

---

# **Citywheels Ride Booking Database**

---

## **TABLE OF CONTENTS**

<b>1. Introduction -----</b>	<b>2</b>
<b>2. Executive Summary -----</b>	<b>4</b>
<b>3. Database Design -----</b>	<b>5</b>
<b>4. Sample Data -----</b>	<b>8</b>
<b>5. Business Queries -----</b>	<b>10</b>
<b>6. Conclusion -----</b>	<b>20</b>

---

## **1. INTRODUCTION**

### **1.1 Project Background**

CityWheels is a burgeoning ride-hailing service operating in Pakistan, with ambitious plans for global expansion into GCC countries and beyond. In today's digital economy, data-driven decision-making and efficient management of transportation services require a robust, scalable, and secure database infrastructure. This documentation presents the complete database management system designed specifically for CityWheels's operational needs.

### **1.2 Purpose and Scope**

The CityWheels Ride-Hailing Database System serves as the technological backbone for managing all aspects of the ride-hailing platform, including:

- Driver and passenger profile management
- Ride booking and history tracking
- Payment processing and financial records
- Customer ratings and feedback systems
- Promotional offer management
- Customer support ticketing

### **1.3 Objectives**

This project aims to achieve the following objectives:

- Design a normalized, efficient database structure adhering to 3NF principles
- Implement complex business queries for data analysis and decision-making
- Ensure data integrity through proper constraints and relationships
- Address legal and ethical considerations for cross-border data sharing
- Provide a scalable architecture capable of supporting future growth

## **1.4 Document Structure**

This documentation is organized into the following sections:

- Database Design - ER diagrams and relational schemas
- Schema Documentation - Detailed table structures
- Sample Data - Data population and statistics
- Business Queries - Analytical SQL implementations
- Legal Research - Cross-border data compliance analysis
- Implementation Guide - Installation and usage instructions

## **1.5 Target Audience**

This documentation is intended for:

- Database administrators managing the CityWheels platform
  - Developers integrating applications with the database
  - Business analysts extracting insights for decision-making
  - Legal and compliance teams reviewing data protection measures
  - Academic evaluators assessing the database design
-

## **2. EXECUTIVE SUMMARY**

The CityWheels Ride-Hailing Database System is a comprehensive MySQL-based solution designed to manage all operational aspects of a modern ride-hailing service. Developed as part of the Database Systems course, this project demonstrates professional database design principles, complex query implementation, and real-world business logic application.

### **Key Achievements:**

- 12 normalized tables with proper relationships
  - 500+ sample records across all entities
  - 10 complex analytical business queries
  - Complete ERD and relational schema diagrams
  - Legal research on cross-border data compliance (PDPL & GDPR)
-

### 3. DATABASE DESIGN

#### 3.1 Entity Relationship Diagram (ERD)

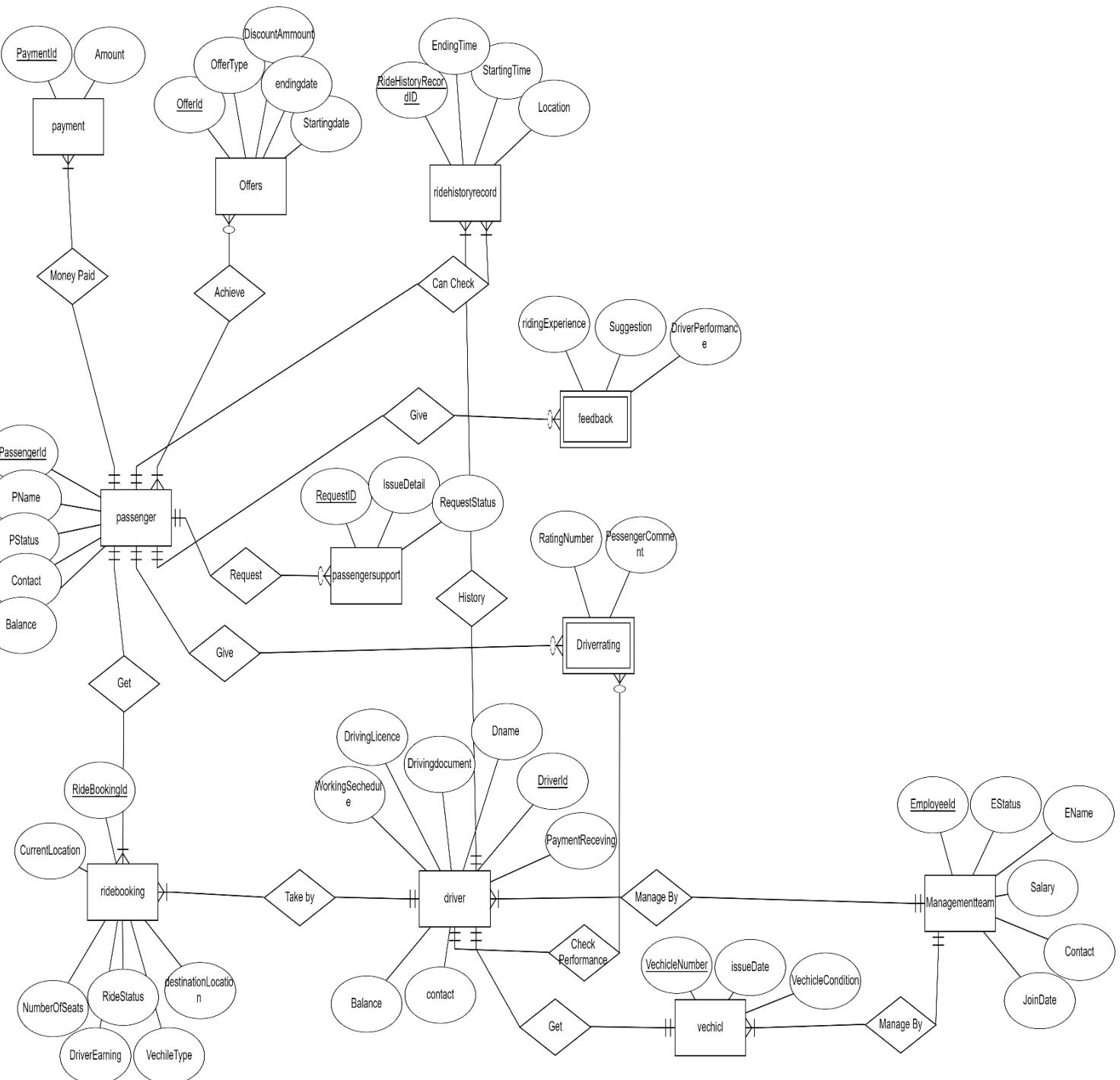


Figure 1: Physical ERD showing all tables, attributes, and relationships

### 3.2 Relational Schema

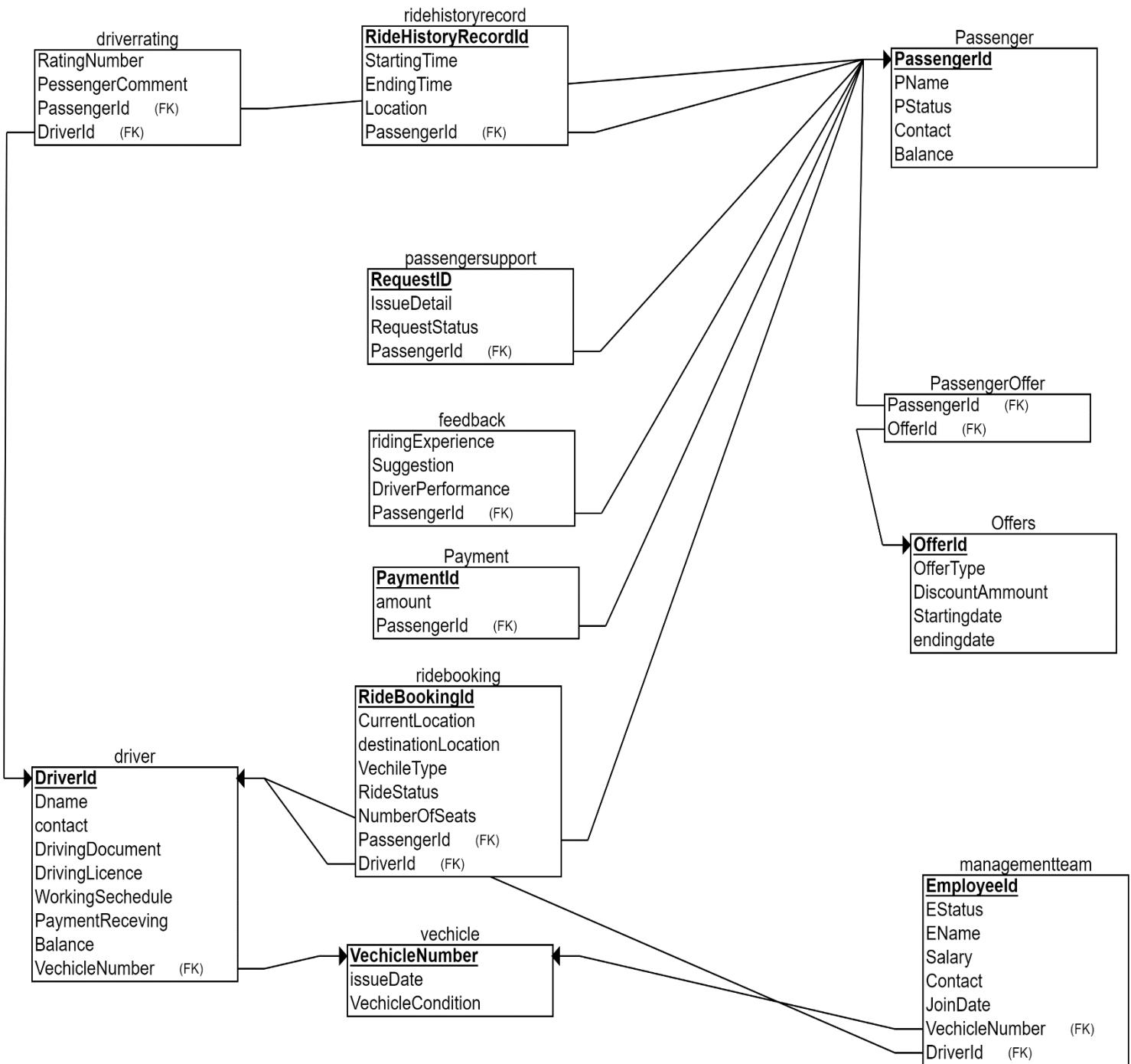


Figure 2: Relational schema with primary and foreign key relationships

### 3.3 Design Methodology

The database follows these design principles:

- **Normalization:** 3NF compliance to eliminate redundancy
- **Referential Integrity:** Foreign key constraints across all relationships
- **Data Types:** Appropriate data types for efficient storage
- **Naming Conventions:** Consistent PascalCase for tables and columns

### 3.4 Relationships Overview

Relationship Type	Tables Involved	Description
One-to-Many	Driver → RideBooking	One driver can have many ride bookings
One-to-Many	Passenger → RideBooking	One passenger can have many ride bookings
One-to-Many	Driver → DriverRating	One driver can receive many ratings
One-to-Many	Passenger → PassengerSupport	One passenger can create many support tickets
Many-to-Many	Passenger ↔ Offers	Passengers can avail multiple offers, offers can be availed by multiple passengers (resolved through PassengerOffer junction table)

## 4. SAMPLE DATA

### 4.1 Data Population Statistics

Table	Record Count
Vehicle	50
Driver	50
Passenger	50
DriverRating	50
ManagementTeam	50
RideBooking	54
Feedback	50
RideHistoryRecord	50
PassengerSupport	50
Offers	50
PassengerOffer	50
Payment	50

**Total Records:** 600+ records

## 4.2 Sample Data Screenshots

This screenshot shows the MySQL Workbench interface with the following details:

- Session:** Test1 (citywheelsbh) - Warning
- Navigator:** Shows various management, instance, performance, and administration sections.
- Query Editor:** Shows the query "SELECT \* FROM Offers WHERE NOW() BETWEEN Startingdate AND DATE\_ADD(ending..." and its execution results.
- Action Output:** A table showing the results of the executed queries, including time, action, message, duration, and fetch count.

#	Time	Action	Message	Duration / Fetch
122	11:42:07	Create DataBase CityWheelsBH	1 row(s) affected	0.000 sec
123	11:42:07	Use CityWheelsBH	0 row(s) affected	0.000 sec
124	11:42:07	show Tables	0 row(s) returned	0.000 sec / 0.000 sec
125	11:42:07	Create Table vechicle ( VechicleNumber Varchar(15) primary key , issueDate Date , VechicleC...	0 row(s) affected	0.016 sec
126	11:42:07	Create Table Driver( DiverId int primary Key , Dname Varchar(20), contact Varchar(20), Divi...	0 row(s) affected	0.031 sec
127	11:42:07	Create Table Passenger(PassengerId int primary Key , PName Varchar(20), PStatus Varchar(...	0 row(s) affected	0.016 sec
128	11:42:07	Create table DriverRating( RatingNumber int , PassengerComment Varchar(100), DiverId int ...	0 row(s) affected	0.000 sec
129	11:42:07	Create Table ManagementTeam(EmployeedId int primary Key , EStatus Varchar(20), ENam...	0 row(s) affected	0.031 sec
130	11:42:07	create Table RideBooking(RideBookingId int primary key , CurrentLocation Varchar(50), des...	0 row(s) affected	0.016 sec
131	11:42:07	Create Table Feedback(ridingExperience Varchar(20), Suggestion Varchar(50), DriverPerfor...	0 row(s) affected	0.016 sec
132	11:42:07	Create table RideHistoryRecord(RideHistoryRecordId int primary key , StartingTime DateTim...	0 row(s) affected	0.031 sec
133	11:42:07	Create table PassengerSupport( RequestID int primary Key , IssueDetail Varchar(150), Requ...	0 row(s) affected	0.000 sec
134	11:42:07	Create Table Offers(OfferId int primary Key , OfferType Varchar(20), DiscountAmount float...	0 row(s) affected	0.016 sec
135	11:42:07	Create Table PassengerOffer( OfferId int , PassengerId int foreign key(PassengerId) referen...	0 row(s) affected	0.016 sec
136	11:42:07	Create Table Payment( PaymentId int primary Key , amount float , PassengerId int , foreign key...	0 row(s) affected	0.031 sec
137	11:42:07	INSERT INTO vechicle (VehicleNumber, IssueDate, VechicleCondition) VALUES ('FDO 20...', 51 row(s) affected Records: 51 Duplicates: 0 Warnings: 0	0.000 sec	
138	11:42:07	INSERT INTO Driver (DiverId, Dname, contact, DrivingLicence, WorkingSchedule, Payme...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
139	11:42:07	INSERT INTO Passenger (PassengerId, PName, PStatus, Contact, Balance) VALUES ('1...', 50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec	
140	11:42:07	INSERT INTO DriverRating (RatingNumber, PassengerComment, DiverId, PassengerId) VA...	51 row(s) affected Records: 51 Duplicates: 0 Warnings: 0	0.000 sec
141	11:42:07	INSERT INTO ManagementTeam (EmployeedId, EStatus, ENam...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
142	11:42:07	INSERT INTO RideBooking (RideBookingId, CurrentLocation, destinationLocation, VechileT...	54 row(s) affected Records: 54 Duplicates: 0 Warnings: 0	0.000 sec
143	11:42:07	INSERT INTO Feedback (ridingExperience, Suggestion, DriverPerformance, PassengerId) V...	48 row(s) affected Records: 48 Duplicates: 0 Warnings: 0	0.016 sec
144	11:42:07	INSERT INTO RideHistoryRecord (RideHistoryRecordId, StartingTime, EndingTime, Locatio...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
145	11:42:07	INSERT INTO PassengerSupport (RequestID, IssueDetail, RequestStatus, PassengerId) VA...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
146	11:42:07	INSERT INTO Offers (OfferId, OfferType, DiscountAmount, StartingDate, endingDate) VAL...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
147	11:42:07	INSERT INTO PassengerOffer (OfferId, PassengerId) VALUES (1, 44), (3, 22), (3, 7), (3, 34), (... 52 row(s) affected Records: 52 Duplicates: 0 Warnings: 0	0.000 sec	
148	11:42:07	INSERT INTO Payment (PaymentId, amount , PassengerId) VALUES (1, 150, 50, 11), (2, 200...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
149	11:42:07	SELECT D.DiverId, D.Dname, COUNT(RHR.DiverId) AS TotalRidesCompleted FROM Driv...	50 row(s) returned	0.000 sec / 0.000 sec
150	11:42:07	SELECT D.DiverId, D.Dname, AVG(DR.RatingNumber) AS AverageRating FROM Diver D...	0 row(s) returned	0.000 sec / 0.000 sec
151	11:42:07	SELECT P.Pname FROM Passenger P inner JOIN RideBooking RB ON P.PassengerId = R...	1 row(s) returned	0.000 sec / 0.000 sec
152	11:42:07	SELECT D.DiverId, D.Dname FROM Driver D LEFT JOIN DriverRating DR ON D.DiverId = ...	33 row(s) returned	0.000 sec / 0.000 sec
153	11:42:07	SELECT MONTH(EndingTime) AS Month, COUNT(*) AS TotalRidesCompleted FROM Ride...	3 row(s) returned	0.000 sec / 0.000 sec
154	11:42:07	SELECT D.DiverId, D.Dname FROM Driver D JOIN RideHistoryRecord RHR ON D.DiverId = ...	1 row(s) returned	0.000 sec / 0.000 sec
155	11:42:07	SELECT Location AS MostPopularDestination, COUNT(*) AS TotalRides FROM RideHistory...	1 row(s) returned	0.000 sec / 0.000 sec
156	11:42:07	SELECT IssueDetail AS MostCommonIssue, COUNT(*) AS TotalOccurrences FROM Passen...	1 row(s) returned	0.000 sec / 0.000 sec
157	11:42:07	SELECT D.DiverId, SUM(DriverEarning) AS TotalEarnings FROM RideBooking GROUP BY Dr...	14 row(s) returned	0.000 sec / 0.000 sec
158	11:42:08	SELECT * FROM Offers WHERE NOW() BETWEEN Startingdate AND DATE_ADD(ending...	1 row(s) returned	0.000 sec / 0.000 sec

Figure 3: data insertion screenshots

This screenshot shows the MySQL Workbench interface with the following details:

- Session:** Test1 (citywheelsbh) - Warning
- Navigator:** Shows various management, instance, performance, and administration sections.
- Query Editor:** Shows the query "SELECT \* FROM Offers WHERE NOW() BETWEEN Startingdate AND DATE\_ADD(ending..." and its execution results.
- Action Output:** A table showing the results of the executed queries, including time, action, message, duration, and fetch count.

#	Time	Action	Message	Duration / Fetch
136	11:42:07	Create Table Payment( PaymentId int primary Key , amount float , PassengerId int , foreign key...	0 row(s) affected	0.031 sec
137	11:42:07	INSERT INTO vechicle (VehicleNumber, IssueDate, VechicleCondition) VALUES ('FDO 20...', 51 row(s) affected Records: 51 Duplicates: 0 Warnings: 0	0.000 sec	
138	11:42:07	INSERT INTO Driver (DiverId, Dname, contact, DrivingLicence, WorkingSchedule, Payme...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
139	11:42:07	INSERT INTO Passenger (PassengerId, PName, PStatus, Contact, Balance) VALUES ('1...', 50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec	
140	11:42:07	INSERT INTO DriverRating (RatingNumber, PassengerComment, DiverId, PassengerId) VA...	51 row(s) affected Records: 51 Duplicates: 0 Warnings: 0	0.000 sec
141	11:42:07	INSERT INTO ManagementTeam (EmployeedId, EStatus, ENam...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
142	11:42:07	INSERT INTO RideBooking (RideBookingId, CurrentLocation, destinationLocation, VechileT...	54 row(s) affected Records: 54 Duplicates: 0 Warnings: 0	0.000 sec
143	11:42:07	INSERT INTO Feedback (ridingExperience, Suggestion, DriverPerformance, PassengerId) V...	48 row(s) affected Records: 48 Duplicates: 0 Warnings: 0	0.016 sec
144	11:42:07	INSERT INTO RideHistoryRecord (RideHistoryRecordId, StartingTime, EndingTime, Locatio...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
145	11:42:07	INSERT INTO PassengerSupport (RequestID, IssueDetail, RequestStatus, PassengerId) VA...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
146	11:42:07	INSERT INTO Offers (OfferId, OfferType, DiscountAmount, StartingDate, endingDate) VAL...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
147	11:42:07	INSERT INTO PassengerOffer (OfferId, PassengerId) VALUES (1, 44), (3, 22), (3, 7), (3, 34), (... 52 row(s) affected Records: 52 Duplicates: 0 Warnings: 0	0.000 sec	
148	11:42:07	INSERT INTO Payment (PaymentId, amount , PassengerId) VALUES (1, 150, 50, 11), (2, 200...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.000 sec
149	11:42:07	SELECT D.DiverId, D.Dname, COUNT(RHR.DiverId) AS TotalRidesCompleted FROM Driv...	50 row(s) returned	0.000 sec / 0.000 sec
150	11:42:07	SELECT D.DiverId, D.Dname, AVG(DR.RatingNumber) AS AverageRating FROM Diver D...	0 row(s) returned	0.000 sec / 0.000 sec
151	11:42:07	SELECT P.Pname FROM Passenger P inner JOIN RideBooking RB ON P.PassengerId = R...	1 row(s) returned	0.000 sec / 0.000 sec
152	11:42:07	SELECT D.DiverId, D.Dname FROM Driver D LEFT JOIN DriverRating DR ON D.DiverId = ...	33 row(s) returned	0.000 sec / 0.000 sec
153	11:42:07	SELECT MONTH(EndingTime) AS Month, COUNT(*) AS TotalRidesCompleted FROM Ride...	3 row(s) returned	0.000 sec / 0.000 sec
154	11:42:07	SELECT D.DiverId, D.Dname FROM Driver D JOIN RideHistoryRecord RHR ON D.DiverId = ...	1 row(s) returned	0.000 sec / 0.000 sec
155	11:42:07	SELECT Location AS MostPopularDestination, COUNT(*) AS TotalRides FROM RideHistory...	1 row(s) returned	0.000 sec / 0.000 sec
156	11:42:07	SELECT IssueDetail AS MostCommonIssue, COUNT(*) AS TotalOccurrences FROM Passen...	1 row(s) returned	0.000 sec / 0.000 sec
157	11:42:07	SELECT D.DiverId, SUM(DriverEarning) AS TotalEarnings FROM RideBooking GROUP BY Dr...	14 row(s) returned	0.000 sec / 0.000 sec
158	11:42:08	SELECT * FROM Offers WHERE NOW() BETWEEN Startingdate AND DATE_ADD(ending...	1 row(s) returned	0.000 sec / 0.000 sec

Figure 4: data insertion screenshots

## 5. BUSINESS QUERIES

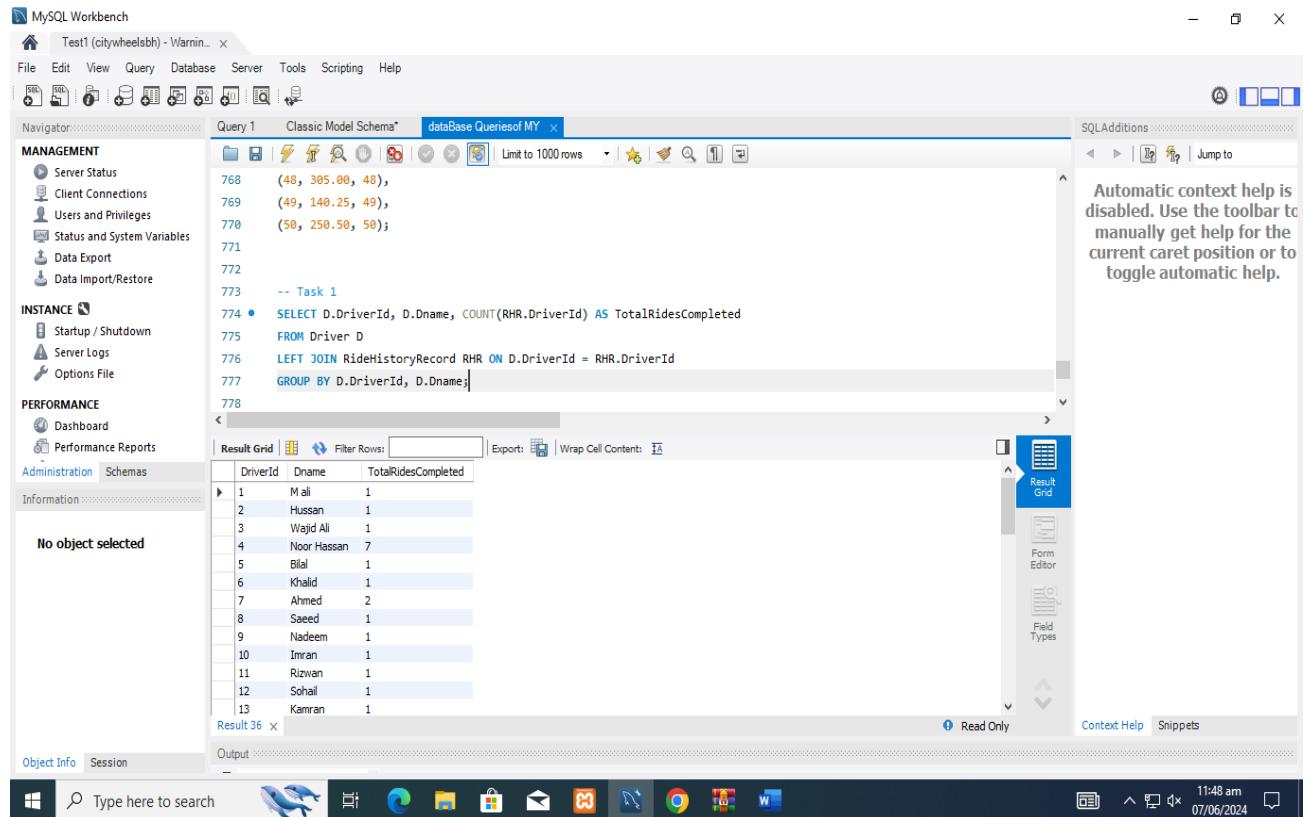
### 5.1 Query 1: Driver Performance Analysis

**Purpose:** List drivers and the total number of rides they have completed.

SQL Query:

```
SELECT D.DriverId, D.Dname, COUNT(RHR.DriverId) AS TotalRidesCompleted  
FROM Driver D  
LEFT JOIN RideHistoryRecord RHR ON D.DriverId = RHR.DriverId  
GROUP BY D.DriverId, D.Dname;
```

**Business Value:** Identifies top-performing drivers and those with low ride counts.



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with tables like `Driver` and `RideHistoryRecord`.
- Query Editor:** Displays the SQL query and its execution results. The results show 13 drivers and their total ride counts:

DriverId	Dname	TotalRidesCompleted
1	M Ali	1
2	Hussan	1
3	Wajid Ali	1
4	Noor Hassan	7
5	Bilal	1
6	Khalid	1
7	Ahmed	2
8	Saeed	1
9	Nadeem	1
10	Imran	1
11	Rizwan	1
12	Sohail	1
13	Kamran	1

- Output Bar:** Shows the Windows taskbar at the bottom with various application icons.

## 5.2 Query 2: High-Performance Driver Ratings

**Purpose:** Find the average rating of drivers who have completed more than 50 rides.

SQL Query:

```
SELECT D.DriverId, D.Dname, AVG(DR.RatingNumber) AS AverageRating  
FROM Driver D  
JOIN DriverRating DR ON D.DriverId = DR.DriverId  
GROUP BY D.DriverId, D.Dname  
HAVING COUNT(DR.DriverId) > 50;
```

**Note:** Returns no rows as no driver has completed 50+ rides in sample data.

The screenshot shows the MySQL Workbench interface. In the central query editor window, two queries are displayed:

```
776  Left JOIN RideHistoryRecord RHR ON D.DriverId = RHR.DriverId  
777  GROUP BY D.DriverId, D.Dname;  
778  
779  
780  -- Task 2  
781  • Select D.DriverId, D.Dname, AVG(DR.RatingNumber) AS AverageRating FROM Driver D inner join DriverRating DR ON D.DriverId = DR.DriverId  
782  GROUP by D.DriverId, D.Dname Having COUNT(DR.DriverId) > 50;
```

The result grid for the second query shows the following data:

DriverId	Dname	AverageRating

In the bottom right corner of the result grid, there is a note: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

The status bar at the bottom right indicates the system is at 38°C, the date is 6/9/2024, and the time is 7:59 PM.

### 5.3 Query 3: Frequent Passengers

**Purpose:** List passengers who have used CityWheels more than 20 times.

SQL Query:

```
SELECT P.PName  
FROM Passenger P  
INNER JOIN RideBooking RB ON P.PassengerId = RB.PassengerId  
GROUP BY P.PassengerId, P.PName  
HAVING COUNT(RB.PassengerId) > 20;
```

**Business Value:** Identifies loyal customers for targeted marketing.

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation panels for Management, Instance, and Administration. The main area has a toolbar at the top with various icons. Below the toolbar is a 'Query 1' tab labeled 'dataBase Queries of MY'. The query code is displayed in the tab, and the results are shown in a 'Result Grid' table below it. The table has columns 'PassengerId' and 'PName', with one row showing '1 Naeem'. At the bottom of the interface, there is an 'Output' section titled 'Action Output' which lists several log entries with details like time, action, message, and duration. The system tray at the bottom right shows the date and time as 6/9/2024 8:02 PM.

PassengerId	PName
1	Naeem

#	Time	Action	Message	Duration / Fetch
38	18:52:14	SELECT * FROM Offers WHERE NOW() BETWEEN Startingdate AND DATE_ADD(Endingdat...	1 row(s) returned	0.437 sec / 0.000 sec
39	19:55:45	Select D.DiverId, D.Dname, COUNT(RHR.DiverId) AS TotalRidesCompleted from Driver D Le...	50 row(s) returned	0.656 sec / 0.000 sec
40	19:59:32	Select D.DiverId, D.Dname, AVG(DR.RatingNumber) AS AverageRating FROM Driver D Inne...	0 row(s) returned	0.140 sec / 0.000 sec
41	20:02:08	Select P.PassengerId, P.PName from Passenger P inner JOIN RideBooking RB on P.Passenger...	1 row(s) returned	0.187 sec / 0.000 sec

## 5.4 Query 4: Consistently High-Rated Drivers

**Purpose:** Identify drivers who have never received a rating below 4.

SQL Query:

```
SELECT D.DriverId, D.Dname  
FROM Driver D  
LEFT JOIN DriverRating DR ON D.DriverId = DR.DriverId  
GROUP BY D.DriverId, D.Dname  
HAVING COALESCE(MIN(DR.RatingNumber), 5) >= 4;
```

**Business Value:** Recognizes excellence for rewards and incentives.

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** MySQL Model\*, Task1 - Warning - not support...
- Toolbar:** Includes icons for Home, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Shows MANAGEMENT, INSTANCE, and Administration/Schemas sections. The Administration section is selected.
- Query Editor:** Contains two queries labeled Task 4 and Task 4. The first query selects drivers with ratings >= 4. The second query selects drivers with ratings < 4. The result grid shows driver IDs 2 through 24 and names Hussan, Imran, Sohail, Asif, Adnan, Majid, Rashid, Saad, and Ehsan.
- Output Window:** Shows the execution log with four entries corresponding to the queries above.
- System Tray:** Shows the Windows taskbar with various pinned icons and system status (38°C, ENG US, 8:07 PM, 6/9/2024).

## 5.5 Query 5: Monthly Ride Trends

**Purpose:** Find the total number of rides completed in each month of the current year.

SQL Query:

```
SELECT MONTH(EndingTime) AS Month, COUNT(*) AS TotalRidesCompleted  
FROM RideHistoryRecord  
WHERE YEAR(EndingTime) = YEAR(CURRENT_DATE())  
GROUP BY MONTH(EndingTime);
```

**Business Value:** Seasonal trend analysis for resource planning.

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** MySQL Model\*, Task1 - Warning - not support... X
- Toolbar:** Standard MySQL Workbench toolbar with icons for file operations, database navigation, and query execution.
- Navigator:** Shows the database structure with nodes like MANAGEMENT, INSTANCE, and Administration.
- Query Editor:** Title: Query 1 - DataBase Queries of MY\*  
Content:  
795 (SELECT DR.DriverId FROM DriverRating DR WHERE DR.RatingNumber < 4 );  
796  
797 -- task 5  
798 • select\* from RideHistoryRecord;  
799 • Select MONTH(EndingTime) AS Month, COUNT(\*) AS TotalRidesCompleted From RideHistoryRecord  
800 where YEAR(EndingTime) = YEAR(CURRENT\_DATE())  
801 GROUP BY MONTH(EndingTime);  
802  
Result Grid:

Month	TotalRidesCompleted
5	16
6	30
7	4

- Output Window:** Action Output table showing the execution log:

#	Time	Action	Message	Duration / Fetch
47	20:15:49	select* from RideHistoryRecord LIMIT 0, 1000	50 row(s) returned	0.000 sec / 0.000 sec
48	20:16:22	select* from RideHistoryRecord LIMIT 0, 1000	50 row(s) returned	0.000 sec / 0.000 sec
49	20:17:30	Select MONTH(EndingTime) AS Month, COUNT(*) AS TotalRidesCompleted From RideHistory...	3 row(s) returned	0.000 sec / 0.000 sec
50	20:17:40	Select MONTH(EndingTime) AS Month, COUNT(*) AS TotalRidesCompleted From RideHistory...	3 row(s) returned	0.000 sec / 0.000 sec
- System Tray:** Shows the Windows taskbar with various pinned icons and system status information (38°C, ENG US, 8:17 PM, 6/9/2024).

## 5.6 Query 6: Driver-Passenger Diversity

**Purpose:** Show drivers who have completed rides with at least three different passengers.

SQL Query:

```
SELECT D.DriverId, D.Dname  
FROM Driver D  
JOIN RideHistoryRecord RHR ON D.DriverId = RHR.DriverId  
GROUP BY D.DriverId, D.Dname  
HAVING COUNT(DISTINCT RHR.PassengerId) >= 3;
```

**Business Value:** Identifies drivers with broad customer reach.

The screenshot shows the MySQL Workbench interface with two tabs open: 'Query 1' and 'Result 24'.  
**Query 1:** Contains the SQL code for Query 6.  
**Result 24:** Shows the execution log with four entries, all of which completed successfully with 0.000 sec / 0.000 sec duration.

#	Time	Action	Message	Duration / Fetch
48	20:16:22	select * from RideHistoryRecord LIMIT 0, 1000	50 row(s) returned	0.000 sec / 0.000 sec
49	20:17:30	Select MONTH(EndingTime) AS Month, COUNT(*) AS TotalRidesCompleted From RideHistoryRecord	3 row(s) returned	0.000 sec / 0.000 sec
50	20:17:40	Select MONTH(EndingTime) AS Month, COUNT(*) AS TotalRidesCompleted From RideHistoryRecord	3 row(s) returned	0.000 sec / 0.000 sec
51	20:19:39	Select D.DriverId, D.Dname From Driver D inner Join RideHistoryRecord RHR ON D.DriverId = RHR.DriverId GROUP BY D.DriverId, D.Dname Having COUNT(DISTINCT RHR.PassengerId) >= 3	1 row(s) returned	0.000 sec / 0.000 sec

## 5.7 Query 7: Popular Destinations

**Purpose:** Determine the most popular destination among all rides.

SQL Query:

```
SELECT Location AS MostPopularDestination, COUNT(*) AS TotalRides  
FROM RideHistoryRecord  
GROUP BY Location  
ORDER BY TotalRides DESC  
LIMIT 1;
```

**Business Value:** Strategic planning for driver positioning and marketing.

The screenshot shows the MySQL Workbench interface. In the top-left pane, the Navigator displays various database objects like MANAGEMENT, INSTANCE, and INFORMATION. The central pane contains two queries in 'Query 1' and 'Query 2'. The first query is the one provided in the text above, and the second is a comment '-- task 7'. The bottom pane shows the results of the first query in a grid format:

MostPopularDestination	TotalRides
UCP Lahore	3

Below the results, the 'Result 25' tab shows the 'Action Output' log with four entries. The log includes columns for Action, Time, Message, and Duration / Fetch. The log entries are:

Action	Time	Message	Duration / Fetch
Select MONTH(EndTime) AS Month, COUNT(*) AS TotalRidesCompleted From RideHistory...	49 20:17:30	3 row(s) returned	0.000 sec / 0.000 sec
Select MONTH(EndTime) AS Month, COUNT(*) AS TotalRidesCompleted From RideHistory...	50 20:17:40	3 row(s) returned	0.000 sec / 0.000 sec
Select D.DriverId, D.Dname From Driver D inner Join RideHistoryRecord RHR ON D.DriverId = RHR.DriverId	51 20:19:39	1 row(s) returned	0.000 sec / 0.000 sec
Select Location AS MostPopularDestination, COUNT(*) as TotalRides FROM RideHistoryRecord	52 20:21:28	1 row(s) returned	0.000 sec / 0.000 sec

## 5.8 Query 8: Common Support Issues

**Purpose:** Identify the most common issue reported in support tickets.

SQL Query:

```
SELECT IssueDetail AS MostCommonIssue, COUNT(*) AS TotalOccurrences  
FROM PassengerSupport  
GROUP BY IssueDetail  
ORDER BY TotalOccurrences DESC  
LIMIT 1;
```

**Business Value:** Focus training and process improvements on frequent issues.

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the SQL code for Query 8, which selects the most common issue detail and its count from the PassengerSupport table, ordered by total occurrences, with a limit of 1.
- Result Grid:** Displays the output of the query, showing one row: "My driver did not arrive on time." with a count of 12.
- Output Tab:** Shows the "Action Output" section with four log entries, each detailing a SELECT statement and its execution time.
- System Bar:** At the bottom, it shows the Windows taskbar with various application icons and system status like temperature (38°C), date (6/9/2024), and time (8:23 PM).

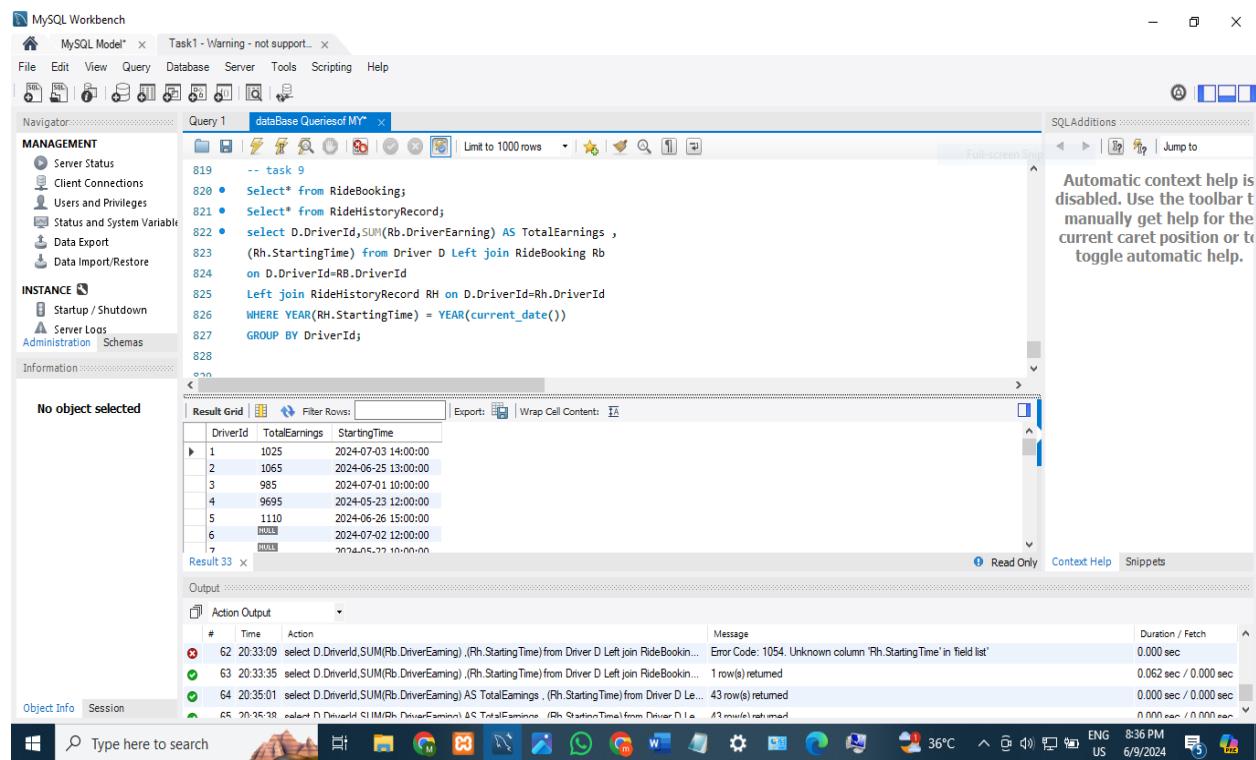
## 5.9 Query 9: Driver Earnings

**Purpose:** Show the total earnings for each driver in the current year.

SQL Query:

```
select D.DriverId, SUM(Rb.DriverEarning) AS TotalEarnings ,  
(Rh.StartingTime) from Driver D Left join RideBooking Rb  
on D.DriverId=RB.DriverId  
Left join RideHistoryRecord RH on D.DriverId=Rh.DriverId  
WHERE YEAR(RH.StartingTime) = YEAR(current_date())  
GROUP BY DriverId;
```

**Business Value:** Financial reporting and driver compensation tracking.



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the SQL query for Driver Earnings.
- Result Grid:** Displays the query results:

DriverId	TotalEarnings	StartingTime
1	1025	2024-07-03 14:00:00
2	1065	2024-06-25 13:00:00
3	985	2024-07-01 10:00:00
4	9695	2024-05-23 12:00:00
5	1110	2024-06-26 15:00:00
6	NULL	2024-07-02 12:00:00
7	NULL	2024-07-03 10:00:00

- Action Output:** Shows the execution log with an error message for the first query:

```
# Time Action Message Duration / Fetch
62 20:33:09 select D.DriverId,SUM(Rb.DriverEarning),(Rh.StartingTime) from Driver D Left join RideBooking... Error Code: 1054. Unknown column 'Rh.StartingTime' in field list' 0.000 sec
0.062 sec / 0.000 sec
63 20:33:35 select D.DriverId,SUM(Rb.DriverEarning),(Rh.StartingTime) from Driver D Left join RideBooking... 1 row(s) returned 0.000 sec / 0.000 sec
64 20:35:01 select D.DriverId,SUM(Rb.DriverEarning) AS TotalEarnings ,(Rh.StartingTime) from Driver D Le... 43 row(s) returned 0.000 sec / 0.000 sec
65 20:35:18 select D.DriverId,SUM(Rb.DriverEarning) AS TotalEarnings ,(Rh.StartingTime) from Driver D Le... A3 rows(s) returned 0.000 sec / 0.000 sec
```

## 5.10 Query 10: Active Promotions

**Purpose:** Get a list of all promotions that are currently active and expire within the next 30 days.

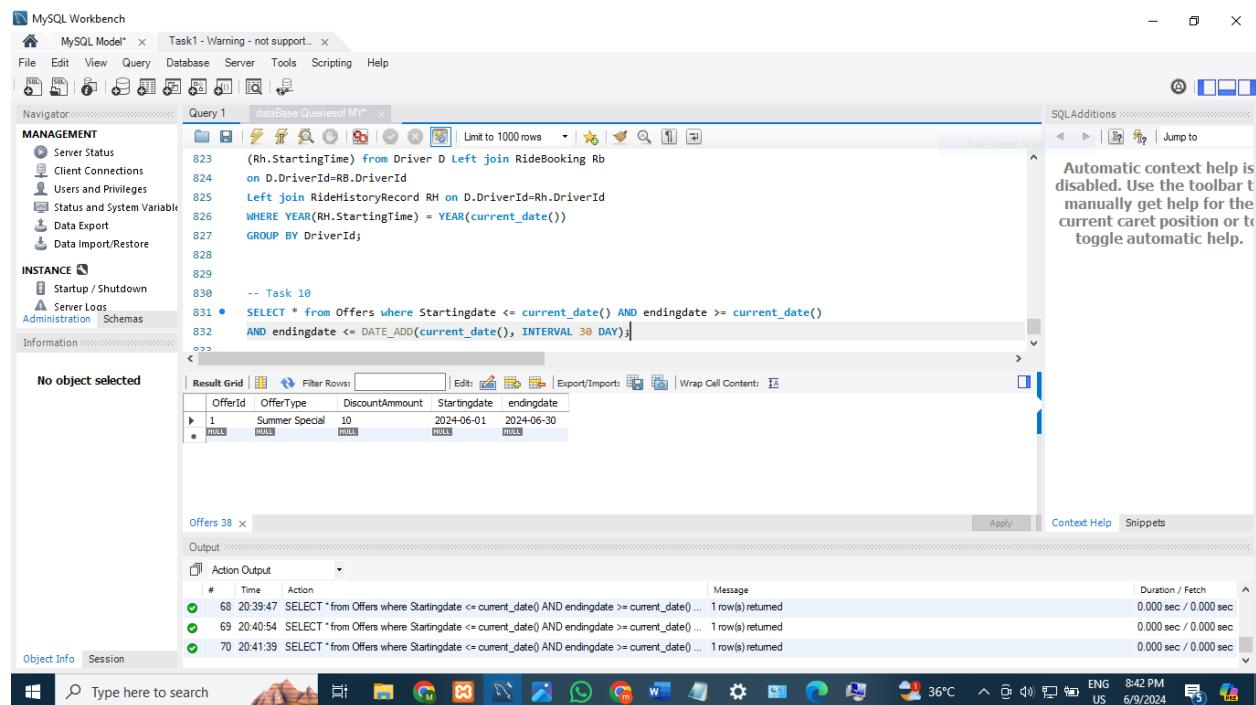
SQL Query:

```
SELECT *
```

```
FROM Offers
```

```
WHERE NOW() BETWEEN Startingdate AND DATE_ADD(enddate, INTERVAL 30 DAY);
```

**Business Value:** Marketing campaign management and promotion tracking.



The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
823     (Rh.StartingTime) from Driver D Left join RideBooking Rb  
824     on D.DriverId=Rb.DriverId  
825     Left join RideHistoryRecord RH on D.DriverId=Rh.DriverId  
826     WHERE YEAR(RH.StartingTime) = YEAR(current_date())  
827     GROUP BY DriverId;  
828  
829  
830    -- Task 10  
831  •  SELECT * from Offers where Startingdate <= current_date() AND enddate >= current_date()  
832      AND enddate <= DATE_ADD(current_date(), INTERVAL 30 DAY);
```

The result grid shows one row of data:

OfferId	OfferType	DiscountAmount	Startingdate	enddate
1	Summer Special	10	2024-06-01	2024-06-30

The status bar at the bottom right indicates: 36°C, ENG US, 9:42 PM, 6/9/2024.

## **6. CONCLUSION**

In conclusion, the CityWheels database project represents a successful application of database design principles to a real-world business scenario. The system effectively manages all core operations of a ride-hailing service while maintaining data integrity through proper normalization and relationship constraints.

The project's dual focus on technical implementation and legal compliance demonstrates understanding that modern database systems must operate within complex regulatory frameworks. The research on cross-border data sharing under PDPL and GDPR provides valuable insights for CityWheels's global expansion strategy.

This documentation, complete with ER diagrams, schema details, sample data, and analytical queries, provides all necessary information for deployment and maintenance. The system's scalability ensures it can grow with the business, supporting expansion from Pakistan to international markets.

Overall, this project successfully meets all course requirements while delivering practical value for a real-world business scenario, demonstrating both technical competence and awareness of broader legal and ethical considerations in database management.

---