- Preprocessing (MySQL Workbench):

First, we decided to aggregate all the tables we needed from MySQL Workbench (collisions, victims, parties), by using the script below:

```
CREATE TABLE main AS
SELECT
    table1.case_id AS case_id,
    table1.killed_victims AS killed_victims,
    table1.injured_victims AS injured_victims,
    table1.primary_collision_factor AS primary_collision_factor,
    table1.pcf_violation_category AS pcf_violation_category,
    table1.pedestrian_action AS pedestrian_action,
    table1.lighting AS lighting_condition,
    table1.alcohol_involved AS alcohol_involved,
    table1.collision_date AS collision_date,
    table2.party_type AS party_type,
    table2.party_race AS party_race,
    table2.party_sex AS party_sex,
    table2.party_sobriety AS party_sobriety,
    table2.party_drug_physical AS party_drug_physical,
    table2.under_influence_ethnicity AS under_influence_ethnicity,
    table2.vehicle_year AS vehicle_year,
    table2.vehicle_make AS vehicle_make,
    table3.victim_degree_of_injury AS victim_degree_of_injury,
    table3.victim_role AS victim_role,
    table3.victim_sex AS victim_sex,
    table3.victim_age AS victim_age
FROM
    (
            SELECT
                  *
            FROM
                  collisions
            WHERE
                  (primary_collision_factor IS NOT NULL) AND
(primary_collision_factor != 2) AND (primary_collision_factor !=
'unknown')
            AND (pcf_violation_category IS NOT NULL) AND
(pcf_violation_category != 21804) AND (pcf_violation_category !=
'unknown')
            AND (pedestrian_action IS NOT NULL)
            AND (lighting IS NOT NULL)
      ) AS table1

    INNER JOIN
```

```
    (
            SELECT
                  *
            FROM
                  parties
            WHERE
                  (vehicle_year > 1929 OR vehicle_year < 2020)
            AND (vehicle_make IS NOT NULL)
            AND (party_drug_physical IS NOT NULL)
            AND (party_race IS NOT NULL)
            AND (under_influence_ethnicity IS NOT NULL)
      ) AS table2 ON table1.case_id = table2.case_id

    INNER JOIN
      (
            SELECT
                  *
            FROM
                  victims
            WHERE
                  (victim_degree_of_injury IS NOT NULL) AND
(victim_degree_of_injury != '7') AND (victim_degree_of_injury != '6')
AND (victim_degree_of_injury != '5') AND (victim_degree_of_injury !=
'M')
            AND (victim_age > -1 OR victim_age < 100)
            AND (victim_sex = 'male' OR victim_sex = 'female')
            AND (victim_role != 'l') AND (victim_role != 'm') AND
(victim_role IS NOT NULL)
      ) AS table3 ON table1.case_id = table3.case_id
```

We took this chance to also clean our dataset from some incorrect values and mistakes that have been made by police officers when populating the SWITRS .
Subsequently, we have used the “Table Data Export Wizard” function available in MySQL Workbench to export the `.json` file.

- Preprocessing (Python):

In this phase, we opened the `.json` file in a Jupyter Notebook to create a better and more purposeful file that could fulfil the design of a truly non-relational database schema.

```
import json
with open('./json/main.json') as m:
    database = json.load(m)
```

We checked again for any unwanted or misleading data, using the script below:

```
 database = [
    x for x in database if
        (x['party_sobriety'] != '')
        and ( x['under_influence_ethnicity'] != '' )
        and ( x['party_drug_physical'] != '' )
        and ( x['vehicle_make'] != '' )
        and ( x['party_race'] != '' )
        and ( x['vehicle_year'] > 1929 or x['vehicle_year'] < 2020 )
        and ( x['victim_role'] != '' )
        and ( x['victim_role'] != 'm' )
        and ( x['victim_role'] != 'l' )
        and ( (x['victim_sex'] == 'male') or (x['victim_sex'] ==
'female') )
        and ( x['victim_age'] > -1 or x['victim_age'] < 100 )
        and ( x['victim_degree_of_injury'] != '6' )
        and ( x['victim_degree_of_injury'] != '7' )
        and ( x['victim_degree_of_injury'] != '5' )
        and ( x['victim_degree_of_injury'] != 'M' )
        and ( x['victim_degree_of_injury'] != '')
        and ( x['primary_collision_factor'] != '' )
        and ( x['primary_collision_factor'] != 2 )
        and ( x['primary_collision_factor'] != 'unknown' )
        and ( x['pcf_violation_category'] != '' )
        and ( x['pcf_violation_category'] != 21804 )
        and ( x['pcf_violation_category'] != 'unknown' )
        and ( x['pedestrian_action'] != '')
        and ( x['lighting_condition'] != '')
]
```

We set the `case_id` field as our main identifier in order to combine/group the information more effectively by using the script below:

```
for item in database:
    item['case_id'] = int(item['case_id'])
    if item['under_influence_ethnicity'] == '0':
        item['under_influence_ethnicity'] = False
    else:
        item['under_influence_ethnicity'] = True
    if item['alcohol_involved'] == 0:
        item['alcohol_involved'] = False
    else:
        item['alcohol_involved'] = True
```

```
guids = set([x['case_id'] for x in database])
result = [{'case_id': x} for x in guids]
```

For this reason, we inserted all the fields with more than one value per `case_id` in arrays (see below):

```
counter = len(result)

for i in result:
    case_id = i['case_id']

    counter -= 1
    if counter % 1000 == 0:
        print(counter)

    for j in database:
        if j['case_id'] == case_id:
            if 'killed_victims' not in i:
                i['killed_victims'] = j['killed_victims']
                i['injured_victims'] = j['injured_victims']
                i['primary_collision_factor'] =
j['primary_collision_factor']
                i['pcf_violation_category'] =
j['pcf_violation_category']
                i['pedestrian_action'] = j['pedestrian_action']
                i['lighting_condition'] = j['lighting_condition']
                i['alcohol_involved'] = j['alcohol_involved']
                i['collision_date'] = j['collision_date']
                i['party_type'] = [j['party_type']]
                i['party_race'] = [j['party_race']]
                i['party_sex'] = [j['party_sex']]
                i['party_sobriety'] = [j['party_sobriety']]
                i['party_drug_physical'] = [j['party_drug_physical']]
                i['vehicle_year'] = [j['vehicle_year']]
                i['vehicle_make'] = [j['vehicle_make']]
                i['victim_degree_of_injury'] =
[j['victim_degree_of_injury']]
                i['victim_role'] = [j['victim_role']]
                i['victim_sex'] = [j['victim_sex']]
                i['victim_age'] = [j['victim_age']]
            else:
                i['party_type'].append(j['party_type'])
                i['party_race'].append(j['party_race'])
                i['party_sex'].append(j['party_sex'])
```

```
                i['party_sobriety'].append(j['party_sobriety'])

i['party_drug_physical'].append(j['party_drug_physical'])
                i['vehicle_year'].append(j['vehicle_year'])
                i['vehicle_make'].append(j['vehicle_make'])

i['victim_degree_of_injury'].append(j['victim_degree_of_injury'])
                i['victim_role'].append(j['victim_role'])
                i['victim_sex'].append(j['victim_sex'])
                i['victim_age'].append(j['victim_age'])
```

At this point, we were ready to dump the `.json` file using the following code...

```
with open('./json/opt_main.json', 'w') as m:
    json.dump(result, m)
```

...and import it into our MongoDB environment, by using this Windows PowerShell command (after having set the MongoDB environment path as global):

```
PS C:\Users\giogi> mongoimport --db traffic --collection california --
drop --jsonArray --batchSize 1 --file opt_main.json
```

Our final preprocessing step was to convert all the date fields to the ISODate format, so as to be able to easily use conditional statements over dates in our queries. Here is the function we used to perform, and save permanently, such modifications.

```
db.california.find().forEach( function(doc) {
   doc['collision_date'] = new Date(doc['collision_date']);
   db.california.save(doc);
})
```