Database homework project

Shamsudinov Nail

Step 1: Guidelines, rules, and CDM

Entities

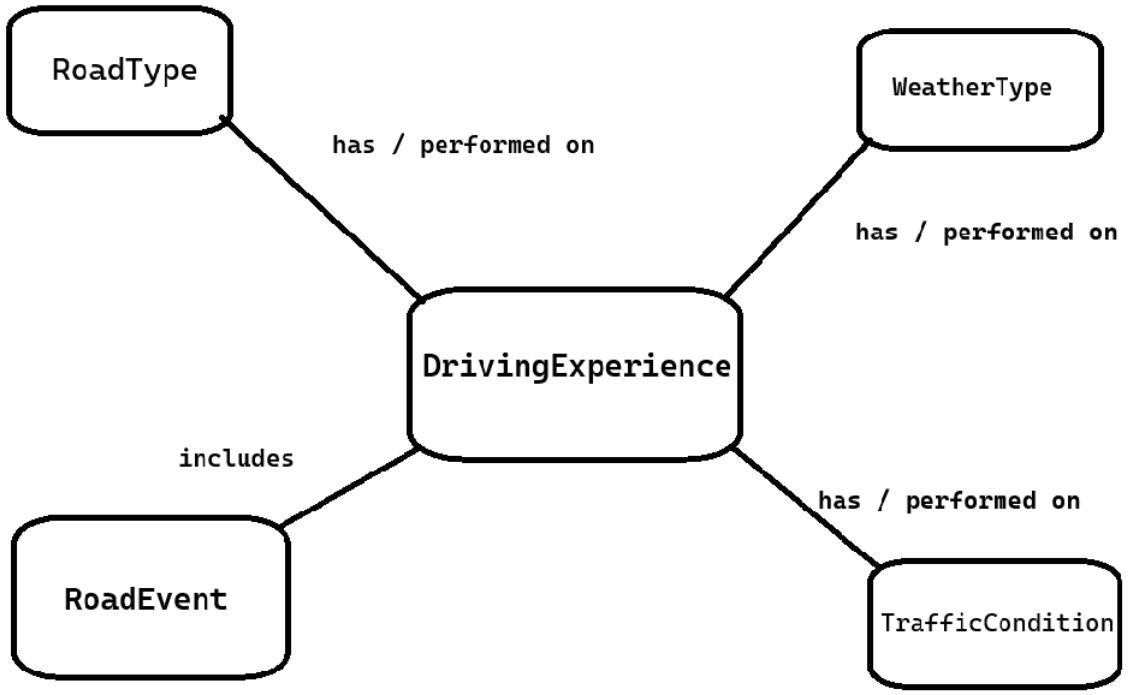
- DrivingExperience
- RoadType
- WeatherType
- TrafficCondition
- RoadEvent

System Rules

Types extracted from my front-end project; Road events are mentioned in the paper-based exam

1. A driving experience has one and only one road type
2. Road Type has 0 or many driving experiences
3. A dr. exp. has 1 and only 1 weather type
4. Weather type has 0 or many dr. exp.
5. A dr. exp. has 1 and only 1 traffic condition
6. Traffic condition has 0 or many dr. exp.
7. A dr. exp. has 0 or many road events
8. Road event has 0 or many dr. exp.

CDM



Text format

[DrivingExperience](has)[RoadType]
[DrivingExperience](has)[WeatherType]
[DrivingExperience](has)[TrafficCondition]
[DrivingExperience](includes)[RoadEvent]

Step 2: Data Dictionary and LDM

Data dictionary

DrivingExperience

Attribute	Description	Type	Size
idExperience	Unique id for experience	INT	4 byte
startDateTime	Date and time of start of exp.	DATETIME	8 byte
endDateTime	Date and time of end of exp.	DATETIME	8 byte
distance	Distance of driving in km.	FLOAT	8 byte
rating	Rating (1-5)	TINYINT	1 byte
notes	Notes of the experience	TEXT	4096 char

RoadType

Attribute	Description	Type	Size
idRoadType	Unique id for road type	INT	4 byte
nameRoadType	Name of the road type	NVARCHAR	32 char

WeatherType

Attribute	Description	Type	Size
idWeatherType	Unique id for weather type	INT	4 byte
nameWeatherType	Name of the weather type	NVARCHAR	32 char

TrafficCondition

Attribute	Description	Type	Size
idTrafficCondition	Unique id for traffic condition	INT	4 byte
nameTrafficCondition	Name of the traffic condition	NVARCHAR	32 char

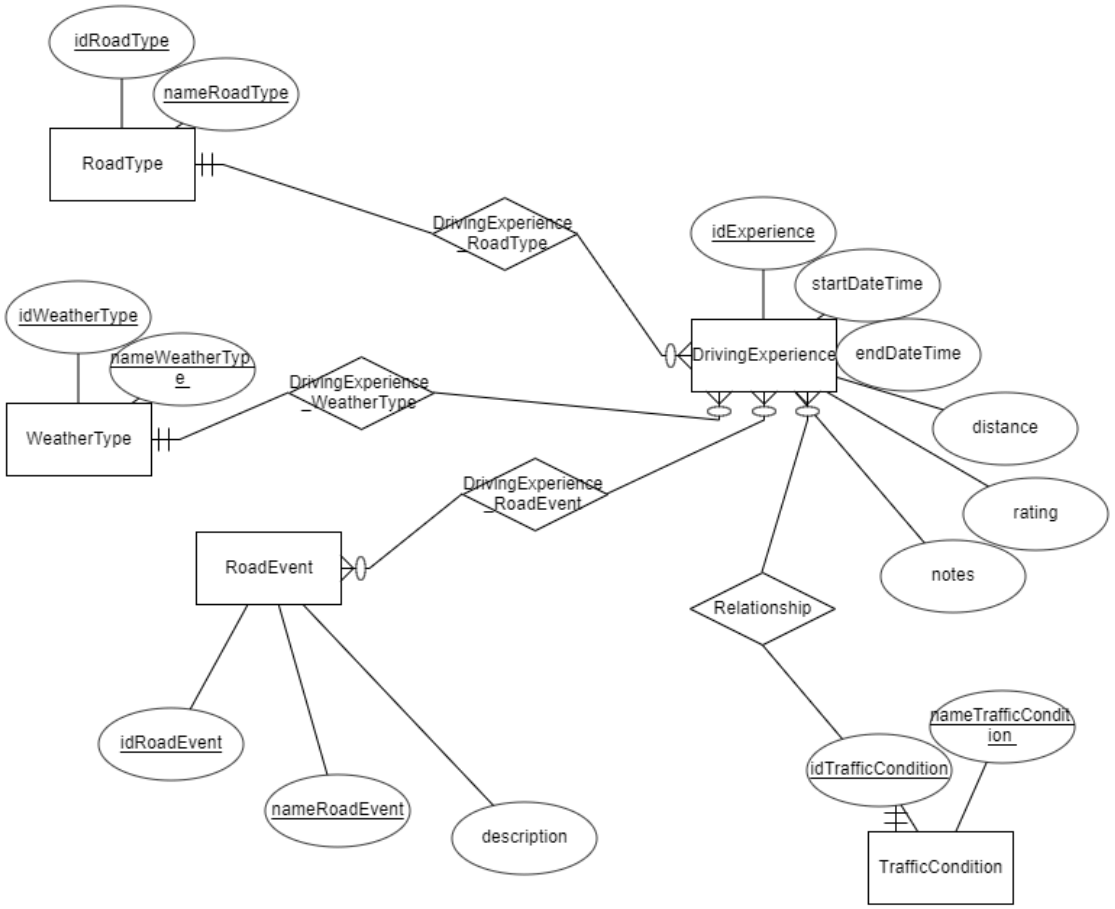
RoadEvent

Attribute	Description	Type	Size
idRoadEvent	Unique id for road event	INT	4 byte
nameRoadEvent	Name of the road event	NVARCHAR	64 char
description	Description of the event	TEXT	4096 char

Text format

[DrivingExperience](idExperience, startDateTime, endDateTime, distance, rating, notes)<INT, DATETIME, DATETIME, FLOAT, TINYINT, TEXT>
[RoadType](idRoadType, nameRoadType)<INT, NVARCHAR(32)>
[WeatherType](idWeatherType, nameWeatherType)<INT, NVARCHAR(32)>
[TrafficCondition](idTrafficCondition, nameTrafficCondition)<INT, NVARCHAR(32)>
[RoadEvent](idRoadEvent, nameRoadEvent, description)<INT, NVARCHAR(64),TEXT>

LDM

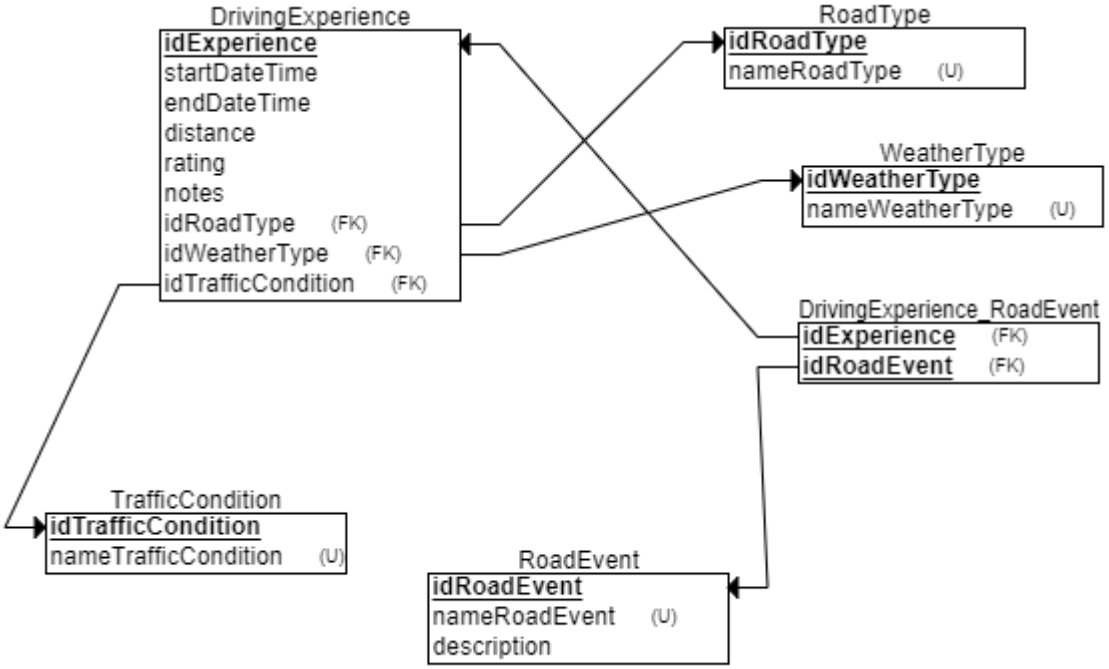


Text format

[DrivingExperience](>011)[RoadType](1-to-many)
[DrivingExperience](>011)[WeatherType](1-to-many)
[DrivingExperience](>011)[TrafficCondition](1-to-many)
[DrivingExperience](>00<)[RoadEvent](many-to-many)

Step 3: PDM and Schema

PDM



Text format

[DrivingExperiences,PK:idExperience,FK:idRoadType](>011)[RoadTypes,PK:idRoadType]
[DrivingExperiences,PK:idExperience,FK:idWeatherType](>011)[WeatherTypes,PK:idWeatherType]
[DrivingExperiences,PK:idExperience,FK:idTrafficCondition](>011)[TrafficConditions,PK:idTrafficCondition]
[DrivingExperiences,PK:idExperience](110<)[DrivingExperience_RoadEvent,FK:idExperience,FK:idRoadEvent]
[DrivingExperience_RoadEvent,FK:idExperience,FK:idRoadEvent](>011)[RoadEvent,PK:idRoadEvent]

Schema

Text format

DrivingExperience (PK: idExperience, startDateTime, endDateTime, distance, rating, notes, FK: idRoadType, FK: idWeatherType, FK: idTrafficCondition)
RoadType (PK: idRoadType, nameRoadType)
WeatherType (PK: idWeatherType, nameWeatherType)
TrafficCondition (PK: idTrafficCondition, nameTrafficCondition)
DrivingExperience_RoadEvent (FK: idExperience, FK: idRoadEvent)
RoadEvent (PK: idRoadEvent, nameRoadEvent, description)

Step 4: SQL

MySQL Queries to create tables

```sql
CREATE TABLE RoadType (
    idRoadType INT PRIMARY KEY AUTO_INCREMENT,
    nameRoadType VARCHAR(32) NOT NULL
);

CREATE TABLE WeatherType (
    idWeatherType INT PRIMARY KEY AUTO_INCREMENT,
    nameWeatherType VARCHAR(32) NOT NULL
);

CREATE TABLE TrafficCondition (
    idTrafficCondition INT PRIMARY KEY AUTO_INCREMENT,
    nameTrafficCondition VARCHAR(32) NOT NULL
);

CREATE TABLE RoadEvent (
    idRoadEvent INT PRIMARY KEY AUTO_INCREMENT,
    nameRoadEvent VARCHAR(64) NOT NULL,
    description TEXT NULL
);

CREATE TABLE DrivingExperience (
    idExperience INT PRIMARY KEY AUTO_INCREMENT,
    startDateTime DATETIME NOT NULL,
    endDateTime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

```
    distance FLOAT NOT NULL,
    rating INT NULL,
    notes TEXT NULL,
    idRoadType INT NOT NULL,
    idWeatherType INT NOT NULL,
    idTrafficCondition INT NOT NULL,
    FOREIGN KEY (idRoadType) REFERENCES RoadType(idRoadType),
    FOREIGN KEY (idWeatherType) REFERENCES
WeatherType(idWeatherType),
    FOREIGN KEY (idTrafficCondition) REFERENCES
TrafficCondition(idTrafficCondition)
);

CREATE TABLE DrivingExperience_RoadEvent (
    idExperience INT NOT NULL,
    idRoadEvent INT NOT NULL,
    FOREIGN KEY (idExperience) REFERENCES
DrivingExperience(idExperience),
    FOREIGN KEY (idRoadEvent) REFERENCES RoadEvent(idRoadEvent)
);
```

Queries:

1. Total distance on each type of road

```
SELECT
    RT.nameRoadType AS RoadType,
    ROUND(SUM(DE.distance), 2) AS TotalDistance
FROM
    DrivingExperience DE
JOIN
    RoadType RT ON DE.idRoadType = RT.idRoadType
GROUP BY
    RT.idRoadType
ORDER BY
     TotalDistance DESC
```

Code explanation:
We fetch all driving experiences, JOIN with the RoadType table (based on foreign key which is the road type's id) and GROUP BY ID of the type of the road. Then we SUM all the experiences' distances corresponding to the aforementioned type and ORDER BY the SUM (DESCending, meaning from the biggest value to the smallest)
Additionally, I ROUNDed the distances up to 2 decimal points because there were some precision errors

RoadType	TotalDistance ▾ 1
City Road	212.10
Rural Road	192.80
Highway	161.50

2. Total № of exp, depending on the weather

```
SELECT
    WT.nameWeatherType AS WeatherType,
    COUNT(DE.idExperience) AS NumberOfExperiences
FROM
    DrivingExperience DE
JOIN
    WeatherType WT ON DE.idWeatherType = WT.idWeatherType
GROUP BY
    WT.idWeatherType
ORDER BY
      NumberOfExperiences DESC
```

Very similar code, now it just COUNTs the number of experiences

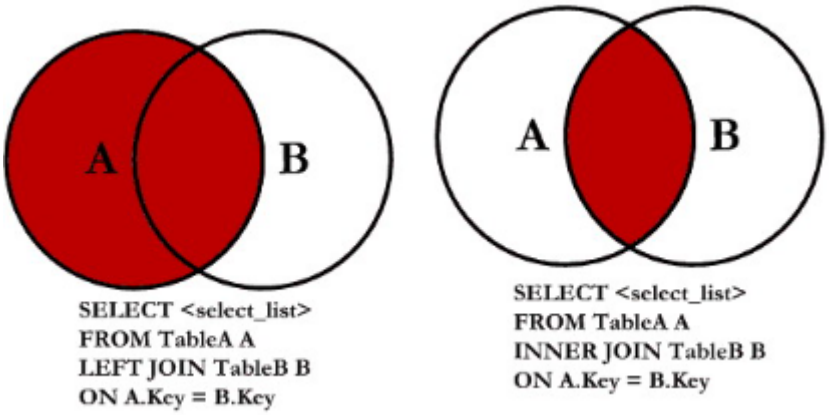
WeatherType	NumberOfExperiences ▾ 1
Sunny	7
Rainy	3
Snowy	2

3. Full data of the experiences

```
SELECT
    DE.idExperience AS ExperienceID,
    DE.startDateTime AS StartDateTime,
    DE.endDateTime AS EndDateTime,
    DE.distance AS Distance,
    DE.rating AS Rating,
    DE.notes AS Notes,
    RT.nameRoadType AS RoadType,
    WT.nameWeatherType AS WeatherType,
    TC.nameTrafficCondition AS TrafficCondition,
    GROUP_CONCAT(RE.nameRoadEvent ORDER BY RE.nameRoadEvent
SEPARATOR ', ') AS RoadEvents
FROM
    DrivingExperience DE
JOIN
    RoadType RT ON DE.idRoadType = RT.idRoadType
JOIN
    WeatherType WT ON DE.idWeatherType = WT.idWeatherType
JOIN
    TrafficCondition TC ON DE.idTrafficCondition =
TC.idTrafficCondition
LEFT JOIN
    DrivingExperience_RoadEvent DER ON DE.idExperience =
DER.idExperience
LEFT JOIN
    RoadEvent RE ON DER.idRoadEvent = RE.idRoadEvent
GROUP BY
    DE.idExperience;
```

Let's look at the query line-by-line; In the first few, we output the information normally, however, when we get to the Road Events we use the GROUP_CONCAT function with SEPARATOR. The reason for that is Road Events being in Many-to-Many relationship with our Driving Experiences table, so there may be several events corresponding to one event. So, if we concat road events with ", " separator, we get the list of them in text format, e.g. "Accident, Road Work".

Then we JOIN our tables; You may notice that when we get to our Road Events we use LEFT JOIN, that is to include Experiences without any events. Let me paraphrase - ordinary JOIN actually corresponds to INNER JOIN; You may see the differences on the following Euler-Vienna diagrams



In the end we GROUP BY id of the experience. We get the following results:

ExperienceID	StartDateTime	EndDateTime	Distance	Rating	Notes	RoadType	WeatherType	TrafficCondition	RoadEvents
1	2024-05-20 08:00:00	2024-05-20 09:00:00	50.5	4	Smooth drive with light traffic	Highway	Sunny	Light Traffic	Accident
2	2024-05-21 10:00:00	2024-05-21 11:30:00	30	3	Rainy weather made driving difficult	City Road	Rainy	Heavy Traffic	Road Work
3	2024-05-22 12:00:00	2024-05-22 13:15:00	45	5	Beautiful day for a drive	Rural Road	Sunny	Moderate Traffic	Animal Crossing
4	2024-05-23 07:30:00	2024-05-23 08:15:00	40.2	4	Light traffic and clear weather	Highway	Sunny	Light Traffic	Accident
5	2024-05-24 09:00:00	2024-05-24 10:45:00	60.7	2	Heavy traffic due to road work	City Road	Sunny	Heavy Traffic	Road Work
6	2024-05-25 11:30:00	2024-05-25 12:30:00	35.4	5	Nice drive with moderate traffic	Rural Road	Snowy	Moderate Traffic	Animal Crossing
7	2024-05-26 06:45:00	2024-05-26 07:30:00	20.8	3	Snowy conditions made driving tricky	Highway	Snowy	Heavy Traffic	Accident
8	2024-05-27 14:00:00	2024-05-27 15:20:00	55.9	4	Encountered animal crossing	City Road	Rainy	Moderate Traffic	Animal Crossing
9	2024-05-28 16:15:00	2024-05-28 17:00:00	42.1	5	Smooth drive with no incidents	Rural Road	Sunny	Light Traffic	Accident, Road Work
10	2024-05-29 18:00:00	2024-05-29 19:00:00	50	4	Evening drive with moderate traffic	Highway	Sunny	Moderate Traffic	Accident
11	2024-05-30 08:30:00	2024-05-30 09:45:00	65.5	3	Morning rush hour traffic	City Road	Rainy	Heavy Traffic	Road Work
12	2024-06-01 10:00:00	2024-06-01 11:30:00	70.3	5	Perfect weather and light traffic	Rural Road	Sunny	Light Traffic	Animal Crossing

PART 2: MongoDB Denormalized Database

Our goal here is to transform our normalized MySQL database into a denormalized MongoDB database via exporting / importing the database via JSON and using MongoDB's aggregation pipelines. Stage explanation:

1. $lookup (experience_events)
   The $lookup stage joins the experience_events collection with drivingExperiences, loading road events where the idExperience matches, and stores these events in an array named events within each drivingExperience document.
2. $unwind (events)
   This stage unwinds an array, resulting in separate documents for each element of the array.
3. $lookup (roadEvents)
   Likewise the first stage, we lookup the roadEvents where idRoadEvent is the same.
4. $unwind (eventDetails)
   Last stage is saved in an array, so we unwind that.
5. $group
   We group everything, all the fields are the same except for events, which are pushed into the array, resulting in the documents with an array of events.
6. $lookup (roadTypes)
   Likewise other lookups, load roadType via the idRoadType.
7. $lookup (trafficConditions)
   Same thing…
8. $lookup (weatherTypes)
   Same…
9. $unwind (roadTypeDetails)
10. $unwind (trafficConditionDetails)
11. $unwind (weatherTypeDetails)
    Results from stages 6-8 are saved as arrays, so we need to

open / “unwind” them into objects.
NOTE: since those were one-to-many, it does not create more than 1 document for each.

12. $addFields
    Using those objects, add string roadType, weatherType and trafficCondition as string / text values.
13. $project
    Eliminate fields that are not required.
14. $sort
    Sort by the ID.

Code:

```
[
  {
    $lookup: {
      from: "experience_events",
      localField: "idExperience",
      foreignField: "idExperience",
      as: "events"
    }
  },
  {
    $unwind: {
      path: "$events",
      preserveNullAndEmptyArrays: true
    }
  },
  {
    $lookup: {
      from: "roadEvents",
      localField: "events.idRoadEvent",
      foreignField: "idRoadEvent",
      as: "eventDetails"
    }
  },
  {
    $unwind: {
      path: "$eventDetails",
      preserveNullAndEmptyArrays: true
    }
  },
  {
    $group: {
      _id: "$_id",
      idExperience: {
        $first: "$idExperience"
      },
      startDateTime: {
        $first: "$startDateTime"
      },
      endDateTime: {
        $first: "$endDateTime"
      },
```

```
    distance: {
      $first: "$distance"
    },
    rating: {
      $first: "$rating"
    },
    notes: {
      $first: "$notes"
    },
    idRoadType: {
      $first: "$idRoadType"
    },
    idWeatherType: {
      $first: "$idWeatherType"
    },
    idTrafficCondition: {
      $first: "$idTrafficCondition"
    },
    events: {
      $push: {
        idRoadEvent: "$events.idRoadEvent",
        nameRoadEvent:
          "$eventDetails.nameRoadEvent",
        description: "$eventDetails.description"
      }
    }
  }
},
{
  $lookup: {
    from: "roadTypes",
    localField: "idRoadType",
    foreignField: "idRoadType",
    as: "roadTypeDetails"
  }
},
{
  $lookup: {
    from: "trafficConditions",
    localField: "idTrafficCondition",
    foreignField: "idTrafficCondition",
    as: "trafficConditionDetails"
  }
},
{
  $lookup: {
    from: "weatherTypes",
    localField: "idWeatherType",
    foreignField: "idWeatherType",
    as: "weatherTypeDetails"
  }
},
{
  $unwind: {
    path: "$roadTypeDetails",
    preserveNullAndEmptyArrays: true
  }
},
{
  $unwind: {
    path: "$trafficConditionDetails",
    preserveNullAndEmptyArrays: true
  }
```

```
  },
  {
    $unwind: {
      path: "$weatherTypeDetails",
      preserveNullAndEmptyArrays: true
    }
  },
  {
    $addFields:
      {
        roadType: "$roadTypeDetails.nameRoadType",
        weatherType:
          "$weatherTypeDetails.nameWeatherType",
        trafficCondition:
          "$trafficConditionDetails.nameTrafficCondition"
      }
  },
  {
    $project: {
      idRoadType: 0,
      idWeatherType: 0,
      idTrafficCondition: 0,
      roadTypeDetails: 0,
      trafficConditionDetails: 0,
      weatherTypeDetails: 0
    }
  },
  {
    $sort:
      {
        idExperience: 1
      }
  }
]
```

Example of an output:

```
_id: ObjectId('664f9d37963a44d12beb6cc8')
idExperience : "9"
startDateTime : "2024-05-28 16:15:00"
endDateTime : "2024-05-28 17:00:00"
distance : "42.1"
rating : "5"
notes : "Smooth drive with no incidents"
▾ events : Array (2)
  ▾ 0: Object
      idRoadEvent : "1"
      nameRoadEvent : "Accident"
      description : "A car accident occurred on the road."
  ▾ 1: Object
      idRoadEvent : "2"
      nameRoadEvent : "Road Work"
      description : "Ongoing road work causing lane closures."
roadType : "Rural Road"
weatherType : "Sunny"
trafficCondition : "Light Traffic"
```