DBMS Project Mid-Project Evaluation

By Group 7:

1. Aditya Choudhary [2020489]
2. Abhijeet Singh [2020486]
3. Amit Malik [2020493]
4. Manav Saini [2020518]

Github Repository: https://github.com/abhijeet486/Sql-Tables-and-scripts

Scope of Project :

This project provides users with a hassle-free, reliable car rental service. It uses the concept of DBMS for handling given and generated info.

- **Stakeholders:** Daily Commuters, Taxi drivers.

- **Entities**:
    1. The entity **Driver** handles all information related to the driver with primary key Driver_ID.
    2. The entity **Passenger** handles all information related to the passenger with primary key Passenger_ID.
    3. The entity **Trip** handles all trips requested by passengers and accepted by the driver with the primary key Trip_ID.
    4. The entity **Payment** handles the payments after the trip has been completed, so it has a dependent relationship with the entity **Trip.**
    5. The entity **Vehicle** contains the attributes of the Drivers Car.
    6. The entity **Booking** is a relationship between Driver, Passenger and Trip. It handles the locations related to Driver & Passenger.

- **Relationships:**
    1. Passenger, Driver, Trip has a relation of Booking
    2. Driver manages the Vehicles
    3. Passenger searches available Drivers
    4. Payment is done after the Trip is Finished
    5. Passenger & Driver can view Previous Trips

Data Population in all tables: Please refer to the Github Repo for the data population scripts and CSV files.

Weak Entity Set:

Payment(Payment_id, Payment_amount, Payment_status, Payment_type)

Reason: The existence of a Payment entity depends upon Trip completion. Once the trip gets completed, then Payment comes into existence. Before that, its existence is meaningless. So, the identifying entity is Trip, and the weak entity is Payment.

Ternary Relationship:

Booking (on Driver, Passenger, Trip)

Reason : Passenger gives a booking request to Driver which when accepted starts the Trip. Relation needed to show the associated Driver, Passenger of Trip.

Entities Participation Type:

1. Total Participation :
   Passenger,Driver in Booking
   Driver in Manage
   Passenger in search_available

2. Partial Participation:
   Driver in search_available
   Vehicle in Manage
   Trip in Previous_Trips

Relationship Roles:

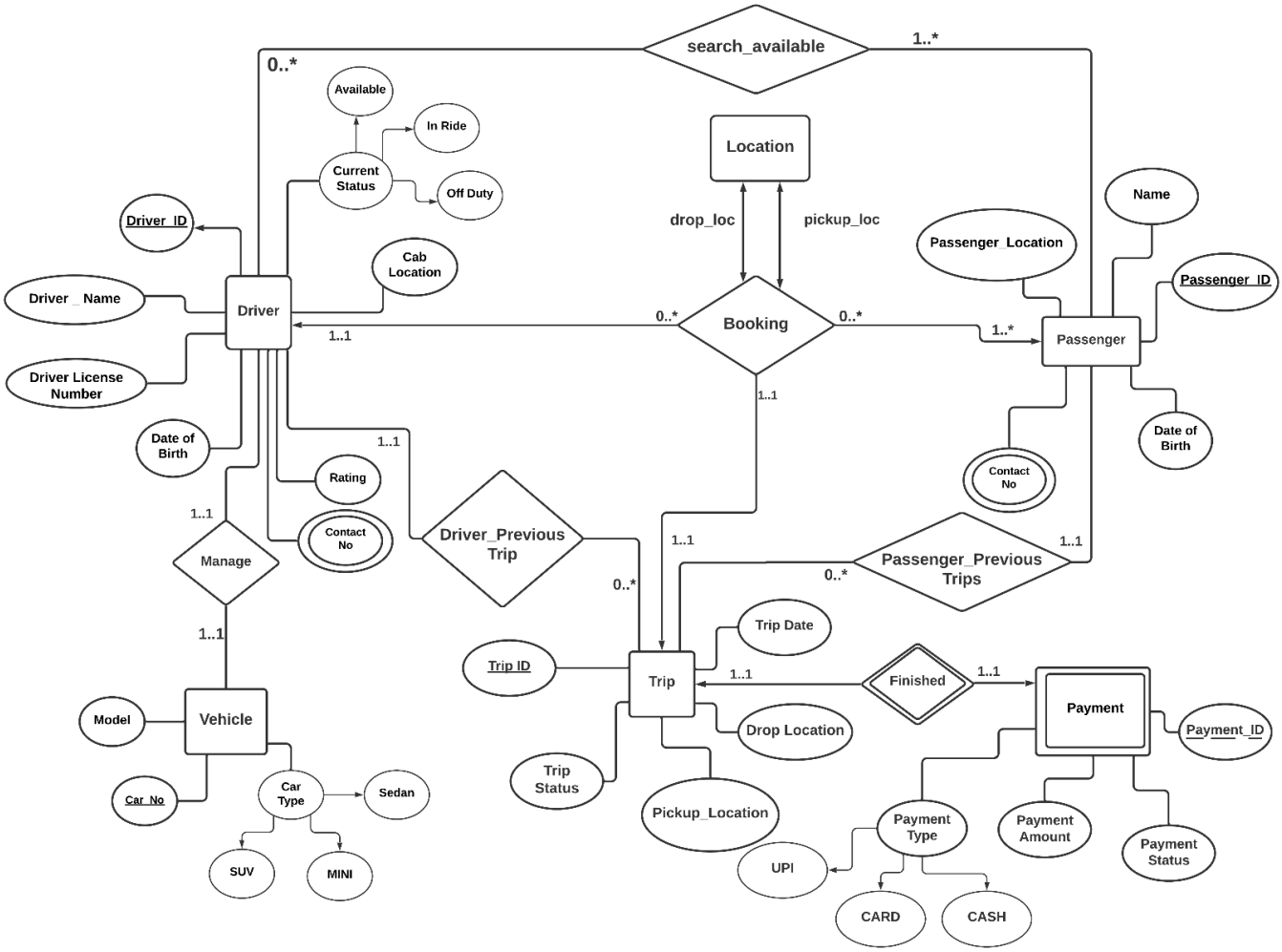
Drop_loc, Pickup_loc in Booking

Reason : Show the drop location and pickup location for the Booking through different roles as both are needed but are different instances of Location.

Relationship Constraints:

1. Drop_loc != Pickup_loc in Booking on Location
2. Driver.Current_Status is Available in search_available

E R Diagram :



[Additional PDF of ER Diagram has been attached in the Github Repository]

Schema

- Vehicle(Car_No : Int(12),Model : Varchar(30) ,Car_Type : Varchar(10), Car_Type in {'Sedan', 'Mini', 'SUV'})
- Location(Address : Varchar(30), Pincode : Int(6))
- Passenger(Passenger_ID : Int(10),Name : Varchar(30), Date_of_Birth : Date(), Contact_No : Int(10),Pickup_Location : Location)
- Driver(Driver_ID : Int(10), Driver_Name : VarChar(30), Driver_License_Number : Int(12), Date_of_Birth : Date(), Contact_No : Int(10), Rating : Int(1), Cab_Location : Location, Current_Status : VarChar(10), Driver_Car_No : Int(12), Current_Status in {'Off Duty','Available','In Ride'}
- Trip(Trip_ID : Int(10), Trip_Status : Varchar(30), Trip_Date_Day : Date() , Trip_Passenger_ID : Int(10), Trip_Driver_ID : Int(10), Drop_Location : Location, Pickup_Location : Location, Foreign Key(Trip_Passenger_ID, Trip_Driver_ID, Drop_Location, Pickup_Location) References Booking(Booking_Passenger_ID, Booking_Driver_ID, Drop_Location, Pickup_Location))
- Booking(Booking_Passenger_ID : Int(10) Foreign Key References ,Passenger(Passenger_ID), Booking_Driver_ID : Int(10) Foreign Key References Driver(Driver_ID)Drop_Location : Location, Pickup_Location : Location, Primary Key(Booking_Passenger_ID, Booking_Driver_ID), check(Drop_Location != Pickup_Location))
- Payment(# Trip_ID : Int(10) Primary Key references Trip(Trip_ID) ON DELETE CASCADE ,Payment_ID : bigint(10), Payment_Type : Varchar(30), Payment_Amount : Int(5), Payment_Status : bool, check Payment_Type in {'CASH', 'CARD', 'UPI', })
- search_available(Driver_ID: bigint(10) Passenger_ID: bigint(10))
- manage(Driver_ID: bigint(10) Car_No: bigint(10))
- finish (Driver_ID: Int(10) Car_No: bigint(10))
- driver_previous_trips(Driver_ID: Int(10) Trip_ID: bigint(10))
- passenger_previous_trips(Driver_ID: bigint(10) Trip_ID: int(10))

Driver

Driver_ID	bigint(10)	Primary Key
Driver_Name	VarChar(30	NOT NULL
Driver_License_Number	Int(12)	UNIQUE
Date_of_Birth	Date()	NOT NULL
Contact_No	Int(10)	NOT NULL
Rating	Int(1)	
Cab_Location	VarChar(30)	NOT NULL
Current_Status	VarChar(10)	{'Off Duty','Available','In Ride'}
Driver_Car_No	bigint(10)	Foreign Key References Vehicle(Car_No)

Location

Pincode	Int(30)	NOT NULL
Address	Varchar(30)	NOT NULL

Booking

Booking_Passenger_ID	Int(10)	Foreign Key References Passenger(Passenger_ID)
Drop_Location	Location	NOT NULL
Pickup_Location	Location	NOT NULL
Booking_Driver_ID	Int(10)	Foreign Key References Driver(Driver_ID)

Passenger

Passenger_ID	bigint(10)	Primary Key
Name	Varchar(30)	NOT NULL
Date_of_Birth	Date()	NOT NULL
Contact_No	bigint(10)	NOT NULL
Pickup_Location	Location	NOT NULL

Vehicle

Car_No	bigint(10)	Primary Key
Model	Varchar(30)	NOT NULL
Car_Type	Varchar(10)	{'Sedan', 'Mini', 'SUV'}

Trip

Trip_ID	Int(10)	NOT NULL
Trip_Status	Varchar(30)	NOT NULL
Drop_Location	Location	NOT NULL
Pickup_Location	Location	NOT NULL
Trip_Date	Date()	NOT NULL
Trip_Passenger_ID	Int(10)	NOT NULL
Trip_Driver_ID	Int(10)	NOT NULL

Foreign Key(Trip_Passenger_ID, Trip_Driver_ID, Drop_Location, Pickup_Location) References Booking(Booking_Passenger_ID, Booking_Driver_ID, Drop_Location, Pickup_Location)

Search_Available

Driver_ID	bigint(10)	Not null, Foreign Key part of Primary Key along with Passenger_ID
Passenger_ID	bigint(10)	Not null, Foreign Key part of Primary Key along with Driver_ID

Manage

Driver_ID	bigint(10)	Not null, Foreign Key part of Primary Key along with Car_No
Car_No	bigint(10)	Not null, Foreign Key part of Primary Key along with Driver_ID

Finish

Trip_ID	Int(10)	Not null, Foreign Key part of Primary Key along with Car_No
Payment_ID	bigint(10)	Not null, Foreign Key part of Primary Key along with Trip_ID

Driver_Previous_Trips

Driver_ID	bigint(10)	Not null, Foreign Key part of Primary Key along with Trip_ID
Trip_ID	Int(10)	Not null, Foreign Key part of Primary Key along with Driver_ID

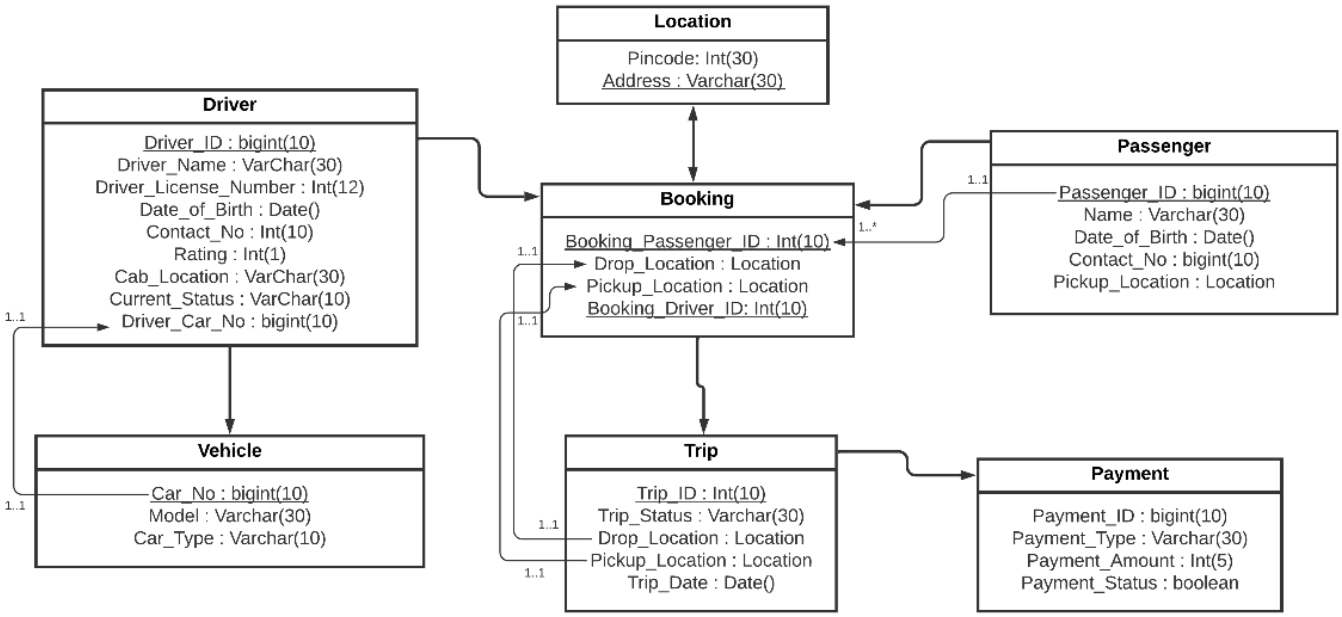
Passenger_PreviousTrips

Passenger_ID	bigint(10)	Not null, Foreign Key part of Primary Key along with Trip_ID
Trip_ID	Int(10)	Not null, Foreign Key part of Primary Key along with Passenger_ID

Payment

Trip_ID	Int(10)	Primary Key references Trip(Trip_ID) ON DELETE CASCADE
Payment_ID	bigint(10)	NOT NULL, UNIQUE
Payment_Type	Varchar(30)	NOT NULL {UPI,CARD,CASH}
Payment_Amount	Int(5)	NOT NULL
Payment_Status	boolean	NOT NULL

Schema Diagram :



Mapping Constraints including Integrity Constraints :

```
# create Table Vehicle(
Car_No : Int(12) Primary Key
Model : Varchar(30) NOT NULL
Car_Type : Varchar(10)
check Car_Type in {'Sedan', 'Mini', 'SUV'}
)
```

```
# create Table Location(
Address : Varchar(30) NOT NULL
Pincode : Int(6) NOT NULL
Primary Key(Address, Pincode)
)
```

```
# create Table Passenger(
Passenger_ID : Int(10) Primary Key
Name : Varchar(30)
Date_of_Birth : Date()
Contact_No : Int(10)
Pickup_Location : Location NOT NULL
)
```

```
# create Table Driver(
Driver_ID : Int(10) Primary Key
Driver_Name : VarChar(30)
Driver_License_Number : Int(12) UNIQUE
Date_of_Birth : Date()
Contact_No : Int(10) NOT NULL
Rating : Int(1)
Cab_Location : Location NOT NULL
Current_Status : VarChar(10)
Driver_Car_No : Int(12) Foreign Key References Vehicle(Car_No)
check Current_Status in {'Off Duty','Available','In Ride'}
)
```

```
create Table Trip(
Trip_ID : Int(10) Primary Key
Trip_Status : Varchar(30) NOT NULL
Trip_Date_Day : Date() NOT NULL
Trip_Passenger_ID : Int(10) NOT NULL
Trip_Driver_ID : Int(10) NOT NULL
Drop_Location : Location NOT NULL
Pickup_Location : Location NOT NULL
Foreign Key(Trip_Passenger_ID, Trip_Driver_ID, Drop_Location, Pickup_Location)
References Booking(Booking_Passenger_ID, Booking_Driver_ID, Drop_Location,
Pickup_Location)
)
```

```
# create Table Booking(
Booking_Passenger_ID : Int(10) Foreign Key References
Passenger(Passenger_ID)
Booking_Driver_ID : Int(10) Foreign Key References Driver(Driver_ID)
Drop_Location : Location NOT NULL
Pickup_Location : Location NOT NULL
Primary Key(Booking_Passenger_ID, Booking_Driver_ID)
check(Drop_Location != Pickup_Location)
)
```

```
# create Table Payment(
# Trip_ID : Int(10) Primary Key references Trip(Trip_ID) ON DELETE CASCADE
Payment_ID : bigint(10) NOT NULL UNIQUE
Payment_Type : Varchar(30)
Payment_Amount : Int(5) NOT NULL
Payment_Status : bool
check Payment_Type in {'CASH', 'CARD', 'UPI', }
)
```

[Additional PDF of both Diagrams has been attached in the Github Repository]

SQL Queries:

Query 1: To Return record that have same matching Values that have a Matching Value in Both Tables(payment and trip)

Sol:
```
SELECT * FROM payment
INNER JOIN trip
ON  Trip_Id_Pay = Trip_ID;
```

Query 2: To Return all records from left table(Payment) and the matched records from the right table(Trip) where Payment_Status = false(no payment done).

Sol:
```
SELECT * FROM payment
LEFT JOIN trip
ON Trip_ID_Pay = Trip_ID
WHERE Payment_Status ='FALSE';
```

Query 3: To Return Payment_ID, Payment_Type, Payment_Amount, Payment_Status From Payment Table Where Payment amount should be more than 1000 and payment type should be in CASH and payment status should be DONE .

Sol:
```
SELECT Payment_ID, Payment_Type, Payment_Amount, Payment_Status
FROM payment
Where Payment_Amount > 1000 AND Payment_Type ='CASH' AND
Payment_Status=’TRUE’;
```

Query 4: To Return Passenger_ID, Name of all passengers who have completed a trip on 22/2/2022 and name starting with “Adi”.

Sol:
```
SELECT passenger.Passenger_ID, passenger.Name
FROM Passenger, Trip
WHERE trip.Trip_Date_Day ="2022-02-22" AND passenger.Name like
"Adi%";
```

Or

SELECT Distinct Passenger_ID, Name
FROM passenger, Trip,booking
WHERE Passenger_ID =Trip_Passenger_ID and Trip_Date_Day ="2021-10-01" AND Name like "Bob%";

Query 5: View Records of all Available Drivers having a Sedan and having a rating between 3 & 5.

Sol: SELECT *
FROM driver ,vehicle
WHERE Driver_Car_Number=Car_no and car_type ="Sedan" AND Rating BETWEEN 3 AND 5;

Query 6: To Return Trip_ID, Driver_ID, Passenger_ID, Trip Date for all Trips where Payment Type is CASH and vehicle was a Sedan or SUV.

Sol: SELECT Trip_ID,Driver_id,Passenger_ID,Trip_Date_Day
FROM payment,trip,booking,driver,passenger
WHERE Trip_Passenger_ID=Passenger_ID and Payment_Type="CASH" and Trip_Driver_ID in (SELECT d2.Driver_Id FROM Driver d2,Vehicle WHERE d2.Driver_Car_Number = Car_No AND Car_Type NOT IN ("Mini"));

Query 7: Show all the drivers who did trip on both dates **2020-02-02** and **2021-01-01.**

Sol: SELECT d.Driver_id,d.Driver_Name
FROM trip t1,driver d
WHERE d.Driver_id=t1.Trip_Driver_ID and t1.Trip_Date_Day='2020-02-02' and exists(select Driver_id from trip t2 where t2.Trip_Driver_ID = t1.Trip_Driver_ID and t2.Trip_Date_Day='2021-01-01');

Query 8: Get Driver Details (ID and Name) who rides a specific car type(say Sedan).

Sol: SELECT Driver_ID, Driver_Name

```
FROM Driver
WHERE Driver_Car_Number  IN (select Car_No from Vehicle where
Car_Type="Sedan") ;
```

Query 9: Show the number of top drivers(whose rating are greater than 8)the average payment made on trips with a top driver according to the rating system.

Sol :

```
SELECT d.Rating as Rating_level,count(*) as Num_top_drivers,
avg(pt.Payment_Amount) as avg_amount
FROM driver d, (select * from payment, trip WHERE Trip_ID =Trip_ID_Pay)
as pt
WHERE pt.Trip_Driver_ID = d.Driver_id
GROUP BY d.Rating
HAVING d.Rating > 8;
```

Query 10 : Create a view of all the elite passengers (an elite passenger is one who has spent more than 5k on trips)

Sol :

```
CREATE VIEW elite_passengers as
with pass_trips(passengerid,trips_count,money_spent) as
(SELECT Trip_Passenger_ID,count(*),sum(Payment_Amount) as
pay_amt from trip ,payment where Trip_ID=Trip_Id_Pay GROUP BY
Trip_Passenger_ID having pay_amt>5000)
SELECT Passenger_ID, Name, pass_trips.trips_count,
pass_trips.money_spent FROM pass_trips, passenger where
Passenger_ID=pass_trips.passengerid;
```

Query 11. Get the trips with more pay than any other trip in a Sedan or the trip with a passenger born in year 2021.

Sol :

```
with tp(id, Pay) as
(select Trip_Driver_ID,Payment_Amount from trip,payment where
Trip_ID = Trip_Id_Pay)
select d.Driver_id, d.Driver_Name from driver d,tp as tp1 where
d.Driver_id=tp1.id and tp1.pay > all(select tp2.pay from tp tp2, (select
Driver_id,Car_Type from driver, vehicle where
```

```
Driver_Car_Number=Car_no) as Dc(d_id, ctype)  where tp2.id=
Dc.d_id and Dc.ctype='SEDAN')
union
select d.Driver_id, d.Driver_Name from driver d,trip ,
payment,passenger p where d.Driver_id=Trip_Driver_ID and
Trip_Passenger_ID=p.Passenger_ID and Trip_Id_Pay=Trip_ID and
p.Date_of_Birth like "2021-__-__";
```