

Hop-On

Team: Storage Warriors

Dhanashree Mane

Arjoo Gangwal

Krutika Ambavane

Sanya Purwar

Guided by Prof. Dr. Shailesh Agarwal | MSIS 2603 Database Management System

# Table of Content

**1. Business Application Description 2**

* 1. **Introduction 2**
  2. **Objective 2**
  3. **Design Description 3**
  4. **Future Scope 3**

**2. User Types or Entities 4**

**3. Tables 5**

**4. Logical Schema – UML Model 6**

**5. Use Cases 7**

**6. Physical Schema – Database Dictionary 8**

**7. Queries 20**

**8. Indexes 30**

**9. Views 31**

**10. Metrics 35**

1. **Project Summary 39**
   1. **Summarize your experience with this exercise 39**
   2. **What was the hardest part of this project? 39**
   3. **What did you learn from the project? 39**
   4. **What problems did you run against in this project? 39**
   5. **How did you solve these problems? 40**
   6. **If you were to do this project again, what methodology would you follow? 40**

# Business application description

### Introduction

Today we see that congestion on the roads and road accident due to drunk driving are increasing enormously. According to New York Times the number of road accidents in the United States have increased by 14% from 2015 to 2017 and hundreds of million dollars have been spent on campaigns to remind the people of the dangers of drunk driving, failing to use seat belts and texting while driving.

One major factor responsible for the congestion is the increasing number of vehicles on the road. Introducing the concept of real time ride sharing will reduce this congestion and the accidents happening as well as curb emission of gases that cause global warming, Hence, more and more companies are looking to implement their own versions of a real time ride sharing system. Hop-On is a real time ridesharing service that provides on demand car transportation. Headquartered in Los Angeles, California, operating in the United States. To use our application the passengers must download the application, sign up, enter a valid phone number, and enter a valid form of payment (either a credit card, or link to an e-wallet account). Passengers can then request a ride from a nearby driver. Once confirmed, the app shows the driver and car details to the passenger. Similarly, the drivers must get registered with us. With our application we create a win-win situation for both the drivers and the passengers. The passengers get a ride to their destination instantly without the hassle of driving or parking and the drivers earn money by filling up empty seats of their car.

### Objective

The objectives of Hop-On are listed as under:

1. **Provide Cheap and Convenient Transportation:** Create an application which enables the passenger to book rides instantly and let them share rides in real time to save money.
2. **Reduce Traffic Congestion:** Our application aims to reduce the traffic on roads during peak hours by introducing safe and efficient ride-sharing.
3. **Help Drivers earn Money:** Provide drivers with means to earn money while sharing rides with the passengers on their convenience.
4. **Reduce the Number of Road Accidents:** Provide people with cheap commute alternatives to reduce drunk driving on roads and ensure safety by hiring drivers only after a thorough background check.

### Design Description

Each ride ID uniquely identifies a passenger.Thus, in a pool every passenger will still have unique ride ID. We haven’t implemented the concept of master ride ID.

A user can be both, passenger and driver.

An incident can be lodged in two ways:

1. the passenger can lodge the complaint directly through the app which will add it to the database
2. the passenger can call the passenger support and the support representative will thereafter enter the incident details into the database

The case of refund is handled using two database tables: payment and ride.

After refunding the amount, the total cost of the ride changes to zero and additional comments get inserted into the payment table indicating that the amount for that ride was refunded.

The passenger/driver support who adds the driver/passenger into the blacklist wont necessarily be the one to remove them from the list.

### Scope

Currently, we are operating only in California. We are offering the passengers LUXURY, XL, X and POOL options for a Ride. We have car types like SUV, Sedan, Hatchback and Minivan which we map respectively based on make and model to the Ride Types. We have two modes of payment currently i.e. Card and PayPal. By Q1 2019, we expect to increase our customer base in the entire United States. As of now, we have a lot of demand for luxury rides. We have estimated to gain increased demands in Pool and X services in the coming 6 months through customer acquisition in California. As we increase in size, we will add new payment modes like Apple Pay and we will be creating a credit system for the customers to use. Each customer will get bonus credits for registering on our application. Also, he can earn credits by inviting friends to register.

# User Types or Entities

Below users or entities would play a pivotal role in our Ridesharing Application:

1. **Passenger**- A person who requests a ride to go from a certain location to another.
2. **Driver**- A driver is a person hired by Hop-On to drive their car around a specific area to provide rides for passengers who book them.
3. **Customer Support Executive**- A person who assists the passengers with all the complaints and issues.
4. **Driver Support Executive** - A person who assists the drivers with all the complaints and issues.
5. **Application** **Administrator** - A person who analyses the data and trends of the business and provides insights to improve the business value.

# Tables

1. Passenger
2. Driver
3. Car
4. Ride
5. Rating
6. Payment Mode
7. PayPal
8. Card
9. Payment
10. Passenger Support
11. Driver Support
12. Passenger Incident
13. Driver Incident
14. Passenger Blacklist
15. Driver Blacklist

# Logical Schema – UML Model

# 

# Use Cases

Below are the use-cases for database users-

1. Passenger
   1. Register/Login to the application.
   2. Request a ride.
   3. Cancel the booked ride.
   4. View ride history.
   5. Rate and tip the Driver.
2. Passenger Support Executive
   1. Raise an incident lodged by Passengers. (accident, lost & found, sexual harassments)
   2. Look-up an incident using Passenger username or ID, Incident ID.
   3. Update the driver as ‘blacklisted’ based on passenger complaints.
   4. Retrieve passenger ride information in case of refund requests.
3. Driver
   1. Register/Login to the application.
   2. Accept a requested ride.
   3. Cancel an accepted ride.
   4. Switch to ‘Non-accepting mode’.
   5. Rate Passenger and view the cost of the ride.
   6. Transfer earnings directly to bank account.
4. Driver Support Executive
   1. Raise an incident lodged by Driver.
   2. Look-up an incident using Driver username or ID, Incident ID.
   3. Update the passenger as ‘blacklisted’ based on driver complaints.
   4. Retrieve the driver’s insurance information using Driver ID.
   5. Retrieve the driver’s vehicle information using Driver ID.
5. Application Admin
   1. Analyze high demand areas/ cities.
   2. Find high demand on ride type.
   3. Find the total number of rides completed daily, monthly and yearly.
   4. Analyze overall revenue growth yearly.
   5. Analyze the revenue generated by city.

# Physical Schema – Database Dictionary

1. **PASSENGER**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| PASSENGER\_ID | INT | **PRIMARY KEY** |  |
| FIRST\_NAME | VARCHAR(50) | NOT NULL |  |
| LAST\_NAME | VARCHAR(50) |  |  |
| PASSENGER\_ADDRESS | VARCHAR(100) |  |  |
| CITY | VARCHAR(50) |  |  |
| ZIP | BIGINT |  |  |
| PHONE\_NO | INT | NOT NULL, UNIQUE |  |
| EMAIL\_ID | VARCHAR(50) | NOT NULL, UNIQUE |  |
| USERNAME | VARCHAR(20) | NOT NULL, UNIQUE |  |
| PASSWORD | VARCHAR(45) | NOT NULL |  |
| PASSENGER\_AVG\_RATING | FLOAT |  |  |
| P\_STATUS | ENUM | NOT NULL |  |

P\_STATUS → {ACTIVE, INACTIVE}

1. **DRIVER**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| DRIVER\_ID | INT | **PRIMARY KEY** |  |
| FIRST\_NAME | VARCHAR(50) | NOT NULL |  |
| LAST\_NAME | VARCHAR(50) |  |  |
| PHONE\_NO | INT | NOT NULL, UNIQUE |  |
| DRIVER\_ADDRESS | VARCHAR(100) |  |  |
| CITY | VARCHAR(50) |  |  |
| ZIP | BIGINT |  |  |
| EMAIL\_ID | VARCHAR(50) | NOT NULL, UNIQUE |  |
| USERNAME | VARCHAR(20) | NOT NULL, UNIQUE |  |
| D\_PASSWORD | VARCHAR(50) | NOT NULL |  |
| LISCENCE\_NO | INT | NOT NULL |  |
| LISCENCE\_VALID\_FROM | DATE | NOT NULL |  |
| LISCENCE\_VALID\_TO | DATE | NOT NULL |  |
| DRIVER\_AVG\_RATING | FLOAT |  |  |
| NON\_ACCEPTANCE | ENUM | NOT NULL |  |
| D\_STATUS | ENUM | NOT NULL |  |

NON\_ACCEPTANCE → {ON, OFF}

D\_STATUS → {ACTIVE, INACTIVE}

1. **CAR**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| CAR\_ID | INT | **PRIMARY KEY** |  |
| DRIVER\_ID | INT | **FOREIGN KEY** |  |
| REGISTRATION\_NO | VARCHAR(50) | NOT NULL, UNIQUE |  |
| MAKE | VARCHAR(50) | NOT NULL |  |
| CAR\_TYPE | ENUM | NOT NULL |  |
| MODEL | VARCHAR(50) |  |  |
| COLOR | VARCHAR(20) |  |  |
| POLICY\_ID | VARCHAR(100) | NOT NULL |  |
| INSURANCE\_VALID\_FROM | DATE |  |  |
| INSURANCE\_VALID\_TO | DATE | NOT NULL |  |
| SMOG\_CHECK | ENUM |  |  |

CAR\_TYPE {SUV, SEDAN, HATCHBACK, MINIVAN}

SMOG\_CHECK {Y, N}

1. **RIDE**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| RIDE\_ID | INT | **PRIMARY KEY** |  |
| PASSENGER\_ID | INT | **FOREIGN KEY** |  |
| CAR\_ID | INT | **FOREIGN KEY** |  |
| RIDE\_TYPE | ENUM | NOT NULL |  |
| RIDE\_STATUS | ENUM | NOT NULL |  |
| RIDE\_SOURCE | VARCHAR(100) | NOT NULL |  |
| RIDE\_DESTINATION | VARCHAR(100) | NOT NULL |  |
| SOURCE\_CITY | VARCHAR(50) | NOT NULL |  |
| DESTINATION\_CITY | VARCHAR(50) | NOT NULL |  |
| START\_DATE | DATE | NOT NULL |  |
| END\_DATE | DATE | NOT NULL |  |
| START\_TIME | TIME |  |  |
| END\_TIME | TIME |  |  |
| NO\_OF\_SEATS | INT |  | SEATS OFFERED IN THE CAR |
| ETA | TIME |  | ESTIMATED TIME OF ARRIVAL |
| PICKUP\_TIME | TIME |  | TIME FOR THE DRIVER TO REACH AT PICKUP LOCATION |
| DRIVER\_LOCATION | VARCHAR(100) |  | CURRENT LOCATION OF THE DRIVER WHEN RIDE IS REQUESTED |
| CANCELLATION\_REASON | ENUM |  |  |
| RIDE\_COST | FLOAT | NOT NULL |  |
| RIDE\_TIP | FLOAT |  |  |
| RIDE\_TOTAL\_COST | FLOAT | NOT NULL |  |

RIDE\_TYPE: {LUXURY, XL, X, POOL}

RIDE\_STATUS: {REQUESTED, CURRENT, COMPLETED, CANCELLED}

CANCELLATION\_REASON: {Driver too far, Changed your mind, Driver asked me to cancel, Passenger did not show up, Driver ETA too short}

1. **RATING**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| RATING\_ID | INT | **PRIMARY KEY** |  |
| RIDE\_ID | INT | **FOREIGN KEY** |  |
| DRIVER\_RATING | FLOAT |  |  |
| PASSENGER\_RATING | FLOAT |  |  |

1. **PAYMENT\_MODE**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| PAYMENT\_MODE\_ID | INT | **PRIMARY KEY** |  |
| PASSENGER\_ID | INT | **FOREIGN KEY** |  |
| MODE\_TYPE | ENUM | NOT NULL |  |

MODE\_TYPE→ {CARD, PAYPAL}

1. **PAYPAL**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| PAYPAL\_ACCOUNT\_NO | INT | **PRIMARY KEY** |  |
| PAYMENT\_MODE\_ID | INT | **FOREIGN KEY** |  |

1. **CARD**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| CARD\_NO | NVARCHAR(25) | **PRIMARY KEY** |  |
| PAYMENT\_MODE\_ID | INT | **FOREIGN KEY** |  |
| CARD\_TYPE | ENUM | NOT NULL |  |
| CARDHOLDER\_NAME | VARCHAR(50) | NOT NULL |  |
| VALID\_THROUGH | DATE | NOT NULL |  |
| ZIP | INT(5) | NOT NULL |  |

CARD TYPE → {CREDIT, DEBIT}

1. **PAYMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| PAYMENT\_ID | INT | **PRIMARY KEY** |  |
| RIDE\_ID | INT | **FOREIGN KEY** |  |
| PAYMENT\_MODE\_ID | INT | **FOREIGN\_KEY** |  |
| AMOUNT | FLOAT | NOT NULL |  |

1. **PASSENGER\_INCIDENT**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| P\_INCIDENT\_ID | INT | **PRIMARY KEY** |  |
| P\_START\_DATE | DATE |  |  |
| P\_END\_DATE | DATE |  |  |
| PASSENGER\_ID | INT | **FOREIGN KEY** |  |
| P\_REP\_ID | INT | **FOREIGN KEY** |  |
| RIDE\_ID | INT | **FOREIGN KEY** |  |
| P\_INCIDENT\_TYPE | ENUM | NOT NULL |  |
| P\_INCIDENT\_PRIORITY | ENUM | NOT NULL |  |
| P\_ADDL\_DESCIPTION | VARCHAR(200) |  |  |
| P\_INCIDENT\_STATUS | ENUM | NOT NULL |  |
| P\_COMMENT | VARCHAR(200) |  |  |

P\_INCIDENT\_TYPE: {LOST AND FOUND, ACCIDENT, UNETHICAL BEHAVIOUR, REFUND ISSUES, OTHER ISSUES}

P\_INCIDENT\_STATUS: {NEW, OPEN, CLOSED, REOPEN}

P\_INCIDENT\_PRIORITY: {HIGH, MEDIUM, LOW}

1. **DRIVER\_INCIDENT**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| D\_INCIDENT\_ID | INT | **PRIMARY KEY** |  |
| D\_START\_DATE | DATE |  |  |
| D\_END\_DATE | DATE |  |  |
| DRIVER\_ID | INT | **FOREIGN KEY** |  |
| D\_REP\_IP | INT | **FOREIGN KEY** |  |
| RIDE\_ID | INT | **FOREIGN KEY** |  |
| D\_INCIDENT\_TYPE | ENUM | NOT NULL |  |
| D\_INCIDENT\_PRIORITY | ENUM | NOT NULL |  |
| D\_ADDL\_DESCIPTION | VARCHAR(200) |  |  |
| D\_INCIDENT\_STATUS | ENUM | NOT NULL |  |
| D\_COMMENT | VARCHAR(200) |  |  |

D\_INCIDENT\_TYPE : { LOST AND FOUND, ACCIDENT, UNETHICAL BEHAVIOUR, OTHER ISSUES  }

D\_INCIDENT\_STATUS : {NEW, OPEN, CLOSED, REOPEN}

D\_INCIDENT\_PRIORITY: {HIGH, MEDIUM, LOW}

1. **PASSENGER\_SUPPORT**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| P\_REP\_ID | INT | **PRIMARY KEY** |  |
| P\_REP\_NAME | VARCHAR(50) | NOT NULL |  |
| P\_REP\_ADDRESS | VARCHAR(100) |  |  |
| P\_REP\_CONTACT | BIGINT, UNIQUE | NOT NULL |  |
| P\_REP\_EMAIL | VARCHAR(50), UNIQUE | NOT NULL |  |
| P\_REP\_USERNAME | VARCHAR(20), UNIQUE | NOT NULL |  |
| P\_REP\_PASSWORD |  | NOT NULL |  |

1. **DRIVER\_SUPPORT**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| D\_REP\_ID | INT | **PRIMARY KEY** |  |
| D\_REP\_NAME | VARCHAR(50) | NOT NULL |  |
| D\_REP\_ADDRESS | VARCHAR(100) |  |  |
| D\_REP\_CONTACT | BIGINT, UNIQUE | NOT NULL |  |
| D\_REP\_EMAIL | VARCHAR(50), UNIQUE | NOT NULL |  |
| D\_REP\_USERNAME | VARCHAR(20), UNIQUE | NOT NULL |  |
| D\_REP\_PASSWORD |  | NOT NULL |  |

1. **PASSENGER\_BLACKLIST**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| P\_BLACKLIST\_ID | INT | **PRIMARY KEY** |  |
| PASSENGER\_ID | INT | **FOREIGN KEY** |  |
| P\_ADDED\_BY | INT | **FOREIGN KEY** | D\_REP\_ID WHO ADDS PASSENGER IN BLACKLIST |
| P\_REMOVED\_BY | INT | **FOREIGN KEY** | D\_REP\_ID WHO REMOVES PASSENGER IN BLACKLIST |
| P\_BLACKLISTED\_FROM | DATE |  |  |
| P\_BLACKLISTED\_TO | DATE |  |  |
| P\_BLACKLIST\_STATUS | ENUM |  |  |

P\_BLACKLIST\_STATUS → {SUSPENDED, BLOCKED, UNBLOCKED}

1. **DRIVER\_BLACKLIST**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TYPE | CONSTRAINT | DESCRIPTION |
| D\_BLACKLIST\_ID | INT | **PRIMARY KEY** |  |
| DRIVER\_ID | INT | **FOREIGN KEY** |  |
| D\_ADDED\_BY | INT | **FOREIGN KEY** | P\_REP\_ID WHO ADDS DRIVER IN BLACKLIST |
| D\_REMOVED\_BY | INT | **FOREIGN KEY** | P\_REP\_ID WHO REMOVES DRIVER IN BLACKLIST |
| D\_BLACKLISTED\_FROM | DATE |  |  |
| D\_BLACKLISTED\_TO | DATE |  |  |
| D\_BLACKLIST\_STATUS | ENUM |  |  |

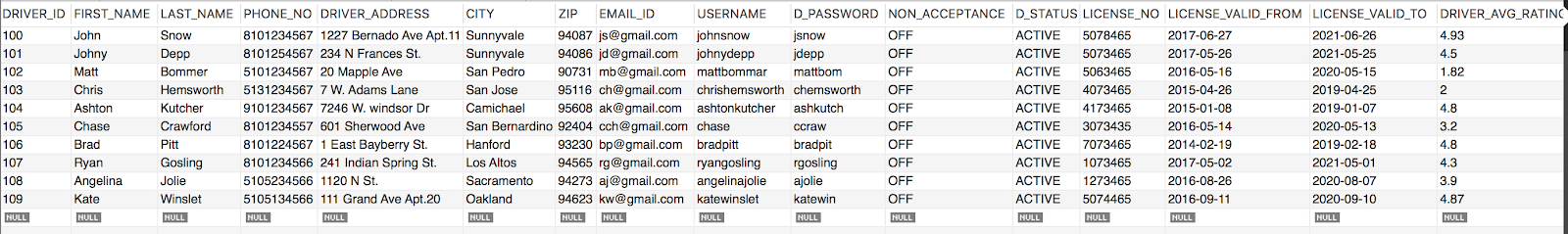
D\_BLACKLIST\_STATUS → {SUSPENDED, BLOCKED, UNBLOCKED}

# Queries

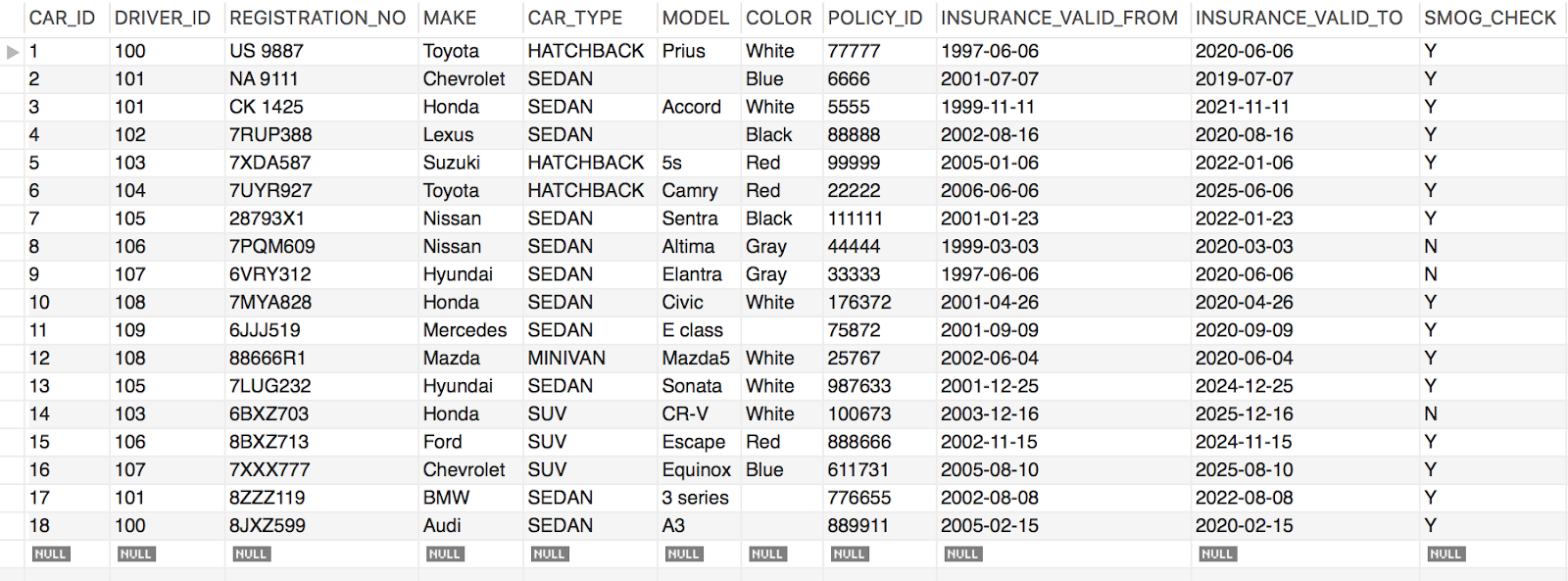
**Passenger:**

****

**DRIVER:**

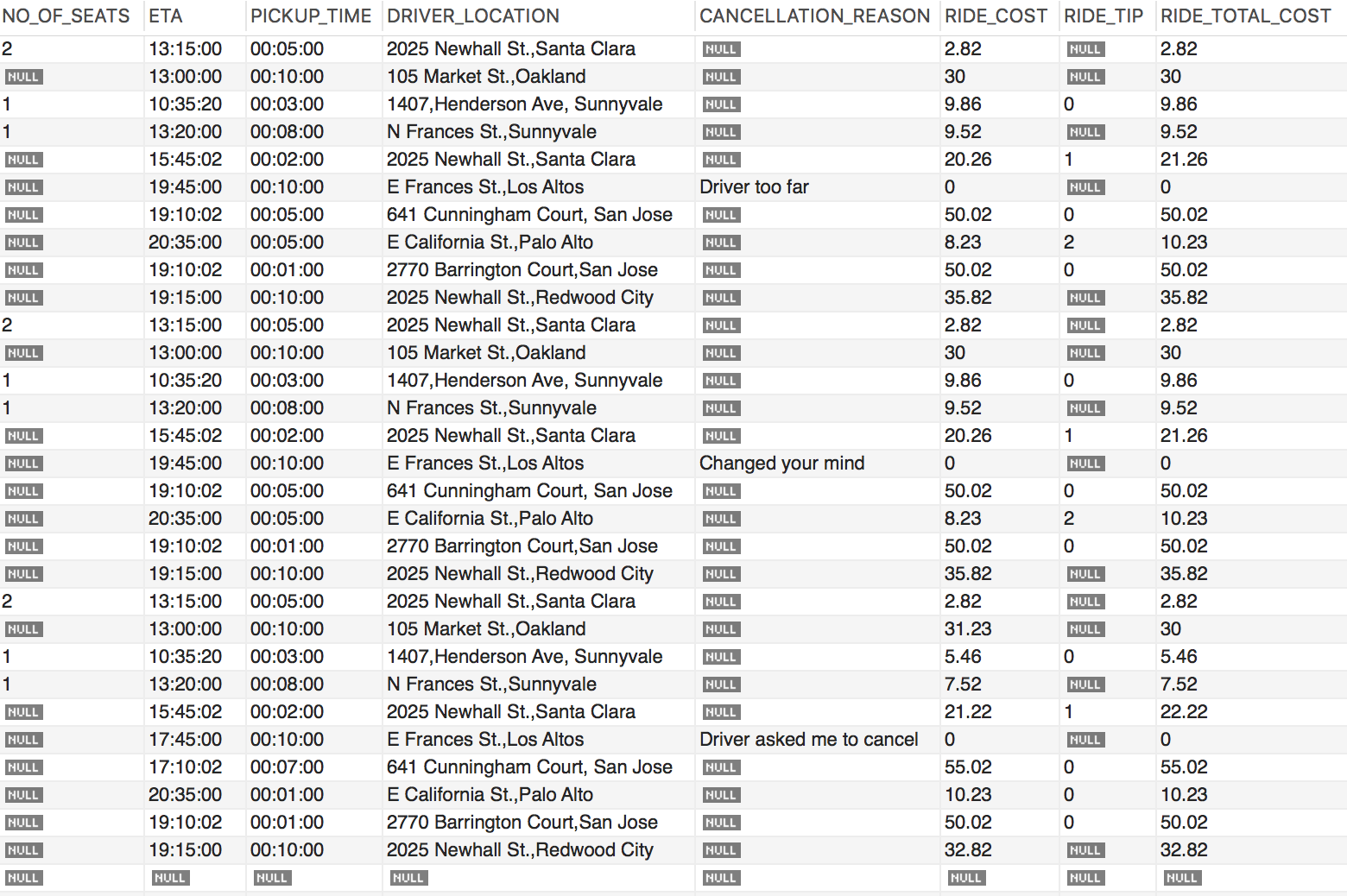
****

**CAR:**

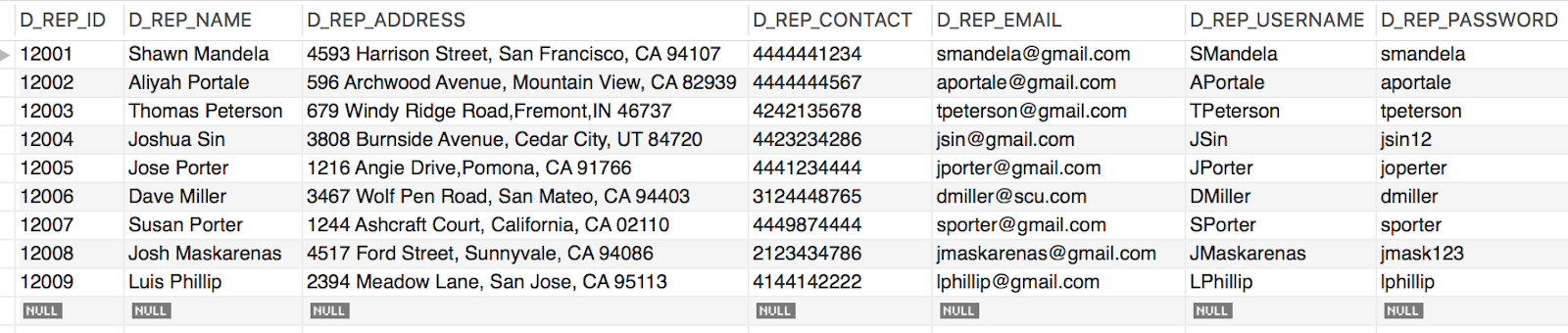
****

**RIDE:**

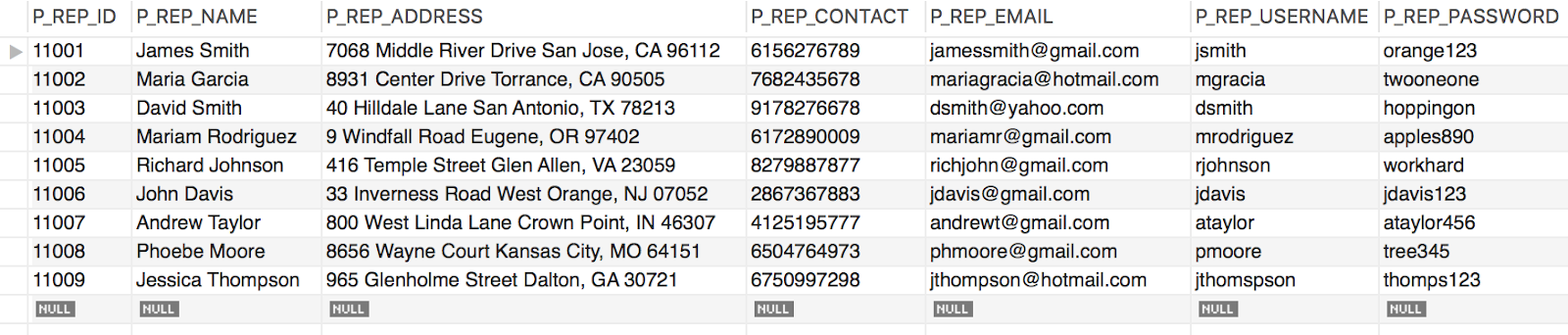
****

****

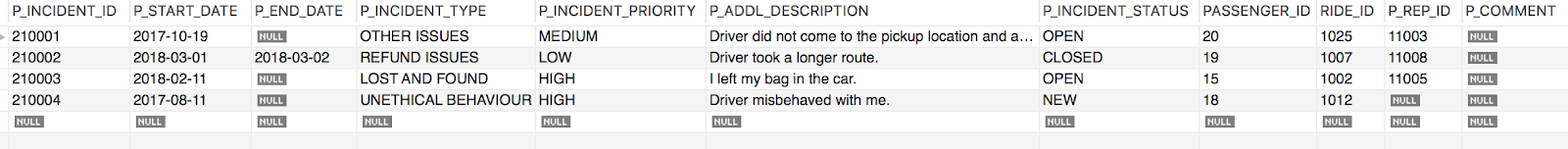
**DRIVER SUPPORT:**

****

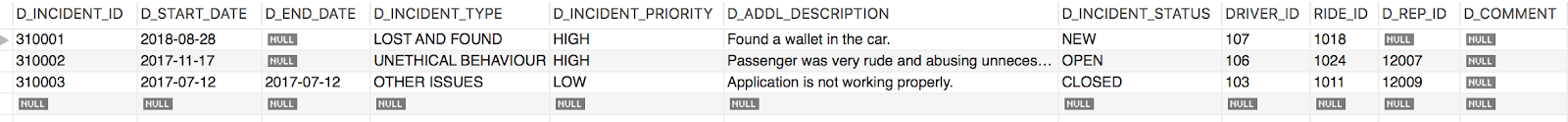
**PASSENGER SUPPORT:**

****

**PASSENGER INCIDENT:**

****

**DRIVER INCIDENT:**

****

**PASSENGER BLACKLIST:**

****

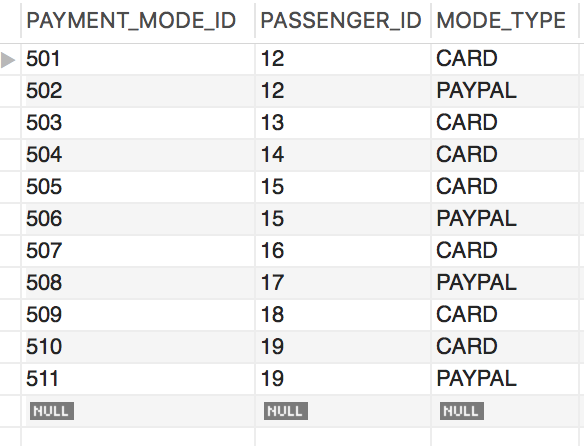
**DRIVER BLACKLIST:**

****

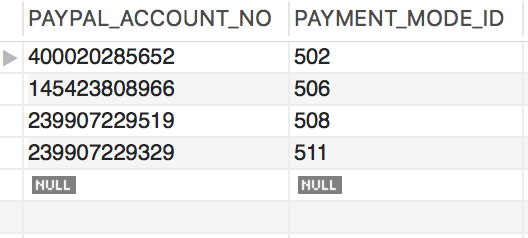
**PAYMENT:**

****

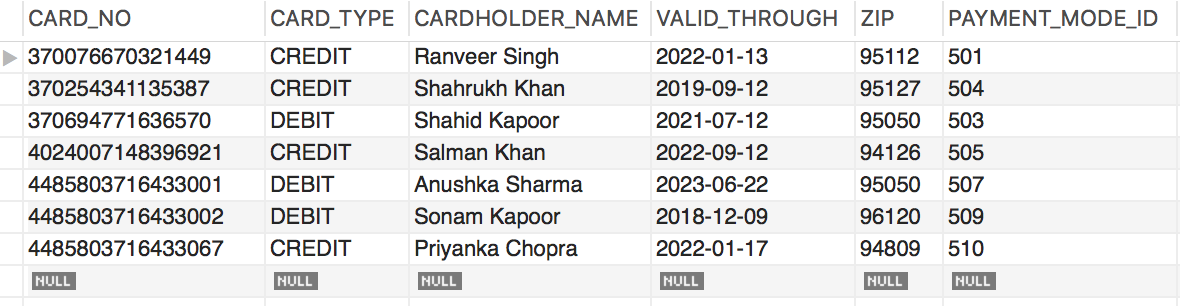
**PAYMENT MODE:**

****

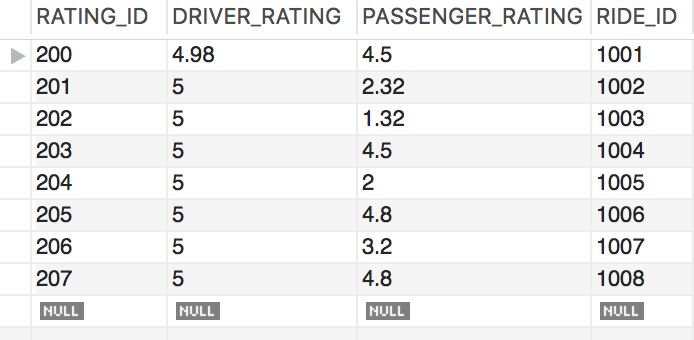
**PAYPAL:**

****

**CARD:**

****

**RATING:**

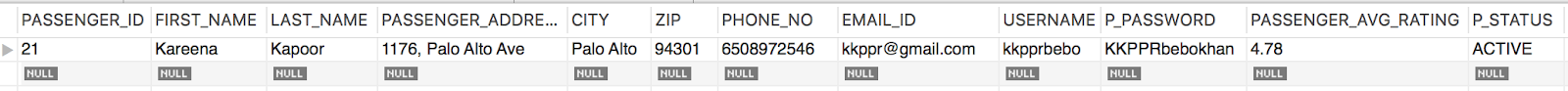
****

**Passenger:**

1. **PASSENGER REGISTERS ON THE APP (USING CONDITIONAL INSERT)**

INSERT INTO PASSENGER(PASSENGER\_ID, FIRST\_NAME, LAST\_NAME, PASSENGER\_ADDRESS, ZIPCODE, PHONE\_NO, EMAIL, USERNAME, P\_PASSWORD, P\_STATUS) VALUES (21, 'Kareena', 'Kapoor', '1176, Palo Alto Ave', 'Palo Alto', 94301, 6508972546 , 'kkppr@gmail.com', 'kkpprbebo', 'KKPPRbebokhan', 4.78,'ACTIVE');

SELECT \* FROM PASSENGER WHERE PASSENGER\_ID = 21;



**2. PASSENGER REQUESTS A RIDE**

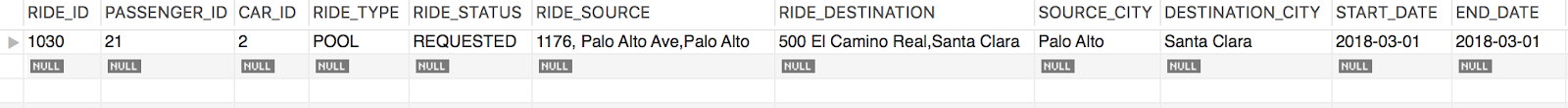
INSERT INTO RIDE(RIDE\_ID,PASSENGER\_ID ,CAR\_ID,RIDE\_TYPE ,RIDE\_STATUS ,RIDE\_SOURCE,RIDE\_DESTINATION ,SOURCE\_CITY,

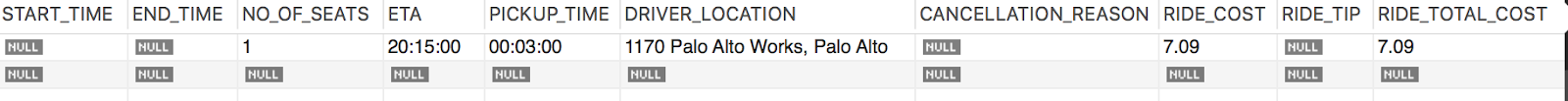
DESTINATION\_CITY ,START\_DATE ,END\_DATE ,START\_TIME ,END\_TIME,NO\_OF\_SEATS ,ETA,PICKUP\_TIME ,

DRIVER\_LOCATION ,CANCELLATION\_REASON ,RIDE\_COST ,RIDE\_TIP ,RIDE\_TOTAL\_COST)

VALUES(1030 ,21 ,002 ,'POOL' ,'REQUESTED' ,'1176, Palo Alto Ave, Palo Alto', '500 El Camino Real, Santa Clara', 'Palo Alto', 'Santa Clara', '2018-03-01', '2018-03-01', NULL, NULL, 1, '20:15:00' , '00:03:00', '1170 Palo Alto Works, Palo Alto', NULL ,7.09,NULL,7.09);

SELECT \* FROM RIDE WHERE RIDE\_ID = 1030



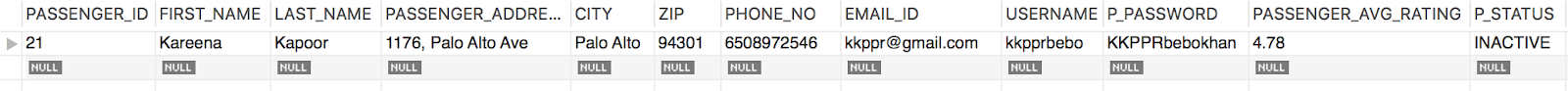


3. **PASSENGER REMOVES HIS ACCOUNT FROM THE APP**

UPDATE PASSENGER SET P\_STATUS = 'INACTIVE'

WHERE PASSENGER\_ID = 21;

SELECT \* FROM PASSENGER WHERE PASSENGER\_ID = 21;



4. **AFTER LODGING A COMPLAINT, PASSENGER WANTS TO CHECK HIS INCIDENT AGAIN FOR A PARTICULAR RIDE.**

SELECT R.RIDE\_ID, R.RIDE\_SOURCE, R.RIDE\_DESTINATION, R.START\_DATE,R.START\_TIME, R.END\_DATE, R.END\_TIME, R.RIDE\_TOTAL\_COST, I.P\_INCIDENT\_ID, I.P\_INCIDENT\_STATUS, I.P\_INCIDENT\_TYPE

FROM PASSENGER\_INCIDENT I, RIDE R

WHERE I.RIDE\_ID = R.RIDE\_ID

AND I.PASSENGER\_ID = 15 AND P\_INCIDENT\_ID = 210003;

(Specifying the incident id because one passenger may have lodged many complaints)

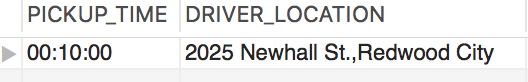


5. **AFTER REQUESTING A RIDE , PASSENGER CHECKS THE LOCATION OF THE DRIVER AND THE TIME DRIVER WILL TAKE TO REACH TO THE PICK-UP SPOT.**

SELECT PICKUP\_TIME, DRIVER\_LOCATION

FROM RIDE

WHERE RIDE\_ID = 1009;



6. **A PASSENGER WANTS TO ADD A PAYMENT MODE FOR HIS ACCOUNT**

START TRANSACTION;

INSERT INTO PAYMENT\_MODE(PAYMENT\_MODE\_ID, PASSENGER\_ID, MODE\_TYPE) VALUES (512,16,'PAYPAL');

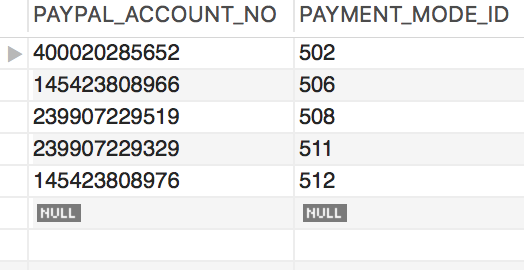
INSERT INTO PAYPAL(PAYMENT\_MODE\_ID, PASSENGER\_ID, MODE\_TYPE) VALUES('145423808976', 512);

COMMIT;

SELECT \* FROM PAYMENT\_MODE;



SELECT \* FROM PAYPAL;

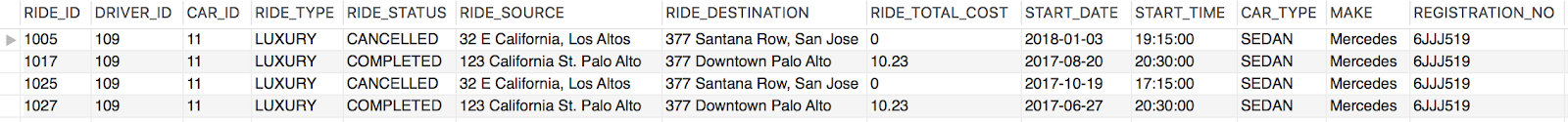


**Driver:**

1. **Check Ride History**

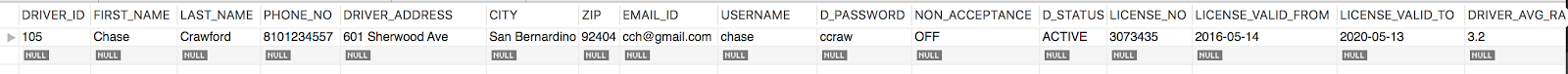
SELECT R.RIDE\_ID,C.DRIVER\_ID,C.CAR\_ID,R.RIDE\_TYPE,R.RIDE\_STATUS,R.RIDE\_SOURCE,R.RIDE\_DESTINATION,R.RIDE\_TOTAL\_COST,R.START\_DATE,R.START\_TIME,C.CAR\_TYPE,C.MAKE,C.REGISTRATION\_NO FROM RIDE R

JOIN CAR C ON C.CAR\_ID = R.CAR\_ID WHERE C.DRIVER\_ID = 109;



1. **Switch to non- accepting mode**

**Before Update:**

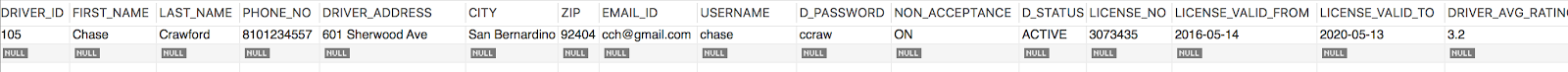


UPDATE DRIVER

SET NON\_ACCEPTANCE = 'ON'

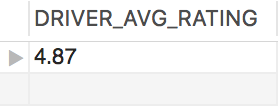
WHERE DRIVER\_ID = 105;

SELECT \* FROM DRIVER WHERE DRIVER\_ID = 105;



3. **Check his/her average rating**

SELECT DRIVER\_AVG\_RATING FROM DRIVER WHERE DRIVER\_ID = 109;



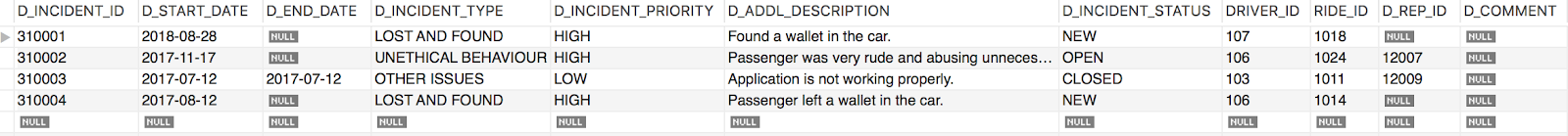
4. **Register an incident to incident table**

INSERT INTO DRIVER\_INCIDENT (D\_INCIDENT\_ID,D\_START\_DATE,D\_END\_DATE, D\_INCIDENT\_TYPE,D\_INCIDENT\_PRIORITY,

D\_ADDL\_DESCRIPTION, D\_INCIDENT\_STATUS, DRIVER\_ID, RIDE\_ID, D\_REP\_ID,D\_COMMENT)

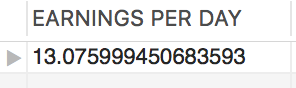
VALUES(310004,'2017-08-12',NULL, 'LOST AND FOUND','HIGH', 'Passenger left a wallet in the car.', 'NEW', '106','1014', NULL,NULL);

SELECT \* FROM DRIVER\_INCIDENT;



5. **Check his earnings for the day**

SELECT SUM(RIDE\_COST\*0.60 + RIDE\_TIP) AS 'EARNINGS PER DAY' FROM RIDE R JOIN CAR C ON C.CAR\_ID = R.CAR\_ID WHERE C.DRIVER\_ID = 109;



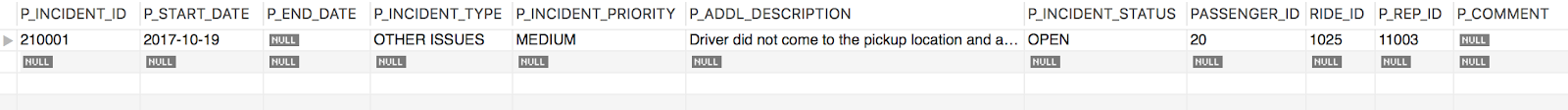
**Passenger Support:**

1. **See all the incidents lodged by a passenger**

SELECT \*

FROM PASSENGER\_INCIDENT

WHERE PASSENGER\_ID = 20;



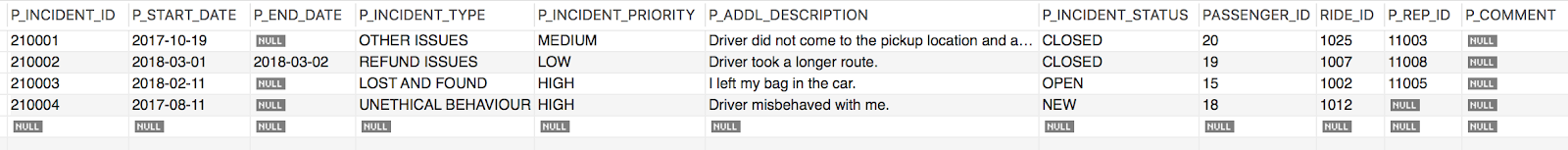
2. **Can change the status of an incident lodged**

UPDATE PASSENGER\_INCIDENT

SET P\_INCIDENT\_STATUS = 'CLOSED'

WHERE P\_INCIDENT\_ID = 210001;

SELECT \* FROM PASSENGER\_INCIDENT;

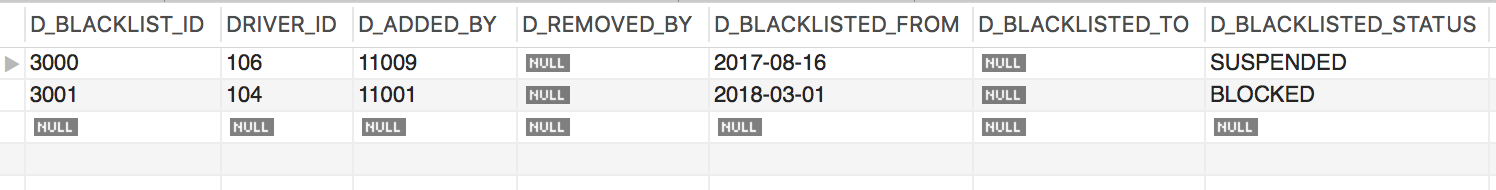


3. **Add/remove a driver from Blacklist**

**ADD-**

INSERT INTO DRIVER\_BLACKLIST(D\_BLACKLIST\_ID, DRIVER\_ID, D\_ADDED\_BY, D\_REMOVED\_BY, D\_BLACKLISTED\_FROM, D\_BLACKLISTED\_TO, D\_BLACKLISTED\_STATUS) VALUES (3001, 104, 11001, NULL, '2018-03-01', NULL, 'BLOCKED');

SELECT \* FROM DRIVER\_BLACKLIST;



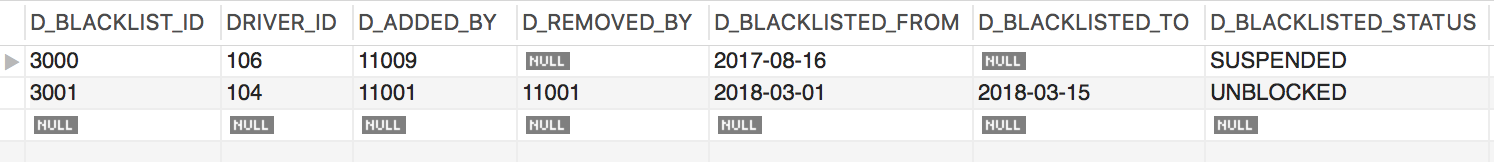
**REMOVE** -

UPDATE DRIVER\_BLACKLIST

SET D\_BLACKLISTED\_STATUS = 'UNBLOCKED' , D\_REMOVED\_BY = 11001, D\_BLACKLISTED\_TO = '2018-03-15'

WHERE DRIVER\_ID = 104 AND D\_BLACKLISTED\_STATUS = 'SUSPENDED' OR D\_BLACKLISTED\_STATUS= 'BLOCKED';

SELECT \* FROM DRIVER\_BLACKLIST;

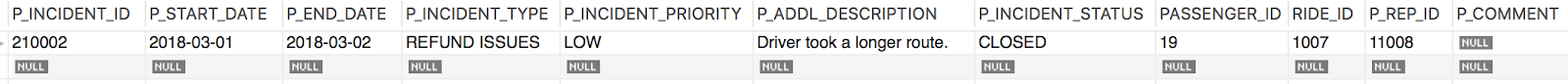


4. **Refund Ride Cost back to the Passenger in case of Refund Issues:**

**A refund issue initiated by the Passenger:**

P\_INCIDENT\_ID: 210002

SELECT \* FROM PASSENGER\_INCIDENT WHERE P\_INCIDENT\_ID = 210002



START TRANSACTION;

UPDATE PASSENGER\_INCIDENT SET P\_COMMENT= 'REFUNDED $10.23 TO THE PASSENGER',

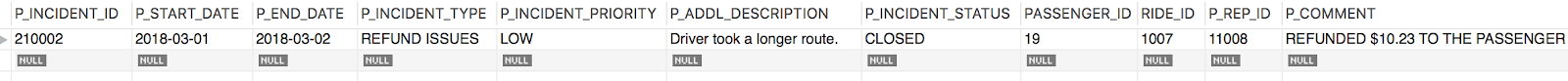
P\_INCIDENT\_STATUS = 'CLOSED'

WHERE P\_INCIDENT\_ID= 210002;

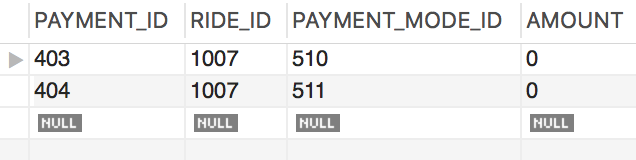
UPDATE PAYMENT SET AMOUNT = 0.00 WHERE RIDE\_ID = 1007;

COMMIT;

SELECT \* FROM PASSENGER\_INCIDENT WHERE P\_INCIDENT\_ID = 210002



SELECT \* FROM PAYMENT WHERE RIDE\_ID = 1007;



**Driver Support:**

**1. Add a passenger to Blacklist**

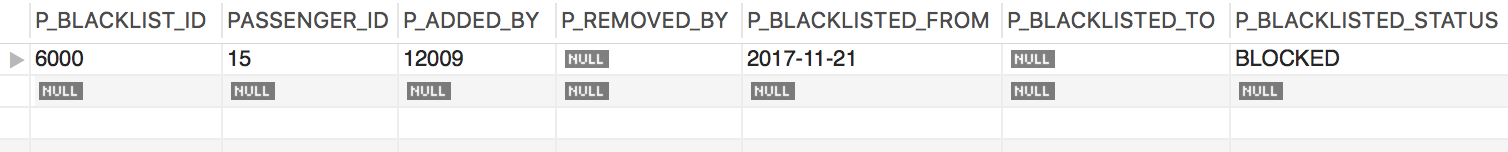
INSERT INTO PASSENGER\_BLACKLIST (P\_BLACKLIST\_ID, PASSENGER\_ID, P\_ADDED\_BY, P\_REMOVED\_BY, P\_BLACKLISTED\_FROM, P\_BLACKLISTED\_TO, P\_BLACKLISTED\_STATUS)

VALUES (6000, 15,12009, NULL, '2017-11-21', NULL, 'BLOCKED'

FROM PASSENGER\_BLACKLIST B1

WHERE NOT EXISTS (SELECT \* FROM PASSENGER\_BLACKLIST B2 WHERE B1.P\_BLACKLIST\_ID= B2.P\_BLACKLIST\_ID);

SELECT \* FROM PASSENGER\_BLACKLIST;

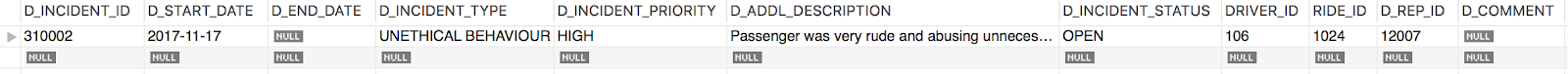


**2. Raise an incident.**

INSERT INTO DRIVER\_INCIDENT (D\_INCIDENT\_ID, D\_START\_DATE, D\_END\_DATE, D\_INCIDENT\_TYPE, D\_INCIDENT\_PRIORITY, D\_ADDL\_DESCRIPTION, D\_INCIDENT\_STATUS, DRIVER\_ID, RIDE\_ID, D\_REP\_ID, D\_COMMENT)

VALUES (310002, '2017-11-17', NULL, 'UNETHICAL BEHAVIOUR', 'HIGH', 'Passenger was very rude and abusing unnecessarily.', 'OPEN', '106', '1024', 12007, NULL);

SELECT \* FROM DRIVER\_INCIDENT WHERE D\_INCIDENT\_ID = 310002;



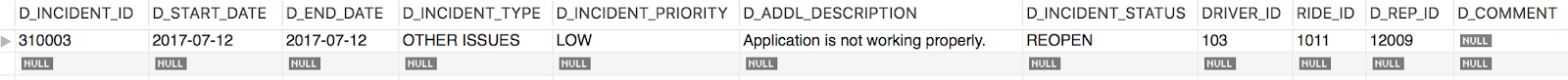
**3. Reopen a closed Incident**

UPDATE DRIVER\_INCIDENT

SET D\_INCIDENT\_STATUS = 'REOPEN'

WHERE D\_INCIDENT\_ID = 310003;

SELECT \* FROM DRIVER\_INCIDENT WHERE D\_INCIDENT\_ID = 310003;

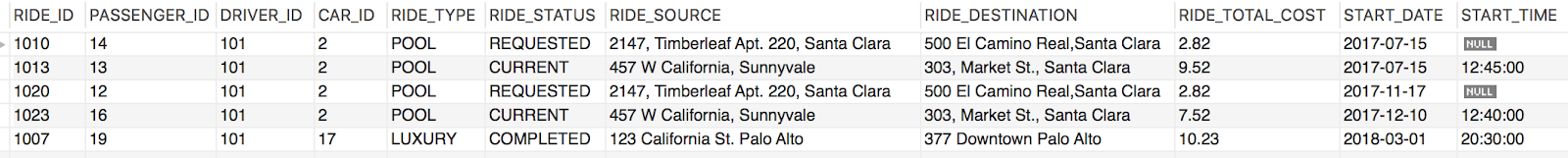


**4. See all the rides of a driver.**

SELECT R.RIDE\_ID, R.PASSENGER\_ID, C.DRIVER\_ID, C.CAR\_ID, R.RIDE\_TYPE, R.RIDE\_STATUS, R.RIDE\_SOURCE, R.RIDE\_DESTINATION, R.RIDE\_TOTAL\_COST, R.START\_DATE, R.START\_TIME

FROM RIDE R,CAR C

WHERE R.CAR\_ID=C.CAR\_ID AND C.DRIVER\_ID =101 ;



# Indexes

CREATE UNIQUE INDEX RIDE ON RIDE(RIDE\_ID);

CREATE UNIQUE INDEX PASSENGER ON PASSENGER(PASSENGER\_ID);

CREATE UNIQUE INDEX DRIVER ON DRIVER(DRIVER\_ID);

# View

**View 1**

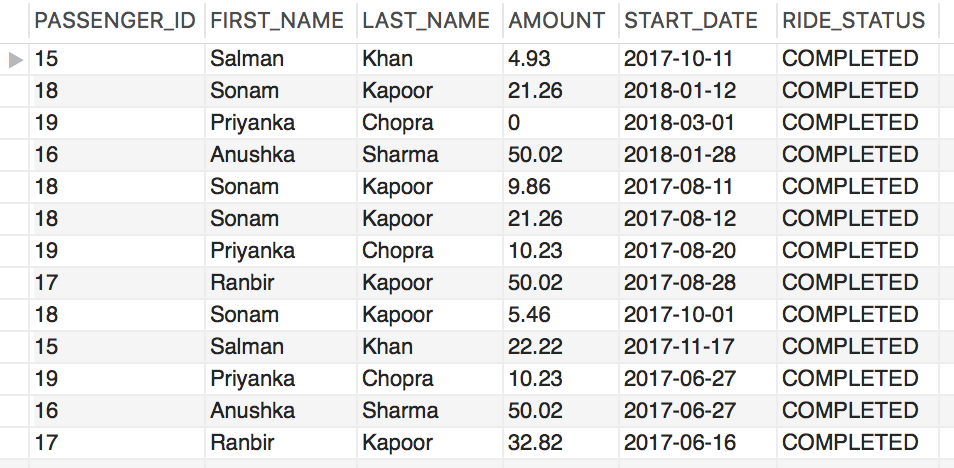
CREATE VIEW **PAYMENT\_INFO** AS

SELECT DISTINCT P.PASSENGER\_ID,P.FIRST\_NAME,P.LAST\_NAME, PA.AMOUNT,R.START\_DATE,R.RIDE\_STATUS

FROM RIDE R, PAYMENT PA, PASSENGER P

WHERE PA.RIDE\_ID= R.RIDE\_ID AND P.PASSENGER\_ID = R.PASSENGER\_ID AND R.RIDE\_STATUS = 'COMPLETED';

SELECT \* FROM PAYMENT\_INFO;

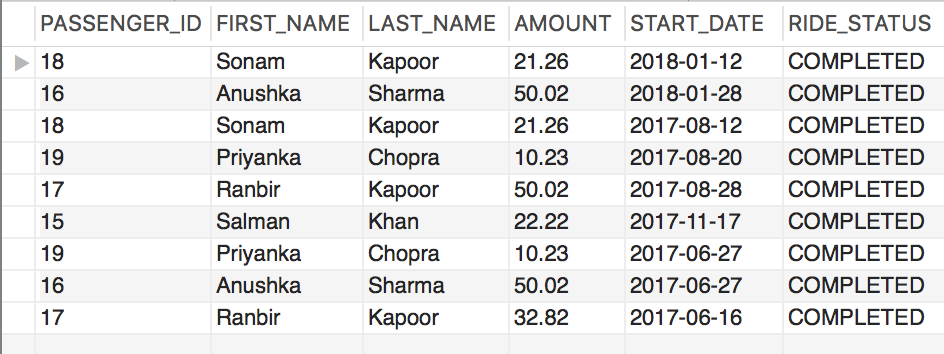


**Find the passenger who has paid more than $10 for a ride**

SELECT \*

FROM PAYMENT\_INFO

WHERE AMOUNT>10.00;

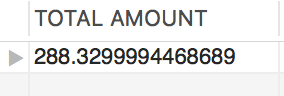


**Find the total amount collected by the passengers in the year 2017?**

SELECT SUM(AMOUNT) AS 'TOTAL AMOUNT'

FROM PAYMENT\_INFO

WHERE START\_DATE > 2016-12-31 & START\_DATE <2018-01-01;



**VIEW 2:**

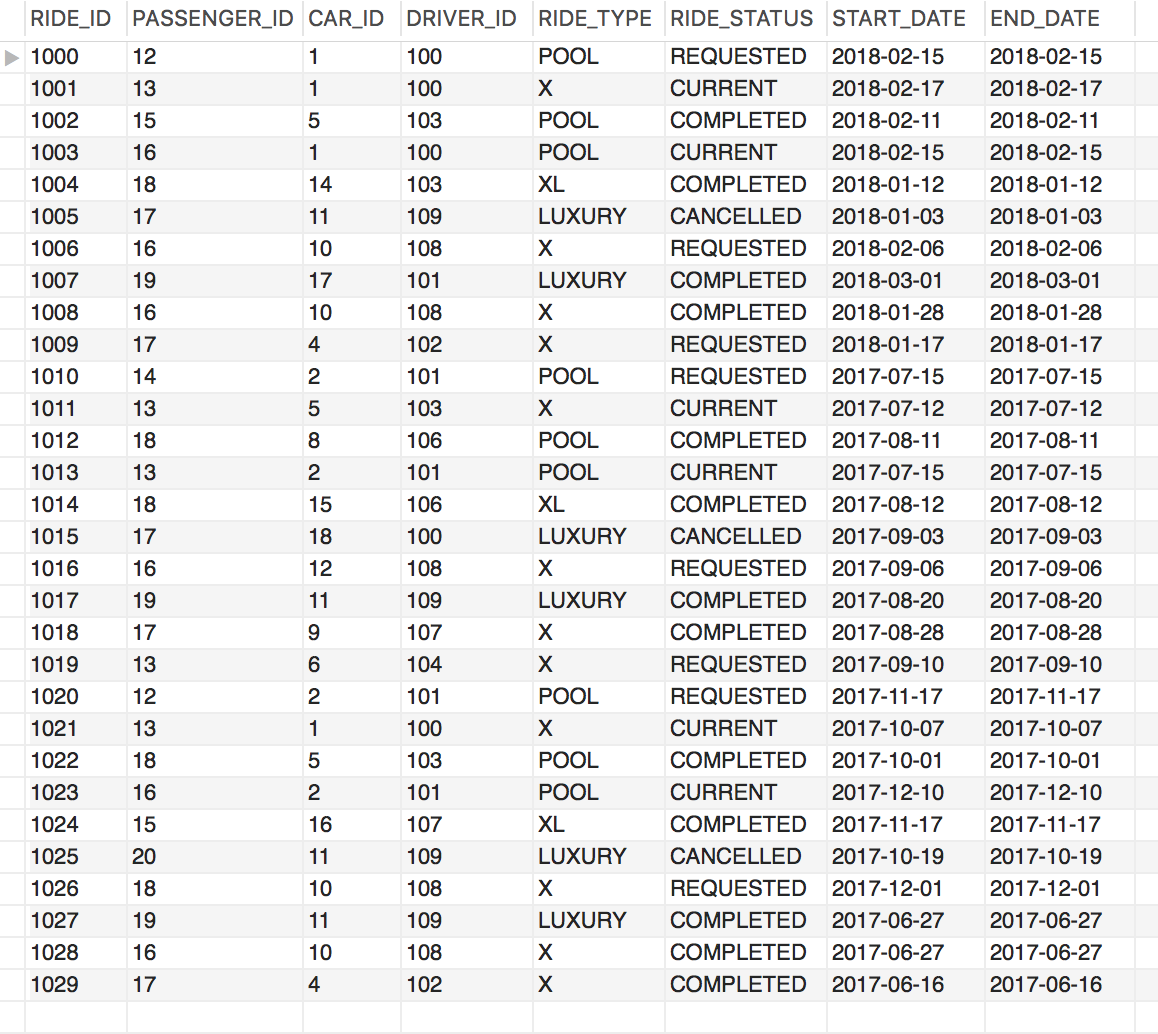
CREATE VIEW **RIDE\_INFO** AS

SELECT R.RIDE\_ID,R.PASSENGER\_ID,C.CAR\_ID,C.DRIVER\_ID,R.RIDE\_TYPE, R.RIDE\_STATUS, R.START\_DATE, R.END\_DATE

FROM RIDE R,CAR C

WHERE C.CAR\_ID = R.CAR\_ID;

SELECT \* FROM RIDE\_INFO;



**Find rides associated with DRIVER\_ID =101**

SELECT \*

FROM RIDE\_INFO

WHERE DRIVER\_ID =101;

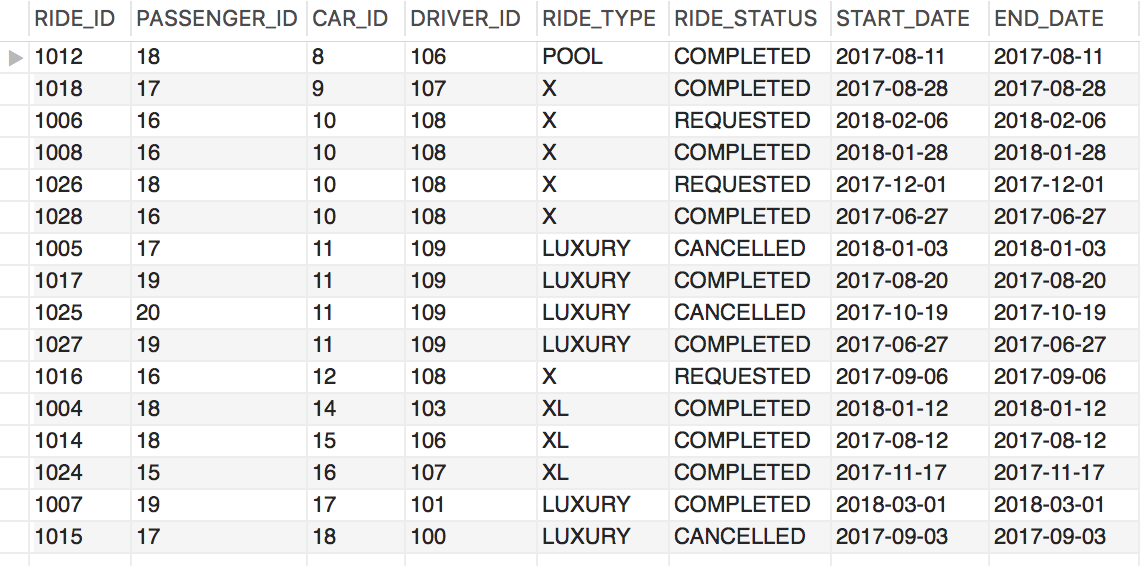


**Find rides where car id is greater than 6**

SELECT \*

FROM RIDE\_INFO

WHERE CAR\_ID > 6;



**VIEW 3:**

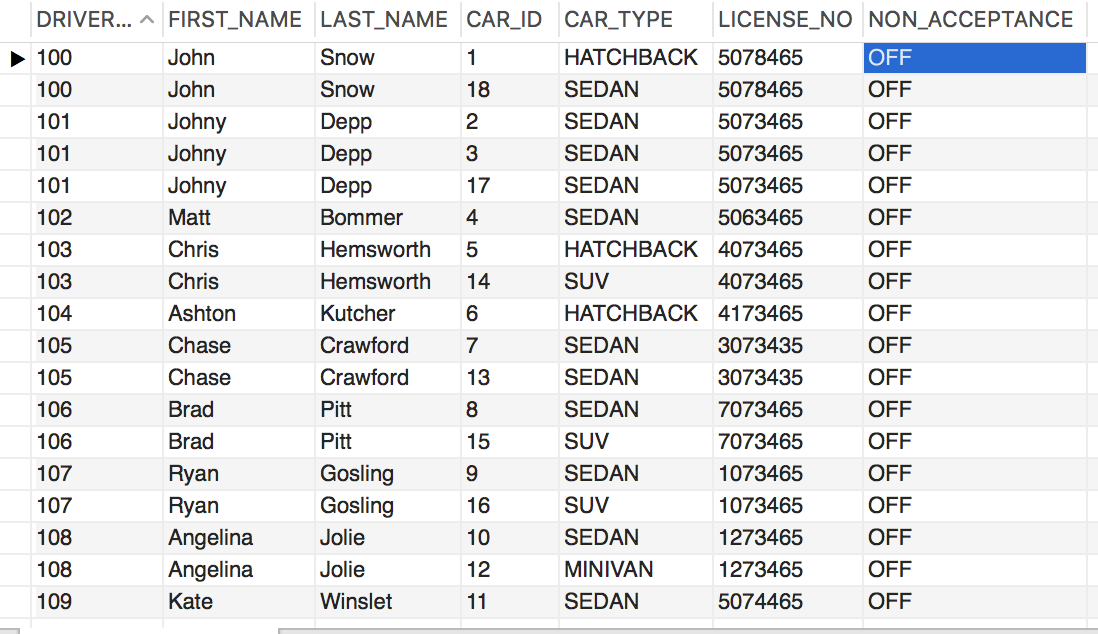
CREATE VIEW **DRIVER\_INFO** AS

SELECT D.DRIVER\_ID, D.FIRST\_NAME, D.LAST\_NAME, C.CAR\_ID,C.CAR\_TYPE, D.LICENSE\_NO,D.NON\_ACCEPTANCE

FROM DRIVER D, CAR C

WHERE C.DRIVER\_ID = D.DRIVER\_ID;

SELECT \* FROM DRIVER\_INFO;



**Find the details of drivers who have more than 1 sedan**

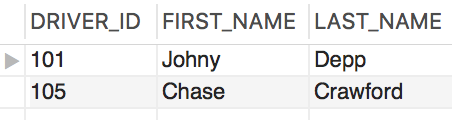
SELECT DRIVER\_ID,FIRST\_NAME,LAST\_NAME

FROM DRIVER\_INFO

WHERE CAR\_TYPE ='SEDAN'

GROUP BY DRIVER\_ID

HAVING COUNT(CAR\_TYPE ) >1;



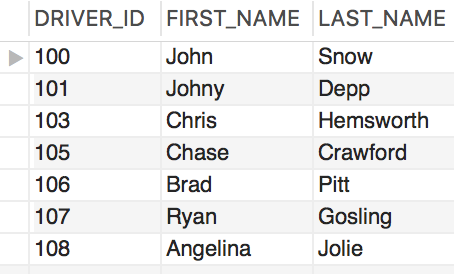
**Find the details of driver have more than 1 car registered.**

SELECT DRIVER\_ID,FIRST\_NAME,LAST\_NAME

FROM DRIVER\_INFO

GROUP BY DRIVER\_ID

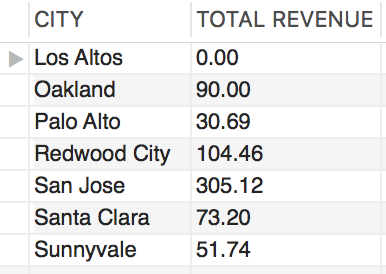
HAVING COUNT(CAR\_TYPE ) >1;



# Metrics

1. **Analyze the total revenue by city.**

SELECT SOURCE\_CITY AS 'CITY', CONVERT(SUM(RIDE\_TOTAL\_COST),DECIMAL(7,2)) AS 'TOTAL REVENUE' FROM RIDE GROUP BY SOURCE\_CITY;



1. **Find total number of rides completed daily, monthly, yearly.**

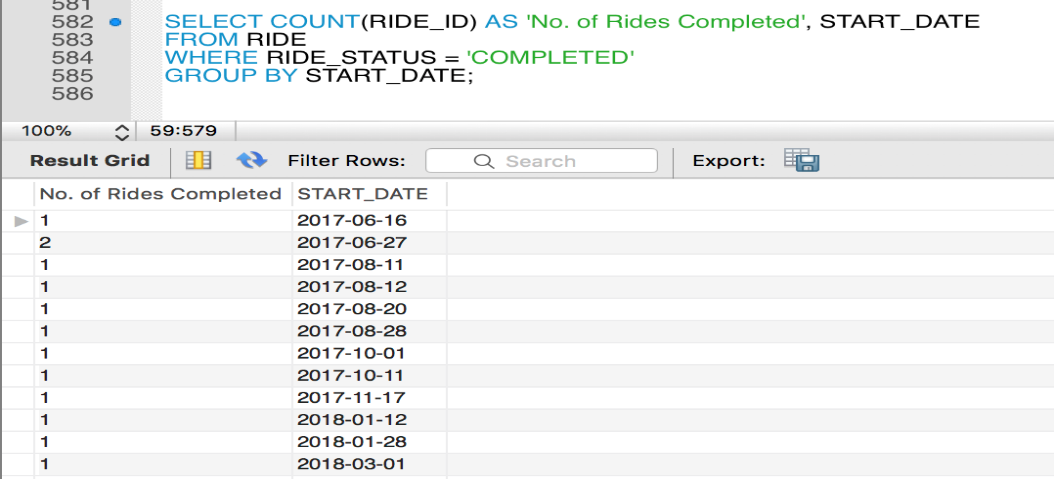
(DAILY)

SELECT COUNT(RIDE\_ID) AS 'No. of Rides Completed', START\_DATE

FROM RIDE

WHERE RIDE\_STATUS = 'COMPLETED'

GROUP BY START\_DATE;

****

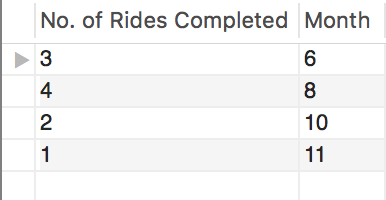
(MONTHLY)

SELECT COUNT(RIDE\_ID) AS 'No. of Rides Completed', MONTH(START\_DATE) AS 'Month'

FROM RIDE

WHERE RIDE\_STATUS = 'COMPLETED' AND YEAR(START\_DATE) = 2017

GROUP BY MONTH(START\_DATE);



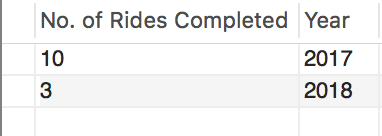
(YEARLY)

SELECT COUNT(RIDE\_ID) AS 'No. of Rides Completed', YEAR(START\_DATE) AS 'Year'

FROM RIDE

WHERE RIDE\_STATUS = 'COMPLETED'

GROUP BY YEAR(START\_DATE);

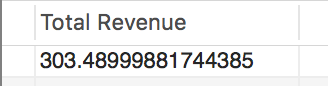
****

1. **Find overall revenue for 2017-18**

SELECT SUM(RIDE\_TOTAL\_COST) AS 'Total Revenue'

FROM RIDE

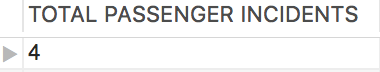
WHERE RIDE\_STATUS = 'COMPLETED';

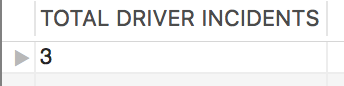
****

1. **Analyze the number of incidents reported by passengers and drivers.**

SELECT COUNT(P\_INCIDENT\_ID) FROM PASSENGER\_INCIDENT;

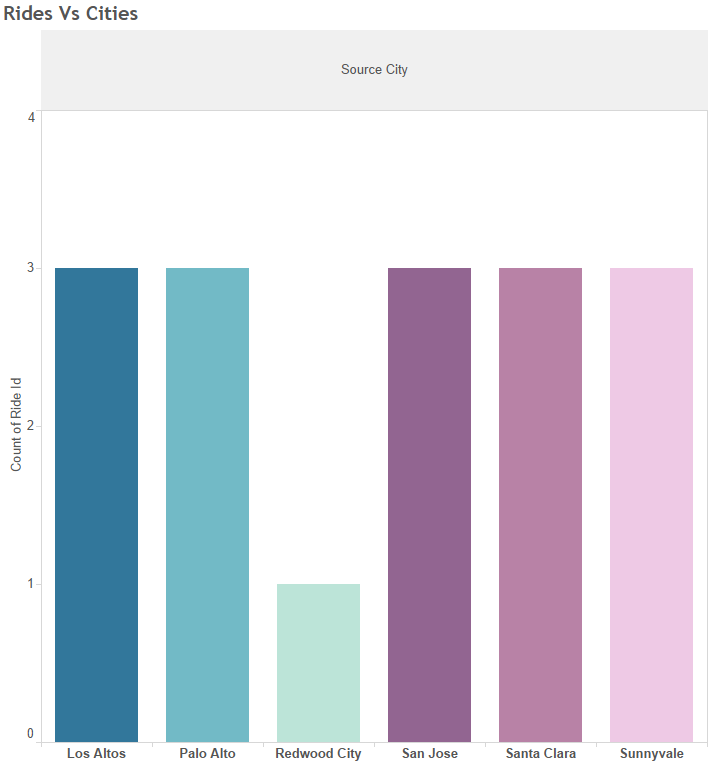
SELECT COUNT(D\_INCIDENT\_ID) FROM DRIVER\_INCIDENT;



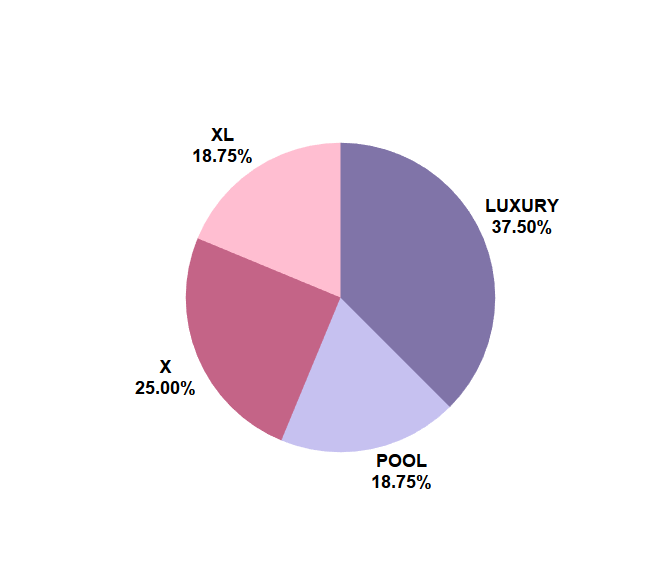
****

**Visualizations:**

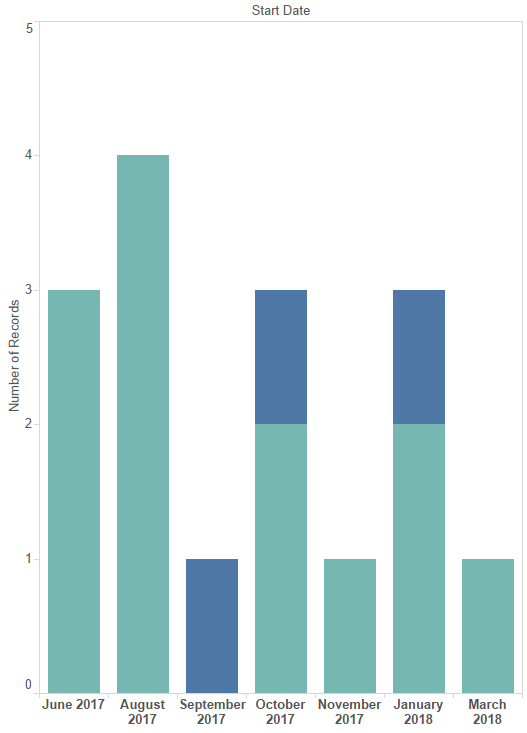
* 1. **Analyze high demand areas/ cities**

****

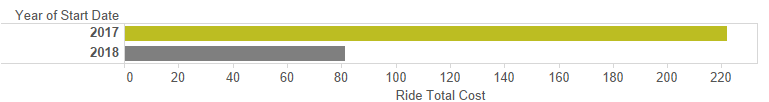
* 1. **Find high demand on ride type**

****

* 1. **Find the total number of rides completed monthly**



* 1. **Analyze overall revenue growth yearly**

****

# Project Summary

**Summarize your experience with this exercise**

We developed the application Hop-On to provide cheap and efficient solution to passengers who want a ride to their destination, drivers who wish to share their rides while earning extra money on the go along with reducing the traffic and accidents on roads and collectively help in reducing the carbon footprint. We have realized that managing data efficiently is an important step in running an organization. While we were designing the database for this application we understood how there can be no limits to what the application can offer and how we can include maximum functionalities into our model.

It was a great learning experience and applying what is taught in the class to an actual business scenario is a wonderful opportunity.

This project has enhanced our understanding of Database Management which will help us in our future challenges.

**What was the hardest part of this project?**

The hardest part of the project according to us was finalizing the UML model in accordance to our business needs. We added the basic features for the app and some complex features including blacklisting but as we moved further into our project we realized there is no end to the features and requirements that we can cater through the database. In addition to that adding attributes to the respective tables based on our use-cases was challenging.

**What did you learn from the project?**

1. We learnt how to create a database schema step by step and how important it is to thoroughly understand each step and then move to the next step.
2. It is very important to understand how to store data efficiently, as while you store data you need to determine its datatype. Choosing the right datatype improves both performance and data integrity.
3. We learnt how to apply our knowledge of DBMS to a real time scenario by understanding the application and its scenarios.
4. Before we start building a database for our application it is very important to understand and create relationships and cardinalities among different entities. Identifying the correct relationships between the entities helps us build an anomaly-free database.

**What problems did you run against in this project?**

1. We encountered a problem while applying the non-nullability constraints for time related attributes.
2. We handled refund scenarios across related tables in the database which was challenging.
3. Currently we have mapped 1 ride id to 1 passenger for a ride, but we faced a problem while mapping a pool ride to all the passengers sharing that ride.

**How did you solve these problems?**

1. We had brainstorming sessions to understand where to apply non nullability constraints.
2. We created a transaction for refund type scenarios as there were updates on multiple tables.
3. Each passenger has a unique RIDE ID even in case of pool

**If you were to do this project again, what methodology would you follow?**

1. Currently we have inserted each ID manually, but if we were to do this project again we would autoincrement the IDs
2. We would add separate table to handle refund requests & credits
3. To address the problem of mapping a pool ride to all the passengers sharing that ride we can add a concept of master ride to handle pool and X rides separately.

**Suggestions how to refine this project for the next class**

To refine this project for the next class we would like to add long distance ride sharing services, connecting big cities like San Francisco to Los Angeles etc. Also, we would try to finalize the requirements and scope of the project such that any further change in the requirements will not result in a lot of rework.