STEP 1A

A Driving Experience can ONLY be driven under ONE AND ONLY ONE Weather condition.

But different Driving Experiences can have the same Weather condition.

A Driving Experience MUST have a Weather condition related to it.

A Weather condition can have 0 Driving Experiences related with it.

A Driving Experience can ONLY be driven on ONE AND ONLY ONE Road type.

But different Driving Experiences can have the same Road type.

A Driving Experience MUST have a Road type related to it.

A Road type can have 0 Driving Experiences related with it.

A Driving Experience can ONLY be driven in ONE AND ONLY ONE Traffic intensity.

But different Driving Experiences can have the same Traffic intensity.

A Driving Experience MUST have a Traffic intensity related to it.

A Traffic intensity can have o a Driving Experiences related with it.

A Driving Experience can Have many Maneuvers made in it.

A Maneuver can be made in different Driving Experiences.

A Driving experience can have 0 Maneuvers made in it.

But a Maneuver can not exist without a Driving experience.

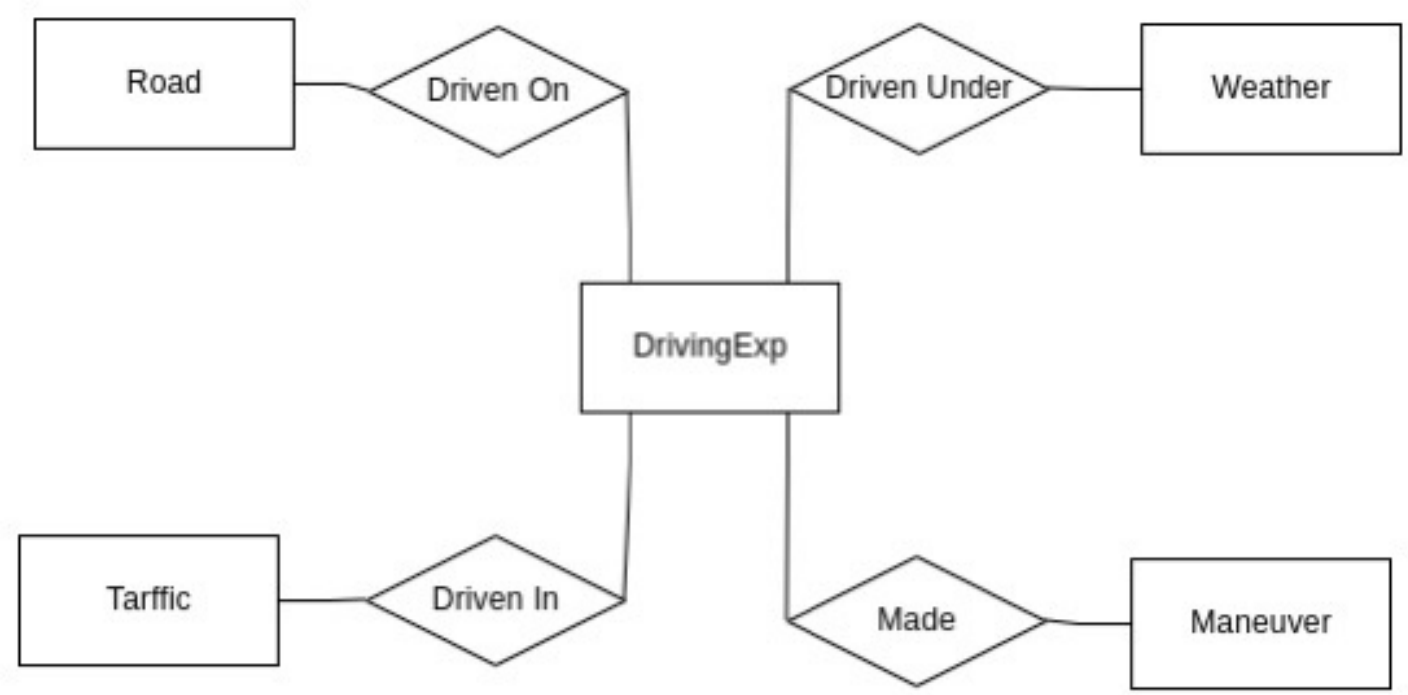
STEP 1B

[DrivingExp](Driven Under)[Weather]

[DrivingExp](Driven In)[Traffic]

[DrivingExp](Made)[Maneuver]

[DrivingExp](Driven on)[Road]



STEP 2A

[DrivingExp](idDrivingExp,startTime,endTime,distance,date,idWeather,,idTraffic,idRoad)
<INT,TIME,TIME,FLOAT,DATE,TINYINT,TINYINT,TINYINT>

[Weather](idWeather,condition)<TINYINT,VARCHAR(20)>

[Traffic](idTraffic,intensity)<TINYINY,VARCHAR(20)>

[Road](idRoad,type)<TINYINT,VARCHAR(20)>

[Maneuver](idManeuver,type)<TINYINT,VARCHAR(20)>

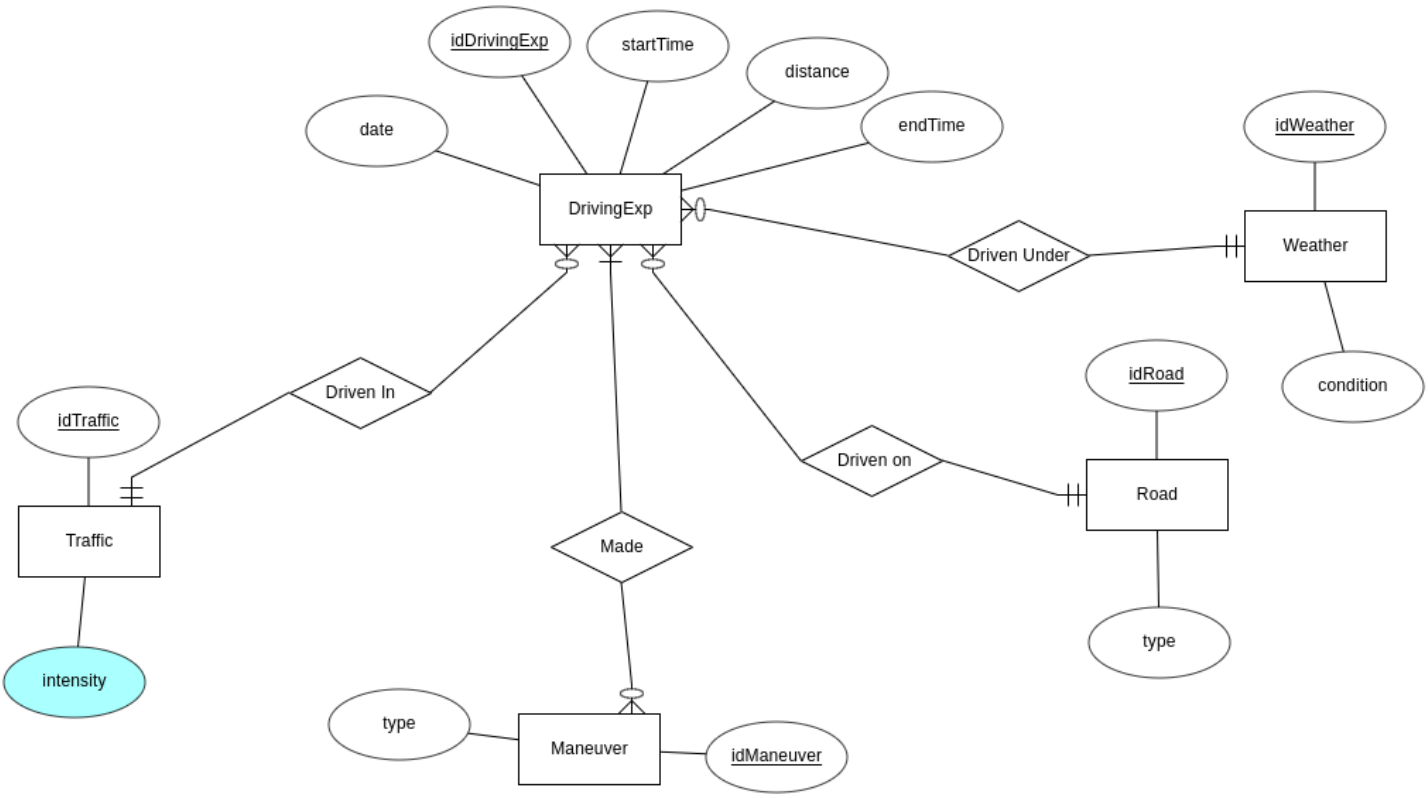
STEP 2B

[DrivingExp](>011)[Traffic](1-to-many)

[DrivingExp](>011)[Weather](1-to-many)

[DrivingExp](>011)[Road](1-to-many)

[DrivingExp](>10<)[Maneuver](many-to-many)



STEP 3A

[DrivingExp,PK:idDrivingExp](>101)[DrivingExp_Maneuver,FK:idDrivingExp,FK:idManeuver]

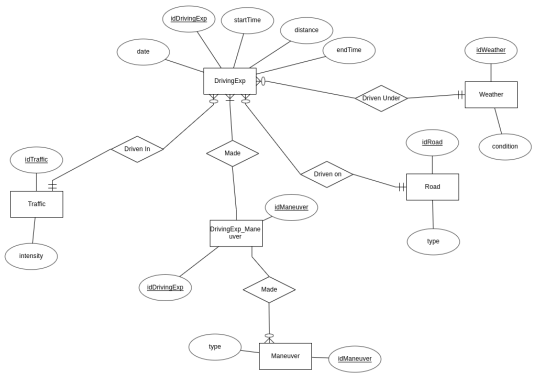
[DrivingExp_Maneuver,FK:idDrivingExp,FK:idManeuver](100<)(Maneuver,PK:idManeuver)

[DrivingExp_Maneuver,FK:idDrivingExp,FK:idWeather](>011)[Weather,PK:idWeather]

[DrivingExp_Maneuver,FK:idDrivingExp,FK:idRoad](>011)[Weather,PK:idRoad]

[DrivingExp_Maneuver,FK:idDrivingExp,FK:idTraffic](>011)[Weather,PK:idTraffic]

PDM



Relational Schema

DrivingExp (PK:idDrivingExp,startTime,endTime,distance,date,FK:idWeather,FK:idTraffic.FK:idRoad)

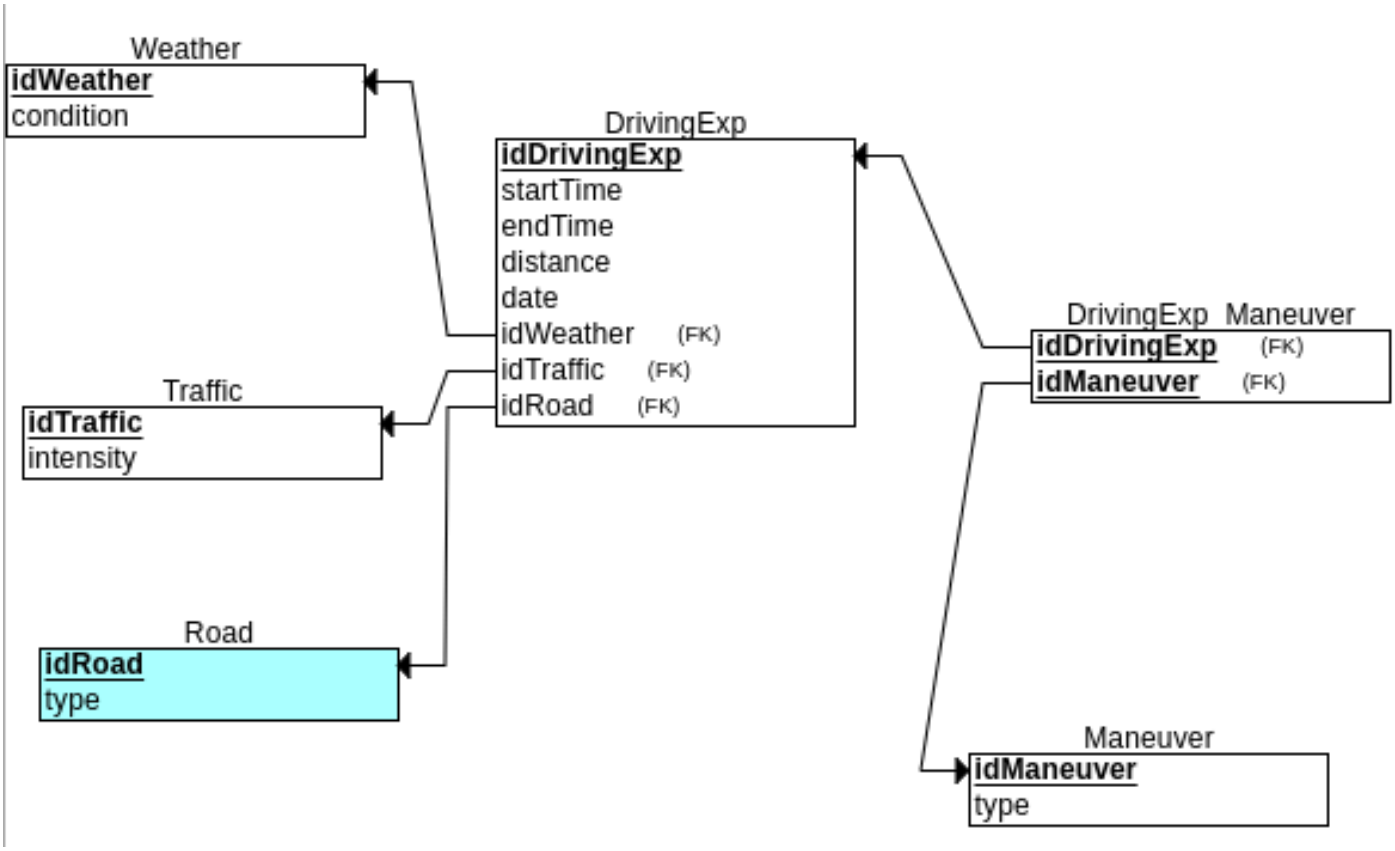
DrivingExp_Maneuver (FK:idDrivingExp,FK:idManeuver)

Weather (PK:idWeather,condition)

Traffic (PK:idTraffic,intensity)

Road (PK: idRoad,type)

Maneuver (PK:idManeuver,type)



STEP 4A

•Here we have a query to find out the total distance driven in all DrivingExperiences.

```
SELECT format(SUM(distance),2) as totalDistance FROM DrivingExp;
```

totalDistance
552.60

•Here we have the total number of Experiences driven in each weather condition.

```
1 SELECT weatherCondition, COUNT(DrivingExp.idDrivingExp) AS num_experiences
2 FROM Weather
3 LEFT JOIN DrivingExp  ON Weather.idWeather = DrivingExp.idWeather
4 GROUP BY Weather.weatherCondition;
```

weatherCondition	num_experiences
sunny	3
rainy	2
foggy	2
cloudy	2
snowy	1

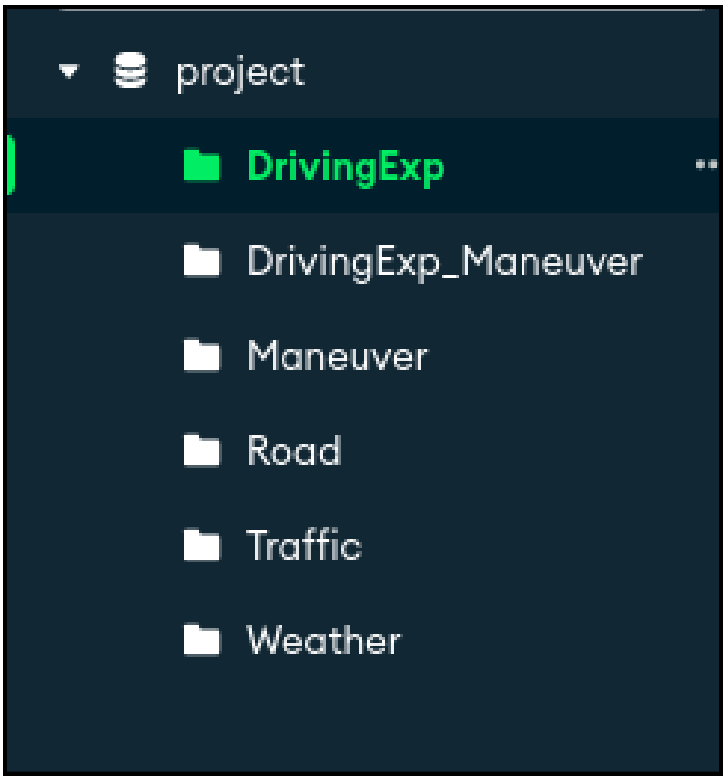
•Here we have a Traffic condition that has not been driven in yet.

```
SELECT * FROM Traffic
WHERE Traffic.idTraffic NOT IN (SELECT idTraffic FROM DrivingExp);
```

idTraffic	intensity
2	moderate

Part 2

Normalized Structure to denormalized structure.



This is the MySQL database converted to MongoDB

We will use some MongoDB aggregation pipeline tools to get the desired result of denormalizing the data.

Some overview of the tools we will use:

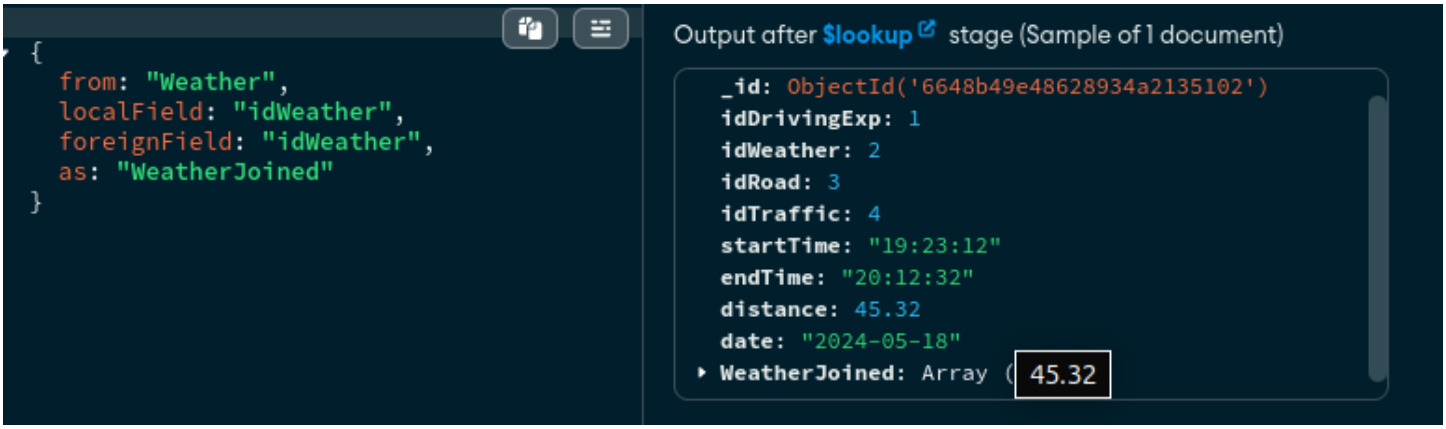
- **$lookup :** Merges data from related tables into the main document, adding details like weather, traffic, and road conditions.
- **$unwind :** Expands nested arrays within the document so each element becomes a separate document, simplifying access to road details.
- **$set :** Transforms document structure by assigning new values to fields, converting road details into a more readable format, like a string representing road type.
- **$project :** Shapes the output document by including or excluding specific fields, removing unnecessary identifiers for a clean denormalized output ready for analysis or presentation.

Let’s do it step-by-step

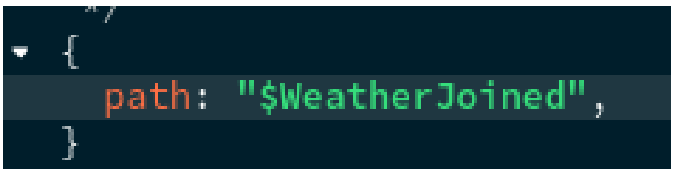
This is an example object before the denormalization:

```
_id: ObjectId('6648b49e48628934a2135102')
idDrivingExp: 1
idWeather: 2
idRoad: 3
idTraffic: 4
startTime: "19:23:12"
endTime: "20:12:32"
distance: 45.32
date: "2024-05-18"
```

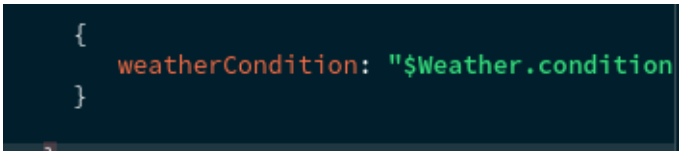
The first step (joining):



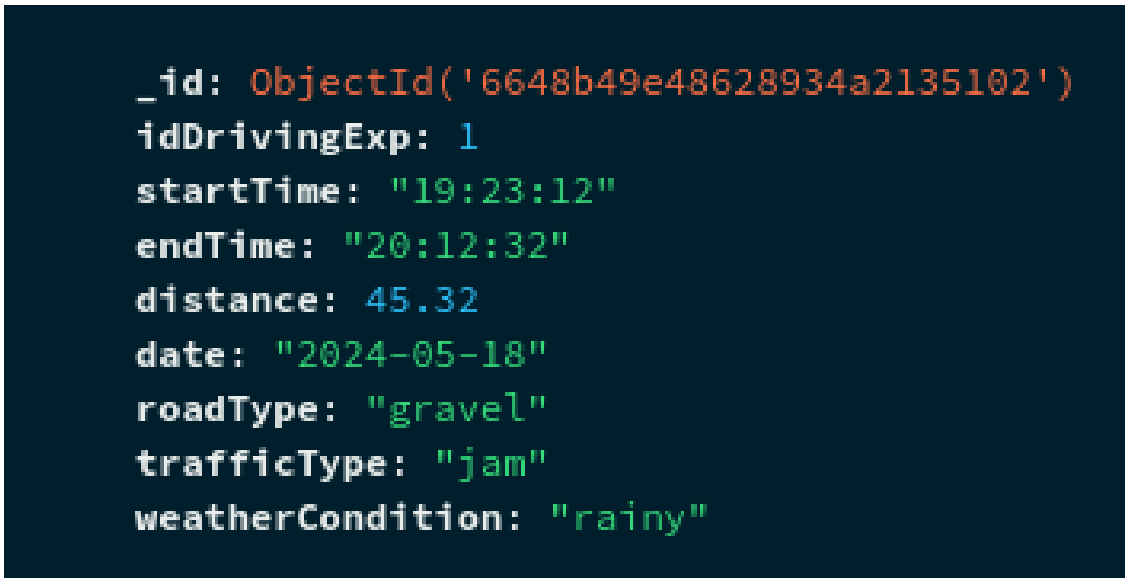
The second step (destructuring the array):



The third stage (matching the id to the value):



After doing this for all fields we get the following object:



As we can see the object now has a denormalized structure , having the corresponding values instead of the foreign keys.