American International University-Bangladesh

Project Name: Elevated Expressway Management System

Course Name: Advance Database Management System

Section: A

Serial No	Name	ID	Contributed Percentage	Topic Names
01	MD. Tahsin Tasnim Aurin	21-45213-2	20%	Queries, Data Insertion, Relation Algebra, Conclusion, PL/SQL(Function, Package), Query writing
02	MD Mehedi Hasan Ratul	21-45007-2	20%	Diagrams (Use Case, Activity, Class, Schema), Normalization, Query writing
03	Saad Bin Sami	21-45246-2	20%	Introduction, Proposal, Scenario Description, Er Diagram, PL/SQL(Cursor, Trigger)
04	Meherab Hasan Borno	21-45236-2	20%	User Interface, Database Connection(CRUD operation), Query writing
05	Palash Sen	20-42969-1	20%	Data insertion, PL/SQL(Record, Procedure)

Content	**Page No.**

1. Introduction

The Elevated Expressway management system operates similarly to toll booths on highways, where vehicles are charged a fee to use the elevated expressway. This fee is determined based on the vehicle type and is stored in the toll rate table. Additionally, records of each vehicle and its owner are maintained to ensure accurate billing. When a vehicle passes through a toll booth, the fee collected, along with the date and vehicle details, is recorded, facilitating the monitoring of vehicle usage, and ensuring fair payment.

Traffic conditions are closely monitored to adjust toll rates as needed, ensuring efficient traffic flow. Investors are engaged in the development of new road sections, contributing funds in exchange for a share of toll revenue. Details of each road section, such as length and construction date, are recorded to inform decision-making regarding road improvement projects.

Users have access to information about toll rates, and system administrators oversee user access and privileges. In summary, the toll management system plays a crucial role in maintaining road operations, ensuring fair billing practices, and providing valuable data for road development and maintenance.

2. Project Proposal

The proposed Elevated Expressway Management System is designed to efficiently manage toll collection operations on elevated expressways. This system will simplify the process of collecting tolls from vehicles entering the elevated expressway by optimizing the toll-rate determination, collection, and traffic monitoring processes. By using technology and data analysis, the system wants to make more money and make traffic move better.

The system comprises several key components, including:

Toll Rate Management: The Toll Rate table stores information about different toll rates corresponding to vehicle types. Each vehicle is associated with a specific toll rate based on its type, allowing for accurate toll calculation during collection.

Toll Collection: The Toll Collection table records events of toll collection, including collection date, vehicle ID, and toll ID. Each time a vehicle passes through the expressway, a toll is collected based on the applicable toll rate, contributing to revenue generation.

Expressway Traffic Monitoring: The Expressway Traffic table tracks traffic conditions, including traffic date and vehicle count. This data is used to analyze traffic patterns, identify congestion hotspots, and optimize traffic flow on the elevated expressway.

Investor Management: The Investor table manages investor information, such as investor ID, name, and share percentage. Investors contribute funds for the development of elevated expressway sections, fostering infrastructure growth and expansion.

Expressway Section Management: The Expressway Section table contains details of expressway sections, including section ID, name, length, and inauguration date. Each section corresponds to a specific record of expressway traffic, assisting better traffic analysis.

User and Admin Management: The User and Admin tables handle user authentication and authorization. Users can access information about toll rates, while admins have authority to manage user privileges and oversee system resources.

The system helps manage traffic better by observing traffic and figuring out trends, which means less traffic congestion and shorter travel times. As more people get used to using the expressway, the less traffic congestion will be seen in the city. It makes more revenue by collecting tolls effectively and adjusting toll rates as needed. It gives useful information for maintaining and keeping highways in good shape. Users can easily find out information about toll rates, and admins can control who has access to the system, making sure everything runs smoothly and safely.

3. Diagram

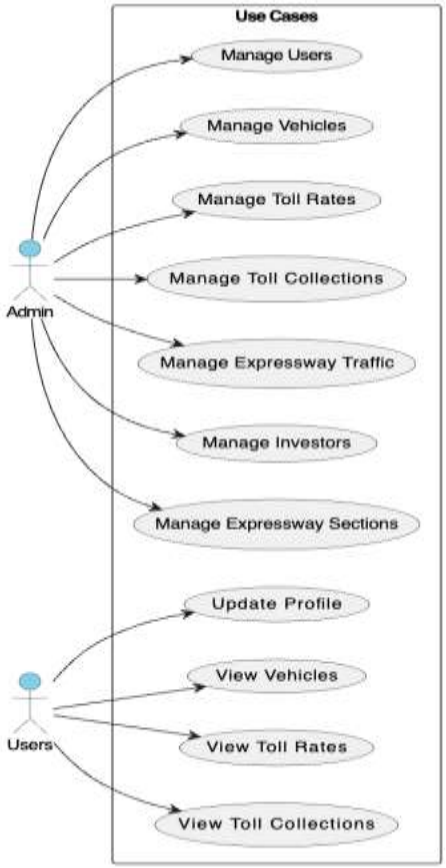


Figure: Use Case Diagram

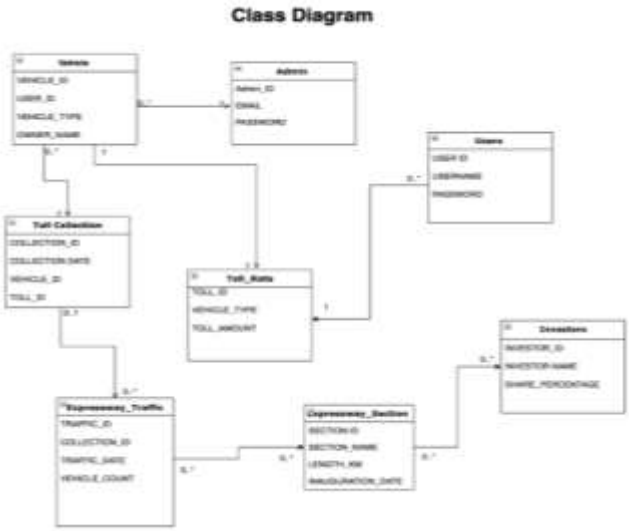


Figure: Class Diagram

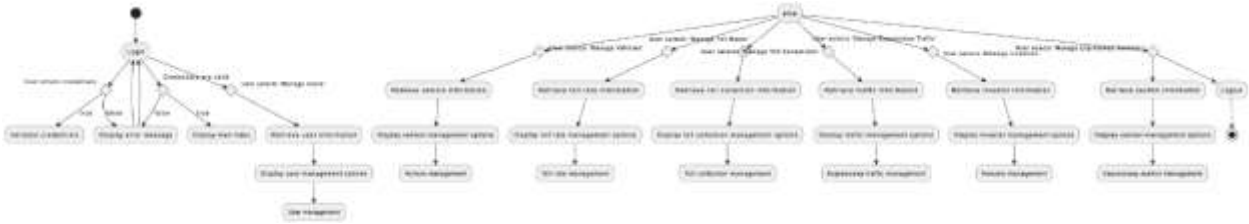


Figure: Activity Diagram

4. User Interface

Figure: Home Page

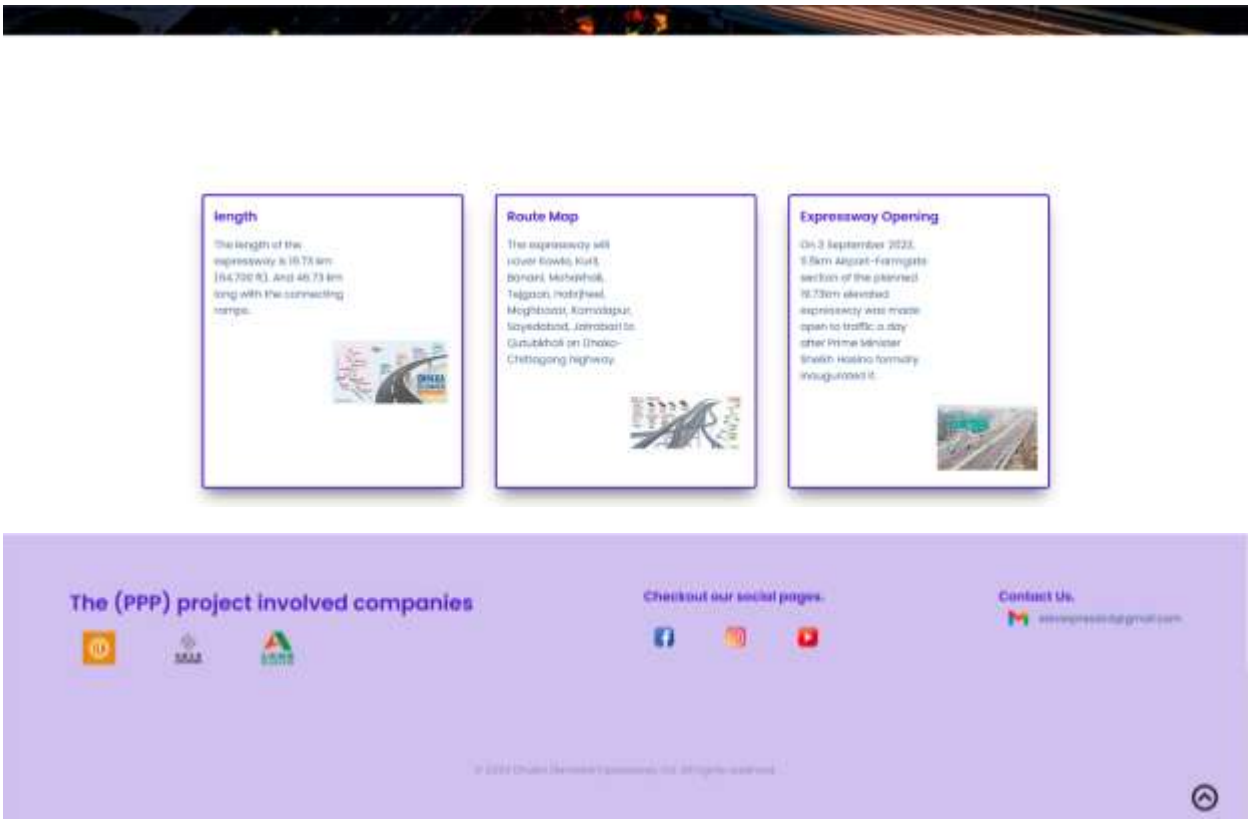


Figure: Information

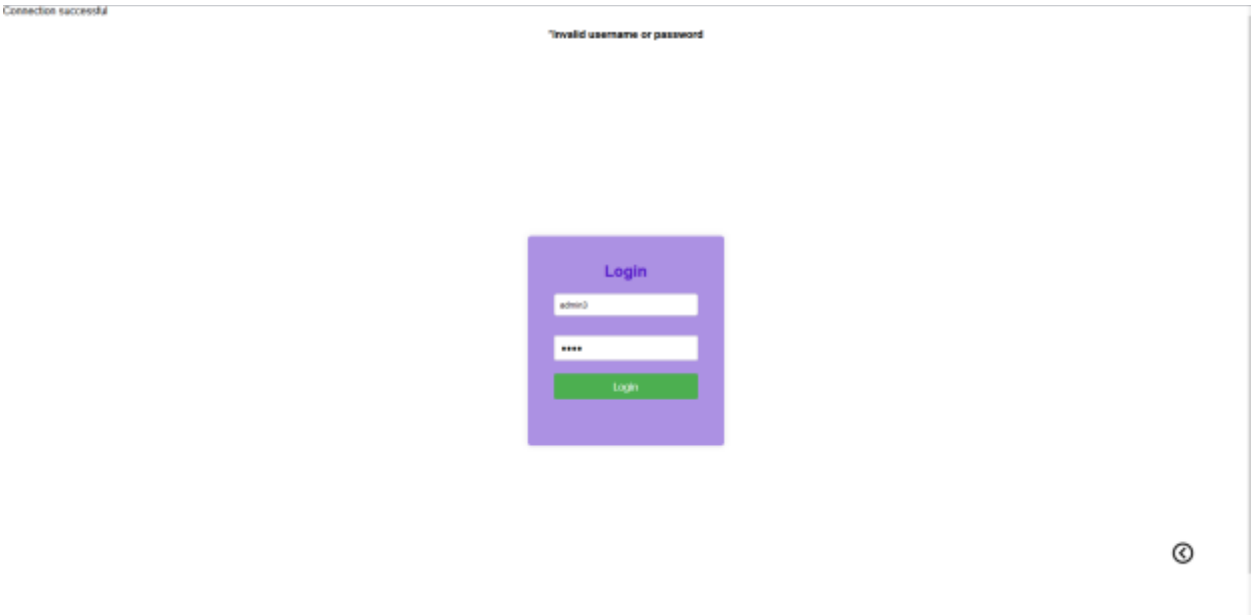


Figure: Login Page

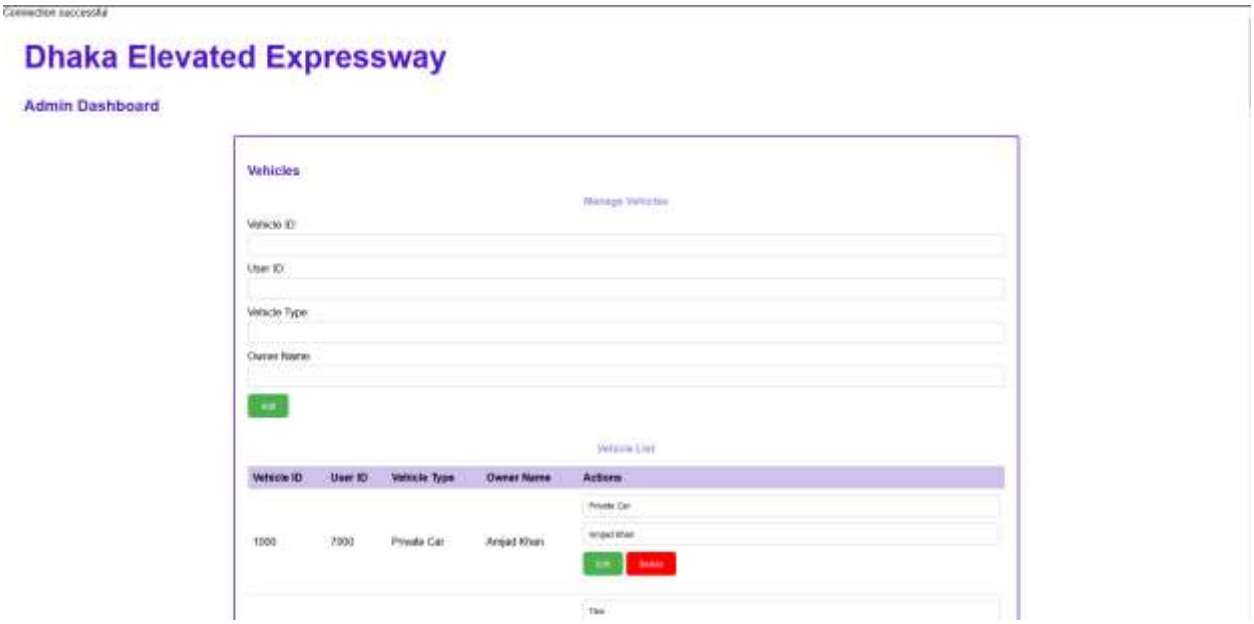


Figure: Admin Dashboard (Vehicles Section)

Figure: Admin Dashboard (Toll Rates Section)

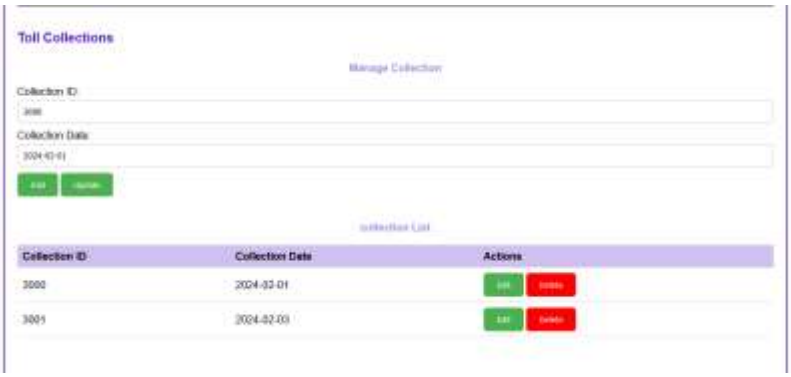


Figure: Admin Dashboard (Toll Collection)

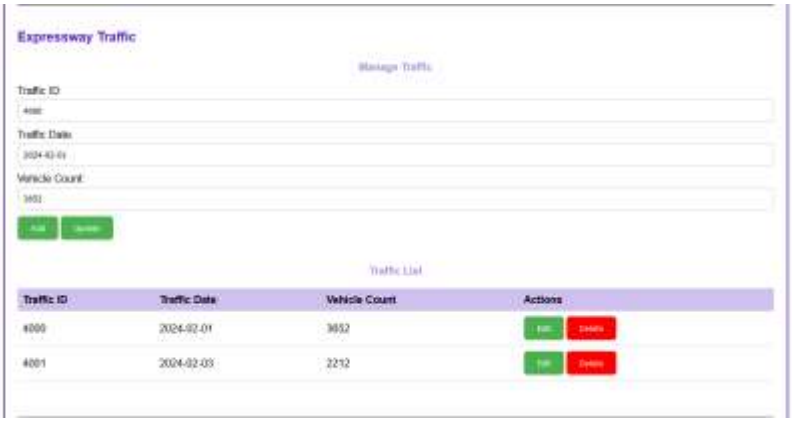


Figure: Admin Dashboard (Expressway Traffic)

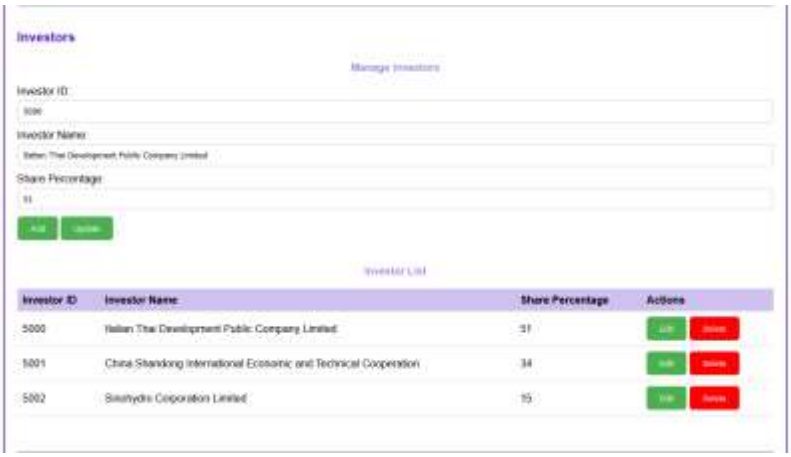


Figure: Admin Dashboard (Investors)

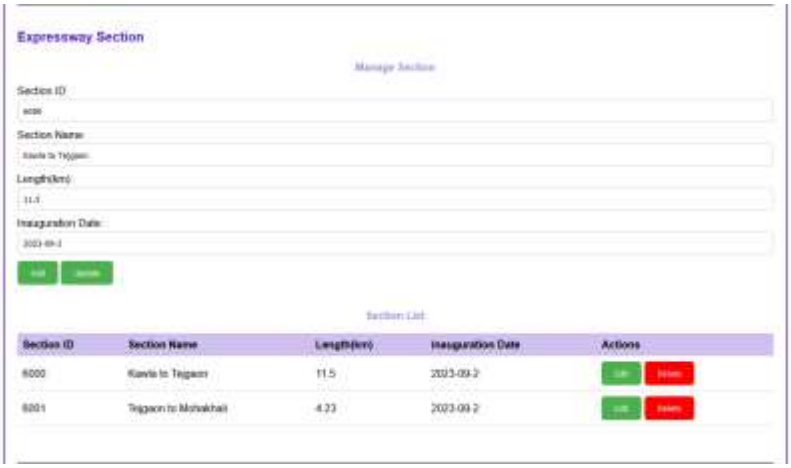


Figure: Admin Dashboard (Expressway Section)

Figure: Admin Dashboard (Users Section)

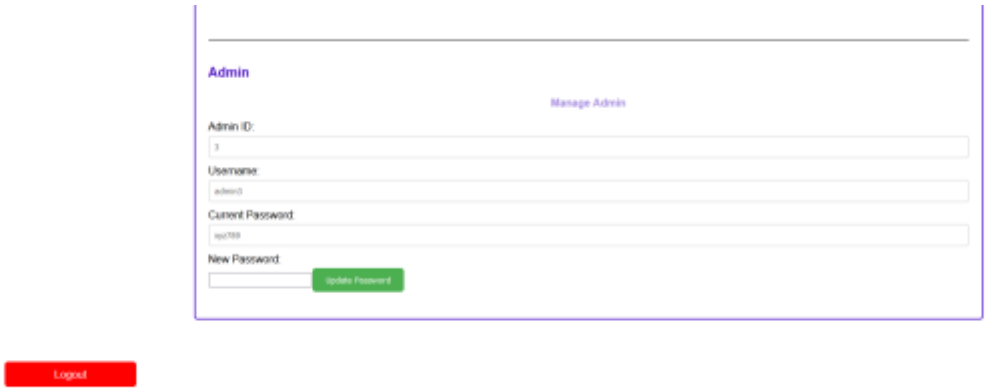


Figure: Admin Dashboard (Admin Info)

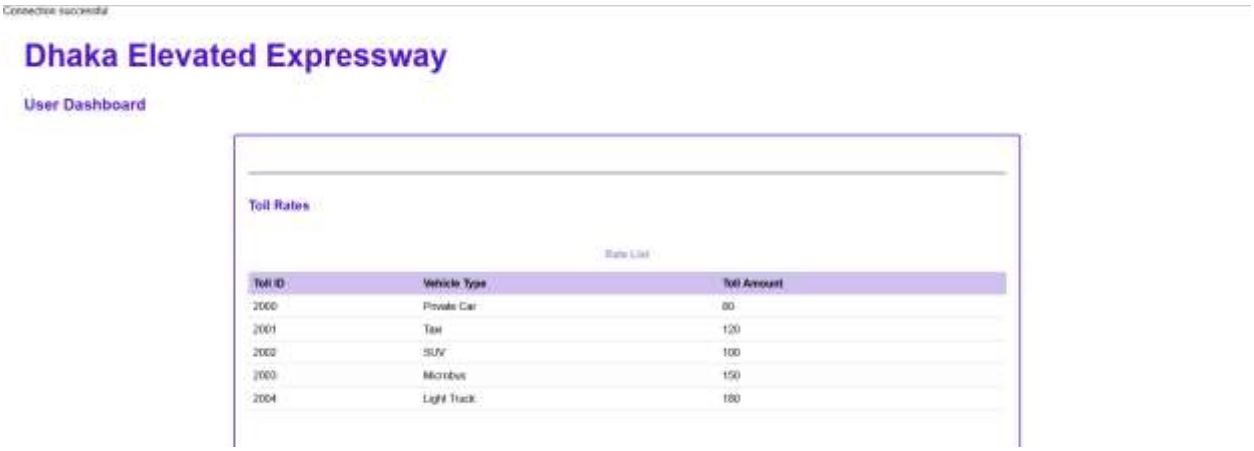


Figure: Users Dashboard (Toll Rates Section)

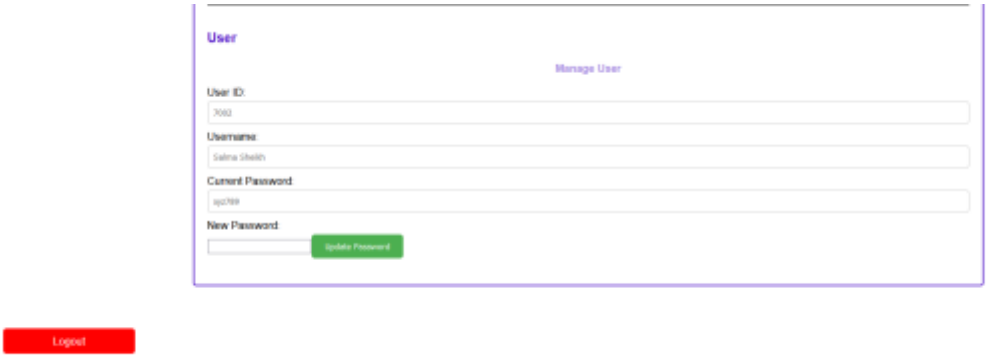


Figure: Users Dashboard (Toll Collection)

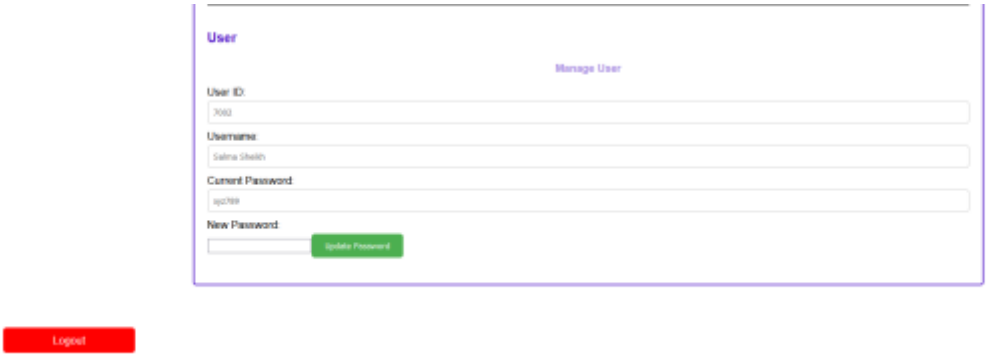


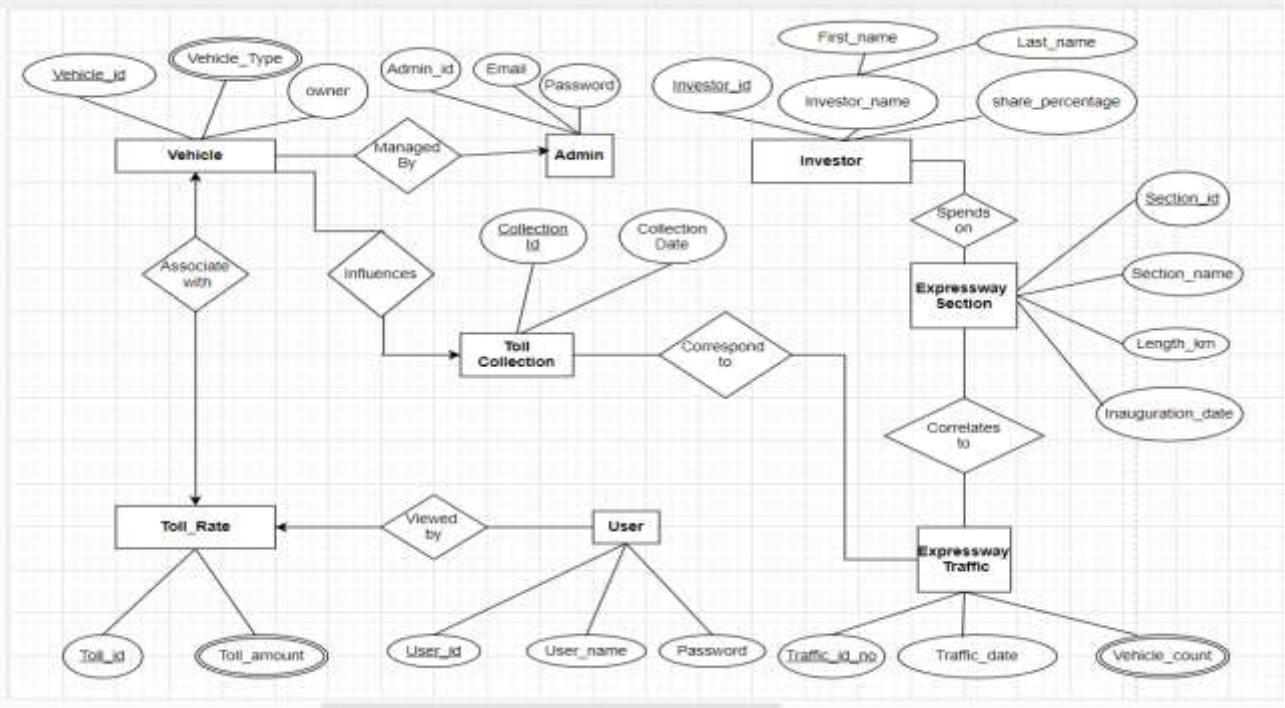
Figure: Users Profile Information

5. Scenario Description

A vehicle entering the elevated expressway will need to pay a fee determined by the toll-rate. This toll-rate table will contain details like toll ID, vehicle type, and the amount to be paid. Depending on what kind of vehicle it is, the toll amount will vary. Each vehicle will have its own information stored in the vehicle table, including vehicle ID, type, and the name of its owner. The toll-rate for each vehicle type will be linked to the toll collection table, which will keep track

of every time a vehicle uses the expressway. This table will have records such as collection ID, collection date, vehicle ID, and the associated toll ID. So, each time a vehicle uses the expressway, a toll will be collected from it, and this will be recorded. Monitoring traffic conditions will also be crucial. The expressway traffic table will note down traffic ID, date, and vehicle count. This helps in understanding how busy the expressway is at different times. Each toll collected will correspond to a specific instance of expressway traffic, making sure everything is accounted for. Investors will be involved in funding various sections of the expressway. Their details, including investor ID, name, and the percentage they own, will be stored in the investor table. Many investors can support multiple sections of the expressway. Each entry in the expressway traffic table will correspond to a specific section of the expressway, recorded in the expressway section table. This table will include section ID, section name, length in kilometers, and the date it was opened. Each section will have its own record of expressway traffic. Users will be able to access information about toll rates. The user table will store user ID, username, and password. Some users will only have access to one toll rate table. Admins will have the authority to manage user access and privileges. They'll oversee multiple users and ensure everything runs smoothly. The admin table will contain details like admin ID, email, and password.

6. ER Diagram



7. Normalization

Influences

UNF:

Vehicle_id, Vehicle_Type, Owner, Collection_Id, Collection_Date

1NF:

Vehicle_Type is a multi-valued attribute.

Vehicle_id, Vehicle_Type, Owner_Name, Collection_Id, Collection_Date

2NF:

1. Vehicle_id, Vehicle_Type, Owner_Name
2. Collection_Id, Collection_Date

3NF:

There is no transitive dependency. Relation already in 3NF.

1. Vehicle_id, Vehicle_Type, Owner_Name
2. Collection_Id, Collection_Date

Table Creation:

1. Vehicle_id, Vehicle_Type, Owner_Name
2. Collection_Id, Collection_Date

Corresponds to

UNF: Collection_Id, Collection_Date, Traffic_id no, Traffic_date, Vehicle_count

1NF:

Vehicle_count is a multi-valued attribute.

1. Collection_Id, Collection_Date, Traffic id no, Traffic_date, Vehicle_count

2NF:

1. Collection_Id, Collection_Date
2. Traffic id, Traffic_date, Vehicle_count

3NF:

There is no transitive dependency. Relation already in 3NF.

1. Collection_Id, Collection_Date
2. Traffic id, Traffic_date, Vehicle_count

Table Creation:

1. Collection_Id, Collection_Date
2. Traffic id, Traffic_date, Vehicle_count

Correlates to

UNF: Traffic_id, Traffic_date, Vehicle_count, Section_id, Section_name, Length_km, Inauguration_date

1NF:

Vehicle_count is a multi-valued attribute.

1. Traffic_id, Traffic_date, Vehicle_count, Section_id, Section_name, Length_km, Inauguration_date

2NF:

1. Traffic_id, Traffic_date, Vehicle_count
2. Section_id, Section_name, Length_km, Inauguration_date

3NF:

There is no transitive dependency. Relation already in 3NF.

1. Traffic_id, Traffic_date, Vehicle_count
2. Section_id, Section_name, Length_km, Inauguration_date

Table Creation:

1. Traffic_id, Traffic_date, Vehicle_count
2. Section_id, Section_name, Length_km, Inauguration_date

Spends on

UNF: Section_id, Section_name, Length_km, Inauguration_date, Investor_id, Investor_name, First_name, Last_name, share_percentage

1NF: No multi-valued attribute.

1. Section_id, Section_name, Length_km, Inauguration_date, Investor_id, Investor_name, First_name, Last_name, share_percentage

2NF:

1. Section_id, Section_name, Length_km, Inauguration_date
2. Investor_id, Investor_name, First_name, Last_name, share_percentage

3NF:

1. Section_id, Section_name, Length_km, Inauguration_date
2. Investor_name, First_name, Last_name
3. Investor_id, share_percentage

Table Creation:

1. Section_id, Section_name, Length_km, Inauguration_date
2. Investor_name, First_name, Last_name
3. Investor_id, share_percentage

Viewed by

UNF: Toll_id, Toll_amount, User_id, Username, Password

1NF:

Toll_amount is a multi-valued attribute.

1. Toll_id, Toll_amount, User_id, Username, Password

2NF:

1. Toll_id, Toll_amount
2. User_id, Username, Password

3NF:

There is no transitive dependency. Relation already in 3NF.

1. Toll_id, Toll_amount
2. User_id, Username, Password

Table Creation:

1. Toll_id, Toll_amount
2. User_id, Username, Password

Managed By

UNF: Vehicle_id, Vehicle_Type, Owner_name, Admin_id, Email, Password

1NF:

Vehicle_Type is a multi-valued attribute.

1. Vehicle_id, Vehicle_Type, Owner_name, Admin_id, Email, Password

2NF:

1. Vehicle_id, Vehicle_Type, Owner
2. Admin_id, Email, Password

3NF:

There is no transitive dependency. Relation already in 3NF.

1. Vehicle_id, Vehicle_Type, Owner_Name
2. Admin_id, Email, Password

Table Creation:

1. Vehicle_id, Vehicle_Type, Owner
2. Admin_id, Email, Password

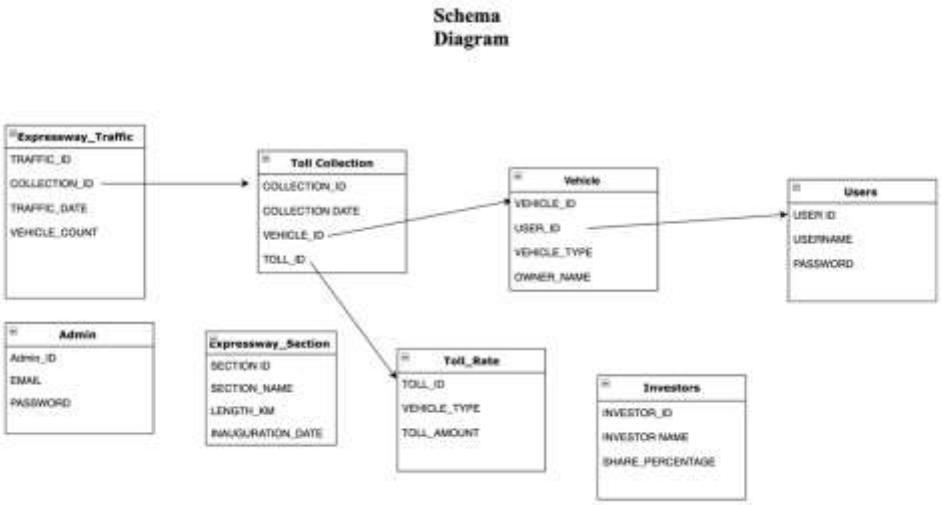
Temporary Tables

1. Vehicle_id, Vehicle_Type, Owner_Name
2. Collection_Id, Collection_Date
3. ~~Collection_Id, Collection_Date~~
4. Traffic id no, Traffic_date, Vehicle_count
5. ~~Traffic_id, Traffic_date, Vehicle_count~~
6. ~~Section_id, Section_name, Length_km, Inauguration_date~~
7. Section_id, Section_name, Length_km, Inauguration_date
8. Investor_name, First_name, Last_name
9. Investor_id, share_percentage
10. ~~Section_id, Section_name, Length_km, Inauguration_date~~
11. ~~Investor_name, First_name, Last_name~~
12. ~~Investor_id, share_percentage~~
13. Toll_id, Toll_amount
14. User_id, Username, Password
15. ~~Vehicle_id, Vehicle_Type, Owner_Name~~
16. Admin_id, Email, Password

Final Tables

1. Vehicle_id, Vehicle_Type, Owner_Name
2. Collection_Id, Collection_Date
3. Traffic id, Traffic_date, Vehicle_count
4. Section_id, Section_name, Length_km, Inauguration_date
5. Investor_name, First_name, Last_name
6. Investor_id, share_percentage
7. Toll_id, Toll_amount
8. Admin_id, Email, Password

8. Schema Diagram



9. Table Creation

<u>Vehicles</u>

```
CREATE TABLE Vehicles (
    vehicle_id Number PRIMARY KEY,
    user_id Number,
    vehicle_type VARCHAR(50) NOT NULL,
    owner_name VARCHAR(100) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE INDEX idx_vehicle_type ON Vehicles(vehicle_type);

create sequence vehicle_id_seq
start with 1000
increment by 1
minvalue 1000
maxvalue 1999
nocycle;
```

```
CREATE INDEX idx_vehicle_type ON Vehicles(vehicle_type);

create sequence vehicle_id_seq
start with 1000
increment by 1
minvalue 1000
maxvalue 1999
nocycle;
```

Figure: Vehicle Index

Toll Rates

CREATE TABLE Toll_Rates (

toll_id Number PRIMARY KEY,

vehicle_type VARCHAR(50) NOT NULL,

toll_amount DECIMAL(10, 2) NOT NULL);

CREATE INDEX idx_toll_amount ON Toll_Rates(toll_amount);

create sequence toll_id_seq

start with 2000

increment by 1

minvalue 2000

maxvalue 2999

nocycle;

```
CREATE TABLE Toll_Rates (
    toll_id Number PRIMARY KEY,
    vehicle_type VARCHAR(50) NOT NULL,
    toll_amount DECIMAL(10, 2) NOT NULL
);

CREATE INDEX idx_toll_amount ON Toll_Rates(toll_amount);
```

Results Explain Describe Saved SQL History

```
Index created.
```

0.00 seconds

Toll Collections

```
CREATE TABLE Toll_Collections (
    collection_id Number PRIMARY KEY,
    collection_date DATE NOT NULL,
    vehicle_id Number,
    toll_id Number,
    FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id),
    FOREIGN KEY (toll_id) REFERENCES Toll_Rates(toll_id)
);

CREATE INDEX idx_collection_date ON Toll_Collections(collection_date);

create sequence collection_id_seq
start with 3000
increment by 1
minvalue 3000
maxvalue 3999
nocycle;
```

```
CREATE TABLE Toll_Collections (
    collection_id Number PRIMARY KEY,
    collection_date DATE NOT NULL,
    vehicle_id INT,
    toll_id INT,
    FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id),
    FOREIGN KEY (toll_id) REFERENCES Toll_Rates(toll_id)
);

CREATE INDEX idx_collection_date ON Toll_Collections(collection_date);
```

Results Explain Describe Saved SQL History

```
Index created.
```

0.00 seconds

Expressway Traffic

CREATE TABLE Expressway_Traffic (

traffic_id Number PRIMARY KEY,

collection_id Number,

traffic_date DATE NOT NULL,

vehicle_count INT NOT NULL,

FOREIGN KEY (**collection_id**) REFERENCES Toll_Collections(collection_id)

);

CREATE INDEX idx_vehicle_count ON Expressway_Traffic(vehicle_count);

create sequence traffic_id_seq

start with 4000

increment by 1

minvalue 4000

maxvalue 4999

nocycle;

```
CREATE TABLE Expressway_Traffic (
    traffic_id Number PRIMARY KEY,
    traffic_date DATE NOT NULL,
    vehicle_count INT NOT NULL
);

CREATE INDEX idx_vehicle_count ON Expressway_Traffic(vehicle_count);

create sequence traffic_id_seq
start with 4000
increment by 1
minvalue 4000
maxvalue 4999
nocycle;
```

Results Explain Describe Saved SQL History

```
Index created.
```

0.00 seconds

Investors

CREATE TABLE Investors (

investor_id Number PRIMARY KEY,

investor_name VARCHAR(100) NOT NULL,

share_percentage DECIMAL(5, 2) NOT NULL

);

CREATE INDEX idx_investor_name ON Investors(investor_name);

create sequence investor_id_seq

start with 5000

increment by 1

minvalue 5000

maxvalue 5999

nocycle;

```
CREATE TABLE Investors (
    investor_id Number PRIMARY KEY,
    investor_name VARCHAR(100) NOT NULL,
    share_percentage DECIMAL(5, 2) NOT NULL
);

CREATE INDEX idx_investor_name ON Investors(investor_name);
```

Results Explain Describe Saved SQL History

```
Index created.
```

0.00 seconds

Expressway Section

CREATE TABLE Expressway_Section (

section_id Number PRIMARY KEY,

section_name VARCHAR(100) NOT NULL,

length_km DECIMAL(6, 2) NOT NULL,

inauguration_date DATE NOT NULL

);

CREATE INDEX idx_section_name ON Expressway_Section(section_name);

create sequence section_id_seq

start with 6000

increment by 1

minvalue 6000

maxvalue 6999

nocycle;

```
CREATE TABLE Expressway_Section (
    section_id Number PRIMARY KEY,
    section_name VARCHAR(100) NOT NULL,
    length_km DECIMAL(6, 2) NOT NULL,
    inauguration_date DATE NOT NULL
);

CREATE INDEX idx_section_name ON Expressway_Section(section_name);
```

Results Explain Describe Saved SQL History

```
Index created.
```

0.11 seconds

Users

CREATE TABLE Users (

user_id Number PRIMARY KEY,

username VARCHAR (50),

password varchar(12) NOT NULL

);

create sequence user_id_seq

start with 7000

increment by 1

minvalue 7000

maxvalue 7999

nocycle;

CREATE INDEX idx_password ON Users(password);

```
create sequence user_id_seq
start with 7000
increment by 1
minvalue 7000
maxvalue 7999
nocycle;

CREATE INDEX idx_password ON Users(password);
```

Results Explain Describe Saved SQL History

```
Index created.
```

0.02 seconds

Admin

CREATE TABLE Admin(

admin_id Number PRIMARY KEY,

email Varchar(50),

password varchar(12) NOT NULL

);

CREATE INDEX idx_admin_email ON Admin(email);

Privileges

Admin Role Creation:

CREATE ROLE Admin;

GRANT ALL PRIVILEGES TO Admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON Vehicles TO Admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON Toll_Rates TO Admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON Toll_Collections TO Admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON Expressway_Traffic TO Admin;

```
GRANT SELECT, INSERT, UPDATE, DELETE ON Investors TO Admin;
GRANT SELECT, INSERT, UPDATE, DELETE ON Expressway_Section TO Admin;
GRANT SELECT, INSERT, UPDATE, DELETE ON Users TO Admin;
```

Users Role Creation:

```
CREATE ROLE Users;

CREATE VIEW Vehicle_Details_View AS
SELECT vehicle_type, owner_name
FROM Vehicles;

GRANT UPDATE (username, password) ON Users TO Users;
GRANT SELECT ON Vehicle_Details_View TO Users;
GRANT SELECT ON Toll_Rates TO Users;
GRANT SELECT ON Toll_Collections TO Users;
```

Admin creation

```
CREATE USER Admin1 IDENTIFIED BY pass123;
CREATE USER Admin2 IDENTIFIED BY abc456;
CREATE USER Admin3 IDENTIFIED BY xyz789;

GRANT Admin TO Admin1, Admin2, Admin3;
GRANT UNLIMITED TABLESPACE TO Admin1, Admin2, Admin3;
```

General Users Creation

```
CREATE USER AmjadKhan IDENTIFIED BY pass123;
CREATE USER TariqAziz IDENTIFIED BY abc456;
CREATE USER SalmaSheikh IDENTIFIED BY xyz789;
CREATE USER SelimAzad IDENTIFIED BY pass456;
CREATE USER FaruqHasan IDENTIFIED BY abc789;
```

GRANT Users TO AmjadKhan, TariqAziz, SalmaSheikh, SelimAzad, FaruqHasan;

10. Data Insertion

Vehicles

INSERT INTO Vehicles (vehicle_id, **user_id**, vehicle_type, owner_name) VALUES (vehicle_id_seq.nextval, 7000, 'Private Car','Amjad Khan');

INSERT INTO Vehicles (vehicle_id, **user_id**, vehicle_type, owner_name) VALUES (vehicle_id_seq.nextval, 7001,'Taxi','Tariq Aziz');

INSERT INTO Vehicles (vehicle_id, **user_id**, vehicle_type, owner_name) VALUES (vehicle_id_seq.nextval, 7002, 'SUV','Salma Sheikh');

INSERT INTO Vehicles (vehicle_id, **user_id**, vehicle_type, owner_name) VALUES (vehicle_id_seq.nextval,7003, 'Microbus','Selim Azad');

INSERT INTO Vehicles (vehicle_id, **user_id**, vehicle_type, owner_name) VALUES (vehicle_id_seq.nextval, 7004,'Light Truck','Faruq Hasan');

Select *

from Vehicles;

```
INSERT INTO Vehicles (vehicle_id, user_id, vehicle_type, owner_name) VALUES
(vehicle_id_seq.nextval,7003, 'Microbus','Selim Azad');

INSERT INTO Vehicles (vehicle_id, user_id, vehicle_type, owner_name) VALUES
(vehicle_id_seq.nextval, 7004,'Light Truck','Faruq Hasan');

Select *
from Vehicles;
```

Results Explain Describe Saved SQL History

VEHICLE_ID	USER_ID	VEHICLE_TYPE	OWNER_NAME
1020	7000	Private Car	Amjad Khan
1021	7001	Taxi	Tariq Aziz
1022	7002	SUV	Salma Sheikh
1023	7003	Microbus	Selim Azad
1024	7004	Light Truck	Faruq Hasan

5 rows returned in 0.00 seconds CSV Export

Toll Rates

INSERT INTO Toll_Rates (toll_id, vehicle_type, toll_amount) VALUES

(toll_id_seq.nextval,'Private Car', 80);

INSERT INTO Toll_Rates (toll_id, vehicle_type, toll_amount) VALUES

(toll_id_seq.nextval,'Taxi', 120);

INSERT INTO Toll_Rates (toll_id, vehicle_type, toll_amount) VALUES

(toll_id_seq.nextval,'SUV', 100);

INSERT INTO Toll_Rates (toll_id, vehicle_type, toll_amount) VALUES

(toll_id_seq.nextval,'Microbus', 150);

INSERT INTO Toll_Rates (toll_id, vehicle_type, toll_amount) VALUES

(toll_id_seq.nextval,'Light Truck', 180);

Select *

from toll_rates;

```
select *
from toll_rates;
```

Results Explain Describe Saved SQL History

TOLL_ID	VEHICLE_TYPE	TOLL_AMOUNT
2000	Private Car	80
2001	Taxi	120
2002	SUV	100
2003	Microbus	150
2004	Light Truck	180

5 rows returned in 0.00 seconds CSV Export

Toll Collections

INSERT INTO Toll_Collections (collection_id, collection_date, **vehicle_id**, **toll_id**) VALUES (collection_id_seq.nextval, TO_DATE('2024-02-01', 'YYYY-MM-DD'), 1020, 2000);

INSERT INTO Toll_Collections (collection_id, collection_date, **vehicle_id**, **toll_id**) VALUES (collection_id_seq.nextval, TO_DATE('2024-02-03', 'YYYY-MM-DD'), 1021, 2001);

INSERT INTO Toll_Collections (collection_id, collection_date, **vehicle_id**, **toll_id**) VALUES (collection_id_seq.nextval, TO_DATE('2024-02-07', 'YYYY-MM-DD'), 1022, 2002);

INSERT INTO Toll_Collections (collection_id, collection_date, **vehicle_id**, **toll_id**) VALUES (collection_id_seq.nextval, TO_DATE('2024-02-13', 'YYYY-MM-DD'), 1023, 2003);

INSERT INTO Toll_Collections (collection_id, collection_date, **vehicle_id**, **toll_id**) VALUES (collection_id_seq.nextval, TO_DATE('2024-02-16', 'YYYY-MM-DD'), 1024, 2004);

Select *

from toll_collections;

```
INSERT INTO Toll_Collections (collection_id, collection_date, vehicle_id, toll_id) VALUES
(3003, TO_DATE('2024-02-13', 'YYYY-MM-DD'), 1023, 2003);

INSERT INTO Toll_Collections (collection_id, collection_date, vehicle_id, toll_id) VALUES
(3004, TO_DATE('2024-02-16', 'YYYY-MM-DD'), 1024, 2004);

Select *
from toll_collections;
```

Results Explain Describe Saved SQL History

COLLECTION_ID	COLLECTION_DATE	VEHICLE_ID	TOLL_ID
3000	01-FEB-24	1020	2000
3001	03-FEB-24	1021	2001
3002	07-FEB-24	1022	2002
3003	13-FEB-24	1023	2003
3004	16-FEB-24	1024	2004

5 rows returned in 0.00 seconds CSV Export

Expressway Traffic

INSERT INTO Expressway_Traffic (traffic_id, collection_id, traffic_date, vehicle_count) VALUES

(traffic_id_seq.nextval, 3000, TO_DATE('2024-02-01', 'YYYY-MM-DD'), 3652);

INSERT INTO Expressway_Traffic (traffic_id, collection_id, traffic_date, vehicle_count) VALUES

(traffic_id_seq.nextval, 3001, TO_DATE('2024-02-03', 'YYYY-MM-DD'), 2212);

INSERT INTO Expressway_Traffic (traffic_id, collection_id, traffic_date, vehicle_count) VALUES

(traffic_id_seq.nextval, 3002, TO_DATE('2024-02-07', 'YYYY-MM-DD'), 2658);

INSERT INTO Expressway_Traffic (traffic_id, collection_id, traffic_date, vehicle_count) VALUES

(traffic_id_seq.nextval, 3003, TO_DATE('2024-02-13', 'YYYY-MM-DD'), 4297);

INSERT INTO Expressway_Traffic (traffic_id, collection_id, traffic_date, vehicle_count) VALUES (traffic_id_seq.nextval, 3004, TO_DATE('2024-02-16', 'YYYY-MM-DD'), 1643);

Select *

from expressway_traffic;

```
select *
from expressway_traffic;
```

Results Explain Describe Saved SQL History

TRAFFIC_ID	COLLECTION_ID	TRAFFIC_DATE	VEHICLE_COUNT
4000	3000	01-FEB-24	3652
4001	3001	03-FEB-24	2212
4002	3002	07-FEB-24	2658
4003	3003	13-FEB-24	4297
4004	3004	16-FEB-24	1643

5 rows returned in 0.00 seconds CSV Export

Investors

INSERT INTO Investors (investor_id, investor_name, share_percentage) VALUES

(investor_id_seq.nextval, 'Italian Thai Development Public Company Limited', 51);

INSERT INTO Investors (investor_id, investor_name, share_percentage) VALUES

(investor_id_seq.nextval, 'China Shandong International Economic and Technical Cooperation Group', 34);

INSERT INTO Investors (investor_id, investor_name, share_percentage) VALUES

(investor_id_seq.nextval, 'Sinohydro Corporation Limited', 15);

select *

from investors;

```
INSERT INTO Investors (investor_id, investor_name, share_percentage) VALUES
(investor_id_seq.nextval, 'Italian Thai Development Public Company Limited', 51);

INSERT INTO Investors (investor_id, investor_name, share_percentage) VALUES
(investor_id_seq.nextval, 'China Shandong International Economic and Technical Cooperation Group', 34);

INSERT INTO Investors (investor_id, investor_name, share_percentage) VALUES
(investor_id_seq.nextval, 'Sinohydro Corporation Limited', 15);

select *
from investors;
```

Results Explain Describe Saved SQL History

INVESTOR_ID	INVESTOR_NAME	SHARE_PERCENTAGE
5000	Italian Thai Development Public Company Limited	51
5001	China Shandong International Economic and Technical Cooperation Group	34
5002	Sinohydro Corporation Limited	15

3 rows returned in 0.00 seconds CSV Export

Expressway Section

INSERT INTO Expressway_Section (section_id, section_name, length_km, inauguration_date) VALUES

(section_id_seq.nextval, 'Kawla to Tejgaon', 11.5, TO_DATE('2023-09-02', 'YYYY-MM-DD'));

INSERT INTO Expressway_Section (section_id, section_name, length_km, inauguration_date) VALUES

(section_id_seq.nextval, 'Tejgaon to Mohakhali', 4.23, TO_DATE('2023-09-02', 'YYYY-MM-DD'));

INSERT INTO Expressway_Section (section_id, section_name, length_km, inauguration_date) VALUES

(section_id_seq.nextval, 'Mohakhali to Kamalapur', 3.5, TO_DATE('2023-09-02', 'YYYY-MM-DD'));

INSERT INTO Expressway_Section (section_id, section_name, length_km, inauguration_date) VALUES

(section_id_seq.nextval, 'Kamalapur to Jatrabari', 5.5, TO_DATE('2023-09-02', 'YYYY-MM-DD'));

INSERT INTO Expressway_Section (section_id, section_name, length_km, inauguration_date) VALUES

(section_id_seq.nextval, 'Jatrabari to Kutubkhali', 7.0, TO_DATE('2023-09-02', 'YYYY-MM-DD'));

select *

from expressway_section;

```
INSERT INTO Expressway_Section (section_id, section_name, length_km, inauguration_date) VALUES
(section_id_seq.nextval, 'Tejgaon to Mohakhali', 4.23, TO_DATE('2023-09-02', 'YYYY-MM-DD'));

INSERT INTO Expressway_Section (section_id, section_name, length_km, inauguration_date) VALUES
(section_id_seq.nextval, 'Mohakhali to Kamalapur', 3.5, TO_DATE('2023-09-02', 'YYYY-MM-DD'));

INSERT INTO Expressway_Section (section_id, section_name, length_km, inauguration_date) VALUES
(section_id_seq.nextval, 'Kamalapur to Jatrabari', 5.5, TO_DATE('2023-09-02', 'YYYY-MM-DD'));

INSERT INTO Expressway_Section (section_id, section_name, length_km, inauguration_date) VALUES
(section_id_seq.nextval, 'Jatrabari to Kutubkhali', 7.0, TO_DATE('2023-09-02', 'YYYY-MM-DD'));

select *
from expressway_section;
```

Results Explain Describe Saved SQL History

SECTION_ID	SECTION_NAME	LENGTH_KM	INAUGURATION_DATE
6000	Kawla to Tejgaon	11.5	02-SEP-23
6001	Tejgaon to Mohakhali	4.23	02-SEP-23
6002	Mohakhali to Kamalapur	3.5	02-SEP-23
6003	Kamalapur to Jatrabari	5.5	02-SEP-23
6004	Jatrabari to Kutubkhali	7	02-SEP-23

5 rows returned in 0.00 seconds CSV Export

Users

INSERT INTO Users (user_id, username, password) VALUES

(user_id_seq.nextval, 'Amjad Khan', 'pass123');

INSERT INTO Users (user_id, username, password) VALUES

(user_id_seq.nextval, 'Tariq Aziz', 'abc456');

INSERT INTO Users (user_id, username, password) VALUES

(user_id_seq.nextval, 'Salma Sheikh', 'xyz789');

INSERT INTO Users (user_id, username, password) VALUES

(user_id_seq.nextval, 'Selim Azad', 'pass456');

INSERT INTO Users (user_id, username, password) VALUES

(user_id_seq.nextval, 'Faruq Hasan', 'abc789');

Select *

from Users;

```
INSERT INTO Users (user_id, username, password) VALUES
(user_id_seq.nextval, 'Amjad Khan', 'pass123');
INSERT INTO Users (user_id, username, password) VALUES
(user_id_seq.nextval, 'Tariq Aziz', 'abc456');
INSERT INTO Users (user_id, username, password) VALUES
(user_id_seq.nextval, 'Salma Sheikh', 'xyz789');
INSERT INTO Users (user_id, username, password) VALUES
(user_id_seq.nextval, 'Selim Azad', 'pass456');
INSERT INTO Users (user_id, username, password) VALUES
(user_id_seq.nextval, 'Faruq Hasan', 'abc789');

Select *
from Users;
```

Results Explain Describe Saved SQL History

USER_ID	USERNAME	PASSWORD
7000	Amjad Khan	pass123
7001	Tariq Aziz	abc456
7002	Salma Sheikh	xyz789
7003	Selim Azad	pass456
7004	Faruq Hasan	abc789

5 rows returned in 0.00 seconds CSV Export

Admin

INSERT INTO Admin (admin_id, email, password) VALUES

(1, 'admin1@gmail.com', 'pass123');

INSERT INTO Admin (admin_id, email, password) VALUES

(2, 'admin2@gmail.com', 'abc456');

INSERT INTO Admin (admin_id, email, password) VALUES

(3, 'admin3@gmail.com', 'xyz789');

select *

from admin;

```
CREATE TABLE Admin(
    admin_id Number PRIMARY KEY,
    email Varchar(50),
    password varchar(12) NOT NULL
);

INSERT INTO Admin (admin_id, email, password) VALUES
(1, 'admin1@gmail.com', 'pass123');
INSERT INTO Admin (admin_id, email, password) VALUES
(2, 'admin2@gmail.com', 'abc456');
INSERT INTO Admin (admin_id, email, password) VALUES
(3, 'admin3@gmail.com', 'xyz789');

select *
```

Results Explain Describe Saved SQL History

ADMIN_ID	EMAIL	PASSWORD
1	admin1@gmail.com	pass123
2	admin2@gmail.com	abc456
3	admin3@gmail.com	xyz789

3 rows returned in 0.00 seconds CSV Export

11. Query Writing

Single Row functions:

1. Use an upper function that converts owner name into upper case.

SELECT UPPER(owner_name) AS upper_case_owner_name

FROM Vehicles;

```
SELECT UPPER(owner_name) AS upper_case_owner_name
FROM Vehicles;
```

Results Explain Describe Saved SQL History

UPPER_CASE_OWNER_NAME
AMJAD KHAN
TARIQ AZIZ
SALMA SHEIKH
SELIM AZAD
FARUQ HASAN

5 rows returned in 0.00 seconds CSV Export

2. Use Round function to convert the length into rounded values.

SELECT section_name, ROUND(length_km) AS rounded_length_km

FROM Expressway_Section;

```
SELECT section_name, ROUND(length_km) AS rounded_length_km
FROM Expressway_Section;
```

Results Explain Describe Saved SQL History

SECTION_NAME	ROUNDED_LENGTH_KM
Kawla to Tejgaon	12
Tejgaon to Mohakhali	4
Mohakhali to Kamalapur	4
Kamalapur to Jatrabari	6
Jatrabari to Kutubkhali	7

5 rows returned in 0.00 seconds CSV Export

3. Use Concat function to join both owner name and their respective vehicles.

SELECT CONCAT(owner_name, vehicle_type) AS owner_vehicle_concatenated

FROM Vehicles;

```
SELECT CONCAT(owner_name, vehicle_type) AS owner_vehicle_concatenated
FROM Vehicles;
```

Results Explain Describe Saved SQL History

OWNER_VEHICLE_CONCATENATED
Amjad KhanPrivate Car
Tariq AzizTaxi
Salma SheikhSUV
Selim AzadMicrobus
Faruq HasanLight Truck

5 rows returned in 0.00 seconds CSV Export

Group functions:

1. Use Max function to check the maximum vehicle count on expressway traffic

SELECT MAX(vehicle_count) AS max_vehicle_count

FROM Expressway_Traffic;

```
SELECT MAX(vehicle_count) AS max_vehicle_count
FROM Expressway_Traffic;
```

Results Explain Describe Saved SQL History

MAX_VEHICLE_COUNT
4297

1 rows returned in 0.01 seconds CSV Export

2. Use Variance function to calculate the variance of vehicle count

SELECT VARIANCE(vehicle_count) AS variance_vehicle_count

FROM Expressway_Traffic;

```
SELECT VARIANCE(vehicle_count) AS variance_vehicle_count
FROM Expressway_Traffic;
```

Results Explain Describe Saved SQL History

VARIANCE_VEHICLE_COUNT
1157195.3

1 rows returned in 0.00 seconds CSV Export

3. Use Average function to calculate the average length.

SELECT AVG(length_km) AS avg_length_km

FROM Expressway_Section;

```
SELECT AVG(length_km) AS avg_length_km
FROM Expressway_Section;
```

Results Explain Describe Saved SQL History

AVG_LENGTH_KM
6.346

1 rows returned in 0.00 seconds CSV Export

Subqueries

1. SELECT toll_amount
FROM Toll_Rates
WHERE vehicle_type='Private Car';

```
SELECT toll_amount
FROM Toll_Rates
WHERE vehicle_type = 'Private Car';
```

Results Explain Describe Saved SQL History

TOLL_AMOUNT
80

1 rows returned in 0.00 seconds CSV Export

2. SELECT Vehicle_type
FROM Vehicles
WHERE vehicle_id = 1001;

```
SELECT vehicle_type
FROM Vehicles
WHERE vehicle_id = 1001;
```

Results Explain Describe Saved SQL History

VEHICLE_TYPE
Taxi

1 rows returned in 0.02 seconds CSV Export

3. SELECT COUNT(*)
FROM Toll_Collections
WHERE Collection_date BETWEEN TO _DATE (‘2024-02-01', ‘YYYY-MM-DD’) AND TO_DATE (‘2024-02-06', ‘YYYY-MM-DD’);

```
SELECT COUNT(*)
FROM Toll_Collections
WHERE collection_date BETWEEN TO_DATE('2024-02-01', 'YYYY-MM-DD') AND TO_DATE('2024-02-06', 'YYYY-MM-DD');
```

Results Explain Describe Saved SQL History

COUNT(*)
2

1 rows returned in 0.00 seconds CSV Export

Join Queries

1. Create an Inner Join of Toll Rates table

SELECT TC.collection_id, TC.collection_date, V.owner_name, V.vehicle_type, TR.toll_amount

FROM Toll_Collections TC

INNER JOIN Vehicles V ON TC.vehicle_id = V.vehicle_id

INNER JOIN Toll_Rates TR ON TC.toll_id = TR.toll_id;

```
SELECT TC.collection_id, TC.collection_date, V.owner_name, V.vehicle_type, TR.toll_amount
FROM Toll_Collections TC
INNER JOIN Vehicles V ON TC.vehicle_id = V.vehicle_id
INNER JOIN Toll_Rates TR ON TC.toll_id = TR.toll_id;
```

Results Explain Describe Saved SQL History

COLLECTION_ID	COLLECTION_DATE	OWNER_NAME	VEHICLE_TYPE	TOLL_AMOUNT
3000	01-FEB-24	Amjad Khan	Private Car	80
3001	03-FEB-24	Tariq Aziz	Taxi	120
3002	07-FEB-24	Salma Sheikh	SUV	100
3003	13-FEB-24	Selim Azad	Microbus	150
3004	16-FEB-24	Faruq Hasan	Light Truck	180

5 rows returned in 0.00 seconds CSV Export

2. Using LEFT JOIN operation retrieve data from Expressway Traffic and Toll collections table

SELECT ET.traffic_id, ET.traffic_date, ET.vehicle_count, TC.collection_date

FROM Expressway_Traffic ET

LEFT JOIN Toll_Collections TC ON ET.collection_id = TC.collection_id;

```
SELECT ET.traffic_id, ET.traffic_date, ET.vehicle_count, TC.collection_date
FROM Expressway_Traffic ET
LEFT JOIN Toll_Collections TC ON ET.collection_id = TC.collection_id;
```

Results Explain Describe Saved SQL History

TRAFFIC_ID	TRAFFIC_DATE	VEHICLE_COUNT	COLLECTION_DATE
4000	01-FEB-24	3652	01-FEB-24
4001	03-FEB-24	2212	03-FEB-24
4002	07-FEB-24	2658	07-FEB-24
4003	13-FEB-24	4297	13-FEB-24
4004	16-FEB-24	1643	16-FEB-24

5 rows returned in 0.00 seconds CSV Export

3. Create a Full Outer Join operation between Expressway Section and Investors

SELECT *

FROM Investors I

FULL OUTER JOIN Expressway_Section ES ON I.investor_id = ES.section_id;

```
SELECT *
FROM Investors I
FULL OUTER JOIN Expressway_Section ES ON I.investor_id = ES.section_id;
```

Results Explain Describe Saved SQL History

INVESTOR_ID	INVESTOR_NAME	SHARE_PERCENTAGE	SECTION_ID	SECTION_NAME	LENGTH_KM	INAUGURATION_DATE
5000	Italian Thai Development Public Company Limited	51	-	-	-	-
5001	China Shandong International Economic and Technical Cooperation Group	34	-	-	-	-
5002	Sinohydro Corporation Limited	15	-	-	-	-
-	-	-	6000	Kavila to Tejgaon	11.5	02-SEP-23
-	-	-	6001	Tejgaon to Mohakhali	4.23	02-SEP-23
-	-	-	6002	Mohakhali to Kamalapur	3.5	02-SEP-23
-	-	-	6003	Kamalapur to Jatrabari	5.5	02-SEP-23
-	-	-	6004	Jatrabari to Kutubkhali	7	02-SEP-23

8 rows returned in 0.00 seconds CSV Export

View Queries

1. Make a view of Toll Collection table

CREATE VIEW Toll_Collection_Details_View AS

SELECT TC.collection_id, TC.collection_date, V.owner_name, V.vehicle_type, TR.toll_amount

FROM Toll_Collections TC

INNER JOIN Vehicles V ON TC.vehicle_id = V.vehicle_id

INNER JOIN Toll_Rates TR ON TC.toll_id = TR.toll_id;

SELECT * FROM Toll_Collection_Details_View;

```
CREATE VIEW Toll_Collection_Details_View AS
SELECT TC.collection_id, TC.collection_date, V.owner_name, V.vehicle_type, TR.toll_amount
FROM Toll_Collections TC
INNER JOIN Vehicles V ON TC.vehicle_id = V.vehicle_id
INNER JOIN Toll_Rates TR ON TC.toll_id = TR.toll_id;

SELECT * FROM Toll_Collection_Details_View;
```

Results Explain Describe Saved SQL History

COLLECTION_ID	COLLECTION_DATE	OWNER_NAME	VEHICLE_TYPE	TOLL_AMOUNT
3000	01-FEB-24	Amjad Khan	Private Car	80
3001	03-FEB-24	Tariq Aziz	Taxi	120
3002	07-FEB-24	Salma Sheikh	SUV	100
3003	13-FEB-24	Selim Azad	Microbus	150
3004	16-FEB-24	Faruq Hasan	Light Truck	180

5 rows returned in 0.00 seconds CSV Export

2. Create an Expressway Traffic View

CREATE VIEW Expressway_Traffic_View AS

SELECT ET.traffic_id, ET.traffic_date, ET.vehicle_count, TC.collection_date

FROM Expressway_Traffic ET

LEFT JOIN Toll_Collections TC ON ET.collection_id = TC.collection_id;

```
CREATE VIEW Expressway_Traffic_View AS
SELECT ET.traffic_id, ET.traffic_date, ET.vehicle_count, TC.collection_date
FROM Expressway_Traffic ET
LEFT JOIN Toll_Collections TC ON ET.collection_id = TC.collection_id;

SELECT * FROM Expressway_Traffic_View;
```

Results Explain Describe Saved SQL History

TRAFFIC_ID	TRAFFIC_DATE	VEHICLE_COUNT	COLLECTION_DATE
4000	01-FEB-24	3652	01-FEB-24
4001	03-FEB-24	2212	03-FEB-24
4002	07-FEB-24	2658	07-FEB-24
4003	13-FEB-24	4297	13-FEB-24
4004	16-FEB-24	1643	16-FEB-24

5 rows returned in 0.00 seconds CSV Export

SELECT * FROM Expressway_Traffic_View;

3. Create an Investors table View

CREATE VIEW Investors_Section_View AS

SELECT I.investor_id, I.investor_name, I.share_percentage, ES.section_name, ES.length_km, ES.inauguration_date

FROM Investors I

FULL OUTER JOIN Expressway_Section ES ON I.investor_id = ES.section_id;

SELECT * FROM Investors_Section_View;

```
CREATE VIEW Investors_Section_View AS
SELECT I.investor_id, I.investor_name, I.share_percentage, E5.section_name, E5.length_km, E5.inauguration_date
FROM Investors I
FULL OUTER JOIN Expressway_Section E5 ON I.investor_id = E5.section_id;

SELECT * FROM Investors_Section_View;
```

Results Explain Describe Saved SQL History

INVESTOR_ID	INVESTOR_NAME	SHARE_PERCENTAGE	SECTION_NAME	LENGTH_KM	INAUGURATION_DATE
5000	Italian Thai Development Public Company Limited	51	-	-	-
5001	China Shandong International Economic and Technical Cooperation Group	34	-	-	-
5002	Sinohydro Corporation Limited	15	-	-	-
-	-	-	Kavla to Tejgaon	11.5	02-SEP-23
-	-	-	Tejgaon to Mohakhali	4.23	02-SEP-23
-	-	-	Mohakhali to Kamalapur	3.6	02-SEP-23
-	-	-	Kamalapur to Jatrabari	5.5	02-SEP-23
-	-	-	Jatrabari to Kutubkhali	7	02-SEP-23

Synonym

1. Create a Vehicle Synonym

CREATE SYNONYM Vehicle_Synonym FOR Vehicles;

SELECT * FROM Vehicle_Synonym;

```
CREATE SYNONYM Vehicle_Synonym FOR Vehicles;
SELECT * FROM Vehicle_Synonym;
```

Results Explain Describe Saved SQL History

VEHICLE_ID	USER_ID	VEHICLE_TYPE	OWNER_NAME
1020	7000	Private Car	Amjad Khan
1021	7001	Taxi	Tariq Aziz
1022	7002	SUV	Salma Sheikh
1023	7003	Microbus	Selim Azad
1024	7004	Light Truck	Faruq Hasan

5 rows returned in 0.00 seconds CSV Export

2. Create a Toll Rates Synonym

CREATE SYNONYM Toll_Rates_Synonym FOR Toll_Rates;

SELECT * FROM Toll_Rates_Synonym;

```
CREATE SYNONYM Toll_Rates_Synonym FOR Toll_Rates;
SELECT * FROM Toll_Rates_Synonym;
```

Results Explain Describe Saved SQL History

TOLL_ID	VEHICLE_TYPE	TOLL_AMOUNT
2000	Private Car	80
2001	Taxi	120
2002	SUV	100
2003	Microbus	150
2004	Light Truck	180

5 rows returned in 0.00 seconds CSV Export

3. Create an Users Synonym

CREATE SYNONYM Users_Synonym FOR Users;

SELECT * FROM Users_Synonym;

```
CREATE SYNONYM Users_Synonym FOR Users;
SELECT * FROM Users_Synonym;
```

Results Explain Describe Saved SQL History

USER_ID	USERNAME	PASSWORD
7000	Amjad Khan	pass123
7001	Tariq Aziz	abc456
7002	Salma Sheikh	xyz789
7003	Selim Azad	pass456
7004	Faruq Hasan	abc789

5 rows returned in 0.00 seconds CSV Export

12. PL/SQL

Functions

1. Create a function to retrieve the count of vehicles based on a specific vehicle type?

```
CREATE OR REPLACE FUNCTION get_vehicle_count(vehicle_type_in IN VARCHAR2)
RETURN NUMBER
IS
   v_count NUMBER;
BEGIN
   SELECT COUNT(*)
   INTO v_count
   FROM Vehicles
   WHERE vehicle_type = vehicle_type_in;
   RETURN v_count;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN 0;
   WHEN OTHERS THEN
      RETURN NULL;
END;
/
```

Autocommit Display 10

```
CREATE OR REPLACE FUNCTION get_vehicle_count(vehicle_type_in IN VARCHAR2)
RETURN NUMBER
IS
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM Vehicles
    WHERE vehicle_type = vehicle_type_in;

    RETURN v_count;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;
    WHEN OTHERS THEN
        RETURN NULL;
END;
/
```

Results Explain Describe Saved SQL History

```
Function created.
```

0.05 seconds

2. Create a function that retrieves the toll amount based on a given vehicle type.

```
CREATE OR REPLACE FUNCTION get_toll_amount(vehicle_type_in IN VARCHAR2)
RETURN DECIMAL
IS
   v_toll_amount DECIMAL(10, 2);
BEGIN
   SELECT toll_amount
   INTO v_toll_amount
   FROM Toll_Rates
   WHERE vehicle_type = vehicle_type_in;
   RETURN v_toll_amount;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN 0.0;
   WHEN OTHERS THEN
      RETURN NULL;
END;
/
```

```
CREATE OR REPLACE FUNCTION get_toll_amount(vehicle_type_in IN VARCHAR2)
RETURN DECIMAL
IS
    v_toll_amount DECIMAL(10, 2);
BEGIN
    SELECT toll_amount
    INTO v_toll_amount
    FROM Toll_Rates
    WHERE vehicle_type = vehicle_type_in;
    RETURN v_toll_amount;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0.0;
    WHEN OTHERS THEN
        RETURN NULL;
END;
/
```

Results Explain Describe Saved SQL History

```
Function created.
```

0.01 seconds

3. Create a function that retrieves the total toll collected on a specific date

```
CREATE OR REPLACE FUNCTION get_total_toll_collected(collection_date_in IN DATE)
RETURN DECIMAL
IS
   v_total_toll DECIMAL(10, 2);
BEGIN
   SELECT SUM(TR.toll_amount)
   INTO v_total_toll
   FROM Toll_Collections TC
   INNER JOIN Toll_Rates TR ON TC.toll_id = TR.toll_id
   WHERE TC.collection_date = collection_date_in;
   RETURN v_total_toll;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN 0.0;
   WHEN OTHERS THEN
      RETURN NULL;
END;
/
```

```
CREATE OR REPLACE FUNCTION get_total_toll_collected(collection_date_in IN DATE)
RETURN DECIMAL
IS
    v_total_toll DECIMAL(10, 2);
BEGIN
    SELECT SUM(TR.toll_amount)
    INTO v_total_toll
    FROM Toll_Collections TC
    INNER JOIN Toll_Rates TR ON TC.toll_id = TR.toll_id
    WHERE TC.collection_date = collection_date_in;
    RETURN v_total_toll;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0.0;
    WHEN OTHERS THEN
        RETURN NULL;
END;
/
```

Results Explain Describe Saved SQL History

```
Function created.
```

0.02 seconds

Procedure

1. Create a procedure that inserts a new vehicle record into the table.

```
CREATE OR REPLACE PROCEDURE insert_vehicle(
   p_user_id IN NUMBER,
   p_vehicle_type IN VARCHAR2,
   p_owner_name IN VARCHAR2
)
IS
BEGIN
   INSERT INTO Vehicles (vehicle_id, user_id, vehicle_type, owner_name)
   VALUES (vehicle_id_seq.nextval, p_user_id, p_vehicle_type, p_owner_name);
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Vehicle inserted successfully.');
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error inserting vehicle: ' || SQLERRM);
END;
/
```

```
CREATE OR REPLACE PROCEDURE insert_vehicle(
    p_user_id IN NUMBER,
    p_vehicle_type IN VARCHAR2,
    p_owner_name IN VARCHAR2
)
IS
BEGIN
    INSERT INTO Vehicles (vehicle_id, user_id, vehicle_type, owner_name)
    VALUES (vehicle_id_seq.nextval, p_user_id, p_vehicle_type, p_owner_name);

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Vehicle inserted successfully.');
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error inserting vehicle: ' || SQLERRM);
END;
/
```

Results Explain Describe Saved SQL History

```
Procedure created.
```

0.01 seconds

2. Define a PL/SQL procedure that performs operations such as inserting, updating, or deleting toll_rates.

```
CREATE OR REPLACE PROCEDURE Insert_Toll_Rate(
   p_toll_id IN NUMBER,
   p_vehicle_type IN VARCHAR2,
   p_toll_amount IN NUMBER
)
IS
BEGIN
   INSERT INTO Toll_Rates(toll_id, vehicle_type, toll_amount)
   VALUES(p_toll_id, p_vehicle_type, p_toll_amount);
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Toll rate inserted successfully.');
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error inserting toll rate: ' || SQLERRM);
END Insert_Toll_Rate;
/
```

```
CREATE OR REPLACE PROCEDURE Insert_Toll_Rate(
    p_toll_id IN NUMBER,
    p_vehicle_type IN VARCHAR2,
    p_toll_amount IN NUMBER
)
IS
BEGIN
    INSERT INTO Toll_Rates(toll_id, vehicle_type, toll_amount)
    VALUES(p_toll_id, p_vehicle_type, p_toll_amount);

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Toll rate inserted successfully.');
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error inserting toll rate: ' || SQLERRM);
END Insert_Toll_Rate;
/
```

Results Explain Describe Saved SQL History

```
Procedure created.
```

3. Define a PL/SQL procedure that inserts a new record into the Toll_Collections table.

```
CREATE OR REPLACE PROCEDURE Insert_Toll_Collection(
   p_collection_id IN NUMBER,
   p_collection_date IN DATE,
   p_vehicle_id IN NUMBER,
   p_toll_id IN NUMBER
)
IS
BEGIN
   INSERT INTO Toll_Collections(collection_id, collection_date, vehicle_id, toll_id)
   VALUES(p_collection_id, p_collection_date, p_vehicle_id, p_toll_id);
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Toll collection record inserted successfully.');
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error inserting toll collection record: ' || SQLERRM);
END Insert_Toll_Collection;
/
```

```
CREATE OR REPLACE PROCEDURE Insert_Toll_Collection(
    p_collection_id IN NUMBER,
    p_collection_date IN DATE,
    p_vehicle_id IN NUMBER,
    p_toll_id IN NUMBER
)
IS
BEGIN
    INSERT INTO Toll_Collections(collection_id, collection_date, vehicle_id, toll_id)
    VALUES(p_collection_id, p_collection_date, p_vehicle_id, p_toll_id);

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Toll collection record inserted successfully.');
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error inserting toll collection record: ' || SQLERRM);
END Insert_Toll_Collection;
/
```

Results Explain Describe Saved SQL History

```
Procedure created.
```

0.00 seconds

Record

1. Define a custom record type that mirrors the structure of the table

```
DECLARE
   TYPE vehicle_record IS RECORD (
      vehicle_id Vehicles.vehicle_id%TYPE,
      user_id Vehicles.user_id%TYPE,
      vehicle_type Vehicles.vehicle_type%TYPE,
      owner_name Vehicles.owner_name%TYPE
   );
   v_vehicle vehicle_record;
BEGIN
   -- Fetch data from the Vehicles table into the record variable
   SELECT *
   INTO v_vehicle
   FROM Vehicles
   WHERE ROWNUM = 1;
   DBMS_OUTPUT.PUT_LINE('Vehicle ID: ' || v_vehicle.vehicle_id);
   DBMS_OUTPUT.PUT_LINE('User ID: ' || v_vehicle.user_id);
   DBMS_OUTPUT.PUT_LINE('Vehicle Type: ' || v_vehicle.vehicle_type);
   DBMS_OUTPUT.PUT_LINE('Owner Name: ' || v_vehicle.owner_name);
END;
/
```

```
DECLARE
    TYPE vehicle_record IS RECORD (
        vehicle_id Vehicles.vehicle_id%TYPE,
        user_id Vehicles.user_id%TYPE,
        vehicle_type Vehicles.vehicle_type%TYPE,
        owner_name Vehicles.owner_name%TYPE
    );
    v_vehicle vehicle_record;
BEGIN
    -- Fetch data from the Vehicles table into the record variable
    SELECT *
    INTO v_vehicle
    FROM Vehicles
    WHERE ROWNUM = 1;
    DBMS_OUTPUT.PUT_LINE('Vehicle ID: ' || v_vehicle.vehicle_id);
    DBMS_OUTPUT.PUT_LINE('User ID: ' || v_vehicle.user_id);
    DBMS_OUTPUT.PUT_LINE('Vehicle Type: ' || v_vehicle.vehicle_type);
    DBMS_OUTPUT.PUT_LINE('Owner Name: ' || v_vehicle.owner_name);
END;
/
```

Results Explain Describe Saved SQL History

```
Vehicle ID: 1020
User ID: 7000
Vehicle Type: Private Car
Owner Name: Amjad Khan

Statement processed.
```

2. Define a custom record that displays the toll rates

```
DECLARE
   TYPE toll_rate_record IS RECORD (
      toll_id Toll_Rates.toll_id%TYPE,
      vehicle_type Toll_Rates.vehicle_type%TYPE,
      toll_amount Toll_Rates.toll_amount%TYPE
   );
   v_toll_rate toll_rate_record;
BEGIN
   SELECT *
   INTO v_toll_rate
   FROM Toll_Rates
```

```
    WHERE ROWNUM = 1; -- Limiting to one row for demonstration purposes
    DBMS_OUTPUT.PUT_LINE('Toll ID: ' || v_toll_rate.toll_id);
    DBMS_OUTPUT.PUT_LINE('Vehicle Type: ' || v_toll_rate.vehicle_type);
    DBMS_OUTPUT.PUT_LINE('Toll Amount: ' || v_toll_rate.toll_amount);
END;
/
```

```
DECLARE
    TYPE toll_rate_record IS RECORD (
        toll_id Toll_Rates.toll_id%TYPE,
        vehicle_type Toll_Rates.vehicle_type%TYPE,
        toll_amount Toll_Rates.toll_amount%TYPE
    );
    v_toll_rate toll_rate_record;
BEGIN
    SELECT *
    INTO v_toll_rate
    FROM Toll_Rates
    WHERE ROWNUM = 1; -- Limiting to one row for demonstration purposes
    DBMS_OUTPUT.PUT_LINE('Toll ID: ' || v_toll_rate.toll_id);
    DBMS_OUTPUT.PUT_LINE('Vehicle Type: ' || v_toll_rate.vehicle_type);
    DBMS_OUTPUT.PUT_LINE('Toll Amount: ' || v_toll_rate.toll_amount);
END;
/
```

Results Explain Describe Saved SQL History

```
Toll ID: 2000
Vehicle Type: Private Car
Toll Amount: 80

Statement processed.
```

0.00 seconds

3. Using a cursor, define a custom record that mirrors the structure of the toll_collections table and then fetch data from the table using a cursor

```
DECLARE
    TYPE toll_collection_record IS RECORD (
        collection_id Toll_Collections.collection_id%TYPE,
        collection_date Toll_Collections.collection_date%TYPE,
        vehicle_id Toll_Collections.vehicle_id%TYPE,
```

```
        toll_id Toll_Collections.toll_id%TYPE
    );
    v_toll_collection toll_collection_record;
    CURSOR toll_collection_cursor IS
        SELECT *
        FROM Toll_Collections;

BEGIN
    OPEN toll_collection_cursor;
    FETCH toll_collection_cursor INTO v_toll_collection;
    WHILE toll_collection_cursor%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE('Collection ID: ' || v_toll_collection.collection_id);
        DBMS_OUTPUT.PUT_LINE('Collection Date: ' || v_toll_collection.collection_date);
        DBMS_OUTPUT.PUT_LINE('Vehicle ID: ' || v_toll_collection.vehicle_id);
        DBMS_OUTPUT.PUT_LINE('Toll ID: ' || v_toll_collection.toll_id);
        FETCH toll_collection_cursor INTO v_toll_collection;
    END LOOP;
    CLOSE toll_collection_cursor;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

```
BEGIN
    OPEN toll_collection_cursor;
    FETCH toll_collection_cursor INTO v_toll_collection;
    WHILE toll_collection_cursor%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE('Collection ID: ' || v_toll_collection.collection_id);
        DBMS_OUTPUT.PUT_LINE('Collection Date: ' || v_toll_collection.collection_date);
        DBMS_OUTPUT.PUT_LINE('Vehicle ID: ' || v_toll_collection.vehicle_id);
        DBMS_OUTPUT.PUT_LINE('Toll ID: ' || v_toll_collection.toll_id);
        FETCH toll_collection_cursor INTO v_toll_collection;
    END LOOP;
    CLOSE toll_collection_cursor;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

Results Explain Describe Saved SQL History

```
Collection ID: 3000
Collection Date: 01-FEB-24
Vehicle ID: 1020
Toll ID: 2000
Collection ID: 3001
Collection Date: 03-FEB-24
Vehicle ID: 1021
Toll ID: 2001
Collection ID: 3002
Collection Date: 07-FEB-24
Vehicle ID: 1022
Toll ID: 2002
Collection ID: 3003
Collection Date: 13-FEB-24
Vehicle ID: 1023
Toll ID: 2003
Collection ID: 3004
Collection Date: 16-FEB-24
Vehicle ID: 1024
Toll ID: 2004

Statement processed.
```

Trigger

1. Show a trigger to calculate the vehicle count.

CREATE OR REPLACE TRIGGER trg_vehicle_count

AFTER INSERT OR DELETE ON Vehicles

BEGIN

DECLARE

```
    vehicle_count NUMBER;
  BEGIN
    SELECT COUNT(*) INTO vehicle_count FROM Vehicles;
    DBMS_OUTPUT.PUT_LINE('Total number of vehicles: ' || vehicle_count);
  END;
END;
```

```
CREATE OR REPLACE TRIGGER trg_vehicle_count
AFTER INSERT OR DELETE ON Vehicles
BEGIN
    DECLARE
        vehicle_count NUMBER;
    BEGIN
        SELECT COUNT(*) INTO vehicle_count FROM Vehicles;
        DBMS_OUTPUT.PUT_LINE('Total number of vehicles: ' || vehicle_count);
    END;
END;
```

Results Explain Describe Saved SQL History

```
Trigger created.
```

0.00 seconds

2. Show a trigger to calculate average toll amount.

```
CREATE OR REPLACE TRIGGER trg_calculate_avg_toll_amount
AFTER INSERT OR UPDATE OR DELETE ON Toll_Rates
DECLARE
  avg_toll_amount NUMBER;
BEGIN
  SELECT AVG(toll_amount) INTO avg_toll_amount FROM Toll_Rates;
  DBMS_OUTPUT.PUT_LINE('Average Toll Amount: ' || avg_toll_amount);
END;
```

```
CREATE OR REPLACE TRIGGER trg_calculate_avg_toll_amount
AFTER INSERT OR UPDATE OR DELETE ON Toll_Rates
DECLARE
    avg_toll_amount NUMBER;
BEGIN
    SELECT AVG(toll_amount) INTO avg_toll_amount FROM Toll_Rates;
    DBMS_OUTPUT.PUT_LINE('Average Toll Amount: ' || avg_toll_amount);
END;
```

Results Explain Describe Saved SQL History

```
Trigger created.
```

0.00 seconds

3. Write a trigger code which would show whether the vehicle Id exists in the vehicle table before inserting the toll.

```
CREATE OR REPLACE TRIGGER check_vehicle_id_exists
BEFORE INSERT ON Toll_Collections
FOR EACH ROW
DECLARE
   v_count NUMBER;
BEGIN
   SELECT COUNT(*)
   INTO v_count
   FROM Vehicles
   WHERE vehicle_id = :NEW.vehicle_id;

   IF v_count = 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Vehicle ID does not exist in the Vehicles table');
   END IF;
END;
/
```

Autocommit Display 10

```
CREATE OR REPLACE TRIGGER check_vehicle_id_exists
BEFORE INSERT ON Toll_Collections
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM Vehicles
    WHERE vehicle_id = :NEW.vehicle_id;

    IF v_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Vehicle ID does not exist in the Vehicles table');
    END IF;
END;
/
```

Results Explain Describe Saved SQL History

```
Trigger created.
```

0.02 seconds

Package

1. Show a package to retrieve vehicle details.

```
CREATE OR REPLACE PACKAGE Vehicle_Details AS
    FUNCTION get_vehicle_owner(vehicle_id IN NUMBER) RETURN VARCHAR2;
  END Vehicle_Details;
  CREATE OR REPLACE PACKAGE BODY Vehicle_Details AS
    FUNCTION get_vehicle_owner(vehicle_id IN NUMBER) RETURN VARCHAR2 IS
    v_owner_name VARCHAR2(100);
  BEGIN
    SELECT owner_name INTO v_owner_name
    FROM Vehicles WHERE vehicle_id = vehicle_id;
    RETURN v_owner_name;
  END get_vehicle_owner;
END Vehicle_Details;
```

Autocommit Display 10

```
CREATE OR REPLACE PACKAGE Vehicle_Details AS
        FUNCTION get_vehicle_owner(vehicle_id IN NUMBER) RETURN VARCHAR2;
END Vehicle_Details;
CREATE OR REPLACE PACKAGE BODY Vehicle_Details AS
        FUNCTION get_vehicle_owner(vehicle_id IN NUMBER) RETURN VARCHAR2 IS
        v_owner_name VARCHAR2(100);
      BEGIN
        SELECT owner_name INTO v_owner_name
        FROM Vehicles WHERE vehicle_id = vehicle_id;
        RETURN v_owner_name;
      END get_vehicle_owner;
END Vehicle_Details;
```

Results Explain Describe Saved SQL History

```
Package Body created.
```

0.00 seconds

2. Use package to find toll amount by vehicle type.

```
CREATE OR REPLACE PACKAGE Toll_Rate_Pkg AS
   FUNCTION get_toll_amount(vehicle_type IN VARCHAR2) RETURN DECIMAL;
END Toll_Rate_Pkg;
/
CREATE OR REPLACE PACKAGE BODY Toll_Rate_Pkg AS
   FUNCTION get_toll_amount(vehicle_type IN VARCHAR2) RETURN DECIMAL IS
      toll_amt DECIMAL(10, 2);
   BEGIN
      SELECT toll_amount INTO toll_amt
      FROM Toll_Rates
      WHERE UPPER(vehicle_type) = UPPER(get_toll_amount.vehicle_type);
      RETURN toll_amt;
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
```

RETURN NULL;

END get_toll_amount;

END Toll_Rate_Pkg;

```
CREATE OR REPLACE PACKAGE Toll_Rate_Pkg AS
    FUNCTION get_toll_amount(vehicle_type IN VARCHAR2) RETURN DECIMAL;
END Toll_Rate_Pkg;
/
CREATE OR REPLACE PACKAGE BODY Toll_Rate_Pkg AS
    FUNCTION get_toll_amount(vehicle_type IN VARCHAR2) RETURN DECIMAL IS
        toll_amt DECIMAL(10, 2);
    BEGIN
        SELECT toll_amount INTO toll_amt
        FROM Toll_Rates
        WHERE UPPER(vehicle_type) = UPPER(get_toll_amount.vehicle_type);
        RETURN toll_amt;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN NULL;
    END get_toll_amount;
END Toll_Rate_Pkg;
```

Results Explain Describe Saved SQL History

```
Package Body created.
```

0.01 seconds

3. Use package to insert toll collection.

CREATE OR REPLACE PACKAGE Toll_Collection_Pkg AS

PROCEDURE insert_toll_collection(collection_date IN DATE, vehicle_id IN NUMBER, toll_id IN NUMBER);

END Toll_Collection_Pkg;

/

CREATE OR REPLACE PACKAGE BODY Toll_Collection_Pkg AS

PROCEDURE insert_toll_collection(collection_date IN DATE, vehicle_id IN NUMBER, toll_id IN NUMBER) IS

BEGIN

INSERT INTO Toll_Collections (collection_id, collection_date, vehicle_id, toll_id)

```
    VALUES (collection_id_seq.nextval, collection_date, vehicle_id, toll_id);
  END insert_toll_collection;
END Toll_Collection_Pkg;
```

Autocommit Display 10

```
CREATE OR REPLACE PACKAGE Toll_Collection_Pkg AS
    PROCEDURE insert_toll_collection(collection_date IN DATE, vehicle_id IN NUMBER, toll_id IN NUMBER);
END Toll_Collection_Pkg;
/
CREATE OR REPLACE PACKAGE BODY Toll_Collection_Pkg AS
    PROCEDURE insert_toll_collection(collection_date IN DATE, vehicle_id IN NUMBER, toll_id IN NUMBER) IS
    BEGIN
        INSERT INTO Toll_Collections (collection_id, collection_date, vehicle_id, toll_id)
        VALUES (collection_id_seq.nextval, collection_date, vehicle_id, toll_id);

    END insert_toll_collection;
END Toll_Collection_Pkg;
```

Results Explain Describe Saved SQL History

```
Package Body created.
```

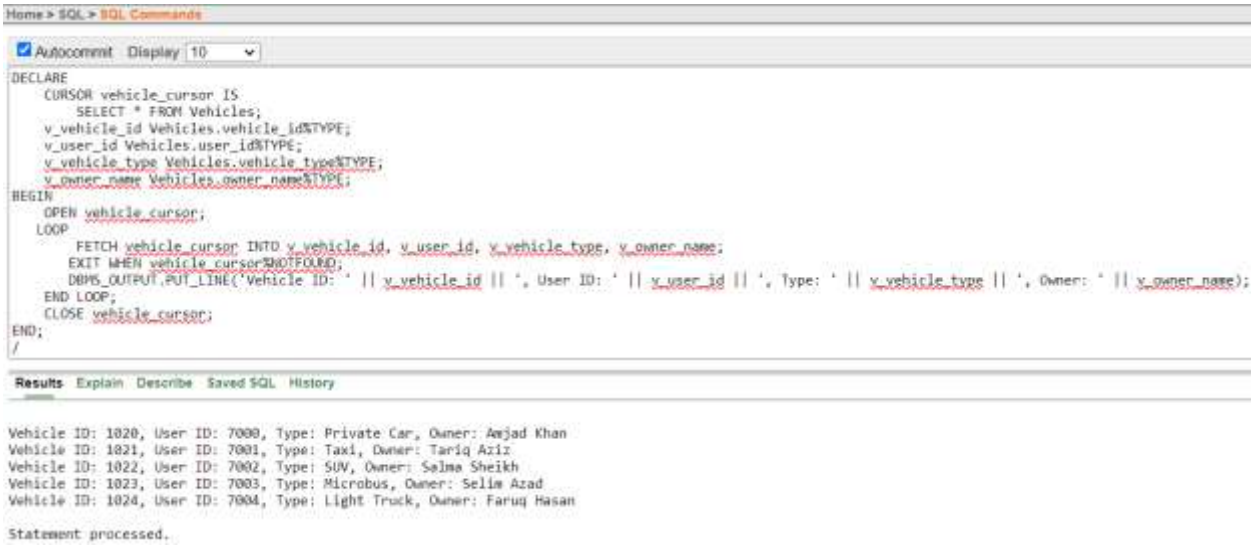
0.01 seconds

Cursor

1. Show a cursor from vehicles table that selects vehicle_id, vehicle_type, and owner_name from the Vehicles table and then prints out each row's information.

```
DECLARE
  CURSOR vehicle_cursor IS
    SELECT * FROM Vehicles;
  v_vehicle_id Vehicles.vehicle_id%TYPE;
  v_user_id Vehicles.user_id%TYPE;
  v_vehicle_type Vehicles.vehicle_type%TYPE;
  v_owner_name Vehicles.owner_name%TYPE;
BEGIN
  OPEN vehicle_cursor;
 LOOP
    FETCH vehicle_cursor INTO v_vehicle_id, v_user_id, v_vehicle_type, v_owner_name;
   EXIT WHEN vehicle_cursor%NOTFOUND;
```

```
    DBMS_OUTPUT.PUT_LINE('Vehicle ID: ' || v_vehicle_id || ', User ID: ' || v_user_id || ', Type: ' || v_vehicle_type || ', Owner: ' || v_owner_name);
  END LOOP;
  CLOSE vehicle_cursor;
END;
/
```



2. Create a PL/SQL block with an explicit cursor for the Toll_Rates table

```
DECLARE
  v_toll_id Toll_Rates.toll_id%TYPE;
  v_vehicle_type Toll_Rates.vehicle_type%TYPE;
  v_toll_amount Toll_Rates.toll_amount%TYPE;
  CURSOR toll_rates_cursor IS
    SELECT * FROM Toll_Rates;
BEGIN
  OPEN toll_rates_cursor;
 LOOP
    FETCH toll_rates_cursor INTO v_toll_id, v_vehicle_type, v_toll_amount;
    EXIT WHEN toll_rates_cursor%NOTFOUND;
```

```
    DBMS_OUTPUT.PUT_LINE('Toll ID: ' || v_toll_id || ', Vehicle Type: ' || v_vehicle_type || ', Toll Amount: ' || v_toll_amount);
  END LOOP;
  CLOSE toll_rates_cursor;
END;
/
```

Home > SQL > SQL Commands

Autocommit Display 10

```
DECLARE
    v_toll_id Toll_Rates.toll_id%TYPE;
    v_vehicle_type Toll_Rates.vehicle_type%TYPE;
    v_toll_amount Toll_Rates.toll_amount%TYPE;
    CURSOR toll_rates_cursor IS
        SELECT * FROM Toll_Rates;
BEGIN
    OPEN toll_rates_cursor;
   LOOP
        FETCH toll_rates_cursor INTO v_toll_id, v_vehicle_type, v_toll_amount;
        EXIT WHEN toll_rates_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Toll ID: ' || v_toll_id || ', Vehicle Type: ' || v_vehicle_type || ', Toll Amount: ' || v_toll_amount);
    END LOOP;
    CLOSE toll_rates_cursor;
END;
/
```

Results Explain Describe Saved SQL History

```
Toll ID: 2000, Vehicle Type: Private Car, Toll Amount: 80
Toll ID: 2001, Vehicle Type: Taxi, Toll Amount: 120
Toll ID: 2002, Vehicle Type: SUV, Toll Amount: 100
Toll ID: 2003, Vehicle Type: Microbus, Toll Amount: 150
Toll ID: 2004, Vehicle Type: Light Truck, Toll Amount: 180

Statement processed.

0.00 seconds
```

3. Create a PL/SQL block using an implicit cursor for the Toll_Collections table

```
DECLARE
  v_collection_id Toll_Collections.collection_id%TYPE;
  v_collection_date Toll_Collections.collection_date%TYPE;
  v_vehicle_id Toll_Collections.vehicle_id%TYPE;
  v_toll_id Toll_Collections.toll_id%TYPE;
BEGIN
  FOR toll_collection IN (SELECT * FROM Toll_Collections) LOOP
    v_collection_id := toll_collection.collection_id;
    v_collection_date := toll_collection.collection_date;
```

```
        v_vehicle_id := toll_collection.vehicle_id;

        v_toll_id := toll_collection.toll_id;

        DBMS_OUTPUT.PUT_LINE('Collection ID: ' || v_collection_id || ', Collection Date: ' || v_collection_date || ', Vehicle ID: ' || v_vehicle_id || ', Toll ID: ' || v_toll_id);

    END LOOP;

END;

/
```

13. Relational Algebra

1. Select all toll rates with amount greater than 100 Taka (Selection)

Ans: $\sigma_{toll\_amount>100}$(Toll_Rates)

2. Find only the vehicle type and owner name from the Vehicles table (Projection)

Ans: $\prod_{(vehicle\_type)(Owner\_Name)}$(Vehicles)

3. Find all the vehicle types from Toll_Rates and Vehicles (Union)

Ans: $\prod_{(vehicle\_type)(Toll\_Rates)} \cup \prod_{(vehicle\_type)(vehicles)}$

4. Find all the vehicle_id that are in Toll_Collections but not in Vehicles (set difference)

Ans: $\pi_{(vehicle\_id)}(Toll\_Collections) - \pi_{(vehicle\_id)}(Vehicles)$

5. rename the attribute username in the Users table to user_name (Rename)

Ans: $\rho\{Users_{(user\_id,user\_name,password)}\}(Users)$

14. Conclusion

The database schema includes tables for vehicles, toll rates, toll collections, expressway traffic, investors, expressway sections, users, and administrators. Various relationships between tables are established using primary and foreign keys to maintain data integrity. Views and synonyms are created to simplify data access and improve query readability. Relational algebra queries are used for data manipulation and retrieval operations. Roles and privileges are granted to users and administrators to control access to database objects. Single-row and group functions are used for data analysis and aggregation. However, there are still scopes of improvement through some future works.

Proposed Future work:

1. Enhance Security: Through implementing more robust security measures such as encryption for sensitive data and restricting access based on user roles, the database can have added layers of security.

2. User Interface Enhancement: By developing a user-friendly interface with advanced features such as data visualization, real-time monitoring, and interactive reporting for better usability.

By addressing these areas of improvement, the project can evolve into a more robust and scalable database management system that meets the evolving needs of users and administrators while ensuring data security, performance, and reliability.