REPORT FOR FAST WHEELS DATABASE PROJECT

Prepared by
Syed Muhammad Mubashir Hassan Gillani
Fatima Mushtaq

Prepared for
Salman Muneer

June 30, 2024

June 30, 2024

Salman Muneer Lecturer
University of Central Punjab
Khayaban-e-Jinnah Road
Lahore,Punjab

Respected Sir Salam:

We are happy to inform you that we have finished our project on designing a database system for Fast Wheels.

The database system was carefully made to manage different parts of the Fast Wheels platform efficiently. It includes detailed information about drivers, passengers, rides, payments, vehicle maintenance, and more, providing a complete solution for the service.

We are very grateful for your guidance and support throughout this project. Your insights and expertise have been very helpful to us.

Thank you again for your mentorship.

Sincerely,
Syed Muhammad Mubashir Hassan Gillani
L1F22BSCS0282
Fatima Mushtaq
L1F22BSCS0272
Students
University of Central Punjab.

Contents

EXECUTIVE SUMMARY

The Fast Wheels database system is designed to manage all aspects of a ride-sharing service efficiently. It keeps detailed records of drivers, passengers, rides, payments, and vehicle maintenance. By tracking driver schedules, handling various payment methods, and storing feedback, the system ensures smooth operations and helps improve service quality.

By organizing all this information in one place, the Fast Wheels database helps provide better service to users and makes managing the ride-sharing platform easier. It automates many tasks, reducing manual work and errors, and ensures accurate data management, which leads to enhanced user satisfaction and a more reliable ride-sharing service.

INTRODUCTION

Project Description

The Fast Wheels database system is a project designed to manage all parts of a ride-sharing service. It helps keep track of important details about drivers, passengers, rides, payments, and vehicle maintenance.

This system makes it easy to handle driver schedules, different payment methods, and passenger feedback. By storing all this information in one place, it helps the ride-sharing service run smoothly and efficiently. This leads to better service for users and makes managing the platform easier and more reliable.

Scope

The Fast Wheels database system covers all essential parts of a ride-sharing service. It includes:

1. **Driver Management:** Keeping track of driver details, schedules, and licenses.
2. **Passenger Management**: Storing passenger information and feedback.
3. **Ride Management**: Organizing ride details, including routes and statuses.
4. **Payment Processing**: Handling different payment methods and recording transactions.
5. **Vehicle Management**: Maintaining vehicle information and service records.
6. **Feedback and Support**: Collecting user feedback and managing support requests.

This system ensures smooth operation, accurate data management, and improved service quality for the ride-sharing platform.

FEATURES

Entities and their Attributes

Driver

- `driverID`: **Unique identifier for each driver.**
- `name`: **Driver's name.**
- `contactNo`: **Driver's contact number.**
- `DrivingLicenseNo`: **Driver's license number.**
- `insuranceNo`: **Driver's insurance number.**
- `VehicleID`: **Foreign key linking to the vehicle the driver operates.**

Passenger

- `passengerID`: **Unique identifier for each passenger.**
- `name`: **Passenger's name.**
- `CardDetails`: **Passenger's card details.**
- `NoOfPassengers`: **Number of passengers traveling.**

DriverSchedule

- `ScheduleID`: **Unique identifier for each schedule entry.**
- `WorkingDate`: **Date of the work schedule.**
- `StartTime`: **Start time of the driver's shift.**
- `EndTime`: **End time of the driver's shift.**
- `Status`: **Status of the schedule (e.g., Active).**
- `DriverID`: **Foreign key linking to the respective driver.**

Payment

- `paymentID`: Unique identifier for each payment.
- `receivingMethod`: Method by which payment was received.
- `amount`: Amount of payment.
- `payingMethod`: Method used to make the payment.
- `DriverID`: Foreign key linking to the driver receiving the payment.
- `PassengerID`: Foreign key linking to the passenger making the payment.

Vehicle

- `vehicleID`: Unique identifier for each vehicle.
- `Name`: Name of the vehicle.
- `Type`: Type of the vehicle (e.g., Sedan, SUV).
- `NoOfSeats`: Number of seats in the vehicle.

VehicleMaintenance

- `SerialNo`: Unique identifier for each maintenance record.
- `DateOfMaintenance`: Date when maintenance was performed.
- `TypeOfMaintenance`: Type of maintenance (e.g., Regular Checkup, Major Repair).
- `DescriptionOfWork`: Detailed description of the maintenance work.
- `cost`: Cost of the maintenance.
- `NextDate`: Date for the next scheduled maintenance.
- `VehicleID`: Foreign key linking to the vehicle that was maintained.

Location

- `locationID`: Unique identifier for each location entry.
- `PickupLocation`: Pickup location.
- `Destination`: Destination location.
- `PassengerID`: Foreign key linking to the passenger associated with the location.

Promotion

- `PromoSerial`: Unique identifier for each promotion.
- `PromoCode`: Promotional code.
- `ExpirationDate`: Expiry date of the promotion.

PaymentDetails

- `SerialNo`: Unique identifier for each payment detail entry.
- `discountedAmount`: Discounted amount applied to the payment.
- `OverTimePrice`: Price charged for overtime.
- `TotalPrice`: Total price after applying discounts and overtime charges.
- `PaymentID`: Foreign key linking to the payment record.
- `PromoSerial`: Foreign key linking to the promotion used.

Rating

- `RateID`: Unique identifier for each rating entry.
- `driverRating`: Rating given to the driver.
- `passengerRating`: Rating given to the passenger.
- `UserGeneralRating`: General rating of the user.
- `DriverID`: Foreign key linking to the rated driver.
- `PassengerID`: Foreign key linking to the rated passenger.
- `UserID`: Foreign key linking to the user who provided the rating.

Feedback

- `FeedbackNo`: Unique identifier for each feedback entry.
- `rideSpecificFeedback`: Feedback specific to a particular ride.
- `generalUserFeedback`: General feedback from the user.
- `UserID`: Foreign key linking to the user who provided the feedback.
- `PassengerID`: Foreign key linking to the passenger providing the feedback.

User

- `UserID`: Unique identifier for each user.
- `UserName`: Name of the user.
- `PassengerID`: Foreign key linking to the passenger (if the user is a passenger).
- `DriverID`: Foreign key linking to the driver (if the user is a driver).

SupportRequest

- `RequestID`: Unique identifier for each support request.
- `IssueDescription`: Description of the issue reported by the user.
- `UserID`: Foreign key linking to the user who made the support request.

Ride

- `RideID`: Unique identifier for each ride.
- `RideStatus`: Status of the ride (e.g., Completed).
- `RideDate`: Date of the ride.
- `DriverID`: Foreign key linking to the driver of the ride.
- `PassengerID`: Foreign key linking to the passenger of the ride.
- `PaymentDetail`: Foreign key linking to the payment details of the ride.
- `LocationID`: Foreign key linking to the ride's location.
- `FeedbackNo`: Foreign key linking to the feedback for the ride.

Relationship Between Entities

Driver and Vehicle

- Relationship: One-to-Many
- Details: One Driver can drive multiple Vehicles
- Foreign Key: `VehicleID` in the `Driver` table references `VehicleID` in the `Vehicle` table.

Driver and DriverSchedule

- Relationship: One-to-Many
- Details: A driver can have multiple schedules, but each schedule is associated with one driver.
- Foreign Key: `DriverID` in the `DriverSchedule` table references `DriverID` in the `Driver` table.

Driver and Payment

- Relationship: One-to-Many
- Details: A driver can receive multiple payments, but each payment is made to one driver.
- Foreign Key: `DriverID` in the `Payment` table references `DriverID` in the `Driver` table.

Passenger and Payment

- **Relationship: One-to-Many**
- **Details: A passenger can make multiple payments, but each payment is made by one passenger.**
- **Foreign Key: `PassengerID` in the `Payment` table references `PassengerID` in the `Passenger` table.**

Passenger and Location

- **Relationship: One-to-Many**
- **Details: A passenger can have multiple locations associated with their rides, but each location is tied to one passenger.**
- **Foreign Key: `PassengerID` in the `Location` table references `PassengerID` in the `Passenger` table.**

Passenger and Feedback

- **Relationship: One-to-Many**
- **Details: A passenger can provide multiple feedback entries, but each feedback is given by one passenger.**
- **Foreign Key: `PassengerID` in the `Feedback` table references `PassengerID` in the `Passenger` table.**

Vehicle and VehicleMaintenance

- **Relationship: One-to-Many**
- **Details: A vehicle can undergo multiple maintenance procedures, but each maintenance record pertains to one vehicle.**
- **Foreign Key: `VehicleID` in the `VehicleMaintenance` table references `VehicleID` in the `Vehicle` table.**

User and SupportRequest

- **Relationship: One-to-Many**
- **Details: A user can make multiple support requests, but each support request is made by one user.**
- **Foreign Key: `UserID` in the `SupportRequest` table references `UserID` in the `User` table.**

User and Rating

- **Relationship: One-to-Many**
- **Details: A user can give multiple ratings, but each rating is given by one user.**
- **Foreign Key: `UserID` in the `Rating` table references `UserID` in the `User` table.**

Driver and Rating

- **Relationship: One-to-Many**
- **Details: A driver can receive multiple ratings, but each rating is given to one driver.**
- **Foreign Key: `DriverID` in the `Rating` table references `DriverID` in the `Driver` table.**

Passenger and Rating

- **Relationship: One-to-Many**
- **Details: A passenger can receive multiple ratings, but each rating is given to one passenger.**
- **Foreign Key: `PassengerID` in the `Rating` table references `PassengerID` in the `Passenger` table.**

Ride and Driver

- **Relationship: Many-to-One**
- **Details: A ride is assigned to one driver, but a driver can handle multiple rides.**
- **Foreign Key: `DriverID` in the `Ride` table references `DriverID` in the `Driver` table.**

Ride and Passenger

- **Relationship: Many-to-One**
- **Details: A ride is assigned to one passenger, but a passenger can take multiple rides.**
- **Foreign Key: `PassengerID` in the `Ride` table references `PassengerID` in the `Passenger` table.**

Ride and PaymentDetails

- **Relationship: Many-to-One**
- **Details: Each ride has one set of payment details, but a set of payment details can be associated with multiple rides.**
- **Foreign Key: `PaymentDetail` in the `Ride` table references `SerialNo` in the `PaymentDetails` table.**

Ride and Location

- **Relationship: Many-to-One**
- **Details: Each ride has one location associated with it, but a location can be associated with multiple rides.**
- **Foreign Key: `LocationID` in the `Ride` table references `LocationID` in the `Location` table.**

Ride and Feedback

- **Relationship: Many-to-One**
- **Details: Each ride can have one feedback entry associated with it, but a feedback entry can be associated with multiple rides.**
- **Foreign Key: `FeedbackNo` in the `Ride` table references `FeedbackNo` in the `Feedback` table.**

PaymentDetails and Payment

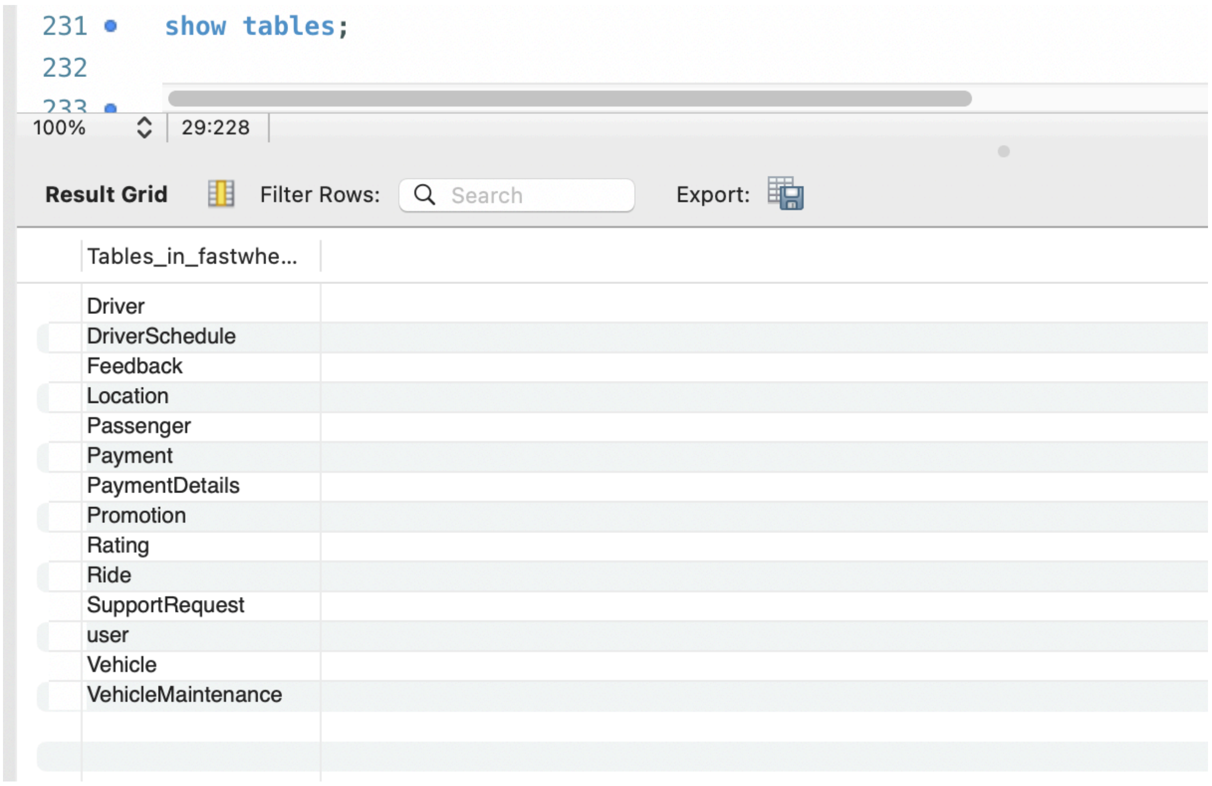
- **Relationship: Many-to-One**
- **Details: A set of payment details is associated with one payment, but one payment can have multiple sets of payment details.**
- **Foreign Key: `PaymentID` in the `PaymentDetails` table references `paymentID` in the `Payment` table.**

PaymentDetails and Promotion

- **Relationship: Many-to-One**
- **Details: A set of payment details can apply one promotion, but a promotion can be used in multiple sets of payment details.**
- **Foreign Key: `PromoSerial` in the `PaymentDetails` table references `PromoSerial` in the `Promotion` table.**

TESTING AND RUNNING

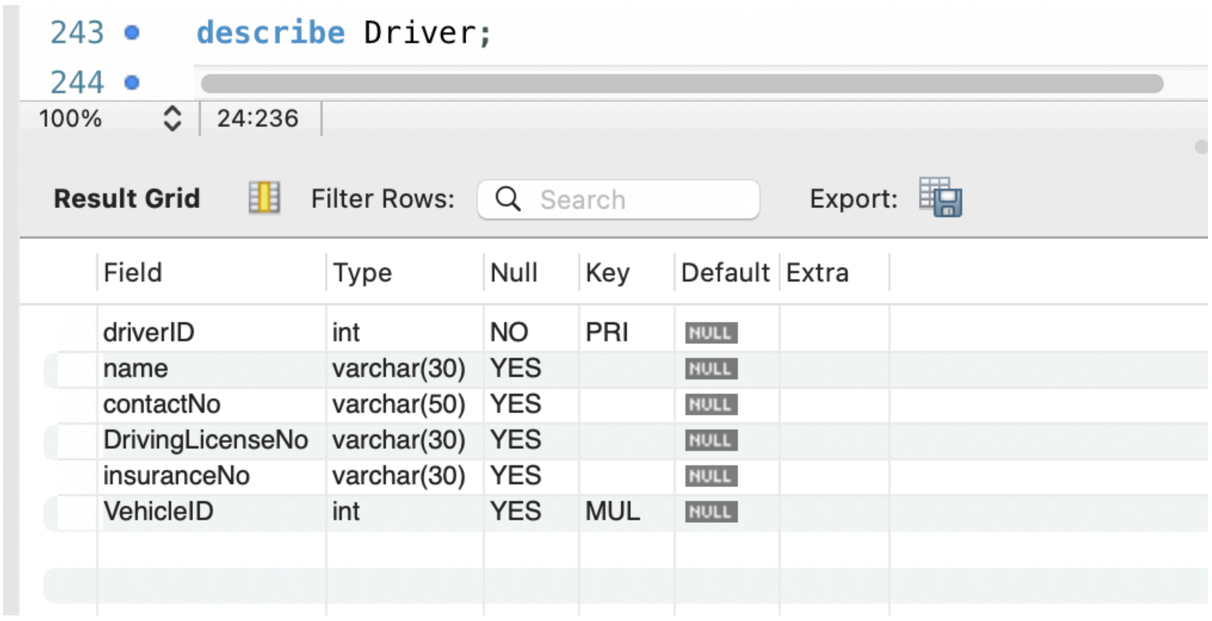
The FastWheels database system is tested and run many times to make sure it works well. Before it starts, we test it a lot to see how it handles things like booking rides and processing payments. This helps us find and fix any problems. After it's up and running, we keep checking it to make sure it keeps working smoothly as more people use it. We update and maintain it regularly to make sure FastWheels can always offer a reliable service to everyone who uses it.



```
show tables;
```

Tables_in_fastwhe...
Driver
DriverSchedule
Feedback
Location
Passenger
Payment
PaymentDetails
Promotion
Rating
Ride
SupportRequest
user
Vehicle
VehicleMaintenance

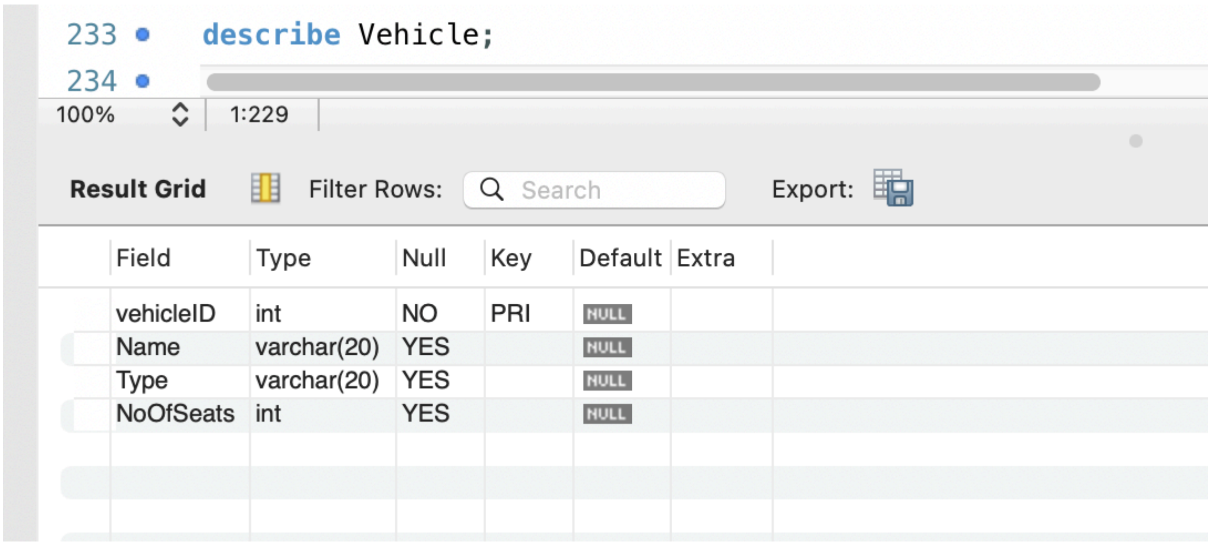
Driver Table



```
describe Driver;
```

Field	Type	Null	Key	Default	Extra
driverID	int	NO	PRI	NULL	
name	varchar(30)	YES		NULL	
contactNo	varchar(50)	YES		NULL	
DrivingLicenseNo	varchar(30)	YES		NULL	
insuranceNo	varchar(30)	YES		NULL	
VehicleID	int	YES	MUL	NULL	

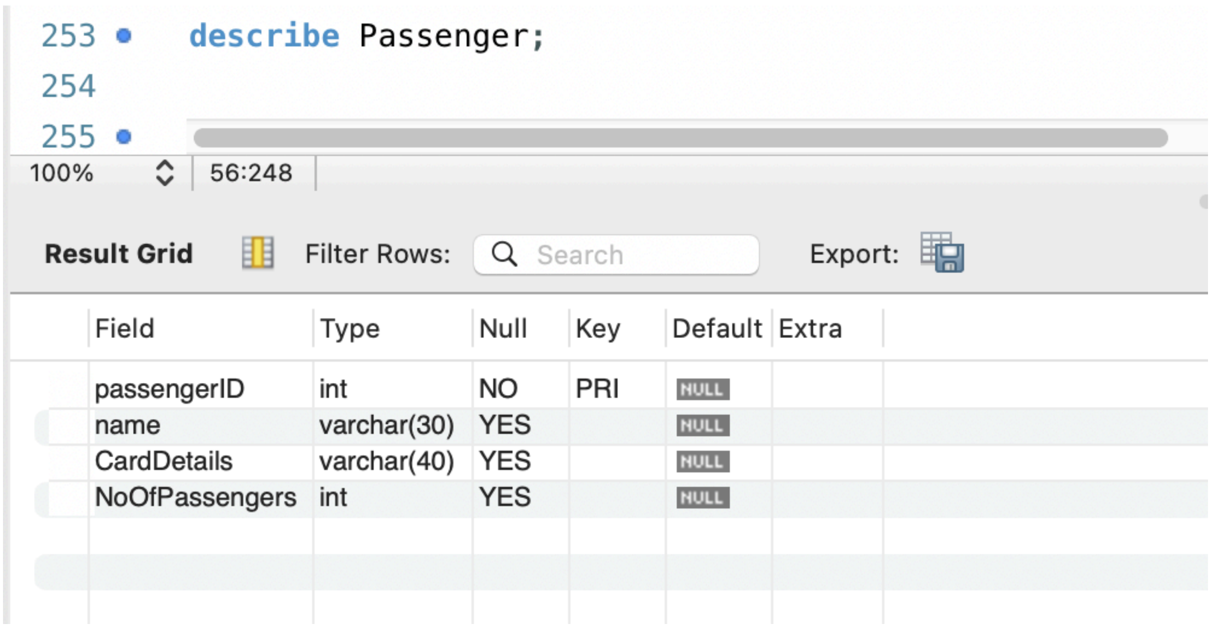
Vehicle Table



```
describe Vehicle;
```

Field	Type	Null	Key	Default	Extra
vehicleID	int	NO	PRI	NULL	
Name	varchar(20)	YES		NULL	
Type	varchar(20)	YES		NULL	
NoOfSeats	int	YES		NULL	

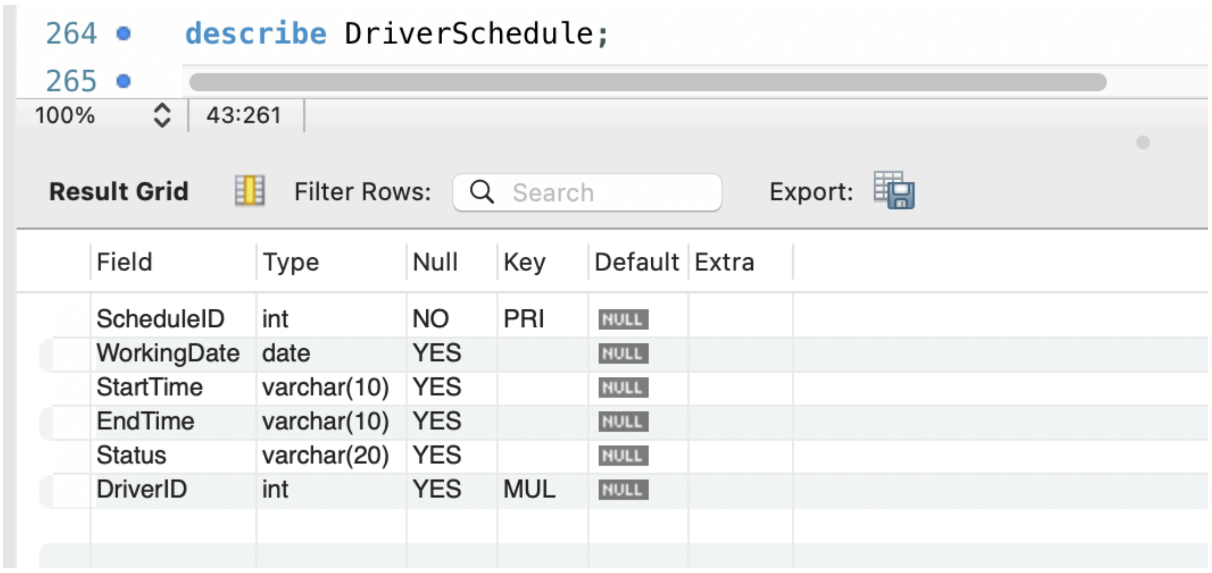
Passenger Table



```
describe Passenger;
```

Field	Type	Null	Key	Default	Extra
passengerID	int	NO	PRI	NULL	
name	varchar(30)	YES		NULL	
CardDetails	varchar(40)	YES		NULL	
NoOfPassengers	int	YES		NULL	

DriverSchedule Table

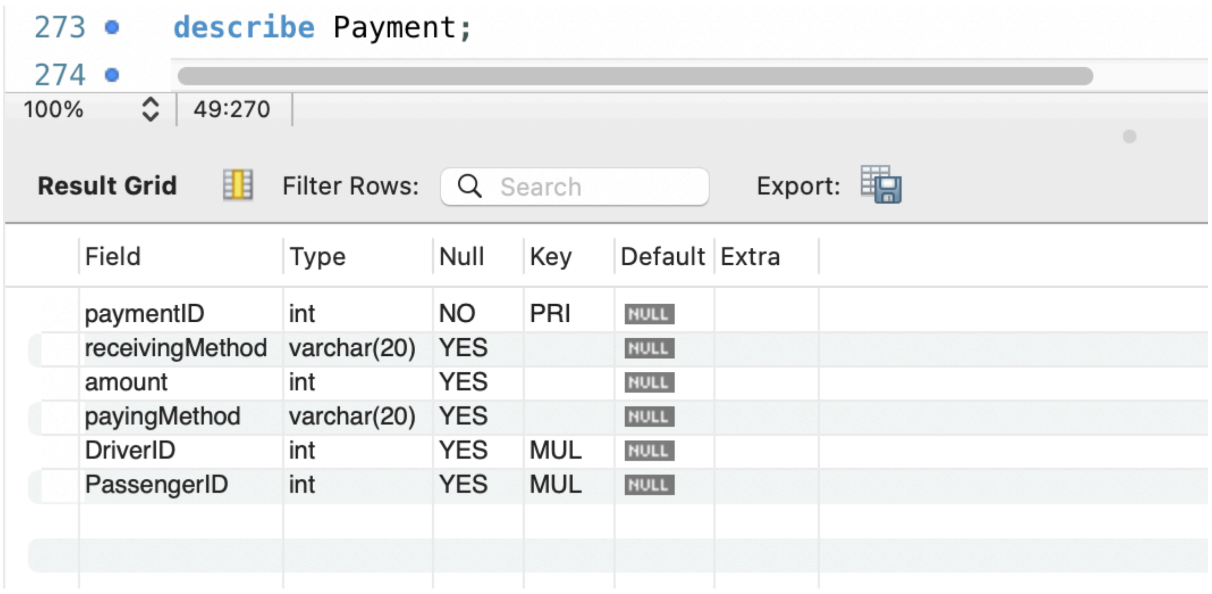


```
describe DriverSchedule;
```

100% 43:261

Field	Type	Null	Key	Default	Extra
ScheduleID	int	NO	PRI	NULL	
WorkingDate	date	YES		NULL	
StartTime	varchar(10)	YES		NULL	
EndTime	varchar(10)	YES		NULL	
Status	varchar(20)	YES		NULL	
DriverID	int	YES	MUL	NULL	

Payment Table

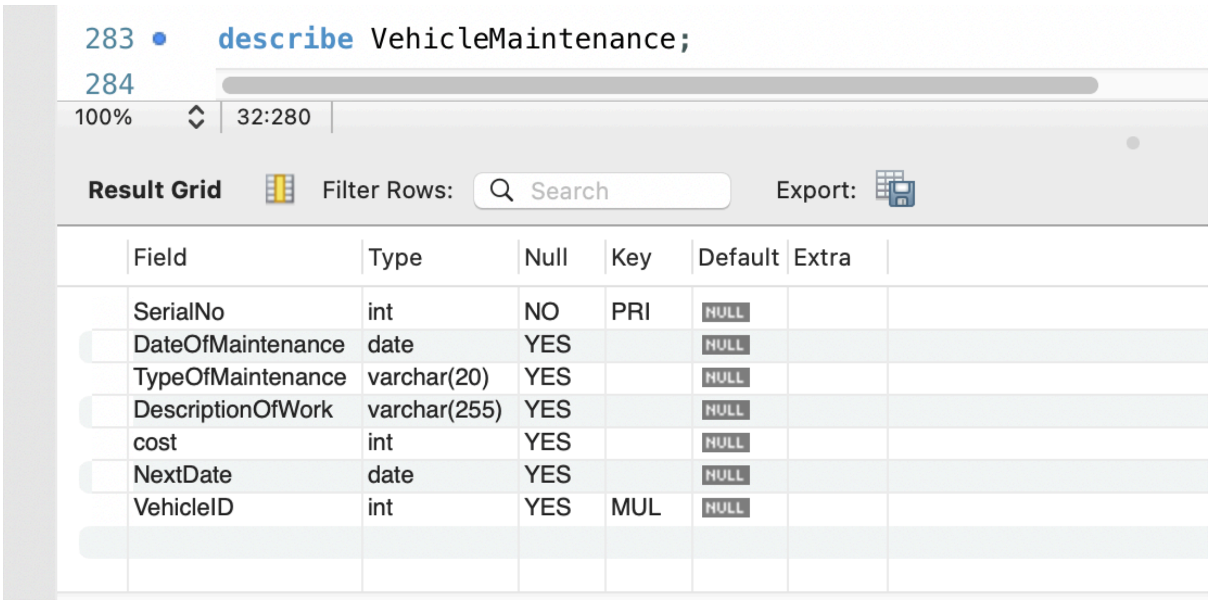


```
describe Payment;
```

100% 49:270

Field	Type	Null	Key	Default	Extra
paymentID	int	NO	PRI	NULL	
receivingMethod	varchar(20)	YES		NULL	
amount	int	YES		NULL	
payingMethod	varchar(20)	YES		NULL	
DriverID	int	YES	MUL	NULL	
PassengerID	int	YES	MUL	NULL	

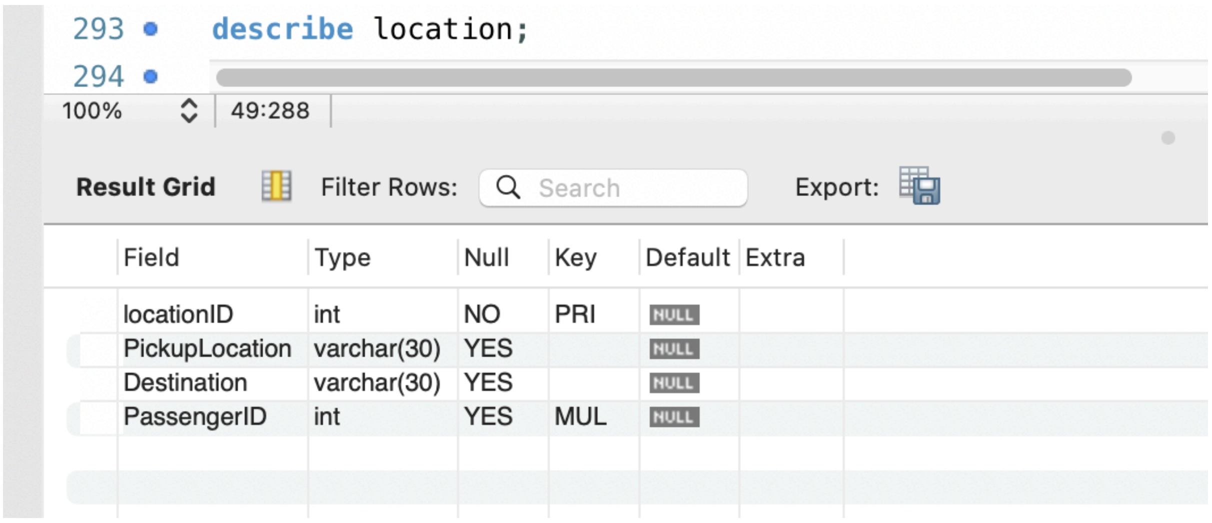
VehicleMaintenance Table



```
describe VehicleMaintenance;
```

Field	Type	Null	Key	Default	Extra
SerialNo	int	NO	PRI	NULL	
DateOfMaintenance	date	YES		NULL	
TypeOfMaintenance	varchar(20)	YES		NULL	
DescriptionOfWork	varchar(255)	YES		NULL	
cost	int	YES		NULL	
NextDate	date	YES		NULL	
VehicleID	int	YES	MUL	NULL	

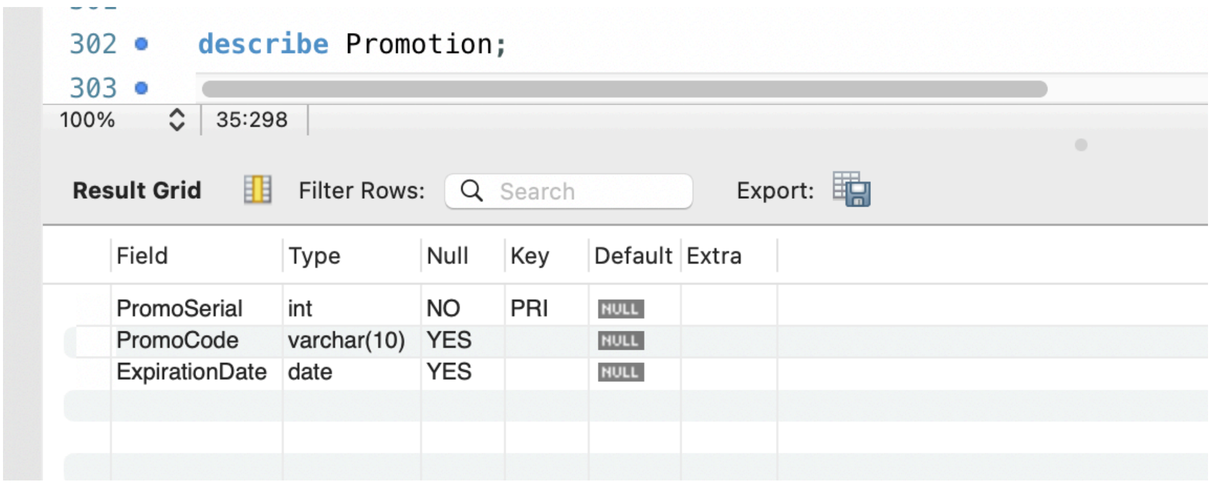
Location Table



```
describe location;
```

Field	Type	Null	Key	Default	Extra
locationID	int	NO	PRI	NULL	
PickupLocation	varchar(30)	YES		NULL	
Destination	varchar(30)	YES		NULL	
PassengerID	int	YES	MUL	NULL	

Promotion Table



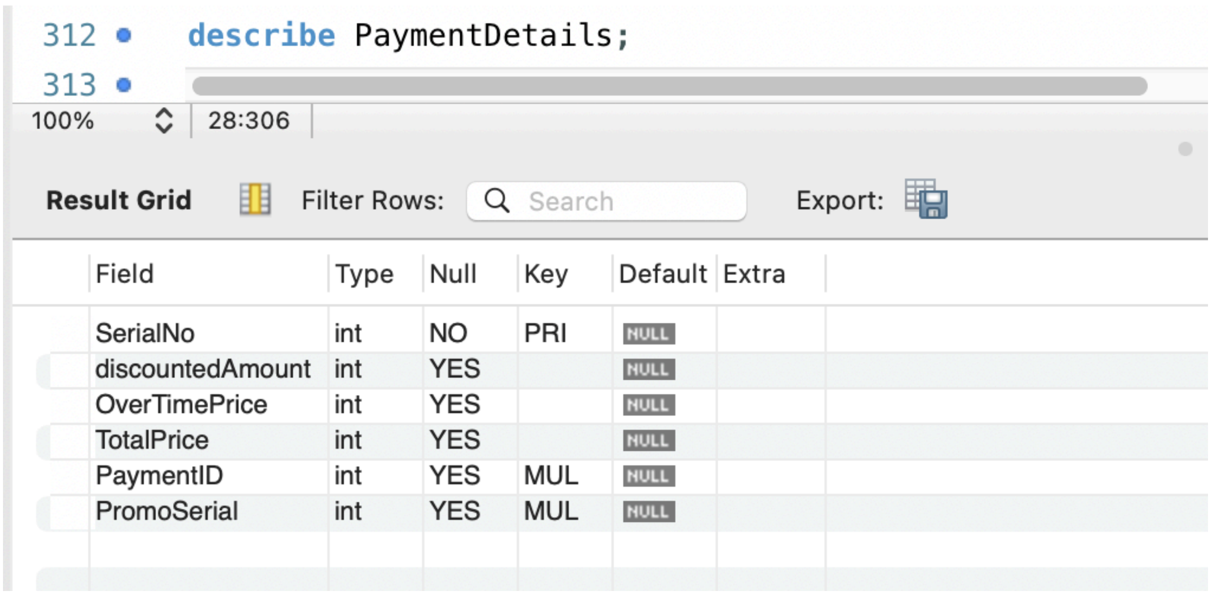
```
describe Promotion;
```

100% 35:298

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
PromoSerial	int	NO	PRI	NULL	
PromoCode	varchar(10)	YES		NULL	
ExpirationDate	date	YES		NULL	

PaymentDetails Table



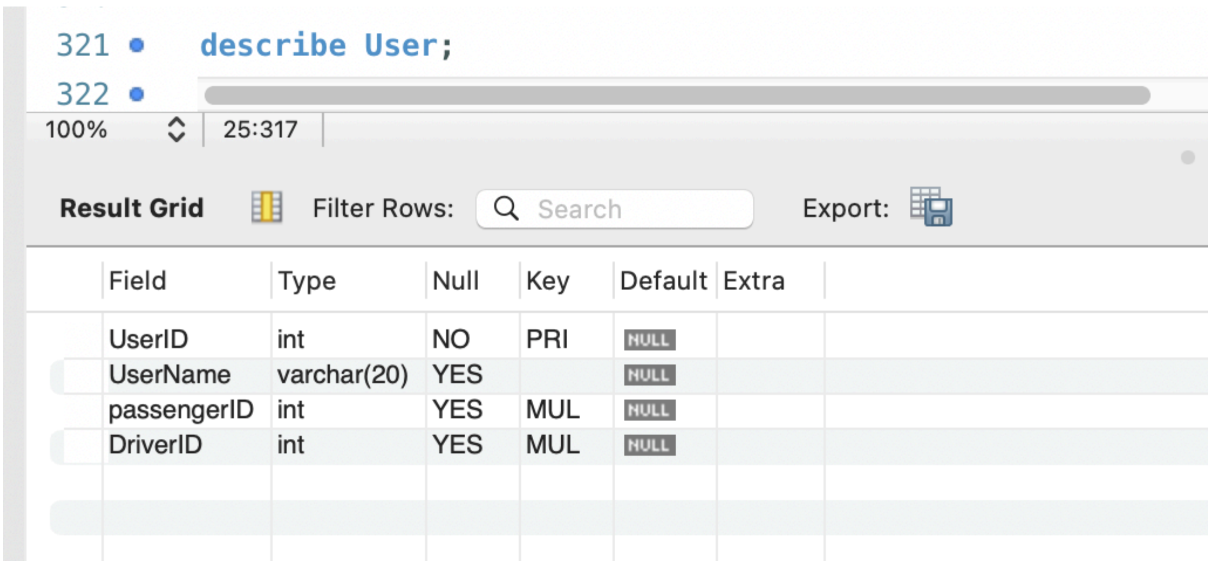
```
describe PaymentDetails;
```

100% 28:306

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
SerialNo	int	NO	PRI	NULL	
discountedAmount	int	YES		NULL	
OverTimePrice	int	YES		NULL	
TotalPrice	int	YES		NULL	
PaymentID	int	YES	MUL	NULL	
PromoSerial	int	YES	MUL	NULL	

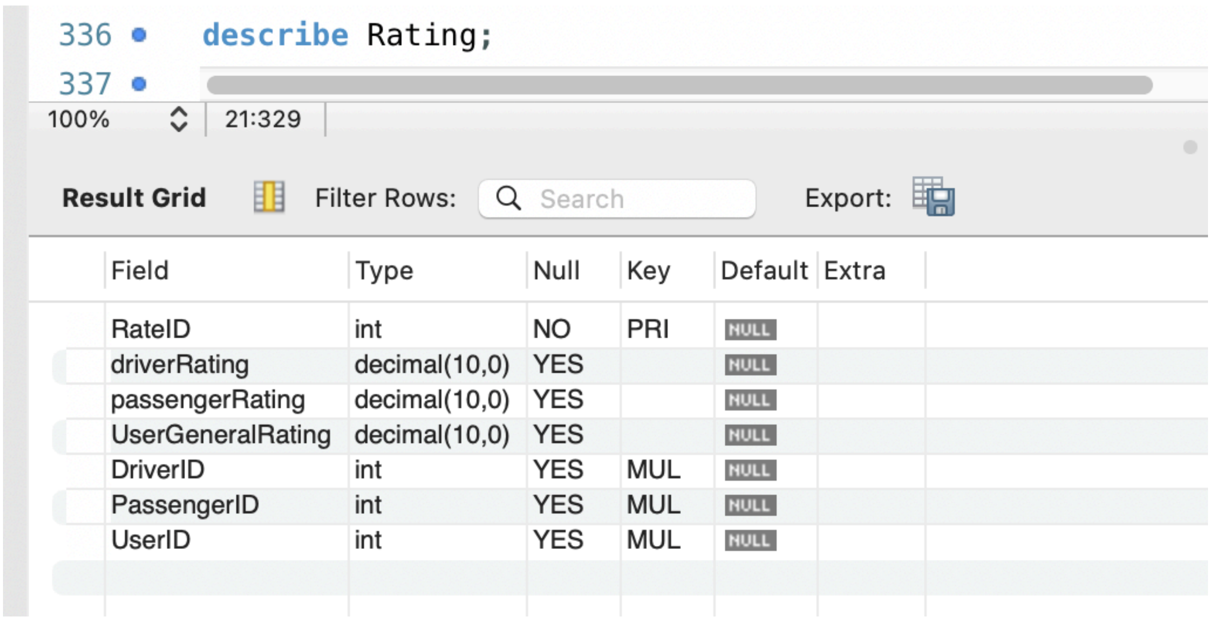
User Table



```
describe User;
```

Field	Type	Null	Key	Default	Extra
UserID	int	NO	PRI	NULL	
UserName	varchar(20)	YES		NULL	
passengerID	int	YES	MUL	NULL	
DriverID	int	YES	MUL	NULL	

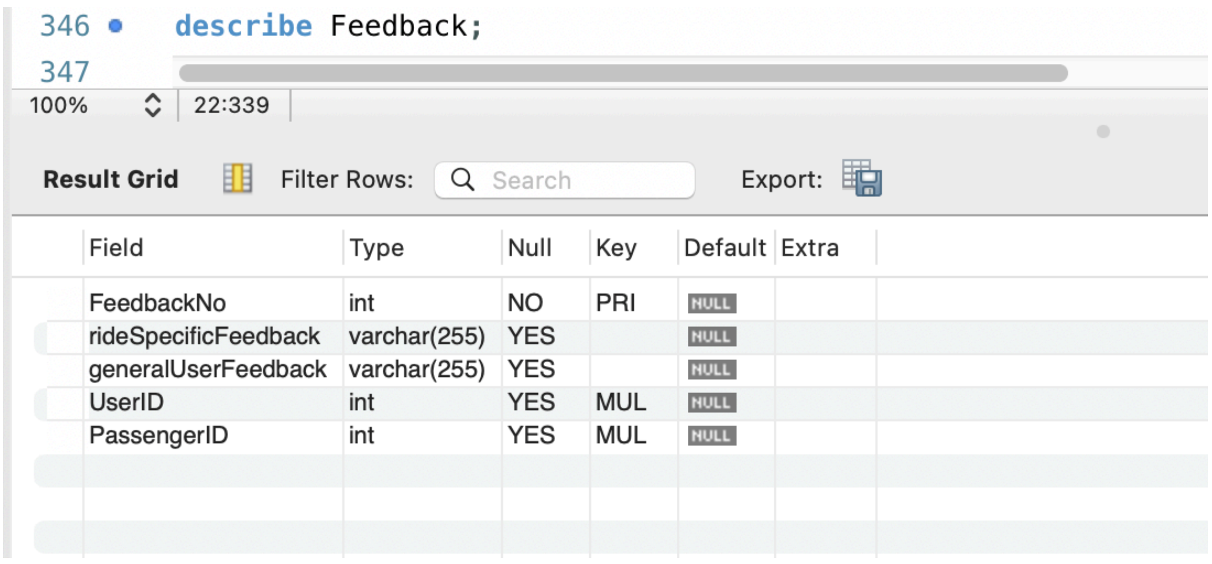
Rating Table



```
describe Rating;
```

Field	Type	Null	Key	Default	Extra
RateID	int	NO	PRI	NULL	
driverRating	decimal(10,0)	YES		NULL	
passengerRating	decimal(10,0)	YES		NULL	
UserGeneralRating	decimal(10,0)	YES		NULL	
DriverID	int	YES	MUL	NULL	
PassengerID	int	YES	MUL	NULL	
UserID	int	YES	MUL	NULL	

Feedback Table

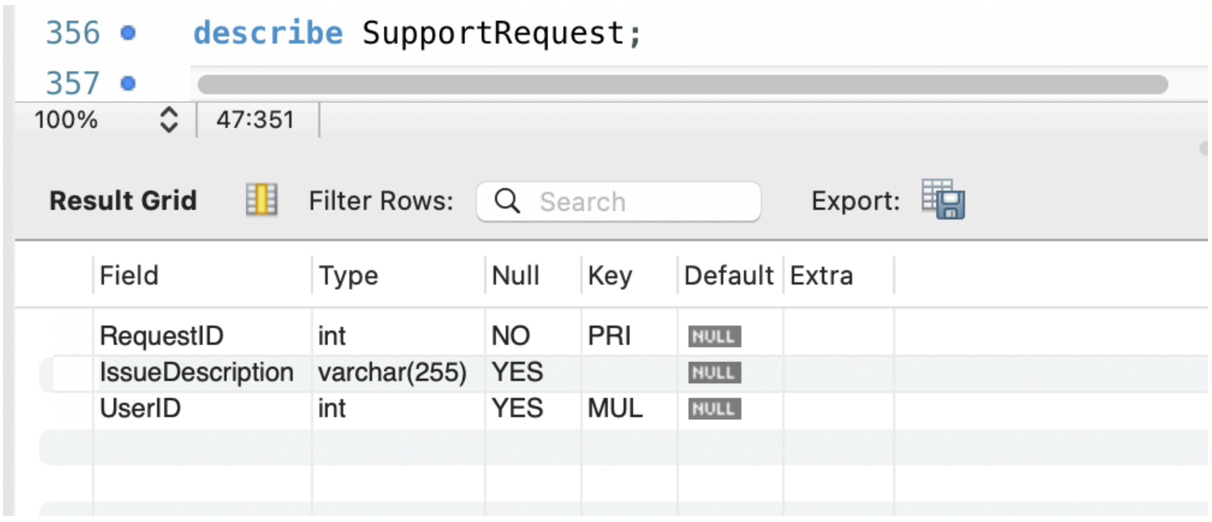


```
describe Feedback;
```

100% 22:339

Field	Type	Null	Key	Default	Extra
FeedbackNo	int	NO	PRI	NULL	
rideSpecificFeedback	varchar(255)	YES		NULL	
generalUserFeedback	varchar(255)	YES		NULL	
UserID	int	YES	MUL	NULL	
PassengerID	int	YES	MUL	NULL	

SupportRequest Table

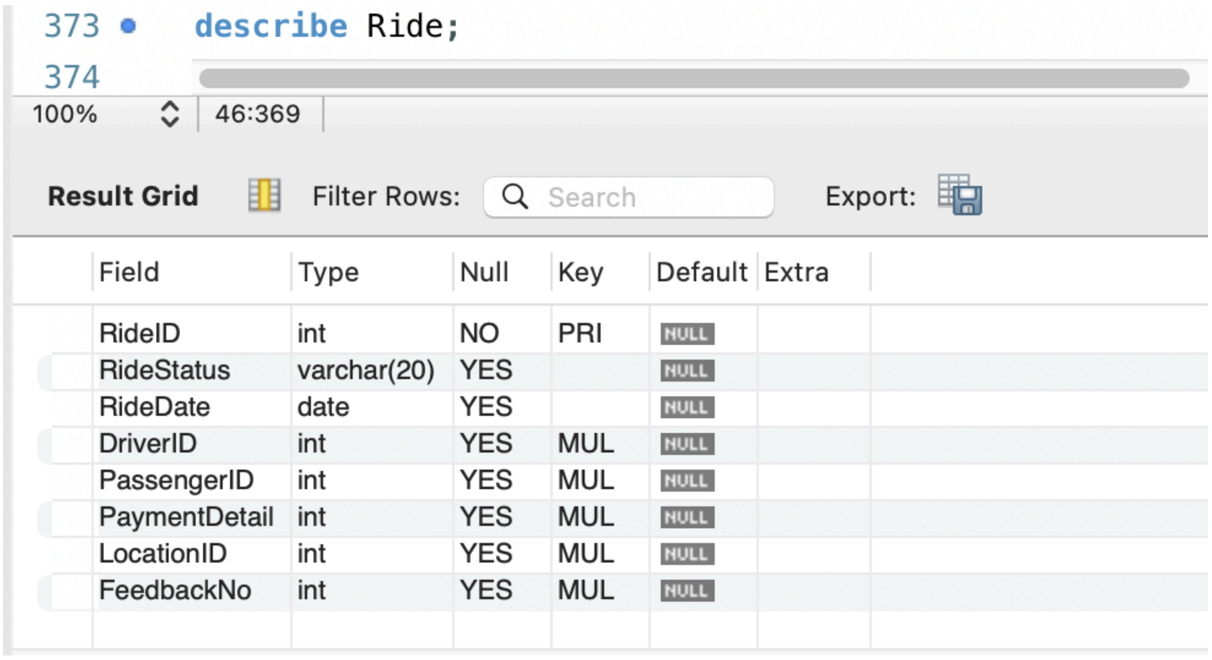


```
describe SupportRequest;
```

100% 47:351

Field	Type	Null	Key	Default	Extra
RequestID	int	NO	PRI	NULL	
IssueDescription	varchar(255)	YES		NULL	
UserID	int	YES	MUL	NULL	

Ride Table



```
describe Ride;
```

100% 46:369

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
RideID	int	NO	PRI	NULL	
RideStatus	varchar(20)	YES		NULL	
RideDate	date	YES		NULL	
DriverID	int	YES	MUL	NULL	
PassengerID	int	YES	MUL	NULL	
PaymentDetail	int	YES	MUL	NULL	
LocationID	int	YES	MUL	NULL	
FeedbackNo	int	YES	MUL	NULL	

BENEFITS

Easy to Manage Information:

- **All-in-One Place**: FastWheels keeps everything about drivers, passengers, rides, payments, vehicles, and maintenance organized in one place, making it simple to find and use.
- **Fewer Mistakes**: By using a structured database, FastWheels makes sure information is accurate and mistakes are less likely.

Better for Customers:

- **Personalized Help**: FastWheels knows a lot about passengers, so they can offer services that feel personal and helpful.
- **Quick Support**: The system handles problems fast, so issues with rides or payments get fixed quickly.

Works Smarter:

- **Schedules and Tracks Well**: FastWheels plans when drivers and vehicles work, making sure everything runs smoothly. They can also track rides in real-time.
- **Keeps Vehicles Safe**: By keeping good records of vehicle upkeep, FastWheels ensures cars are in good shape and less likely to break down.

Manages Money Better:

- **Gets Payments Right**: The database makes sure payments are recorded accurately, so bills are right and money is managed well.
- **Uses Discounts Well**: It helps keep track of discounts and special offers, making marketing campaigns work better.

Grows and Changes Easily:

- **Handles More Business**: As FastWheels grows, the database can grow with it. It can handle more drivers, passengers, and vehicles without a problem.
- **Changes When Needed**: If they want to add new things or change how things work, the system can be updated to fit new ideas.

Follows Rules:

- **Keeps Good Records**: The database helps FastWheels keep the right records, which is important for following laws and rules.
- **Does Things Right**: By being clear about how they work, FastWheels can follow all the rules they need to.

CONCLUSION

In conclusion, the FastWheels database system brings many advantages for managing a transportation service efficiently. It helps keep track of drivers, passengers, rides, payments, vehicles, and maintenance in one organized place, reducing mistakes and improving customer service. With better scheduling, quick support, and accurate financial management, FastWheels can offer personalized services while making smart decisions based on data. As the business grows, the system can expand easily and adapt to new needs, all while keeping information safe and following regulations. Overall, FastWheels is well-equipped to provide a reliable and customer-focused transportation experience.