col_date DATE,
col_severity VARCHAR2(30),
col_time DATE,
hit_run VARCHAR2(30),
jurisdiction INTEGER,
officer_id VARCHAR2(10),
process_date DATE,
tow_away INTEGER,
col_type VARCHAR2(30),
PRIMARY KEY (case_id),
FOREIGN KEY (loc_num) REFERENCES Location(loc_num),
FOREIGN KEY(con_num) REFERENCES Condition(con_num),
FOREIGN KEY(pcf_num) REFERENCES PCF(pcf_num)
);


CREATE TABLE Party_involve(
party_id INTEGER,
case_id INTEGER NOT NULL,
at_fault INTEGER,
phone VARCHAR2(3),
fin_resp VARCHAR2(3),
haz_mat VARCHAR2(3),
move_pre VARCHAR2(3),
age INTEGER,
drug_phy VARCHAR2(3),
```

```
sobriety VARCHAR2(3),
party_type VARCHAR2(15),
party_num INTEGER,
sex VARCHAR2(6),
PRIMARY KEY (party_id),
FOREIGN KEY (case_id) REFERENCES Case(case_id)
);

CREATE TABLE Associate_victim(
vic_id INTEGER,
party_id INTEGER NOT NULL,
vic_age INTEGER,
ejected INTEGER,
vic_role INTEGER,
deg_injury VARCHAR2(50),
vic_seat INTEGER,
vic_sex VARCHAR2(6),
PRIMARY KEY (vic_id),
FOREIGN KEY (party_id) REFERENCES Party_involve(party_id)
);

CREATE TABLE Have (
other_fac_num INTEGER,
party_id INTEGER,
PRIMARY KEY (other_fac_num, party_id),
FOREIGN KEY (party_id) REFERENCES Party_involve(party_id),
FOREIGN KEY (other_fac_num) REFERENCES Other_fac_en(other_fac_num)
);

CREATE TABLE Have_ps(
party_id INTEGER,
safety_equip_num INTEGER,
PRIMARY KEY (party_id, safety_equip_num),
FOREIGN KEY (party_id) REFERENCES Party_involve(party_id),
FOREIGN KEY (safety_equip_num) REFERENCES Safety_equip_en(safety_equip_num)
);

CREATE TABLE Have_vs(
vic_id INTEGER,
safety_equip_num INTEGER,
PRIMARY KEY (vic_id, safety_equip_num),
FOREIGN KEY (vic_id) REFERENCES Associate_victim(vic_id),
FOREIGN KEY (safety_equip_num) REFERENCES Safety_equip_en(safety_equip_num)
);


CREATE TABLE Take(
```

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
ve_num INTEGER,
party_id INTEGER,
school_bus_rel VARCHAR2(5),
PRIMARY KEY (ve_num, party_id),
FOREIGN KEY (party_id) REFERENCES Party_involve(party_id),
FOREIGN KEY (ve_num) REFERENCES Vehicle(ve_num)
);

CREATE TABLE Under_r(
case_id INTEGER,
road_num INTEGER,
PRIMARY KEY (road_num, case_id),
FOREIGN KEY (road_num) REFERENCES Road_en(road_num),
FOREIGN KEY (case_id) REFERENCES Case(case_id)
);


CREATE TABLE Under_w(
case_id INTEGER,
wea_num INTEGER,
PRIMARY KEY (wea_num, case_id),
FOREIGN KEY (wea_num) REFERENCES Weather_en(wea_num),
FOREIGN KEY (case_id) REFERENCES Case(case_id)
);
```

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Data Loading/Cleaning

In “milestone2.ipynb”, we dropped the duplicates of cases and then combine “party_num” and “case_id” in the victims2018.csv to assign “party_id” to each victim. We extracted “road condition”, “weather condition”, “safety equipment”, “other associated factors” (where one case/party/victim may involve two values of them) to independent small tables and created the corresponding relationship tables. We concluded independent “location”, “PCF” and “condition” table from collisions2018.csv, added unique index to them and connected them with case by this index.We found that ‘TIME’ type will lead to error in query (using Oracle SQL), but we realized that ‘DATE’ datatype could include time. So we combined “collision_date” into “collision_time” (which now includes both time and date of the collision). Finally we got 17 tables corresponding to the 17 tables of the DLL. We used Oracle EPFL to import our data.

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	544739
2003	538952
2004	538294
2005	532724
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 take. To illustrate the whole row of the most popular, we sort the table and take the first row.

SQL statement

```
SELECT *
FROM
   (
   (SELECT ve_make,   COUNT(ve_make) AS N_VEHICLE
    FROM (vehicle INNER JOIN take ON vehicle.ve_num = take.ve_num)
    GROUP BY ve_make)
    ORDER BY N_VEHICLE DESC
   )
WHERE ROWNUM = 1
```

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

VE_MAKE	N_VEHICLE
FORD	1129700

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. We count the total number of cases and cases under “dark” lighting, then calculate the fraction.

SQL statement

```
SELECT DISTINCT
CONCAT(ROUND((SELECT COUNT(*) AS c
        FROM case, condition
        WHERE lighting LIKE '%dark%' AND case.con_num = condition.con_num)*100 /
             (SELECT COUNT (*) FROM case), 2),'%') as fraction
FROM condition
```

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

FRACTION
27.98%

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 “under_w”

SQL statement

```
SELECT count(*) AS N_collisions
FROM
   (SELECT *
```

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
    FROM under_w, weather_en
      WHERE weather_en.wea_num = under_w.wea_num AND weather_en.weather_con =
'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

(We consider grouping and finding the largest row as two tasks)

5.a

```
SELECT TO_CHAR(COL_DATE, 'D') AS WEEK_DAY, COUNT(*) AS N_COLLISONS
FROM CASE
GROUP BY TO_CHAR(COL_DATE, 'D')
ORDER BY TO_CHAR(COL_DATE, 'D') ASC
```

5.b

```
SELECT *
FROM
(SELECT TO_CHAR(COL_DATE, 'D') AS WEEK_DAY, COUNT(*) AS N_COLLISONS
FROM CASE
GROUP BY TO_CHAR(COL_DATE, 'D')
ORDER BY COUNT(*) DESC)
WHERE ROWNUM = 1
```

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

5.a

WEEK_DAY	N_COLLISIONS
1	428287
2	516798
3	535742
4	536068
5	536813
6	614852
7	509497

5.b

WEEK_DAY	N_COLLISIONS
6	614852

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Query 6:

Description of logic:

List all weather types and their corresponding number of collisions in descending order of the collisions. We first join the weather_en table with the under_w table to know which case is under which weather condition, then we group cases by weather_con and list weather and the count number.

SQL statement

```
SELECT WEATHER_CON AS WEATHER, COUNT(*) AS COUNT
FROM WEATHER_EN
INNER JOIN UNDER_W
ON WEATHER_EN.wea_num = UNDER_W.wea_num
GROUP BY WEATHER_CON
ORDER BY COUNT(*) DESC
```

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

WEATHER	COUNT
clear	2941037
cloudy	548249
raining	223752
fog	21259
wind	13952
snowing	8530
other	6960

Query 7:

Description of logic:

Count the number of parties that are at-fault, with financial responsibility and loose material. The attributes “at-fault” and “fin_resp” are in the table of party, we can filter them directly, but we need to find the road_num connected with the case from the case_id of party_involve table and the under_r table, then use road_en table to know if the road condition is “road_loose”.

We first extract the road_num of “road_loose”, then join it with the under_r table to find which cases are under such road condition. Then we filter the party_id with its case_id of party_involve table who is at fault and with financial responsibility. Finally we join the two table on the same case_id and count the number of distinct parties.

SQL statement

```
SELECT COUNT(DISTINCT PARTY_ID) AS N_PARTIES
FROM
   (SELECT CASE_ID
   FROM UNDER_R
   INNER JOIN (SELECT ROAD_NUM
            FROM ROAD_EN
            WHERE ROAD_EN.ROAD_CON = 'loose material') road_loose
```

```
    ON UNDER_R.ROAD_NUM = road_loose.ROAD_NUM) case_loose
    INNER JOIN (SELECT PARTY_ID, CASE_ID
                FROM PARTY_INVOLVE
                WHERE AT_FAULT = 1 AND FIN_RESP = 'Y') party_atfault
  ON case_loose.CASE_ID = party_atfault.CASE_ID
```

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)
WHERE rownum = 1;
```

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_ids who use belt using table have_vs and safety_equip_en. Then we count the unique vic_ids and use this number as the numerator. Finally we get the fraction and format it to percentage.

SQL statement

```
SELECT CONCAT(ROUND(a.fraction*100.0,2),'%') AS fraction
FROM(SELECT DISTINCT
      (SELECT  COUNT(vic_id) AS count
         FROM
           (SELECT h1.vic_id as vic_id
              FROM have_vs h1, safety_equip_en s1
              WHERE h1.safety_equip_num = s1.safety_equip_num
                 AND s1.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
1.06%

Query 10:

Description of logic:

Compute the fraction of collisions happening for each hour of the day, and display as ratio as percentage 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 percentage.

SQL statement

```
SELECT h_count.hour, CONCAT(ROUND((h_count.count/sum_count.sum*100.0),2),'%') AS
fraction
FROM(SELECT
        DISTINCT EXTRACT(hour from cast(col_time as timestamp)) as hour, count(*) as
count
      FROM case
         GROUP BY EXTRACT(hour from cast(col_time as timestamp))
         ORDER BY hour ASC) h_count,
   (SELECT sum(h_count.count) AS sum
      FROM
      (SELECT
```

```
        DISTINCT EXTRACT(hour from cast(col_time as timestamp)) as hour, count(*) as
count
        FROM case
        GROUP BY EXTRACT(hour from cast(col_time as timestamp))
        ORDER BY hour ASC) h_count) sum_count
```

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

HOUR	FRACTION
0	1.91%
1	1.83%
2	1.81%
3	1.15%
4	0.98%
5	1.45%
6	2.62%
7	5.17%
8	5.23%
9	4.09%
10	4.23%
11	4.89%
12	5.78%
13	5.78%
14	6.55%
15	7.75%
16	7.33%
17	7.91%
18	6.30%
19	4.43%
20	3.49%
21	3.28%
22	2.86%
23	2.38%
(null)	0.81%

General Comments

We made necessary modifications based on the feedback of Deliverable 1, and achieved the required results in Deliverable 2 accordingly.

However, Besides, we also have several questions, about which we hope to get your feedback as well.

We found that TIME type will lead to error in query (in Oracle SQL) so we use DATE type. There are also other types like timestamp/datetime and so on, and we are not sure if this type is the best choice.

Is it better to combine “condition” into the case? For it has only two attributes now. We found that we may need more join operations based on this design, is it acceptable?