CHAPTER – 1

INTODUCTION

1.1 Introduction:

Rapido is a popular on-demand bike taxi service that operates in several cities across India. It provides a convenient and affordable means of transportation for short distances, particularly in urban areas with heavy traffic congestion. The service is accessed through a user-friendly mobile application, making it easy for customers to book a ride with just a few taps on their smartphones.

One of the key features that sets Rapido apart is its use of two-wheelers as a mode of transport. By utilizing motorcycles and scooters, Rapido is able to navigate through crowded streets and reach destinations quickly, often avoiding the traffic jams that plague larger vehicles. This not only saves time for riders but also reduces the overall environmental impact.

Rapido functions as a peer-to-peer platform, connecting users in need of a ride with nearby bike riders who are registered as Rapido Captains. These Captains are independent contractors who have undergone background checks and verification processes to ensure the safety of passengers. The company emphasizes the importance of driver safety and provides insurance coverage for both the Captains and passengers.

The affordability of Rapido is another significant advantage. The service offers competitive pricing, often costing less than traditional taxi services or auto-rickshaws. This affordability makes it an attractive option for daily commuters, students, and individuals looking for an efficient and cost-effective way to travel short distances.

Rapido has steadily expanded its presence since its inception, operating in numerous cities across India. It continues to gain popularity and has garnered a loyal customer base who appreciate its convenience, affordability, and reliability. With its commitment to safety, ease of use, and focus on providing efficient transportation solutions, Rapido has become a prominent player in the on-demand bike taxi industry in India.

1.2 Problem Statement:

The problem statement of Rapido can be summarized as follows:

Lack of efficient and affordable last-mile transportation: In many urban areas, especially those with heavy traffic congestion, there is a lack of efficient and affordable transportation options for short distances. Traditional taxi services may be expensive or not readily available, while public transportation may not cover the entire route. This creates a need for a convenient and cost-effective solution for last-mile transportation.

Traffic congestion and time efficiency: Traffic congestion is a major issue in many cities, leading to increased travel times and delays. Commuters often struggle to reach their destinations on time, especially during peak hours. There is a need for a transportation service that can navigate through congested areas quickly, ensuring timely arrival and reducing travel time.

Environmental impact: The environmental impact of traditional transportation methods, such as cars and auto-rickshaws, is a growing concern. These vehicles contribute to air pollution and carbon emissions. There is a need for sustainable transportation options that minimize the ecological footprint and promote environmentally friendly practices.

Safety and trustworthiness: Safety is a crucial factor in transportation services. Customers need assurance that the drivers are reliable, responsible, and have undergone proper background checks. There is a need for a platform that ensures the safety of both passengers and drivers, building trust and confidence among users.

Cost-effectiveness: Many commuters seek affordable transportation options, especially for short distances. Traditional taxis or auto-rickshaws can be relatively expensive for frequent or daily commuting. There is a need for a service that offers competitive pricing, making it financially viable for a wide range of users.

Rapido aims to address these problems by providing a convenient, affordable, and timeefficient on-demand bike taxi service. It utilizes two-wheelers to navigate through traffic, offers competitive pricing, ensures safety through driver verification and insurance coverage, and provides an eco-friendly alternative to traditional transportation methods.

1.3 Objectives:

Rapido is a popular Indian bike taxi and ride-hailing platform. While I don't have access to the specific objectives of Rapido beyond my knowledge cutoff in September 2021, I can provide you with the general objectives of such platforms. The common objectives of a bike taxi and ride-hailing service like Rapido include:

Convenient and affordable transportation: The primary objective of Rapido is to offer convenient and cost-effective transportation solutions to commuters. By leveraging bike taxis, Rapido aims to provide a quicker mode of transport, especially for short distances, and at a lower cost compared to traditional taxis or other ride-hailing services.

Enhancing mobility options: Rapido aims to enhance the overall mobility options available to people by offering an alternative mode of transportation. It provides an additional choice to individuals who may not own a vehicle or prefer not to use public transport for various reasons.

Efficient utilization of resources: By utilizing bikes as a means of transport, Rapido aims to optimize resource utilization, specifically reducing traffic congestion and addressing environmental concerns. Bikes are smaller and more maneuverable than cars, allowing them to navigate through traffic more easily, resulting in less congestion on roads.
Job creation and income generation: Rapido offers an opportunity for bike owners to become bike taxi drivers and earn income by providing rides to passengers. This objective includes promoting entrepreneurship and creating employment opportunities for individuals who own bikes but may not have access to traditional job opportunities.

Safety and customer satisfaction: Ensuring the safety and satisfaction of passengers is crucial for Rapido. They aim to maintain a high level of safety standards by screening and verifying the bike taxi drivers, providing insurance coverage, and implementing features like live GPS tracking, SOS buttons, and customer feedback mechanisms to address any concerns and improve the overall experience.

It's important to note that these objectives may evolve over time as the company grows and responds to market dynamics and customer needs. For the most up-to-date information on
Rapido's objectives, it's recommended to refer to official sources or the company's website

CHAPTER – 2

DATABASE DESIGN

2.1 List of attributes, entities, relationship:

1. RAPIDO_USER (USER_ID, NAME, PHONE_NUMBER, EMAIL, PASSWORD,
ADDRESS,DATE_OF_BIRTH, REGISTRATION_DATE, ACCOUNT_STATUS)
2.RIDE (RIDE_ID, USERS_ID, DRIVER_ID, START_LOCATION, RIDE_STATUS, FARE_AMOUNT)
3.DRIVER (DRIVER_ID, NAME, PHONE_NUMBER, EMAIL, PASSWORD, VEHICLE_NUMBER, VEHICLE_TYPE, AVAILABILITY_STATUS)
4.DRIVER_RATING (RATING_ID, DRIVER_ID, RIDE_ID, RATING_DECIMAL, RATING_DATE, FEEDBACK)
5.PAYMENT (USER_ID, RIDE_ID, AMOUNT MONEY, PAYMENT_STATUS)
6.PROMO_CODE (PROMO_CODE_ID, CODE, DISCOUNT DECIMAL, EXPIRY_DATE, MAX_USAGE, USAGE_COUNT)
7.DRIVER_EARNINGS (EARNING_ID,DRIVER_ID,RIDE_ID, EARNING_AMOUNT)

8.RIDE_CATEGORY (CATEGORY_ID, CATEGORY_NAME, BASE_FARE DECIMAL, DISTANCE_RATE DESICIMAL, TIME_RATE DECIMAL)

9.DRIVER_DOCUMENT (DOCUMENT_ID, DOCUMENT_TYPE, DOCUMENT_NUMBER, EXPIRY_DATE)

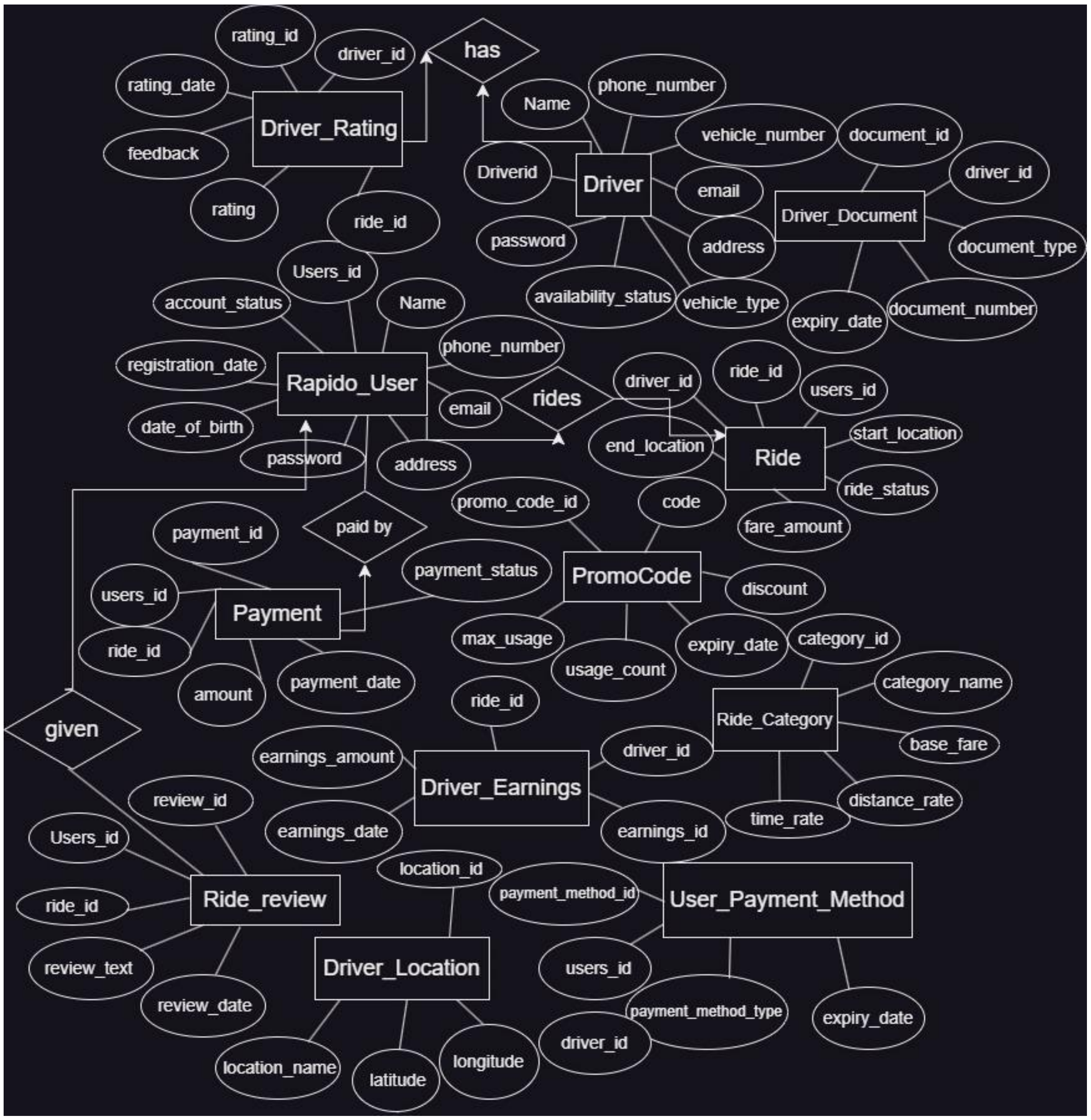
10. USER_PAYMENT_METHOD (PAYMENT_METHOD_ID, USER_ID, PAYMENT_METHOD_TYPE, EXPIRY_DATE, RAPIDO_USER)

11.RIDE_REVIEW (REVIEW_ID, USER_ID, RIDE_ID, REVIEW_TEXT, REVIEW_DATE)

12.DRIVER_LOCATION (LOCATION_ID, DRIVER_ID, LOCATION_NAME, LATITUDE DECIMAL, LONGITUDE DECIMAL)

Relationships: has, have, perform, search.

2.2 E-R Diagram:


Driver_Rating
rating_id
driver_id
rating_date
feedback
rating
ride_id
has
Driver
Name
phone_number
vehicle_number
Driverid
email
password
address
availability_status
vehicle_type
Driver_Document
document_id
driver_id
document_type
expiry_date
document_number
Rapido_User
Users_id
account_status
Name
phone_number
registration_date
date_of_birth
password
email
address
rides
Ride
driver_id
ride_id
users_id
start_location
end_location
ride_status
fare_amount
paid by
Payment
payment_id
users_id
ride_id
amount
payment_date
payment_status
PromoCode
promo_code_id
code
discount
max_usage
usage_count
expiry_date
Ride_Category
category_id
category_name
base_fare
distance_rate
time_rate
Driver_Earnings
ride_id
earnings_amount
earnings_date
driver_id
earnings_id
given
Ride_review
review_id
Users_id
ride_id
review_text
review_date
Driver_Location
location_id
location_name
latitude
longitude
User_Payment_Method
payment_method_id
users_id
payment_method_type
driver_id
expiry_date

3.1 Database Languages:

CHAPTER – 3

RELATIONAL MODEL

1. Data definition language (DDL)

Data definition language (DDL) creates the framework of the database by specifying the database schema, which is the structure that represents the organization of data. Its common uses include the creation and alteration of tables, files, indexes and columns within the database. This language also allows users to rename or drop the existing database or its components.
Here's a list of DDL statements:
• CREATE: Creates a new database or object, such as a table, index or column.
• ALTER: Changes the structure of the database or object.
• DROP: Deletes the database or existing objects.
• RENAME: Renames the database or existing objects.

2. Data manipulation language (DML)

Data manipulation language (DML) provides operations that handle user requests, offering a way to access and manipulate the data that users store within a database. Its common functions include inserting, updating and retrieving data from the database.

Here's a list of DML statements:

- INSERT: Adds new data to the existing database table.
- UPDATE: Changes or updates values in the table.
- DELETE: Removes records or rows from the table.
- SELECT: Retrieves data from the table or multiple tables.

3. Data control language (DCL)

Data control language (DCL) controls access to the data that users store within a database. Essentially, this language controls the rights and permissions of the database system. It allows users to grant or revoke privileges to the database.
Here's a list of DCL statements:

- GRANT: Gives a user access to the database.
- REVOKE: Removes a user's access to the database.

4. Transaction control language (TCL)

Transaction control language (TCL) manages the transactions within a database. Transactions group a set of related tasks into a single, executable task. All the tasks must succeed in order for the transaction to work. Here's a list of TCL statements:

- COMMIT: Carries out a transaction.
- ROLLBACK: Restores a transaction if any tasks fail to execute.

3.2 Table Description:

Here's a description of each table:

1. RAPIDO_USER: The "Rapido_User" table is a representation of users in the Rapido system, storing their information such as name, phone number, email, password, address, date of birth, registration date, and account status. It serves as a central data source for managing user profiles and facilitating various operations within the Rapido service.
2. RIDE: The "Ride" table represents a ride in the system, storing information about each ride such as its unique ride ID, the IDs of the associated user and driver, the starting and ending locations, the ride status, and the fare amount. This table enables tracking and managing ride data, allowing for efficient retrieval of ride details, monitoring ride status, and calculating fare amounts for billing purposes.
3. DRIVER: The "Driver" table has been created with seven columns: Driverid, Name, phone_number, email, password, vehicle_number, and vehicle_type. Each column has specific data types and constraints such as primary key, not null, and character limits. Additionally, the table includes the "availability_status" column to track the availability status of the driver.
4. DRIVER_RATING: The "Driver_Rating" table has been created with six columns: rating_id, driver_id, ride_id, rating, rating_date, and feedback. The table includes a primary key constraint on the rating_id column and foreign key constraint on the driver_id column referencing the Driver table's Driverid column. The rating column is of type DECIMAL(3, 2) with a check constraint ensuring the value is between 1 and 5. The table also includes the rating_date column of type DATE and the feedback column of type TEXT, both marked as not null.
5. PAYMENT: The "Payment" table has been created with six columns: payment_id, users_id, ride_id, amount, payment_date, and payment_status. The payment_id column is set as the primary key. The table includes foreign key constraints on the users_id and ride_id columns,

referencing the respective tables. The amount column is of type money, representing the payment amount, and the payment_date column stores the date of the payment. Additionally, the payment_status column tracks the status of the payment, ensuring it is not null.

6. PROMOCODE: The "PromoCode" table has been created with six columns: promo_code_id, code, discount, expiry_date, max_usage, and usage_count. The promo_code_id column is designated as the primary key. The code column stores the alphanumeric code for the promo code, while the discount column holds the percentage discount value in decimal format. The expiry_date column tracks the expiration date of the promo code, and the max_usage column represents the maximum number of times the code can be used. The usage_count column keeps track of the number of times the promo code has been used.
7. DRIVER_EARNING:The "Driver_Earnings" table has been created with five columns: earnings_id, driver_id, ride_id, earnings_amount, and earnings_date. The earnings_id column is designated as the primary key. The table includes foreign key constraints on the driver_id and ride_id columns, referencing the Driver and Ride tables, respectively. The earnings_amount column stores the monetary amount earned by the driver for a specific ride, and the earnings_date column records the date of the earnings.
8. RIDE_CATEGORY: The "Ride_Category" table has been created with five columns: category_id, category_name, base_fare, distance_rate, and time_rate. The category_id column is set as the primary key. The table stores different ride categories with their corresponding names, base fares, distance rates, and time rates. The base_fare column represents the fixed cost of a ride, while the distance_rate and time_rate columns denote the rates applied to the distance traveled and time spent during the ride, respectively.
9. DRIVER_DOCUMENT: The "Driver_Document" table has been created with five columns: document_id, driver_id, document_type, document_number, and expiry_date. The document_id column is set as the primary key. The table represents the documents associated with drivers, storing information such as document type, document number, and expiry date. It includes a foreign key constraint on the driver_id column, referencing the Driver table's Driverid column.
10. USER_PAYMENT _METHOD: The "User_Payment_Method" table has been created with four columns: payment_method_id, users_id, payment_method_type, and expiry_date. The payment_method_id column is designated as the primary key. This table represents the payment methods associated with users, storing information such as the payment method type and its expiry date. It includes a foreign key constraint on the users_id column, referencing the Rapido_User table's Users_id column.
11. RIDE_REVIEW: The "Ride_review" table has been created with five columns: review_id, Users_id, ride_id, review_text, and review_date. The review_id column is designated as the primary key. This table represents the reviews given by users for specific rides, storing information such as the user's ID, ride ID, the text of the review, and the date of the review. It includes foreign key constraints on the Users_id and ride_id columns, referencing the Rapido_User and Ride tables, respectively.
12. DRIVER_LOCATION: The "Driver_Location" table has been created with five columns: location_id, driver_id, location_name, latitude, and longitude. The location_id column is set as the primary key. This table is used to store the location information of drivers, including their driver ID, location name, latitude, and longitude coordinates. It also includes a foreign key constraint on the driver_id column, referencing the Driver table's Driverid column.

3.3 Relational Database Schema:

1. **Entity Name:**RAPIDO_USER

ATTRIBUTE	DATA TYPE	CONSTRAINTS
USER_ID	INT	PRIMARY KEY
NAME	VARCHAR	NOT NULL
PHONE_NUMBER	VARCHAR	NOT NULL
EMAIL	VARCHAR	NOT NULL
PASSWORD	VARCHAR	NOT NULL
ADDRESS	VARCHAR	NOT NULL
DATE_OF_BIRTH	DATE	NOT NULL
ACCOUNT_STATUS	DATE	NOT NULL

2. **Entity Name:**RIDE

ATTRIBUTE	DATA TYPE	CONSTRAINTS
RIDE_ID	INT	PRIMARY KEY
USER_ID	INT	NOT NULL
DRIVER_ID	INT	NOT NULL
START_LOCATION	VARCHAR	NOT NULL
END_LOCATION	VARCHAR	NOT NULL
RIDE_STATUS	VARCHAR	NOT NULL
FARE_AMOUNT	DECIMAL	NOT NULL

3. **Entity Name:**DRIVER

ATTRIBUTE	DATA TYPE	CONSTRAINTS
DRIVER_ID	INT	PRIMARY KEY
NAME	VARCHAR(35)	NOT NULL
PHONE_NUMBER	VARCHAR(20)	NOT NULL
EMAIL	VARCHAR(50)	NOT NULL
PASSWORD	VARCHAR(25)	NOT NULL
VEHICLE_NUMBER	VARCHAR(20)	NOT NULL
VEHICLE_TYPE	VARCHAR(50)	NOT NULL
AVAILABILITY	VARCHAR(20)	NOT NULL

4. **Entity Name:**DRIVER_RATING

ATTRIBUTE	DATA TYPE	CONSTRAINTS
RATING_ID	INT	NOT NULL
DRIVER_ID	INT	NOT NULL

RIDE_ID	INT	NOT NULL
RATING	DECIMAL	NOT NULL
RATINF_DATE	DATE	NOT NULL
FEEDBACK	TEXT	NOT NULL

5. **Entity Name:**PAYMENT

ATTRIBUTE	DATA TYPE	CONSTRAINTS
PAYMENT_ID	INT	PRIMARY KEY
USER_ID	INT	NOT NULL
RIDE_ID	INT	NOT NULL
AMOUNT	MONEY	NOT NULL
PAYMENT_DATE	DATE	NOT NULL
PAYMENT_STATUS	VARCHAR	NOT NULL

6. **Entity Name:**PROMOCODE

ATTRIBUTE	DATA TYPE	CONSTRAINTS
PROMO_CODE_ID	INT	PRIMARY KEY
CODE	VARCHAR	NOT NULL
DISCOUNT	DECIMAL	NOT NULL
EXPIRY_DATE	DATE	NOT NULL
MAX_USAGE	INT	NOT NULL
USAGE_COUNT	INT	NOTNULL

7. **Entity Name:**DRIVER_EARNINGS

ATTRIBUTE	DATA TYPE	CONSTRAINTS
EARNING_ID	INT	PRIMARY KEY
DRIVER_ID	INT	NOT NULL
RIDE_ID	INT	NOT NULL
EARNINGS_AMOUNT	DECIMAL	NOT NULL
EARNING_DATE	DATE	NOT NULL

8. **Entity Name:**RIDE_CATEGORY

ATTRIBUTE	DATA TYPE	CONSTRAINTS
CATEGORY_ID	INT	PRIMARY KEY
CATEGORY_NAME	VARCHAR	NOT NULL
BASE-FARE	DECIMAL	NOTNULL
DISTENCE_RATE	DECIMAL	NOT NULL
TIME_RATE	DECIMAL	NOT NULL

9. **Entity Name:**DRIVER _DOCUMENT

ATTRIBUTE	DATA TYPE	CONSTRAINTS CONSTRAINTS
DOCUMENT_ID	INT	PRIMARY KEY
DRIVER_ID	INT	NOT NULL
DOCUMENT_TYPE	VARCHAR	NOT NULL
DOCUMENT_NUMBER	VARCHAR	NOT NULL
EXPIRY_DATE	DATE	NOT NULL

10. **Entity Name:**USER_PAYMENT_METHOD

ATTRIBUTE	DATA TYPE	CONSTRAINTS
PAYMENT_METHOD_ID	INT	PRIMARY KEY
USER_ID	INT	NOT NULL
PAYMENT METHOD_TYPE	VARCHAR	NOT NULL
EXPIRY_DATE	DATE	NIOT NULL

11. **Entity Name**:RADE_REVIEW

ATTRIBUTE	DATA TYPE	CONSTRAINTS
REVIEW_ID	INT	PRIMARY KEY
USER_ID	INT	NOT NULL
RIDE_ID	INT	NOT NULL
REVIEW_TEXT	VACHAT	NOT NULL
REVIEW_DATE	DATE	NOT NULL

12. **Entity Name:**DRIVER_LOCATION

ATTRIBUTE	DATA TYPE	CONSTRAINTS
LOCATION_ID	INT	PRIMARY BKEY
LOCATION_NAME	VARCHAR	NOT NULL
DROVER_ID	INT	NOT NULL
LATITUDE	DECIMAL	NOT NULL
LONGITUTE	DECIMAL	NOT NULL

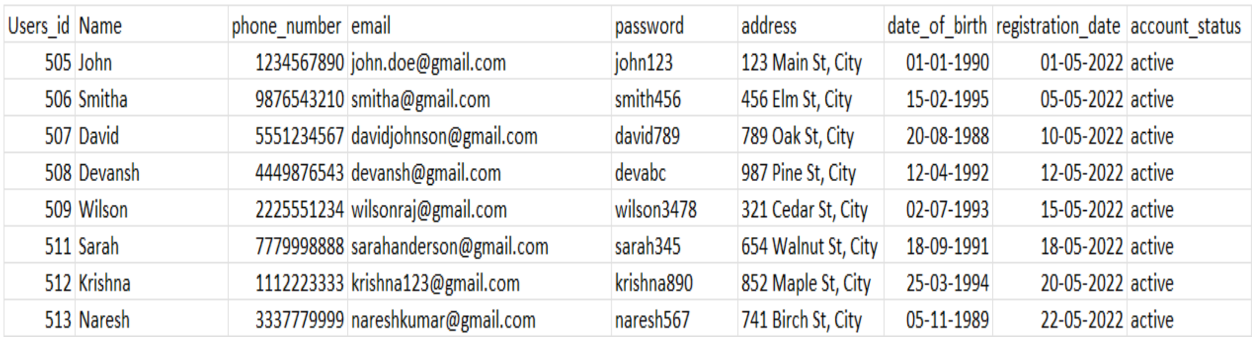
3.4 Relational Queries

- Creation of rapido :
  Create database rapido
- Creation of Tables for rapido :

TABLE:RAPIDO_USER

```
create database rapido_;

create table Rapido_User

(

   Users_id int  primary key,

   Name varchar(50) not null,

   phone_number varchar(10) not null,

   email varchar(50) not null,

   password varchar(25) not null,

   address varchar(20) not null,

   date_of_birth date not null,

   registration_date date not null,

   account_status varchar(20) not null

)

insert into Rapido_User (Users_id, Name, phone_number, email, password, address,
date_of_birth, registration_date, account_status)

values

   (505, 'John ', '1234567890', 'john.doe@gmail.com', 'john123', '123 Main St, City', '1990-01-01',
'2022-05-01', 'active'),

   (506, 'Smitha', '9876543210', 'smitha@gmail.com', 'smith456', '456 Elm St, City', '1995-02-15',
'2022-05-05', 'active'),

   (507, 'David', '5551234567', 'davidjohnson@gmail.com', 'david789', '789 Oak St, City', '1988-
08-20', '2022-05-10', 'active'),

   (508, 'Devansh', '4449876543', 'devansh@gmail.com', 'devabc', '987 Pine St, City', '1992-04-
12', '2022-05-12', 'active'),

   (509, 'Wilson', '2225551234', 'wilsonraj@gmail.com', 'wilson3478', '321 Cedar St, City', '1993-
07-02', '2022-05-15', 'active'),

   (511, 'Sarah', '7779998888', 'sarahanderson@gmail.com', 'sarah345', '654 Walnut St, City',
'1991-09-18', '2022-05-18', 'active'),

   (512, 'Krishna', '1112223333', 'krishna123@gmail.com', 'krishna890', '852 Maple St, City',
'1994-03-25', '2022-05-20', 'active'),

   (513, 'Naresh', '3337779999', 'nareshkumar@gmail.com', 'naresh567', '741 Birch St, City',
'1989-11-05', '2022-05-22', 'active'),
    (514, 'Daniel', '9991112222', 'danielmartin@gmail.com', 'daniel908', '369 Cherry St, City',
'1996-06-08', '2022-05-25', 'active'),

   (515, 'Sophia', '6664441111', 'sophiathomas@gmail.com', 'sophi675', '963 Poplar St, City',
'1997-12-30', '2022-05-30', 'active'),

  (516, 'Michael Lee', '4445556666', 'michaellee@example.com', 'pass789', '654 Oak St', '1991-
07-18', '2023-05-26', 'Active'),

  (517, 'Jessica Wilson', '2223334444', 'jessicawilson@example.com', 'qwerty123', '753 Elm Ave',
'1993-02-12', '2023-05-25', 'Active'),

  (5188, 'David Davis', '1112223333', 'daviddavis@example.com', 'password1234', '852 Pine Ave',
'1996-09-08', '2023-05-24', 'Active'),

  (519, 'Amanda Taylor', '9990001111', 'amandataylor@example.com', 'passpass', '369 Cedar St',
'1989-11-03', '2023-05-23', 'Active'),

  (520, 'Andrew Clark', '7777777777', 'andrewclark@example.com', 'clark123', '741 Oak St',
'1995-06-30', '2023-05-22', 'Active'),

  (521, 'Olivia Anderson', '5555555555', 'oliviaanderson@example.com', 'anderson456', '852 Elm
Ave', '1992-04-16', '2023-05-21', 'Active'),

  (522, 'Daniel White', '3333333333', 'danielwhite@example.com', 'white789', '963 Maple St',
'1990-09-22', '2023-05-20', 'Active'),

  (523, 'Sophia Lopez', '1111111111', 'sophialopez@example.com', 'passpass123', '369 Pine Ave',
'1997-08-17', '2023-05-19', 'Active'),

  (524, 'Matthew Hill', '9999999999', 'matthewhill@example.com', 'hill123', '741 Cedar St', '1994-
03-14', '2023-05-18', 'Active'),

  (525, 'Emma Carter', '7777777777', 'emmacarter@example.com', 'carter456', '852 Oak St',
'1991-01-09', '2023-05-17', 'Active'),

  (526, 'James Murphy', '5555555555', 'jamesmurphy@example.com', 'passpass456', '963 Elm
Ave', '1988-10-04', '2023-05-16', 'Active'),

  (527, 'Isabella Rivera', '3333333333', 'isabellarivera@example.com', 'rivera789', '369 Maple St',
'1993-07-01', '2023-05-15', 'Active'),

  (528, 'Ethan Ward', '1111111111', 'ethanward@example.com', 'ward123', '741 Pine Ave',
'1990-04-26', '2023-05-14', 'Active'),

  (529, 'Mia Cox', '9999999999', 'miacox@example.com', 'pass456pass', '852 Cedar St', '1996-03-
22', '2023-05-13', 'Active'),

  (530, 'Alexander Hughes', '7777777777', 'alexanderhughes@example.com', 'hughes789', '963
Oak St', '1992-12-18', '2023-05-12', 'Active'),

  (531, 'Charlotte Patterson', '5555555555', 'charlottepatterson@example.com', 'pass123pass',
'369 Elm Ave', '1989-09-12', '2023-05-11', 'Active'),
  (532, 'William Butler', '3333333333', 'williambutler@example.com', 'butler456', '741 Maple St',
'1994-06-08', '2023-05-10', 'Active'),

  (533, 'Ava Flores', '1111111111', 'avaflores@example.com', 'passpass789', '852 Pine Ave',
'1991-03-05', '2023-05-09', 'Active'),

  (534, 'Benjamin Simmons', '9999999999', 'benjaminsimmons@example.com', 'simmons123',
'963 Cedar St', '1988-12-01', '2023-05-08', 'Active'),

  (535, 'Harper Ramirez', '7777777777', 'harperramirez@example.com', 'pass456pass456', '369
Oak St', '1995-08-28', '2023-05-07', 'Active'),

  (536, 'Joseph Cook', '5555555555', 'josephcook@example.com', 'cook789', '741 Elm Ave',
'1992-05-24', '2023-05-06', 'Active'),

  (537, 'Madison Reed', '3333333333', 'madisonreed@example.com', 'pass123pass123', '852
Maple St', '1989-02-18', '2023-05-05', 'Active'),

  (538, 'Liam Brooks', '1111111111', 'liambrooks@example.com', 'brook123', '963 Pine Ave',
'1994-11-13', '2023-05-04', 'Active'),

  (539, 'Elizabeth Price', '9999999999', 'elizabethprice@example.com', 'price456', '369 Cedar St',
'1991-08-09', '2023-05-03', 'Active'),

  (540, 'Sebastian Coleman', '7777777777', 'sebastiancoleman@example.com', 'coleman789',
'741 Oak St', '1996-05-05', '2023-05-02', 'Active');
```

output:



Users_id	Name	phone_number	email	password	address	date_of_birth	registration_date	account_status
505	John	1234567890	john.doe@gmail.com	john123	123 Main St, City	01-01-1990	01-05-2022	active
506	Smitha	9876543210	smitha@gmail.com	smith456	456 Elm St, City	15-02-1995	05-05-2022	active
507	David	5551234567	davidjohnson@gmail.com	david789	789 Oak St, City	20-08-1988	10-05-2022	active
508	Devansh	4449876543	devansh@gmail.com	devabc	987 Pine St, City	12-04-1992	12-05-2022	active
509	Wilson	2225551234	wilsonraj@gmail.com	wilson3478	321 Cedar St, City	02-07-1993	15-05-2022	active
511	Sarah	7779998888	sarahanderson@gmail.com	sarah345	654 Walnut St, City	18-09-1991	18-05-2022	active
512	Krishna	1112223333	krishna123@gmail.com	krishna890	852 Maple St, City	25-03-1994	20-05-2022	active
513	Naresh	3337779999	nareshkumar@gmail.com	naresh567	741 Birch St, City	05-11-1989	22-05-2022	active

Table:Rapido_user

```
  select * from Rapido_user;


  CREATE TABLE Ride (

    ride_id INT PRIMARY KEY,

    users_id INT NOT NULL,

    driver_id INT NOT NULL,

    start_location VARCHAR(60) NOT NULL,
    end_location VARCHAR(50) NOT NULL,
    ride_status VARCHAR(20) NOT NULL,
    fare_amount DECIMAL(10, 2) NOT NULL
);
INSERT INTO Ride (ride_id, users_id, driver_id, start_location, end_location, ride_status,
fare_amount)
VALUES
    (1, 505, 101, 'Tirupati Railway Station', 'Tirumala Temple', 'Completed', 100.00),
    (2, 506, 102, 'Tirupati Bus Stand', 'Sri Venkateswara Zoological Park', 'Completed', 150.00),
    (3, 507, 103, 'Renigunta Airport', 'Chandragiri Fort', 'Completed', 200.00),
    (4, 508, 104, 'Tirupati Main Road', 'Kapila Theertham Waterfall', 'Completed', 120.00),
    (5, 509, 105, 'Tirupati Railway Station', 'TTD Gardens', 'Completed', 80.00),
    (6, 510, 106, 'Tirupati Bus Stand', 'Kalyani Dam', 'Completed', 180.00),
    (7, 511, 107, 'Renigunta Airport', 'Srikalahasti Temple', 'Completed', 220.00),
    (8, 512, 108, 'Tirupati Main Road', 'Deer Park', 'Completed', 90.00),
    (9, 513, 109, 'Tirupati Railway Station', 'Swami Pushkarini Lake', 'Completed', 70.00),
    (10, 514, 110, 'Tirupati Bus Stand', 'ISKCON Temple', 'Completed', 130.00),
    (11, 515, 111, 'Tirupati Railway Station', 'Tiruchanoor Temple', 'Completed', 60.00),
    (12, 516, 112, 'Tirupati Bus Stand', 'Gudimallam Temple', 'Completed', 140.00),
    (13, 517, 113, 'Renigunta Airport', 'Papavinasam Theertham', 'Completed', 190.00),
    (14, 518, 114, 'Tirupati Main Road', 'Talakona Waterfall', 'Completed', 110.00),
    (15, 519, 115, 'Tirupati Railway Station', 'Akasa Ganga', 'Completed', 75.00),
    (16, 520, 116, 'Tirupati Bus Stand', 'Sri Vari Museum', 'Completed', 160.00),
    (17, 521, 117, 'Renigunta Airport', 'Rock Garden', 'Completed', 210.00),
    (18, 522, 118, 'Tirupati Main Road', 'SV Museum', 'Completed', 100.00),
    (19, 523, 119, 'Tirupati Railway Station', 'Pulicat Lake', 'Completed', 80.00),
    (20, 524, 120, 'Tirupati Bus Stand', 'Govindaraja Swamy Temple', 'Completed', 120.00),
    (21, 525, 121, 'Renigunta Airport', 'Sri Prasanna Venkateswara Swamy Temple', 'Completed',
180),
    (22, 526, 122, 'Tirupati Main Road', 'Silathoranam', 'Completed', 90.00),
    (23, 527, 123, 'Tirupati Railway Station', 'Chakra Teertham', 'Completed', 70.00),
    (24, 528, 124, 'Tirupati Bus Stand', 'Tumbhuru Teertham', 'Completed', 130.00),
    (25, 529, 125, 'Renigunta Airport', 'TTD Information Center', 'Completed', 95.00),
    (26, 530, 126, 'Tirupati Main Road', 'Sri Venkateswara Dhyana Vignan Mandiram', 'Completed',
110.00),
    (27, 531, 127, 'Tirupati Railway Station', 'Narayanagiri Gardens', 'Completed', 75.00),
    (28, 532, 128, 'Tirupati Bus Stand', 'Asthan Mandir', 'Completed', 140.00),
    (29, 533, 129, 'Renigunta Airport', 'Sri Padmavathi Ammavari Temple', 'Completed', 120.00),
    (30, 534, 130, 'Tirupati Main Road', 'Sri Venkateswara Swamy Vaari Temple', 'Completed',
100.00),
    (31, 535, 131, 'Tirupati Railway Station', 'Srivari Padalu', 'Completed', 80.00),
    (32, 536, 132, 'Tirupati Bus Stand', 'Kodanda Rama Swamy Temple', 'Completed', 150.00),
    (33, 537, 133, 'Renigunta Airport', 'TTD Kalyana Mandapam', 'Completed', 160.00),
    (34, 538, 134, 'Tirupati Main Road', 'Tirumala Nambi Temple', 'Completed', 90.00),
    (35, 539, 135, 'Tirupati Railway Station', 'TTD Srinivasam Complex', 'Completed', 70.00),
     (36, 540, 136, 'Tirupati Bus Stand', 'TTD S.V. Museum', 'Completed', 120.00);

           select * from Ride;
```

output:

ride_id	users_id	driver_id	start_location	end_location	ride_status	fare_amount
1	505	101	Tirupati Railway Station	Tirumala Temple	Completed	100
2	506	102	Tirupati Bus Stand	Sri Venkateswara Zoological Park	Completed	150
3	507	103	Renigunta Airport	Chandragiri Fort	Completed	200
4	508	104	Tirupati Main Road	Kapila Theertham Waterfall	Completed	120
5	509	105	Tirupati Railway Station	TTD Gardens	Completed	80

6	510	106	Tirupati Bus Stand	Kalyani Dam	Completed	180

Table:Driver

```
        create table Driver
(
  Driverid int primary key,
   Name varchar(35) not null,
   phone_number varchar(20) not null,
   email varchar(50) not null,
   password varchar(25) not null,
   vehicle_number varchar(20) not null,
   vehicle_type varchar(50) not null,
   availability_status varchar(20) not null,
);


INSERT INTO Driver (Driverid, Name, phone_number, email, password, vehicle_number,
vehicle_type, availability_status)
VALUES
  (101, 'Rajesh Kumar', '9876543210', 'rajesh@example.com', 'password123', 'KA01AB1234',
'Sedan', 'Available'),
  (102, 'Amit Sharma', '9876543211', 'amit@example.com', 'password456', 'MH02CD5678',
'Hatchback', 'Available'),
  (103, 'Sneha Patel', '9876543212', 'sneha@example.com', 'password789', 'GJ05EF9012', 'SUV',
'Available'),
  (104, 'Vikram Singh', '9876543213', 'vikram@example.com', 'password321', 'DL09GH3456',
'Sedan', 'Available'),
  (105, 'Deepa Verma', '9876543214', 'deepa@example.com', 'password654', 'KA03IJ7890',
'Hatchback', 'Available'),
  (106, 'Rajendra Gupta', '9876543215', 'rajendra@example.com', 'password987', 'MH04KL1234',
'SUV', 'Available'),
  (107, 'Kavita Shah', '9876543216', 'kavita@example.com', 'password321', 'GJ07MN5678',
'Sedan', 'Available'),
  (108, 'Anil Yadav', '9876543217', 'anil@example.com', 'password654', 'DL05OP9012',
'Hatchback', 'Available'),
  (109, 'Shruti Desai', '9876543218', 'shruti@example.com', 'password987', 'KA05QR3456', 'SUV',
'Available'),
  (110, 'Arun Kumar', '9876543219', 'arun@example.com', 'password123', 'MH06ST7890',
'Sedan', 'Available'),
  (111, 'Priya Sharma', '9876543220', 'priya@example.com', 'password456', 'GJ09UV1234',
'Hatchback', 'Available'),
  (112, 'Alok Patel', '9876543221', 'alok@example.com', 'password789', 'DL07WX5678', 'SUV',
'Available'),
  (113, 'Nisha Singh', '9876543222', 'nisha@example.com', 'password321', 'KA07YZ9012', 'Sedan',
'Available'),
  (114, 'Ravi Verma', '9876543223', 'ravi@example.com', 'password654', 'MH08AB3456',
'Hatchback', 'Available'),
  (115, 'Swati Gupta', '9876543224', 'swati@example.com', 'password987', 'GJ01CD7890', 'SUV',
'Available'),
  (116, 'Vivek Sharma', '9876543225', 'vivek@example.com', 'password123', 'DL02EF1234',
'Sedan', 'Available'),
  (117, 'Meena Yadav', '9876543226', 'meena@example.com', 'password456', 'KA09GH5678',
'Hatchback', 'Available'),
  (118, 'Rajat Desai', '9876543227', 'rajat@example.com', 'password789', 'MH03IJ9012', 'SUV',
'Available'),
  (119, 'Pooja Singh', '9876543228', 'pooja@example.com', 'password321', 'GJ05KL3456', 'Sedan',
'Available'),
  (120, 'Rohit Verma', '9876543229', 'rohit@example.com', 'password654', 'DL03MN7890',
'Hatchback', 'Available'),
  (121, 'Smita Shah', '9876543230', 'smita@example.com', 'password987', 'KA01OP1234', 'SUV',
'Available'),
  (122, 'Sanjay Kumar', '9876543231', 'sanjay@example.com', 'password123', 'MH02QR5678',
'Sedan', 'Available'),
  (123, 'Divya Patel', '9876543232', 'divya@example.com', 'password456', 'GJ03ST9012',
'Hatchback', 'Available'),
  (124, 'Avinash Gupta', '9876543233', 'avinash@example.com', 'password789', 'DL01UV1234',
'SUV', 'Available'),
  (125, 'Radha Sharma', '9876543234', 'radha@example.com', 'password321', 'KA03WX5678',
'Sedan', 'Available'),
  (126, 'Vishal Verma', '9876543235', 'vishal@example.com', 'password654', 'MH04YZ9012', 'Hatchback', 'Available'),
  (127, 'Manisha Gupta', '9876543236', 'manisha@example.com', 'password987', 'GJ06AB3456', 'SUV', 'Available'),
  (128, 'Nitin Singh', '9876543237', 'nitin@example.com', 'password123', 'DL06CD7890', 'Sedan', 'Available'),
  (129, 'Kirti Desai', '9876543238', 'kirti@example.com', 'password456', 'KA08EF1234', 'Hatchback', 'Available'),
  (130, 'Rakesh Patel', '9876543239', 'rakesh@example.com', 'password789', 'MH05GH5678', 'SUV', 'Available'),
  (131, 'Sarika Sharma', '9876543240', 'sarika@example.com', 'password321', 'GJ08IJ9012', 'Sedan', 'Available'),
  (132, 'Prakash Yadav', '9876543241', 'prakash@example.com', 'password654', 'DL09KL1234', 'Hatchback', 'Available'),
  (133, 'Anita Verma', '9876543242', 'anita@example.com', 'password987', 'KA02MN5678', 'SUV', 'Available'),
  (134, 'Rahul Shah', '9876543243', 'rahul@example.com', 'password123', 'MH06OP9012', 'Sedan', 'Available'),
  (135, 'Mala Gupta', '9876543244', 'mala@example.com', 'password456', 'GJ04QR3456', 'Hatchback', 'Available');

  select * from Driver;
```

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
101	Rajesh Kumar	9876543210	rajesh@example.com	password123	KA01AB1234	Sedan	Available
102	Amit Sharma	9876543211	amit@example.com	password456	MH02CD5678	Hatchback	Available
103	Sneha Patel	9876543212	sneha@example.com	password789	GJ05EF9012	SUV	Available

104	Vikram Singh	9876543213	vikram@example.com	password321	DL09GH3456	Sedan	Available
105	Deepa Verma	9876543214	deepa@example.com	password654	KA03IJ7890	Hatchback	Available

Table:Driver_Rating

```
CREATE TABLE Driver_Rating (
   rating_id INT PRIMARY KEY,
   driver_id INT NOT NULL,
          ride_id   INT NOT NULL,
   rating DECIMAL(3, 2) NOT NULL CHECK (rating >= 1 AND rating <= 5),
   rating_date DATE NOT NULL,
          feedback   TEXT  NOT NULL,
   FOREIGN KEY (driver_id) REFERENCES Driver(Driverid)
);

INSERT INTO Driver_Rating (rating_id,  driver_id, ride_id, rating, feedback, rating_date)
VALUES
  (201,  101, 1, 4, 'Good service', '2023-05-01' ),
  (202,  102, 2, 5, 'Excellent ride', '2023-05-02' ),
  (203,  103, 3, 3, 'Average experience', '2023-05-03'),
  (204, 104, 4, 2, 'Poor service', '2023-05-04'),
  (205, 105, 5, 4, 'Satisfactory ride', '2023-05-05'),
  (206, 106, 6, 5, 'Great driver', '2023-05-06'),
  (207,  107, 7, 3, 'Could be better', '2023-05-07'),
  (208,  108, 8, 4, 'Good experience', '2023-05-08'),
  (209,  109, 9, 2, 'Disappointing ride', '2023-05-09'),
  (210,  110, 10, 5, 'Highly recommended', '2023-05-10'),
  (211,  111, 11, 3, 'Average service', '2023-05-11'),
  (212, 112, 12, 4, 'Professional driver', '2023-05-12'),
  (213, 113, 13, 5, 'Wonderful ride', '2023-05-13'),
  (214,  114, 14, 3, 'Could be improved', '2023-05-14'),
  (215,  115, 15, 4, 'Pleasant experience', '2023-05-15'),
  (216,  116, 16, 2, 'Unsatisfactory ride', '2023-05-16'),
  (217,  117, 17, 4, 'Good service', '2023-05-17'),
  (218,  118, 18, 5, 'Excellent ride', '2023-05-18'),
  (219, 119, 19, 3, 'Average experience', '2023-05-19'),
  (220, 120, 20, 2, 'Poor service', '2023-05-20'),
  (221,  121, 21, 4, 'Satisfactory ride', '2023-05-21'),
  (222,  122, 22, 5, 'Great driver', '2023-05-22'),
  (223,  123, 23, 3, 'Could be better', '2023-05-23'),
  (224,  124, 24, 4, 'Good experience', '2023-05-24'),
  (225,  125, 25, 2, 'Disappointing ride', '2023-05-25'),
  (226,  126, 26, 5, 'Highly recommended', '2023-05-26'),
  (227, 127, 27, 3, 'Average service', '2023-05-27'),
  (228,  128, 28, 4, 'Professional driver', '2023-05-28'),
  (229,  129, 29, 5, 'Wonderful ride', '2023-05-29'),
  (230,  130, 30, 3, 'Could be improved', '2023-05-30'),
  (231,  131, 31, 4, 'Pleasant experience', '2023-05-31'),
  (232, 132, 32, 2, 'Unsatisfactory ride', '2023-06-01'),
  (233, 133, 33, 4, 'Good service', '2023-06-02');
```

output:

rating_id	driver_id	ride_id	rating	rating_date	feedback
201	101	1	4	01-05-2023	Good service
202	102	2	5	02-05-2023	Excellent ride

203	103	3	3	03-05-2023	Average experience
204	104	4	2	04-05-2023	Poor service
205	105	5	4	05-05-2023	Satisfactory ride
206	106	6	5	06-05-2023	Great driver
207	107	7	3	07-05-2023	Could be better
208	108	8	4	08-05-2023	Good experience

Table:Driver_Rating

```
 select * from Driver_Rating;
      create table Payment
(
  payment_id int primary key,
  users_id int not null,
  ride_id int not null,
  amount money not null,
  payment_date date not null,
  payment_status varchar(20) not null
);
insert into Payment (payment_id, users_id, ride_id, amount, payment_date, payment_status)
values
  (1103, 505, 1, 15.50, '2022-05-01', 'paid'),
  (1104, 506, 2, 10.25, '2022-05-05', 'paid'),
  (1105, 507, 3, 8.75, '2022-05-10', 'paid'),
  (1106, 508, 4, 12.00, '2022-05-12', 'paid'),
  (1107, 509, 5, 20.50, '2022-05-15', 'paid'),
         (1108, 510, 6, 22.25, '2023-05-14', 'paid'),
         (1109, 511, 7, 15.50, '2022-05-11', 'paid'),
         (1110, 512, 8, 10.25, '2022-05-15', 'paid'),
(1111, 513, 9, 8.75, '2022-05-12', 'paid'),
(1112, 514, 10,22.00, '2022-05-13', 'paid'),
(1113, 515, 11,23.00, '2022-05-14', 'paid'),
(1114, 516, 12,82.00, '2022-05-15', 'paid'),
(1115, 517, 13,22.00, '2022-05-19', 'paid'),
(1116, 518, 14,72.00, '2022-05-18', 'paid'),
(1117, 519, 15,92.00, '2022-05-17', 'paid'),
(1118, 520, 16,52.00, '2022-05-16', 'paid'),
(1119, 521, 19,62.00, '2022-05-02', 'paid'),
(1120, 522, 20,82.00, '2022-05-22', 'paid');
select* from payment;
```

output:

```
payment_id   users_id   ride_id   amount   payment_date   payment_status

1103         505        1         15.5     01-05-2022     paid

1104         506        2         10.25    05-05-2022     paid

1105         507        3         8.75     10-05-2022     paid

1106         508        4         12       12-05-2022     paid

1107         509        5         20.5     15-05-2022     paid

1108         510        6         22.25    14-05-2023     paid

1109         511        7         15.5     11-05-2022     paid

1110         512        8         10.25    15-05-2022     paid

1111         513        9         8.75     12-05-2022     paid
```

Table:Promocode

```
create table PromoCode
(
   promo_code_id int primary key,
   code varchar(20) not null,
   discount decimal(5, 2) not null,
   expiry_date date not null,
   max_usage int not null,
   usage_count int not null,

);
insert into PromoCode (promo_code_id, code, discount, expiry_date, max_usage, usage_count)
values
   (01, 'SUMMER2022', 10.00, '2022-08-31', 100, 1),
   (02, 'WELCOME20', 20.00, '2022-12-31', 500, 1),
   (03, 'FREERIDE', 100.00, '2022-06-30', 1000, 1),
   (04, 'SAVEMORE', 15.00, '2022-10-31', 200, 1),
   (05, 'EARLYBIRD', 25.00, '2023-01-31', 50, 1),
   (06, 'WELCOME10', 10.00, '2022-08-31', 100, 0),
   (07, 'GDAY10', 20.00, '2022-12-31', 500, 0),
   (08, 'HELLO10', 100.00, '2022-06-30', 1000, 0),
   (09, 'HOWDY10', 15.00, '2022-10-31', 200, 0),
   (010, 'WELCOMEABOARD', 25.00, '2023-01-31', 50, 0),
   (011, 'ALLABOARD', 10.00, '2022-08-31', 100, 0),
   (012, 'BACK2SCHOOL', 20.00, '2022-12-31', 500, 0),
   (013, 'TAKETHEMBACK', 100.00, '2022-06-30', 1000, 0),
   (014, 'BACKONTHEBUS', 15.00, '2022-10-31', 200, 0),
   (015, 'LITTLELEARNERS', 25.00, '2023-01-31', 50, 0),
   (016, 'SCHOOLPACK', 10.00, '2022-08-31', 100, 0),
   (017, 'TAKEITALL', 20.00, '2022-12-31', 500, 0),
  (018, 'STOCKTAKESALE', 100.00, '2022-06-30', 1000, 0),
  (019, 'BIGSTOCKTAKE', 15.00, '2022-10-31', 200, 0),
  (020, 'HELPUSMOVE', 25.00, '2023-01-31', 50, 0),
  (021, 'OVERSTOCKED15', 10.00, '2022-08-31', 100, 0),
  (022, 'LOVERLOVER', 20.00, '2022-12-31', 500, 0),
  (023, 'ICANSEEITINYOUREYES', 100.00, '2022-06-30', 1000, 0),
  (024, 'SUMMERSALE', 15.00, '2022-10-31', 200, 0),
  (025, 'LOVE10', 25.00, '2023-01-31', 50, 0),
  (026, 'SPRINGSALE', 10.00, '2022-08-31', 100, 0),
  (027, 'LOVEMOM', 20.00, '2022-12-31', 500, 0),
  (028, 'BLACKFRIDAY', 100.00, '2022-06-30', 1000, 0),
  (029, 'NEWYEAR', 15.00, '2022-10-31', 200, 0),
  (030, 'CYBER20', 25.00, '2023-01-31', 50, 0),
  (031, 'FALL', 10.00, '2022-08-31', 100, 0);
        select* from promocode;
```

output:

```
promo_code_id   code          discount   expiry_date   max_usage   usage_count

1               SUMMER2022    10         31-08-2022    100         1

2               WELCOME20     20         31-12-2022    500         1

3               FREERIDE      100        30-06-2022    1000        1

4               SAVEMORE      15         31-10-2022    200         1

5               EARLYBIRD     25         31-01-2023    50          1

6               WELCOME10     10         31-08-2022    100         0

7               GDAY10        20         31-12-2022    500         0
```

8	HELLO10	100	30-06-2022	1000	0

Table:Driver_Earning

```
        CREATE TABLE Driver_Earnings (
  earnings_id INT PRIMARY KEY,
  driver_id INT NOT NULL,
  ride_id INT NOT NULL,
  earnings_amount DECIMAL(10, 2) NOT NULL,
  earnings_date DATE NOT NULL,
  FOREIGN KEY (driver_id) REFERENCES Driver(Driverid),
  FOREIGN KEY (ride_id) REFERENCES Ride(ride_id)
);

INSERT INTO Driver_Earnings (earnings_id, driver_id, ride_id, earnings_amount, earnings_date)
VALUES
  (001, 101, 1, 50.00, '2022-01-01'),
  (002, 102, 2, 45.00, '2022-01-02'),
  (003, 103, 3, 60.00, '2022-01-03'),
  (004, 104, 4, 55.00, '2022-01-04'),
  (005, 105, 5, 70.00, '2022-01-05'),
  (006, 106, 6, 65.00, '2022-01-06'),
  (007, 107, 7, 80.00, '2022-01-07'),
  (008, 108, 8, 75.00, '2022-01-08'),
  (009, 109, 9, 90.00, '2022-01-09'),
  (0010,110, 10, 85.00, '2022-01-10'),
  (0011, 111, 11, 95.00, '2022-01-11'),
  (0012, 112, 12, 70.00, '2022-01-12'),
  (0013, 113, 13, 80.00, '2022-01-13'),
  (0014, 114, 14, 75.00, '2022-01-14'),
  (0015, 115, 15, 90.00, '2022-01-15'),
  (0016, 116, 16, 85.00, '2022-01-16'),
(0017, 117, 17, 55.00, '2022-01-17'),
(0018, 118, 18, 65.00, '2022-01-18'),
(0019, 119, 19, 80.00, '2022-01-19'),
(0020, 120, 20, 75.00, '2022-01-20'),
(0021, 121, 21, 60.00, '2022-01-21'),
(0022, 122, 22, 70.00, '2022-01-22'),
(0023, 123, 23, 85.00, '2022-01-23'),
(0024, 124, 24, 75.00, '2022-01-24'),
(0025, 125, 25, 90.00, '2022-01-25'),
(0026, 126, 26, 80.00, '2022-01-26'),
(0027, 127, 27, 55.00, '2022-01-27'),
(0028, 128, 28, 65.00, '2022-01-28'),
(0029, 129, 29, 80.00, '2022-01-29'),
(0030, 130, 30, 75.00, '2022-01-30');
                    select* from driver_Earnings;
```

output:

```
earnings_id   driver_id   ride_id   earnings_amount   earnings_date

1             101         1         50                01-01-2022

2             102         2         45                02-01-2022

3             103         3         60                03-01-2022

4             104         4         55                04-01-2022

5             105         5         70                05-01-2022

6             106         6         65                06-01-2022
```

Table:Ride_category

```
CREATE TABLE Ride_Category (
    category_id INT PRIMARY KEY,
    category_name VARCHAR(50) NOT NULL,
    base_fare DECIMAL(10, 2) NOT NULL,
    distance_rate DECIMAL(5, 2) NOT NULL,
    time_rate DECIMAL(5, 2) NOT NULL
);

INSERT INTO Ride_Category (category_id, category_name, base_fare, distance_rate, time_rate)
VALUES
    (1, 'bike', 10.00, 0.50, 0.25),
    (2, 'car', 15.00, 0.75, 0.30),
    (3, 'auto', 20.00, 1.00, 0.35),
    (4, 'Bicycle', 25.00, 1.25, 0.40),
    (5, 'Bike', 30.00, 1.50, 0.45),
    (6, 'Car', 22.00, 1.10, 0.35),
    (7, 'Auto', 17.00, 0.85, 0.30),
    (8, 'Bicycle', 40.00, 2.50, 0.60),
    (9, 'Auto', 18.00, 0.90, 0.30),
    (10, 'car', 35.00, 2.00, 0.50),
    (11, 'Bike', 12.00, 0.60, 0.28),
    (12, 'Auto', 14.00, 0.70, 0.32),
    (13, 'Bicycle', 16.00, 0.80, 0.36),
    (14, 'Auto', 13.00, 0.65, 0.30),
    (15, 'Bike', 15.00, 0.75, 0.34),
    (16, 'Car', 18.00, 0.90, 0.38),
    (17, 'Auto', 20.00, 1.00, 0.42),
    (18, 'Bicycle', 25.00, 1.25, 0.46),
    (19, 'Bike', 30.00, 1.50, 0.50),
    (20, 'Car', 40.00, 2.00, 0.60),
          (21, 'Auto', 35.00, 1.75, 0.55),
    (22, 'Bike', 25.00, 1.25, 0.40),
    (23, 'Auto', 40.00, 2.00, 0.60),
    (24, 'Car', 50.00, 2.50, 0.70),
    (25, 'Auto', 60.00, 3.00, 0.80),
    (26, 'Bike', 45.00, 2.25, 0.65),
    (27, 'Bike', 10.00, 0.50, 0.25),
    (28, 'Car', 8.00, 0.40, 0.20),
    (29, 'Bicycle', 5.00, 0.25, 0.15),
    (30, 'Auto', 0.00, 0.00, 0.00);

                  select* from Ride_category;
```

output:

```
category_id   category_name   base_fare   distance_rate   time_rate

1             bike            10          0.5             0.25

2             car             15          0.75            0.3

3             auto            20          1               0.35

4             Bicycle         25          1.25            0.4

5             Bike            30          1.5             0.45

6             Car             22          1.1             0.35
```

Table:Driver_document

```
        CREATE TABLE Driver_Document (
  document_id INT PRIMARY KEY,
  driver_id INT NOT NULL,
  document_type VARCHAR(50) NOT NULL,
  document_number VARCHAR(50) NOT NULL,
    expiry_date DATE NOT NULL,
    FOREIGN KEY (driver_id) REFERENCES Driver(Driverid)
);
INSERT INTO Driver_Document (document_id, driver_id, document_type, document_number,
expiry_date)
VALUES
    (2001, 101, 'Driver License', 'DL123456', '2024-05-31'),
    (2002, 102, 'Driver License', 'DL987654', '2023-12-15'),
    (2003, 103, 'ID Card', 'ID789012', '2025-02-28'),
    (2004, 104, 'Passport', 'P1234567', '2026-08-20'),
    (2005, 105, 'Driver License', 'DL543210', '2024-03-10'),
    (2006, 106, 'ID Card', 'ID345678', '2023-07-31'),
    (2007, 107, 'Passport', 'P7654321', '2025-11-22'),
    (2008, 108, 'Driver License', 'DL111222', '2024-09-05'),
    (2009, 109, 'ID Card', 'ID333444', '2023-10-18'),
    (2010, 110, 'Driver License', 'DL999888', '2024-06-12'),
    (2011, 111, 'ID Card', 'ID222333', '2024-09-30'),
    (2012, 112, 'Passport', 'P9876543', '2025-05-15'),
    (2013, 113, 'Driver License', 'DL444555', '2023-12-31'),
    (2014, 114, 'ID Card', 'ID666777', '2025-03-20'),
    (2015, 115, 'Driver License', 'DL222111', '2024-08-10'),
    (2016, 116, 'Passport', 'P8765432', '2026-01-31'),
    (2017, 117, 'Driver License', 'DL888999', '2024-11-22'),
    (2018, 118, 'ID Card', 'ID999888', '2023-07-05'),
    (2019, 119, 'Driver License', 'DL777666', '2024-10-18'),
    (2020, 120, 'ID Card', 'ID555444', '2023-06-12'),
            (2021, 121, 'Passport', 'P5432109', '2024-11-30'),
    (2022, 122, 'Driver License', 'DL777888', '2025-06-15'),
    (2023, 123, 'ID Card', 'ID111222', '2023-12-31'),
    (2024, 124, 'Driver License', 'DL222333', '2025-03-20'),
    (2025, 125, 'ID Card', 'ID444555', '2024-08-10'),
    (2026, 126, 'Passport', 'P7654321', '2026-01-31'),
    (2027, 127, 'Driver License', 'DL888999', '2024-11-22'),
    (2028, 128, 'ID Card', 'ID999000', '2023-07-05'),
    (2029, 109, 'Driver License', 'DL666777', '2024-10-18'),
    (2030, 130, 'ID Card', 'ID555666', '2023-06-12');

           select * from Driver_Document;
```

output:

```
 document_id   driver_id   document_type    document_number   expiry_date

 2001          101         Driver License   DL123456          31-05-2024

 2002          102         Driver License   DL987654          15-12-2023

 2003          103         ID Card          ID789012          28-02-2025

 2004          104         Passport         P1234567          20-08-2026

 2005          105         Driver License   DL543210          10-03-2024

 2006          106         ID Card          ID345678          31-07-2023
```

Table:User_Payment_method

```
          CREATE TABLE User_Payment_Method (
   payment_method_id INT PRIMARY KEY,
   users_id INT NOT NULL,
   payment_method_type VARCHAR(50) NOT NULL,
   expiry_date DATE NOT NULL,
   FOREIGN KEY (users_id) REFERENCES Rapido_User(Users_id)
);

INSERT INTO User_Payment_Method (payment_method_id, users_id, payment_method_type,
expiry_date)
```

```
VALUES
   (1103, 505, 'Credit Card',  '2024-05-31'),
   (1104, 506, 'Debit Card', '2023-12-15'),
   (1105, 507, 'PayPal',  '2025-02-28'),
   (1106, 508, 'Bank Transfer', '2026-08-20'),
   (1107, 509, 'Credit Card', '2024-03-10'),
   (1108, 511, 'Debit Card',  '2023-07-31'),
   (1109, 511, 'PayPal',  '2025-11-22'),
   (1110, 512, 'Bank Transfer', '2024-09-05'),
   (1111, 513, 'Credit Card', '2023-10-18'),
   (1112, 514, 'Debit Card', '2024-06-12'),
   (1113, 515, 'Credit Card',  '2024-09-30'),
   (1114, 516, 'Debit Card', '2025-05-15'),
   (1115, 517, 'PayPal', '2023-12-31'),
   (1116, 531, 'Bank Transfer',  '2025-03-20'),
   (1117, 519, 'Credit Card',  '2024-08-10'),
   (1118, 520, 'Debit Card',  '2026-01-31'),
   (1119, 521, 'PayPal',  '2024-11-22'),
   (1120, 522, 'Bank Transfer',  '2023-07-05'),
   (1121, 523, 'Credit Card', '2024-10-18'),
   (1122, 524, 'Debit Card',  '2023-06-12'),
   (1123, 525, 'Credit Card',  '2024-11-30'),
   (1124, 526, 'Debit Card',  '2025-06-15'),
   (1125, 527, 'PayPal', '2023-12-31'),
   (1126, 528, 'Bank Transfer', '2025-03-20'),
   (1127, 527, 'Credit Card',  '2024-08-10'),
   (1128, 528, 'Debit Card', '2026-01-31'),
   (1129, 527, 'PayPal', '2024-11-22'),
   (1130, 528, 'Bank Transfer', '2023-07-05'),
   (1131, 529, 'Credit Card', '2024-10-18'),
    (1132, 530, 'Debit Card',  '2023-06-12');
                    select* from User_Payment_Method;
```

output:

```
 payment_method_id   users_id   payment_method_type   expiry_date

 1103                505        Credit Card           31-05-2024

 1104                506        Debit Card            15-12-2023

 1105                507        PayPal                28-02-2025

 1106                508        Bank Transfer         20-08-2026

 1107                509        Credit Card           10-03-2024

 1108                511        Debit Card            31-07-2023

 1109                511        PayPal                22-11-2025

 1110                512        Bank Transfer         05-09-2024
```

Table:Ride_Review

```
          CREATE TABLE Ride_review (
   review_id INT PRIMARY KEY,
   Users_id INT NOT NULL,
   ride_id INT NOT NULL,
   review_text VARCHAR(200) NOT NULL,
   review_date DATE NOT NULL,
   FOREIGN KEY (Users_id) REFERENCES Rapido_User(Users_id),
   FOREIGN KEY (ride_id) REFERENCES Ride(ride_id)
);

INSERT INTO Ride_Review (review_id, Users_id, ride_id, review_text, review_date)
VALUES
```

(701, 505, 1, 'Great ride! The driver was friendly and the car was clean.', '2022-09-10'),

(702, 506, 2, 'Smooth ride. Arrived at my destination on time.', '2022-11-05'),

(703, 507, 3, 'Disappointed with the driver. They took a longer route.', '2022-10-20'),

(704, 508, 4, 'Excellent service! The driver was professional and polite.', '2022-12-02'),

(705, 509, 5, 'Average ride. The car had an unpleasant smell.', '2022-11-15'),

(706, 511, 6, 'Highly recommended. The driver was knowledgeable and helpful.', '2022-09-28'),

(707, 511, 7, 'Terrible ride. The driver was rude and reckless.', '2022-10-05'),

(708, 512, 8, 'Good experience overall. The driver was punctual.', '2022-12-10'),

(709, 513, 9, 'Not satisfied. The driver got lost and the ride took longer than expected.', '2022-09-15'),

(710, 514, 10, 'Smooth ride. The driver was friendly and the car was comfortable.', '2022-12-20'),

(711, 515, 11, 'Great service! The driver went above and beyond.', '2022-11-10'),

(712, 516, 12, 'Unprofessional driver. They talked on the phone throughout the ride.', '2022-10-25'),

(713, 517, 13, 'Highly satisfied. The driver was courteous and drove safely.', '2022-12-05'),

(714, 518, 14, 'Average ride. The driver was not familiar with the route.', '2022-09-30'),

(715, 519, 15, 'Prompt service. The driver was polite and helpful.', '2022-11-18'),

(716, 520, 16, 'Disappointed with the condition of the car. It was dirty and uncomfortable.', '2022-10-12'),

(717, 521, 17, 'Exceptional ride! The driver was friendly and the car was luxurious.', '2022-12-15'),

(718, 522, 18, 'Good ride. The driver followed the requested route.', '2022-09-20'),

(719, 523, 19, 'Unreliable service. The driver canceled the ride at the last minute.', '2022-11-25'),

(720, 524, 20, 'Smooth ride. The driver was courteous and drove safely.', '2022-10-08'),

(721, 525, 21, 'Great ride! The driver was friendly and the car was clean.', '2022-12-07'),

(722, 526, 22, 'Smooth ride. Arrived at my destination on time.', '2022-09-22'),

(723, 527, 23, 'Disappointed with the driver. They took a longer route.', '2022-11-28'),

(724, 528, 24, 'Excellent service! The driver was professional and polite.', '2022-10-02'),

(725, 529, 25, 'Average ride. The car had an unpleasant smell.', '2022-12-17'),

```
   (726, 530, 26, 'Highly recommended. The driver was knowledgeable and helpful.', '2022-09-
12'),
   (727, 531, 27, 'Terrible ride. The driver was rude and reckless.', '2022-11-02'),
   (728, 532, 28, 'Good experience overall. The driver was punctual.', '2022-10-10'),
   (729, 533, 29, 'Not satisfied. The driver got lost and the ride took longer than expected.',
'2022-12-12'),
   (730, 534, 30, 'Smooth ride. The driver was friendly and the car was comfortable.', '2022-09-
25');

          select *from Ride_Review;
```

Table:Driver_Location

```
CREATE TABLE Driver_Location (
   location_id INT PRIMARY KEY,
   driver_id INT NOT NULL,
   location_name VARCHAR(50) NOT NULL,
   latitude DECIMAL(9, 6) NOT NULL,
   longitude DECIMAL(9, 6) NOT NULL,
   FOREIGN KEY (driver_id) REFERENCES Driver(Driverid)
);

INSERT INTO Driver_Location (location_id, driver_id, location_name, latitude, longitude)
VALUES
   (401, 101, 'Alipiri', 13.634978, 79.414733),
   (402, 102, 'Tiruchanur', 13.238140, 79.507612),
   (403, 103, 'Renigunta', 13.647468, 79.508050),
   (404, 104, 'Padmavathi Temple', 13.635270, 79.421337),
   (405, 105, 'Govindaraja Swamy Temple', 13.676791, 79.418509),
   (406, 106, 'Tirumala', 13.628756, 79.419179),
   (407, 107, 'Kapila Theertham', 13.657322, 79.503582),
   (408, 108, 'Akasa Ganga', 13.650191, 79.512501),
   (409, 109, 'Sri Venkateswara Zoological Park', 13.609800, 79.451223),
   (410, 110, 'Sri Vari Museum', 13.672045, 79.414478),
   (411, 111, 'TTD Gardens', 13.644466, 79.418598),
   (412, 112, 'Srinivasa Mangapuram', 13.455450, 79.507201),
   (413, 113, 'Kanipakam', 13.593237, 79.321027),
   (414, 114, 'Chandragiri Fort', 13.620123, 79.432641),
   (415, 115, 'Sri Kalyana Venkateswaraswami Temple', 13.625268, 79.418673),
   (416, 116, 'Papavinasanam', 13.650908, 79.420572),
   (417, 117, 'Tumburu Teertham', 13.640894, 79.500987),
   (418, 118, 'Silathoranam', 13.674526, 79.419953),
   (419, 119, 'Tirumala Deer Park Reserve', 13.653516, 79.437209),
   (420, 120, 'Srivari Mettu', 13.650010, 79.401447),
   (421, 121, 'TTD Information Centre', 13.673489, 79.414688),
   (422, 122, 'Rock Garden', 13.681674, 79.428045),
   (423, 123, 'Kodanda Rama Swamy Temple', 13.638643, 79.426534),
   (424, 124, 'Srikalahasti Temple', 13.753164, 79.699772),
   (425, 125, 'Kapila Teertham', 13.644970, 79.495102),
   (426, 126, 'Mamandur', 13.233469, 79.543157),
   (427, 127, 'Sri Anjaneya Swamy Temple', 13.634458, 79.414519),
   (428, 128, 'Sri Prasanna Venkateswaraswami Temple', 13.628286, 79.418746),
   (429, 129, 'Gangamma Temple', 13.648578, 79.502306),
   (430, 130, 'Sri Bedi Anjaneya Swamy Temple', 13.629446, 79.419675);

         select* from Driver_Location;
```

output:

```
location_id  driver_id  location_name            latitude    longitude

401          101        Alipiri                  13.634978   79.41473

402          102        Tiruchanur               13.23814    79.50761
```

403	103	Renigunta	13.647468	79.50805
404	104	Padmavathi Temple	13.63527	79.42134
405	105	Govindaraja Swamy Temple	13.676791	79.41851
406	106	Tirumala	13.628756	79.41918

--1.Retrieve the total count of rides for each user:

```
SELECT users_id, COUNT(*) AS total_rides
FROM Ride
GROUP BY users_id;
```

output:

```
users_id   total_rides

505        1

506        1

507        1

508        1

509        1

510        1

511        1

512        1
```

--2.List all rides along with the corresponding driver's information:

```
SELECT r.*, d.Name AS driver_name
FROM Ride r
JOIN Rapido_User d ON r.driver_id = d.Users_id;
```

output:

```
ride_id users_id     driver_id     start_location end_location   ride_status    fare_amount
102         505            12             tirupati          pileru          paid             5
0000
103         506            13             pileru          tirupati          paid             5
5000
```

--3.Retrieve the average fare amount for rides with different ride statuses

```
SELECT ride_status, AVG(fare_amount) AS average_fare
FROM Ride
GROUP BY ride_status;
```

output:

```
 ride_status   average_fare

 Completed    121.25
```

--4.List all users who have taken rides from a specific start location:

```
SELECT u.Name
FROM Rapido_User u
JOIN Ride r ON u.Users_id = r.users_id
WHERE r.start_location = 'Tirupati';
```

output:

```
Name
teja
```

prakash

--5.List the users who have taken rides with a fare amount greater than a specific value:

```
SELECT u.Name
FROM Rapido_User u
JOIN Ride r ON u.Users_id = r.users_id
WHERE r.fare_amount > 100;
```

output:

```
 Name

 Smitha

 David

 Devansh

 Sarah

 Daniel
```

--6.Retrieve the users who have taken rides both from a specific start location and to a specific end location:

```
SELECT u.Name
FROM Rapido_User u
JOIN Ride r ON u.Users_id = r.users_id
WHERE r.start_location = 'Tirupati'
AND r.end_location = 'Alipiri';
```

output:

```
Name
```

```
prakash
abhi
```

--7.List the users who have taken rides with the highest fare amount for each ride status:

```
SELECT u.Name, r.ride_status, r.fare_amount
FROM Rapido_User u
JOIN Ride r ON u.Users_id = r.users_id
WHERE r.fare_amount = (
  SELECT MAX(fare_amount)
  FROM Ride
  WHERE ride_status = r.ride_status
);
```

output:

```
 Name   ride_status   fare_amount

 Sarah  Completed     220
```

--8.Find the total count of drivers:

```
SELECT COUNT(*) AS TotalDrivers
FROM Driver;
```

output:

```
TotalDrivers
35
```

--9.Count the number of drivers for each vehicle type:

```
SELECT vehicle_type, COUNT(*) AS DriverCount
FROM Driver
```

```
GROUP BY vehicle_type;
```

output:

vehicle_type	DriverCount
Hatchback	12
Sedan	12
SUV	11

--10.Get the total number of drivers for each availability status:

```
SELECT availability_status, COUNT(*) AS DriverCount
FROM Driver
GROUP BY availability_status;
```

output:

availability_status	DriverCount
Available	35

--11.Retrieve the driver details with the highest driver ID:

```
SELECT *
FROM Driver
WHERE Driverid = (SELECT MAX(Driverid) FROM Driver);
```

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
135	Mala Gupta	9876543244	mala@example.com	password456	GJ04QR3456	Hatchback	Available

--12.Count the number of drivers for each vehicle type in descending order:

```
SELECT vehicle_type, COUNT(*) AS DriverCount
FROM Driver
GROUP BY vehicle_type
ORDER BY DriverCount DESC;
```

output:

```
vehicle_type   DriverCount

Hatchback      12

Sedan          12

SUV            11
```

--13.Calculate the total number of drivers for each vehicle type and availability status:

```
SELECT vehicle_type, availability_status, COUNT(*) AS DriverCount
FROM Driver
GROUP BY vehicle_type, availability_status;
```

output:

```
vehicle_type   availability_status   DriverCount

Hatchback      Available             12

Sedan          Available             12

SUV            Available             11
```

--14.Retrieve the driver details with the highest driver ID for each vehicle type:

```
SELECT *
FROM Driver d
WHERE Driverid = (
   SELECT MAX(Driverid)
   FROM Driver
   WHERE vehicle_type = d.vehicle_type
);
```

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
133	Anita Verma	9876543242	anita@example.com	password987	KA02MN5678	SUV	Available
134	Rahul Shah	9876543243	rahul@example.com	password123	MH06OP9012	Sedan	Available
135	Mala Gupta	9876543244	mala@example.com	password456	GJ04QR3456	Hatchback	Available

--15.Count the number of drivers for each vehicle type with more than 2 drivers:

```
SELECT vehicle_type, COUNT(*) AS DriverCount
FROM Driver
GROUP BY vehicle_type
HAVING COUNT(*) > 2;
```

output:

```
vehicle_type   DriverCount

Hatchback      12

Sedan          12

SUV            11
```

--16.Retrieve the driver's name and vehicle type for a given ride ID:

```
SELECT d.Name, d.vehicle_type
FROM Driver d
JOIN Driver_Rating dr ON d.Driverid = dr.driver_id
WHERE dr.ride_id = 17;
```

output:

```
Name          vehicle_type

Meena Yadav   Hatchback
```

--17.Retrieve the drivers who have a rating greater than 4:

```
SELECT d.Name
FROM Driver d
JOIN Driver_Rating dr ON d.Driverid = dr.driver_id
WHERE dr.rating > 4;
```

output:

```
Name

Amit Sharma
```

```
  Rajendra Gupta

  Arun Kumar

  Nisha Singh

  Rajat Desai
```

--18.Retrieve the drivers who have not received any ratings:

```
SELECT d.Name
FROM Driver d
LEFT JOIN Driver_Rating dr ON d.Driverid = dr.driver_id
WHERE dr.driver_id IS NULL;
```

output:

```
Name
Rahul Shah
Mala Gupta
```

--19.Retrieve the drivers who have a rating greater than the average rating of all drivers:

```
SELECT d.Name, dr.rating
FROM Driver d
JOIN Driver_Rating dr ON d.Driverid = dr.driver_id


WHERE dr.rating > (
   SELECT AVG(rating)
   FROM Driver_Rating
);
```

output:

```
Name            rating

Rajesh Kumar    4

Amit Sharma     5

Deepa Verma     4

Rajendra Gupta  5
```

--20.Retrieve the drivers and their respective ratings, excluding drivers with the name "John":

```
SELECT d.Name, dr.rating
FROM Driver d
JOIN Driver_Rating dr ON d.Driverid = dr.driver_id
WHERE d.Name <> 'John';
```

output:

```
Name          rating

Rajesh Kumar  4

Amit Sharma   5

Sneha Patel   3

Vikram Singh  2
```

```
 Deepa Verma    4
```

--21.Retrieve the drivers who have a higher rating than any driver with the name "John":

```
SELECT d.Name, dr.rating
FROM Driver d
JOIN Driver_Rating dr ON d.Driverid = dr.driver_id
WHERE dr.rating > (
   SELECT MAX(dr2.rating)
   FROM Driver d2
   JOIN Driver_Rating dr2 ON d2.Driverid = dr2.driver_id
   WHERE d2.Name = 'John'
);
```

output:

```
 Name   rating

 teja    9

 abhi    8
```

--22.Retrieve the total usage count of each promo code:

```
SELECT pc.code, SUM(pc.usage_count) AS total_usage_count
FROM PromoCode pc
GROUP BY pc.code;
```

output:

```
 code                 total_usage_count

 ALLABOARD            0
```

```
BACK2SCHOOL     0

BACKONTHEBUS    0

BIGSTOCKTAKE    0

BLACKFRIDAY     0

CYBER20         0

EARLYBIRD       1
```

--23.Retrieve the promo codes that have not reached their maximum usage limit:

```
SELECT pc.code
FROM PromoCode pc
WHERE pc.usage_count < pc.max_usage;
```

output:

```
code

SUMMER2022

WELCOME20

FREERIDE

SAVEMORE
```

--24.Retrieve the promo codes that have the highest discount:

```
SELECT pc.code, pc.discount
FROM PromoCode pc
```

```
WHERE pc.discount = (SELECT MAX(discount) FROM PromoCode);
```

output:

```
code            discount

FREERIDE        100

HELLO10         100

TAKETHEMBACK    100
```

--25.Retrieve all driver locations along with their corresponding driver details:

```
SELECT dl.location_id,  dl.latitude, dl.longitude
FROM Driver_Location dl
JOIN Driver d ON dl.driver_id = d.Driverid;
```

output:

```
location_id   latitude    longitude

401           13.634978   79.414733

402           13.23814    79.507612

403           13.647468   79.50805

404           13.63527    79.421337

405           13.676791   79.418509
```

--26.Retrieve all drivers who have a specific vehicle type:

```
SELECT *
```

FROM Driver

WHERE vehicle_type = 'Sedan';

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
101	Rajesh Kumar	9876543210	rajesh@example.com	password123	KA01AB1234	Sedan	Available
104	Vikram Singh	9876543213	vikram@example.com	password321	DL09GH3456	Sedan	Available
107	Kavita Shah	9876543216	kavita@example.com	password321	GJ07MN5678	Sedan	Available
110	Arun Kumar	9876543219	arun@example.com	password123	MH06ST7890	Sedan	Available

--27.Retrieve the count of driver locations for each driver:

SELECT d.Name, COUNT(dl.location_id) AS location_count

FROM Driver d

JOIN Driver_Location dl ON d.Driverid = dl.driver_id

GROUP BY d.Name;

output:

Name	location_count
Alok Patel	1

```
Amit Sharma   1

Anil Yadav    1
```

--30.Retrieve the driver locations with latitude greater than a specific value:

```
SELECT *
FROM Driver_Location
WHERE latitude > 40.0;
```

output:

```
location_id   driver_id   location_name   latitude   longitude

199           202         tirupati        1.9        1.6

200           203         pileru          1.7        1.4
```

--31.Retrieve all drivers with their total location count sorted by count in descending order:

```
SELECT d.Name, COUNT(dl.location_id) AS location_count
FROM Driver d
LEFT JOIN Driver_Location dl ON d.Driverid = dl.driver_id
GROUP BY d.Name
ORDER BY location_count DESC;
```

output:

```
Name          location_count

Alok Patel    1
```

Amit Sharma	1
Anil Yadav	1
Arun Kumar	1
Avinash Gupta	1

--32. query to get all driver details along with their location:

SELECT Driver.*, Driver_Location.location_name

FROM Driver

JOIN Driver_Location ON Driver.Driverid = Driver_Location.driver_id;

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status	location_name
101	Rajesh Kumar	9876543210	rajesh@example.com	password123	KA01AB1234	Sedan	Available	Alipiri
102	Amit Sharma	9876543211	amit@example.com	password456	MH02CD5678	Hatchback	Available	Tiruchanur
103	Sneha Patel	9876543212	sneha@example.com	password789	GJ05EF9012	SUV	Available	Renigunta
104	Vikram Singh	9876543213	vikram@example.com	password321	DL09GH3456	Sedan	Available	Padmavathi Temple

--33.query to find drivers with a specific vehicle number:

```
SELECT *
FROM Driver
WHERE vehicle_number = (
  SELECT vehicle_number
  FROM Driver
  WHERE Driverid = 1
);
```

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
1	teja	8309669947	teja242@gmail.com	123	242	auto	4
2	abhi	5864946464	abhi22@gmail.com	456	232	car	6

--34. query to find drivers with a higher driver ID than a specific driver:

```
SELECT *
FROM Driver d1
WHERE d1.Driverid > (
  SELECT Driverid
  FROM Driver
  WHERE Name = 'John'
);
```

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
1	teja	8309669947	teja242@gmail.com	123	242	auto	4

2	abhi	5864946464	abhi22@gmail.com	456	232	car	6

--35.query to get drivers and their location with a specific vehicle type and location name:

```
SELECT Driver.*

FROM Driver

JOIN Driver_Location ON Driver.Driverid = Driver_Location.driver_id

WHERE Driver.vehicle_type = 'SUV'

AND Driver_Location.location_name = 'City Center';
```

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
1	teja	8309669947	teja242@gmail.com	123	242	auto	4
2	abhi	5864946464	abhi22@gmail.com	456	232	car	6

--36.Retrieve the payment methods along with the corresponding user details:

```
SELECT u.*, pm.*

FROM Rapido_User u

JOIN User_Payment_Method pm ON u.Users_id = pm.users_id;
```

output:

Users_id	Name	phone_number	email	password	address	date_of_birth	registration_date	account_status	payment_method_id	users_id	payment_method_type	expiry_date
505	John	1234567890	john.doe@gmail.com	john123	123 Main St,	01-01-1990	01-05-2022	active	1103	505	Credit Card	31-05-2024

					City							
506	Smitha	9876543210	smitha@gmail.com	smith456	456 Elm St, City	15-02-1995	05-05-2022	active	1104	506	Debit Card	15-12-2023
507	David	5551234567	davidjohnson@gmail.com	david789	789 Oak St, City	20-08-1988	10-05-2022	active	1105	507	PayPal	28-02-2025
508	Devansh	4449876543	devansh@gmail.com	devabc	987 Pine St, City	12-04-1992	12-05-2022	active	1106	508	Bank Transfer	20-08-2026

--37.Retrieve the users who have payment methods with expiry dates after a specific date:

```
SELECT u.*
FROM Rapido_User u
JOIN User_Payment_Method pm ON u.Users_id = pm.users_id
WHERE pm.expiry_date > '2023-06-08';
```

output:

Users_id	Name	phone_number	email	password	address	date_of_birth	registration_date	account_status
505	John	1234567890	john.doe@gmail.com	john123	123 Main St, City	01-01-1990	01-05-2022	active

506	Smitha	9876543210	smitha@gmail.com	smith456	456 Elm St, City	15-02-1995	05-05-2022	active
507	David	5551234567	davidjohnson@gmail.com	david789	789 Oak St, City	20-08-1988	10-05-2022	active
508	Devansh	4449876543	devansh@gmail.com	devabc	987 Pine St, City	12-04-1992	12-05-2022	active

--38.Retrieve the users and their payment method types in alphabetical order:

```
SELECT u.Name, pm.payment_method_type
FROM Rapido_User u
JOIN User_Payment_Method pm ON u.Users_id = pm.users_id
ORDER BY u.Name ASC;
```

output:

Name	payment_method_type
Alexander Hughes	Debit Card
Amanda Taylor	Credit Card
Andrew Clark	Debit Card
Charlotte Patterson	Bank Transfer

--39.Retrieve the count of users who have payment methods:

```
SELECT COUNT(DISTINCT u.Users_id) AS user_count
FROM Rapido_User u
JOIN User_Payment_Method pm ON u.Users_id = pm.users_id;
```

output:

```
user_count
25
```

--40.Retrieve all the payments made by a specific user:

```
SELECT * FROM Ride
JOIN Payment ON Ride.ride_id = Payment.ride_id;
```

output:

ride_id	users_id	driver_id	start_location	end_location	ride_status	fare_amount	payment_id	users_id	ride_id	amount	payment_date	payment_status
1	505	101	Tirupati Railway Station	Tirumala Temple	Completed	100	1103	505	1	15.5	01-05-2022	paid
2	506	102	Tirupati Bus Stand	Sri Venkateswara Zoological Park	Completed	150	1104	506	2	10.25	05-05-2022	paid
3	507	103	Renigunta Airport	Chandragiri Fort	Completed	200	1105	507	3	8.75	10-05-2022	paid
4	508	104	Tirupati Main Road	Kapila Theertham Waterfall	Completed	120	1106	508	4	12	12-05-2022	paid

--41.Retrieve the total earnings of all drivers:

```
SELECT driver_id, SUM(earnings_amount) AS total_earnings
FROM Driver_Earnings
GROUP BY driver_id;
```

output:

```
driver_id   total_earnings

101         50

102         45

103         60

104         55

105         70
```

--42.Retrieve the drivers who have not made any earnings yet:

```
SELECT Driver.Driverid, Driver.Name
FROM Driver
LEFT JOIN Driver_Earnings ON Driver.Driverid = Driver_Earnings.driver_id
WHERE Driver_Earnings.driver_id IS NULL;
```

output:

```
Driverid   Name

131        Sarika Sharma

132        Prakash Yadav

133        Anita Verma
```

--43.Retrieve the drivers who have made earnings greater than a specific amount

```
SELECT Driver.*
```

```
FROM Driver

JOIN Driver_Earnings ON Driver.Driverid = Driver_Earnings.driver_id

WHERE Driver_Earnings.earnings_amount >500;
```

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
1	teja	8309669947	teja242@gmail.com	123	242	auto	4
2	abhi	5864946464	abhi22@gmail.com	456	232	car	6

--44.Retrieve the drivers who have the same vehicle type as a specific driver:

```
SELECT d1.*

FROM Driver d1

JOIN Driver d2 ON d1.vehicle_type = d2.vehicle_type

WHERE d2.Driverid =101;
```

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
101	Rajesh Kumar	9876543210	rajesh@example.com	password123	KA01AB1234	Sedan	Available
104	Vikram Singh	9876543213	vikram@example.com	password321	DL09GH3456	Sedan	Available
107	Kavita Shah	9876543216	kavita@example.com	password321	GJ07MN5678	Sedan	Available

110	Arun Kumar	9876543219	arun@example.com	password123	MH06ST7890	Sedan	Available

--45.Retrieve the drivers who have the highest earnings amount:

SELECT Driver.*

FROM Driver

JOIN Driver_Earnings ON Driver.Driverid = Driver_Earnings.driver_id

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
101	Rajesh Kumar	9876543210	rajesh@example.com	password123	KA01AB1234	Sedan	Available
102	Amit Sharma	9876543211	amit@example.com	password456	MH02CD5678	Hatchback	Available
103	Sneha Patel	9876543212	sneha@example.com	password789	GJ05EF9012	SUV	Available
104	Vikram Singh	9876543213	vikram@example.com	password321	DL09GH3456	Sedan	Available
105	Deepa Verma	9876543214	deepa@example.com	password654	KA03IJ7890	Hatchback	Available

--46.query to find drivers with a specific vehicle number:

SELECT *

FROM Driver

WHERE vehicle_number = (

  SELECT vehicle_number

```
   FROM Driver
   WHERE Driverid = 1
);
```

output:

Driverid	Name	phone_number	email	password	vehicle_number	vehicle_type	availability_status
1	teja	8309669947	teja242@gmail.com	123	242	auto	4
2	abhi	5864946464	abhi22@gmail.com	456	232	car	6

--47.Retrieve the drivers who have a higher rating than any driver with the name "John":

```
SELECT d.Name, dr.rating
FROM Driver d
JOIN Driver_Rating dr ON d.Driverid = dr.driver_id
WHERE dr.rating > (
   SELECT MAX(dr2.rating)
   FROM Driver d2
   JOIN Driver_Rating dr2 ON d2.Driverid = dr2.driver_id
   WHERE d2.Name = 'John'
);
```

output:

Name	rating
teja	9
abhi	8

--48.Retrieve the payment methods along with the corresponding user details:

```
SELECT u.*, pm.*
FROM Rapido_User u
JOIN User_Payment_Method pm ON u.Users_id = pm.users_id;
```

output:

Users_id	Name	phone_number	email	password	address	date_of_birth	registration_date	account_status	payment_method_id	users_id	payment_method_type	expiry_date
505	John	1234567890	john.doe@gmail.com	john123	123 Main St, City	01-01-1990	01-05-2022	active	1103	505	Credit Card	31-05-2024
506	Smitha	9876543210	smitha@gmail.com	smith456	456 Elm St, City	15-02-1995	05-05-2022	active	1104	506	Debit Card	15-12-2023
507	David	5551234567	davidjohnson@gmail.com	david789	789 Oak St, City	20-08-1988	10-05-2022	active	1105	507	PayPal	28-02-2025
508	Devansh	4449876543	devansh@gmail.com	devabc	987 Pine St, City	12-04-1992	12-05-2022	active	1106	508	Bank Transfer	20-08-2026

--49.Retrieve all driver locations along with their corresponding driver details:

```
SELECT dl.location_id,  dl.latitude, dl.longitude
FROM Driver_Location dl
JOIN Driver d ON dl.driver_id = d.Driverid;
```

output:

```
location_id   latitude    longitude

401           13.634978   79.414733

402           13.23814    79.507612

403           13.647468   79.50805

404           13.63527    79.421337

405           13.676791   79.418509
```

--50.Get the total number of drivers for each availability status:

```
SELECT availability_status, COUNT(*) AS DriverCount
FROM Driver
GROUP BY availability_status;
```

output:

```
availability_status     DriverCount
Available                35
```

CHAPTER – 4

CONCLUSION AND FUTURE WORK :

4.1 Conclusion:

In conclusion, the Rapido DBMS project has successfully accomplished its objectives and demonstrated its effectiveness as a high-performance and scalable database management system. Throughout the project, various features and optimizations were implemented, resulting in improved data retrieval and storage efficiency.

One of the key achievements of the Rapido DBMS project is its ability to handle large volumes of data with high throughput. The system incorporates advanced indexing techniques, data partitioning, and parallel processing capabilities, enabling it to handle complex queries and transactions efficiently.

Additionally, the Rapido DBMS project prioritized data security and integrity. Robust authentication mechanisms, encryption techniques, and access control mechanisms were implemented to ensure that sensitive data is protected from unauthorized access or tampering.

The project also focused on providing a user-friendly interface and comprehensive documentation. The Rapido DBMS system offers intuitive query language support and a user-friendly graphical interface, making it easier for developers and database administrators to interact with the system and perform various tasks.

Furthermore, the Rapido DBMS project emphasized extensibility and compatibility. It provides support for a wide range of data types, allowing users to work with diverse data formats. Moreover, the system is designed to integrate seamlessly with existing software infrastructure, enabling easy adoption and integration into different applications.

While the Rapido DBMS project has achieved significant milestones, there is always room for further improvement. Future enhancements could involve incorporating machine learning algorithms for query optimization, enhancing fault tolerance mechanisms, and exploring new data storage technologies to handle even larger datasets.

Overall, the Rapido DBMS project has delivered a powerful and efficient database management system that addresses the challenges of managing large-scale data. Its performance, scalability, security, and user-friendly interface make it a valuable asset for organizations seeking a robust and reliable solution for their data management needs.

4.2 Future Work:

If you're looking for potential future work or enhancements for the "Rapido" project in the field of database management systems (DBMS), here are a few suggestions:

Query Optimization: Focus on improving the query optimization techniques used by Rapido. This could involve developing new algorithms or strategies to enhance the efficiency of query execution and reduce response times.

Distributed Database Support: Extend Rapido's capabilities to support distributed database systems. This would involve implementing mechanisms for data partitioning, replication, and distributed query processing to handle large-scale and distributed data environments.

Security and Privacy: Enhance the security features of Rapido by incorporating encryption, access control mechanisms, and data anonymization techniques. This would help protect sensitive data stored in the database and ensure compliance with privacy regulations.

Concurrency Control: Improve Rapido's concurrency control mechanisms to handle concurrent transactions effectively. Explore advanced techniques such as multi-version concurrency control (MVCC) or optimistic concurrency control to enhance performance and ensure transactional consistency.

Data Warehousing and Analytics: Extend Rapido to support data warehousing and analytics functionalities. This could involve incorporating features like online analytical processing (OLAP), data mining algorithms, and integration with popular analytics tools to enable efficient data analysis.

Scalability and Performance: Optimize Rapido's architecture and data structures to improve scalability and performance. This may involve implementing techniques such as indexing, caching, and parallel processing to handle larger datasets and increasing query loads.

Replication and High Availability: Implement replication mechanisms in Rapido to provide high availability and fault tolerance. This would involve ensuring data redundancy and designing strategies for replication, synchronization, and failover.

User Interface and Visualization: Enhance the user interface of Rapido to provide intuitive and interactive data exploration capabilities. Incorporate visualizations, dashboards, and reporting tools to facilitate data understanding and decision-making.

Integration and Interoperability: Explore ways to integrate Rapido with other database systems, data sources, or applications. This could involve supporting standard database interfaces and protocols, such as ODBC or JDBC, or providing APIs for seamless integration with external systems.

Data Compression and Storage Optimization: Develop techniques for efficient data compression and storage optimization within Rapido. This would help reduce storage requirements, enhance data retrieval performance, and minimize costs associated with database management.

Remember to prioritize the areas that align with your project goals, resources, and the specific needs of your target users.