DBMS PROJECT G3-GROUP-13

UBER DATABASE MANAGEMENT SYSTEM

GROUP MEMBERS :-

1 . Parv Khetawat (202301157)

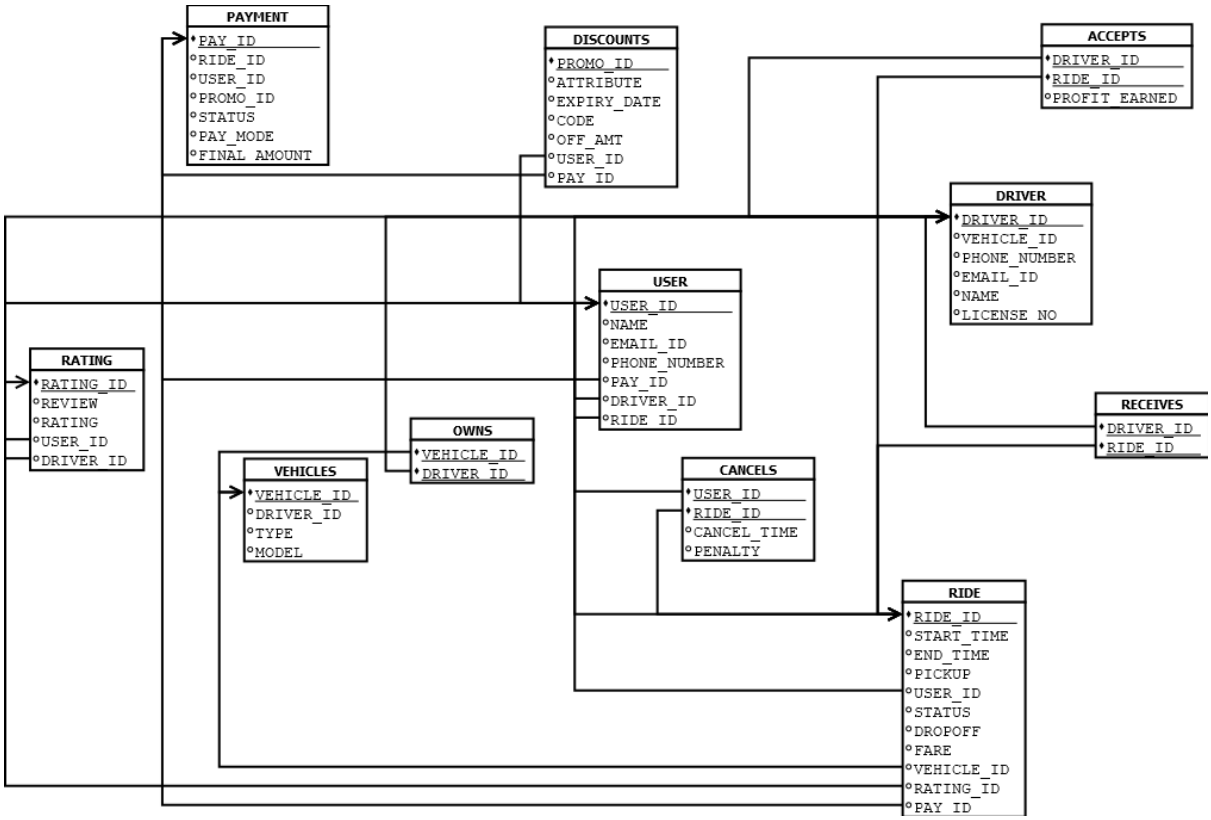
2 . Smit Limbasiya (202301139)

3 . Shrey Shah (202301165)

4 . Siddharth vala (202301180)

5 . Aayush Mittal (202301145)

❖ REALTIONAL DIAGRAM



❖ Functional Dependencies (FDs)
(As minimal FD set)

- USER_ID→ NAME, EMAIL_ID, PHONE_NUMBER, PAY_ID, DRIVER_ID, RIDE_ID
- DRIVER_ID → VEHICLE_ID, PHONE_NUMBER, EMAIL_ID, NAME, LICENSE_NO
- RIDE_ID → START_TIME, END_TIME, PICKUP, USER_ID, STATUS, DROPOFF, FARE, VEHICLE_ID, RATING_ID, PAY_ID
- VEHICLE_ID → DRIVER_ID, TYPE, MODEL
- RATING_ID → REVIEW, RATING, USER_ID, DRIVER_ID
- PAY_ID → RIDE_ID, USER_ID, PROMO_ID, STATUS, PAY_MODE, FINAL_AMOUNT
- PROMO_ID → ATTRIBUTE, EXPIRY_DATE, CODE, OFF_AMT
- (PROMO_ID, USER_ID) → PAY_ID
- (DRIVER_ID, RIDE_ID) → PROFIT_EARNED
- (USER_ID, RIDE_ID) → CANCEL_TIME, PENALTY

❖ BCNF RULE

➔ **A relation is in BCNF if for every non-trivial FD X → A, X is a Candidate or Superkey.**

- **User Table** :- UserID is the Primary Key.
- **Driver** :- DriverID is the Primary Key.
- **Ride** :- RideID is the Primary Key.
- **Vehicle Table** :- Vehicle_ID is the Primary Key.
- **Rating** :- Rating_ID is the Primary Key.
- **Payment** :- Pay_ID is the Primary Key.
- **Promo Table** :- Promo_ID is the Primary Key.
- **Payment_Promo Table** :- Composite Key (Promo_ID, UserID).
- **Accepts Table** :- Composite Key (Driver_ID, Ride_ID).
- **Cancels Table** :- Composite Key (UserID, Ride_ID).

➔ Here we can see all FD'S have satisfy BCNF Rule that all Non-primary dependency must depend on Candidate key or Superkey.

➔ Therefore , we can see this table's are in BCNF form.

❖ DDL SCRIPT

```
-- 1. Create schema
CREATE SCHEMA ride_sharing;
SET search_path TO ride_sharing;

-- 2. Tables

-- USER
CREATE TABLE "USER" (
    USER_ID INT PRIMARY KEY,
    NAME VARCHAR(100),
    EMAIL_ID VARCHAR(100) UNIQUE,
    PHONE_NUMBER VARCHAR(15),
    PAY_ID INT,
    DRIVER_ID INT,
    RIDE_ID INTdia
);

-- DRIVER
CREATE TABLE DRIVER (
    DRIVER_ID INT PRIMARY KEY,
    VEHICLE_ID INT,
    PHONE_NUMBER VARCHAR(15),
    EMAIL_ID VARCHAR(100),
    NAME VARCHAR(100),
    LICENSE_NO VARCHAR(50)
);

-- VEHICLES
CREATE TABLE VEHICLES (
    VEHICLE_ID INT PRIMARY KEY,
    DRIVER_ID INT,
    TYPE VARCHAR(50),
    MODEL VARCHAR(50)
);

-- RIDE
CREATE TABLE RIDE (
```

```
    RIDE_ID INT PRIMARY KEY,
    START_TIME TIMESTAMP,
    END_TIME TIMESTAMP,
    PICKUP VARCHAR(255),
    USER_ID INT,
    STATUS VARCHAR(50),
    DROPOFF VARCHAR(255),
    FARE DECIMAL(10,2),
    VEHICLE_ID INT,
    RATING_ID INT,
    PAY_ID INT
);

-- PAYMENT
CREATE TABLE PAYMENT (
    PAY_ID INT PRIMARY KEY,
    RIDE_ID INT,
    USER_ID INT,
    PROMO_ID INT,
    STATUS VARCHAR(50),
    PAY_MODE VARCHAR(50),
    FINAL_AMOUNT DECIMAL(10,2)
);

-- DISCOUNTS
CREATE TABLE DISCOUNTS (
    PROMO_ID INT PRIMARY KEY,
    ATTRIBUTE VARCHAR(100),
    EXPIRY_DATE DATE,
    CODE VARCHAR(50),
    OFF_AMT DECIMAL(10,2),
    USER_ID INT,
    PAY_ID INT
);

-- RATING
CREATE TABLE RATING (
    RATING_ID INT PRIMARY KEY,
    REVIEW TEXT,
    RATING INT CHECK (RATING BETWEEN 1 AND 5),
    USER_ID INT,
    DRIVER_ID INT
```

```
);

-- CANCELS
CREATE TABLE CANCELS (
    USER_ID INT,
    RIDE_ID INT,
    CANCEL_TIME TIMESTAMP,
    PENALTY DECIMAL(10,2),
    PRIMARY KEY (USER_ID, RIDE_ID)
);

-- OWNS
CREATE TABLE OWNS (
    VEHICLE_ID INT,
    DRIVER_ID INT,
    PRIMARY KEY (VEHICLE_ID, DRIVER_ID)
);

-- ACCEPTS
CREATE TABLE ACCEPTS (
    DRIVER_ID INT,
    RIDE_ID INT,
    PROFIT_EARNED DECIMAL(10,2),
    PRIMARY KEY (DRIVER_ID, RIDE_ID)
);

-- RECEIVES
CREATE TABLE RECEIVES (
    DRIVER_ID INT,
    RIDE_ID INT,
    PRIMARY KEY (DRIVER_ID, RIDE_ID)
);

-- 3. Foreign Keys

-- USER
ALTER TABLE "USER"
    ADD FOREIGN KEY (PAY_ID) REFERENCES PAYMENT(PAY_ID),
    ADD FOREIGN KEY (DRIVER_ID) REFERENCES
DRIVER(DRIVER_ID),
    ADD FOREIGN KEY (RIDE_ID) REFERENCES RIDE(RIDE_ID);
```

```
-- RIDE
ALTER TABLE RIDE
    ADD FOREIGN KEY (USER_ID) REFERENCES "USER"(USER_ID),
    ADD FOREIGN KEY (VEHICLE_ID) REFERENCES
VEHICLES(VEHICLE_ID),
    ADD FOREIGN KEY (RATING_ID) REFERENCES
RATING(RATING_ID),
    ADD FOREIGN KEY (PAY_ID) REFERENCES PAYMENT(PAY_ID);

-- DISCOUNTS
ALTER TABLE DISCOUNTS
    ADD FOREIGN KEY (USER_ID) REFERENCES "USER"(USER_ID),
    ADD FOREIGN KEY (PAY_ID) REFERENCES PAYMENT(PAY_ID);

-- RATING
ALTER TABLE RATING
    ADD FOREIGN KEY (USER_ID) REFERENCES "USER"(USER_ID),
    ADD FOREIGN KEY (DRIVER_ID) REFERENCES
DRIVER(DRIVER_ID);

-- CANCELS
ALTER TABLE CANCELS
    ADD FOREIGN KEY (USER_ID) REFERENCES "USER"(USER_ID),
    ADD FOREIGN KEY (RIDE_ID) REFERENCES RIDE(RIDE_ID);

-- OWNS
ALTER TABLE OWNS
    ADD FOREIGN KEY (VEHICLE_ID) REFERENCES
VEHICLES(VEHICLE_ID),
    ADD FOREIGN KEY (DRIVER_ID) REFERENCES
DRIVER(DRIVER_ID);

-- ACCEPTS
ALTER TABLE ACCEPTS
    ADD FOREIGN KEY (DRIVER_ID) REFERENCES
DRIVER(DRIVER_ID),
    ADD FOREIGN KEY (RIDE_ID) REFERENCES RIDE(RIDE_ID);

-- RECEIVES
ALTER TABLE RECEIVES
    ADD FOREIGN KEY (DRIVER_ID) REFERENCES
DRIVER(DRIVER_ID),
```

```
    ADD FOREIGN KEY (RIDE_ID) REFERENCES RIDE(RIDE_ID);
```