Name : İrem

Surname : ÖZTÜRK

Student Number : 22092090064

Department : Computer Engineering

Email Address : iremozturk36@gmail.com

Project Name : Smart Public Transport and Transportation Assistant

1 -) PROJECT INTRODUCTION

Problem Statement & Proposed Solution :

I encountered the following problems while using public transportation in my daily life, and I developed the Smart Public Transport and Transportation Assistant System to solve these problems:

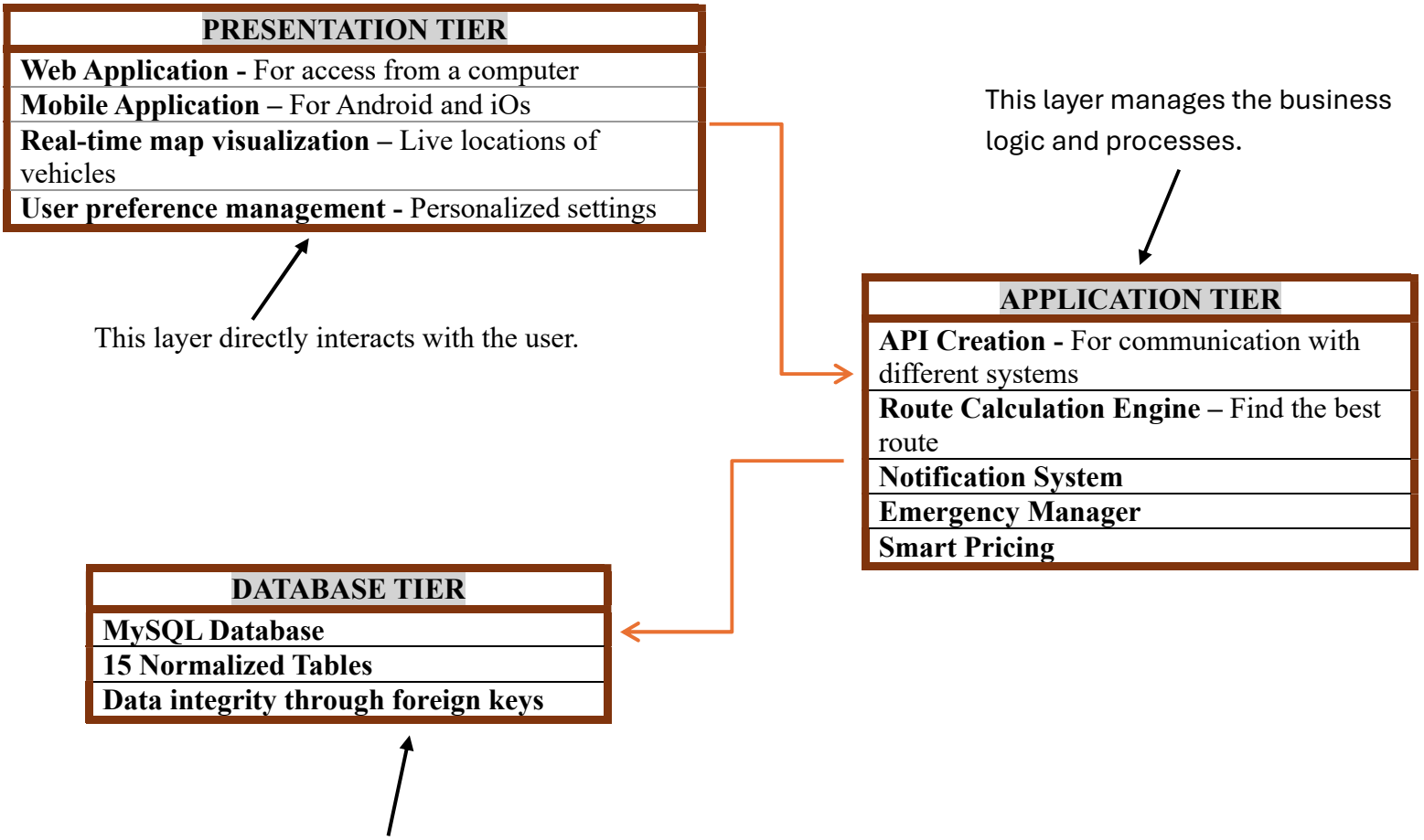
Problems	The Solution I Developed	Parts related to the SQL table
I don't know where the vehicle is	Real-time vehicle tracking	• Vehicles.real_time_location
I have to ride in crowded vehicles	Density information	• Vehicles.current_capacity • Vehicles.capacity • Trips.crowd_level
I can't find the best route	Personalized route suggestion	• Trips + Routes + JOIN queries
I can't find a vehicle with disabled access	Disabled access filter	• Vehicles.is_accessible • Stations.is_accessible
I don't know when I will arrive	Estimated arrival time	• Routes.estimated_duration • Schedule.arrival_time
I am unaware of delays	Emergency notifications	• Notifications table • Emergency table
Discounts are confusing	Automatic price calculation	• Pricing table • User_Types.discount_rate
Night travel is unsafe	Safe route at night	• schedule.day_type • Time_Slots.slot_name
I can't track my travel history	History and analysis	• Trips table • Crowd_Analytics table

- **Target Users :** Students , tourists , busy professionals , special needs users: (disabled, elderly, pregnant women) , municipal Employees , comfort-oriented users

2 -) SYSTEM ARCHITECTURE

Three-Tier Architecture Design :

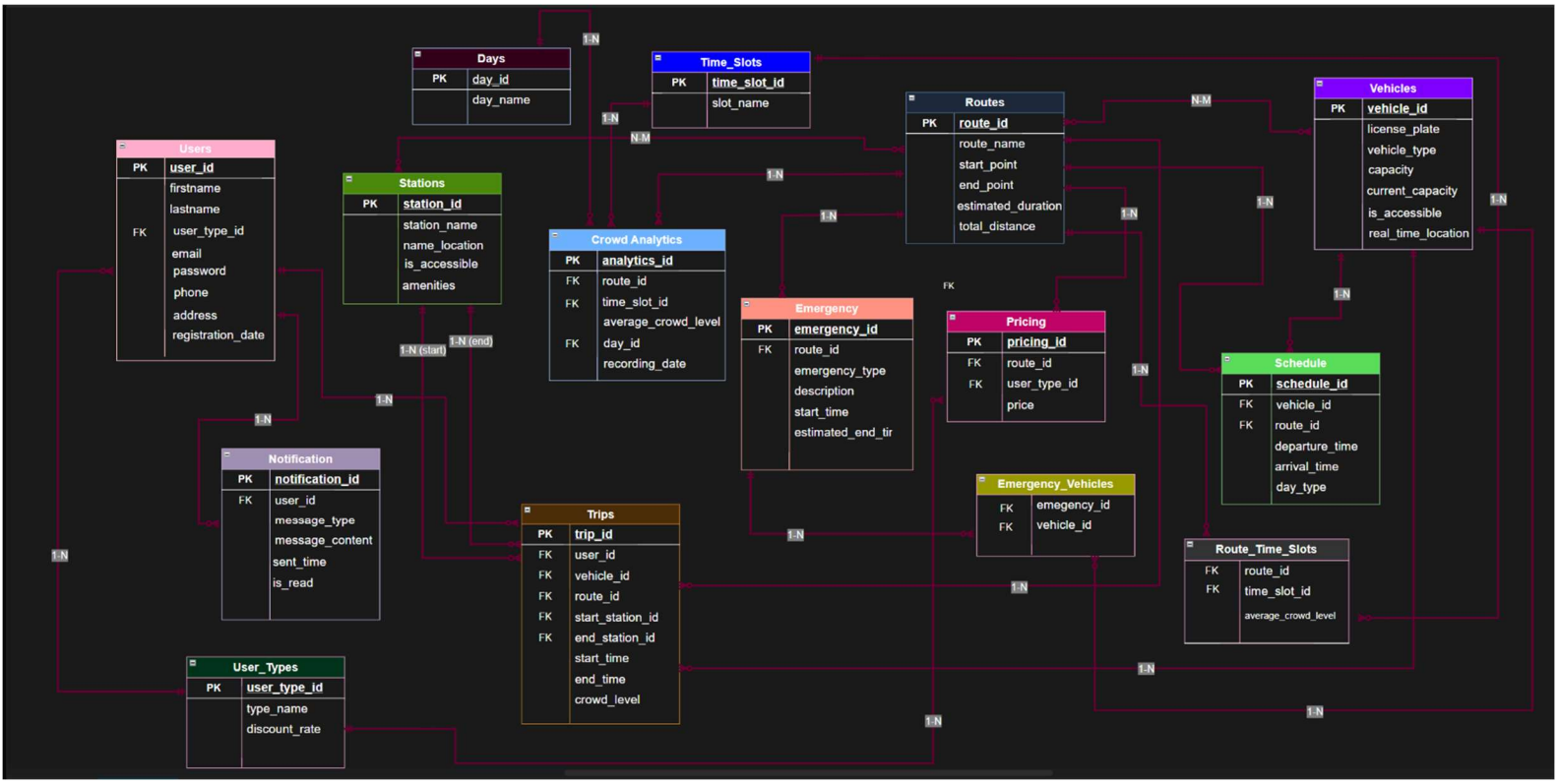
The system is designed using a 3-tier architecture for scalability, maintainability, and separation of concerns :

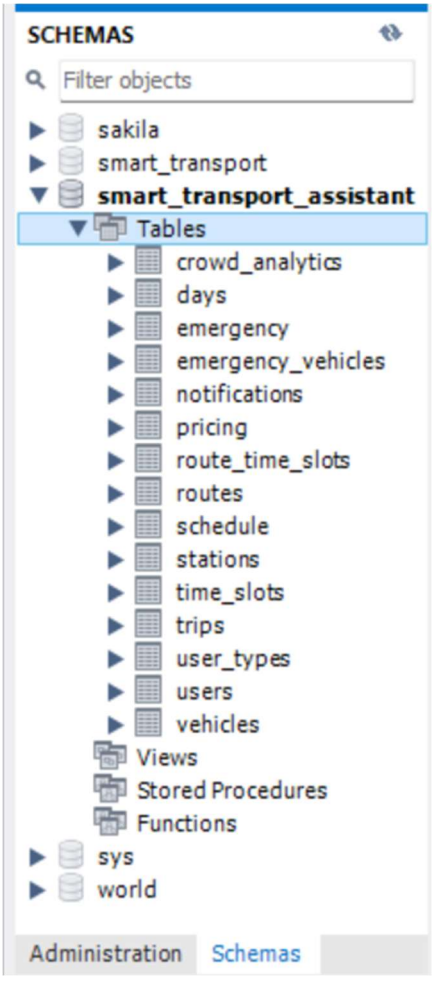


Implemented in this Project : For this database course project, only the Database Tier has been implemented. The Presentation and Application tiers are described conceptually to demonstrate how the complete system would function in real-world deployment.

3 -) DATABASE DESIGN

Entity-Relationship Diagram :





The database consists of 15 interrelated tables :

→**Core Entities**

Users: Registered system users with personal information
Vehicles: Public transport vehicles with real-time status
Routes: Transportation paths with distance and duration
Stations: Public transport stops with amenities

→**Transaction Entities**

Trips: Records of user journeys with analytics data
Schedule: Vehicle timetables and departure information
Pricing: Fare structure based on routes and user types

→**System Management Entities**

Notifications: User alerts and system messages
Emergency: Incident reports and disruption management
Crowd_Analytics: Historical passenger density data

→**Reference Tables**

User_Types: User categories with discount rates
Days: Day names for scheduling
Time_Slots: Time periods for analytics

→**Junction Tables**

Emergency_Vehicles: Many-to-many relationship between emergencies and vehicles
Route_Time_Slots: Many-to-many relationship for crowd analytics

4 -) NORMALIZATION PROCESS

First Normal Form (1NF)

- Eliminated multi-valued attributes
- Created Emergency_Vehicles junction table for affected vehicles
- Separated time_slot and day_of_week into lookup tables

Second Normal Form (2NF)

- Removed partial dependencies
- Created User_Types table for user categorization
- Pricing table now fully dependent on composite key (route_id, user_type_id)

Third Normal Form (3NF)

- Eliminated transitive dependencies
- Moved estimated_duration from Trips to Routes table
- Created separate Route_Time_Slots for crowd level data

Normal Form	Table Name	Problem	Correction
1NF	Emergency	Multi-valued attribute in **affected_vehicles**	Created **Emergency_Vehicles** junction table
1NF	Crowd_Analytics	**time_slot** and **day_of_week** as text without referential integrity	Created **Time_Slots** and **Days** lookup tables
2NF	Pricing	**price** partially dependent only on **route_id**	Added **User_Types** table, **user_type** as FK
3NF	Trips	**estimated_duration** transitively dependent via **route_id**	Moved column to **Routes** table
3NF	Crowd_Analytics	**average_crowd_level** with transitive dependency	Created **Route_Time_Slots** junction table

→ First, I ensured a single value in each cell using 1NF.
→Then, I removed partial dependencies using 2NF.
→Finally, I cleaned transitive dependencies using 3NF.

I applied all the fixes in Assignment 5.

Now my database is:

- Non-repetitive
- Consistent
- Efficient
- Resilient

5 -) SQL IMPLEMENTATION

Database Creation :

```
1
2 •  CREATE DATABASE IF NOT EXISTS smart_transport_assistant;
3 •  USE smart_transport_assistant;
```

Table Creation Examples :

```
CREATE TABLE IF NOT EXISTS Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    firstname VARCHAR(50) NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    user_type_id INT,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    phone VARCHAR(20),
    address TEXT,
    registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_type_id) REFERENCES User_Types(user_type_id)
);
```

```
CREATE TABLE IF NOT EXISTS Pricing (
    pricing_id INT PRIMARY KEY AUTO_INCREMENT,
    route_id INT NOT NULL,
    user_type_id INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (route_id) REFERENCES Routes(route_id),
    FOREIGN KEY (user_type_id) REFERENCES User_Types(user_type_id)
);
```

Data Population :

The database was populated with realistic sample data including:

- 5 user types (Student, Elderly, Disabled, Regular, Tourist)
- 7 days of the week
- 4 time slots for analytics
- 5 stations with accessibility information
- 5 vehicles with real-time locations
- 5 transportation routes
- 5 users with different profiles
- Multiple pricing records with discounts
- Sample trips, notifications, and emergencies

user_type_id	type_name	discount_rate
1	Student	0.50
2	Elderly	0.00
3	Disabled	0.00
4	Regular	1.00
5	Tourist	0.80

day_id	day_name
5	Friday
1	Monday
6	Saturday
7	Sunday
4	Thursday
2	Tuesday
3	Wednesday

time_slot_id	slot_name
3	Evening Peak (16:00-20:00)
2	Midday (09:00-16:00)
1	Morning Peak (06:00-09:00)
4	Night (20:00-06:00)

route_id	route_name	start_point	end_point	estimated_duration	total_distance
1	Blue Line	Central Station	University Campus	45	12.50
2	Red Line	City Hospital	Shopping Mall	30	8.20
3	Green Line	Old Town Square	Central Station	20	5.70
4	Express Route	Central Station	Shopping Mall	15	6.30
5	Night Line	University Campus	City Hospital	35	10.10

SQL Operations Demonstrated :

→Data Manipulation Language (DML)

DML commands are the SQL commands I use to manage data in a database. That is, they are the tools I use to add new records to tables, update existing data, and delete records.

```
INSERT INTO Users (firstname, lastname, user_type_id, email, password, phone, address) VALUES
('Ahmet', 'Yılmaz', 1, 'ahmet@gmail.com', 'SecurePass123!', '5551112233', '123 Main St, Istanbul'),
('Ayşe', 'Kaya', 2, 'ayse@hotmail.com', 'AysePass456!', '5552223344', '456 Park Ave, Istanbul'),
('Mehmet', 'Demir', 3, 'mehmet@outlook.com', 'Mehmet789!', '5553334455', '789 Oak Rd, Istanbul'),
('Zeynep', 'Çelik', 4, 'zeynep@gmail.com', 'ZeynepPass012!', '5554445566', '101 Pine St, Istanbul'),
('John', 'Smith', 5, 'john@gmail.com', 'TouristPass345!', '5555556677', 'Tourist Hotel, Istanbul');
```

```
UPDATE Users
SET phone = '5559998877'
WHERE user_id = 1;
```

```
DELETE FROM Notifications WHERE notification_id = 5;
```

→ Data Query Language (DQL)

DQL refers to the SQL commands I use to read, view, and analyze data in a database. In other words, they are the tools I use to query, filter, sort, and combine existing information in tables.

```
SELECT * FROM Users
WHERE user_type_id IN (
    SELECT user_type_id FROM User_Types WHERE discount_rate < 1.00
);
```

```
SELECT * FROM Stations ORDER BY station_name;
```

```
SELECT route_id, AVG(crowd_level) as avg_crowd
FROM Trips
GROUP BY route_id
HAVING AVG(crowd_level) > 50.00;
```

→ Comparison and Logical Operators

These operators are the tools I use in my SQL queries to filter data, create conditions, and perform logical comparisons.

```
-- Comparison operators (=, <>, >, <, >=, <=)
SELECT * FROM Vehicles WHERE vehicle_type = 'bus';
SELECT * FROM Pricing WHERE price > 10.00;
SELECT * FROM Trips WHERE crowd_level <= 60.00;

-- Logical operators (AND, OR, NOT)
SELECT * FROM Vehicles WHERE is_accessible = TRUE AND vehicle_type <> 'metro';
SELECT * FROM Users WHERE user_type_id = 1 OR user_type_id = 2;
SELECT * FROM Notifications WHERE NOT is_read;
```

6 -) BUSINESS RULES IMPLEMENTATION

User Management Rules

- Passwords stored securely (VARCHAR(255) for hashing)
- Email uniqueness enforced through UNIQUE constraint
- Registration date automatically recorded via DEFAULT CURRENT_TIMESTAMP

Vehicle and Route Rules

- Real-time location tracking using MySQL POINT data type
- Crowd level calculation: (current_capacity / capacity) * 100
- Accessibility filtering through is_accessible boolean flag

Pricing Rules

- Dynamic pricing based on User_Types.discount_rate
- Automatic discount application through JOIN operations
- Different fares for different routes and user categories

Emergency Management

- Immediate notification to affected users
- Junction table for multiple affected vehicles
- Estimated resolution time tracking

➔ In short, the business rules I implemented in SQL are:

- **Data Integrity Rules:** UNIQUE, FOREIGN KEY, NOT NULL constraints
- **Automatic Data:** DEFAULT values, AUTO_INCREMENT
- **Calculation Rules:** Formulas in SELECT statements
- **Filtering Rules:** WHERE conditions for business logic
- **Relationship Rules:** Junction tables for N-M relationships
- **Validation Rules:** Referential integrity through foreign keys

7 -) MY SQL CODES

```sql
CREATE DATABASE IF NOT EXISTS smart_transport_assistant;
USE smart_transport_assistant;


CREATE TABLE IF NOT EXISTS User_Types (
    user_type_id INT PRIMARY KEY AUTO_INCREMENT,
    type_name VARCHAR(50) NOT NULL UNIQUE,
    discount_rate DECIMAL(5,2) DEFAULT 0.00
);


CREATE TABLE IF NOT EXISTS Days (
    day_id INT PRIMARY KEY AUTO_INCREMENT,
    day_name VARCHAR(20) NOT NULL UNIQUE
);


CREATE TABLE IF NOT EXISTS Time_Slots (
    time_slot_id INT PRIMARY KEY AUTO_INCREMENT,
    slot_name VARCHAR(50) NOT NULL UNIQUE
);


CREATE TABLE IF NOT EXISTS Stations (
    station_id INT PRIMARY KEY AUTO_INCREMENT,
    station_name VARCHAR(100) NOT NULL,
    name_location VARCHAR(255),
    is_accessible BOOLEAN DEFAULT FALSE,
    amenities TEXT
);


CREATE TABLE IF NOT EXISTS Vehicles (
    vehicle_id INT PRIMARY KEY AUTO_INCREMENT,
    license_plate VARCHAR(20) UNIQUE NOT NULL,
    vehicle_type ENUM('bus', 'metro', 'tram') NOT NULL,
    capacity INT NOT NULL,
    current_capacity INT DEFAULT 0,
    is_accessible BOOLEAN DEFAULT FALSE,
    real_time_location POINT
);


CREATE TABLE IF NOT EXISTS Routes (
    route_id INT PRIMARY KEY AUTO_INCREMENT,
    route_name VARCHAR(100) NOT NULL,
    start_point VARCHAR(255),
    end_point VARCHAR(255),
    estimated_duration INT,
    total_distance DECIMAL(10,2)
);


CREATE TABLE IF NOT EXISTS Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    firstname VARCHAR(50) NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    user_type_id INT,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    phone VARCHAR(20),
    address TEXT,
    registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_type_id) REFERENCES User_Types(user_type_id)
);


CREATE TABLE IF NOT EXISTS Pricing (
    pricing_id INT PRIMARY KEY AUTO_INCREMENT,
    route_id INT NOT NULL,
    user_type_id INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (route_id) REFERENCES Routes(route_id),
    FOREIGN KEY (user_type_id) REFERENCES User_Types(user_type_id)
);


CREATE TABLE IF NOT EXISTS Schedule (
    schedule_id INT PRIMARY KEY AUTO_INCREMENT,
    vehicle_id INT NOT NULL,
    route_id INT NOT NULL,
    departure_time TIME NOT NULL,
    arrival_time TIME NOT NULL,
    day_type ENUM('weekday', 'weekend', 'holiday'),
    FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id),
    FOREIGN KEY (route_id) REFERENCES Routes(route_id)
);


CREATE TABLE IF NOT EXISTS Notifications (
    notification_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    message_type VARCHAR(50) NOT NULL,
    message_content TEXT NOT NULL,
    sent_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    is_read BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);


CREATE TABLE IF NOT EXISTS Emergency (
    emergency_id INT PRIMARY KEY AUTO_INCREMENT,
    route_id INT NOT NULL,
    emergency_type VARCHAR(50) NOT NULL,
    description TEXT,
    start_time DATETIME NOT NULL,
    estimated_end_time DATETIME,
    FOREIGN KEY (route_id) REFERENCES Routes(route_id)
);


CREATE TABLE IF NOT EXISTS Trips (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    vehicle_id INT NOT NULL,
    route_id INT NOT NULL,
    start_station_id INT NOT NULL,
    end_station_id INT NOT NULL,
    start_time DATETIME NOT NULL,
    end_time DATETIME,
    crowd_level DECIMAL(5,2),
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id),
    FOREIGN KEY (route_id) REFERENCES Routes(route_id),
    FOREIGN KEY (start_station_id) REFERENCES Stations(station_id),
    FOREIGN KEY (end_station_id) REFERENCES Stations(station_id)
);


CREATE TABLE IF NOT EXISTS Crowd_Analytics (
    analytics_id INT PRIMARY KEY AUTO_INCREMENT,
    route_id INT NOT NULL,
    time_slot_id INT NOT NULL,
    day_id INT NOT NULL,
    average_crowd_level DECIMAL(5,2),
    recording_date DATE NOT NULL,
    FOREIGN KEY (route_id) REFERENCES Routes(route_id),
    FOREIGN KEY (time_slot_id) REFERENCES Time_Slots(time_slot_id),
    FOREIGN KEY (day_id) REFERENCES Days(day_id)
);


CREATE TABLE IF NOT EXISTS Emergency_Vehicles (
    emergency_id INT NOT NULL,
    vehicle_id INT NOT NULL,
    PRIMARY KEY (emergency_id, vehicle_id),
    FOREIGN KEY (emergency_id) REFERENCES Emergency(emergency_id),
    FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id)
);


CREATE TABLE IF NOT EXISTS Route_Time_Slots (
    route_id INT NOT NULL,
    time_slot_id INT NOT NULL,
    average_crowd_level DECIMAL(5,2),
    PRIMARY KEY (route_id, time_slot_id),
    FOREIGN KEY (route_id) REFERENCES Routes(route_id),
    FOREIGN KEY (time_slot_id) REFERENCES Time_Slots(time_slot_id)
);


SHOW TABLES;
```

8 -) PRACTICAL SCENARIOS

Tourist Scenario :

A tourist arrives in Istanbul and uses the system to:

- Register with tourist privileges (20% discount)
- Find accessible vehicles with available capacity
- Plan route from Central Station to Shopping Mall
- Receive English notifications about delays
- Get emergency alerts about traffic conditions

```
USE smart_transport_assistant;
```

David enter the system as a tourist.

```
INSERT INTO Users (firstname, lastname, user_type_id, email, password, phone, address, registration_date)
VALUES ('David', 'Miller', 5, 'david.miller@mail.com', 'TouristPass1!', '+123456789', 'Hotel Istanbul', NOW());
```

```
SELECT vehicle_id, license_plate, vehicle_type, capacity, current_capacity, is_accessible
FROM Vehicles
WHERE current_capacity < capacity * 0.7;
```

He find available and suitable vehicles.

```
INSERT INTO Trips (user_id, vehicle_id, route_id, start_station_id, end_station_id, start_time, end_time, crowd_level)
VALUES (7, 4, 4, 1, 5, NOW(), DATE_ADD(NOW(), INTERVAL 15 MINUTE), 45.00);
```

He is launching a service on the Express Route.

```
UPDATE Vehicles
SET current_capacity = current_capacity + 1
WHERE vehicle_id = 4;
```

David gets into the vehicle and the capacity is updated.

```
INSERT INTO Notifications (user_id, message_type, message_content, sent_time, is_read)
VALUES (7, 'info', 'Welcome to Istanbul Public Transport', NOW(), FALSE);
```

English notification for tourists

```
SELECT p.price, r.route_name, ut.type_name, ut.discount_rate
FROM Pricing p
JOIN Routes r ON p.route_id = r.route_id
JOIN User_Types ut ON p.user_type_id = ut.user_type_id
WHERE p.route_id = 4 AND p.user_type_id = 5;
```

Tourist discount is being calculated.

```
SELECT station_id, station_name, is_accessible, amenities
FROM Stations
WHERE station_id = 1 OR station_id = 4 OR station_id = 5;
```

Places to visit

```
SELECT trip_id, crowd_level, start_time
FROM Trips
ORDER BY crowd_level ASC;
```

He displays the emptiest vehicles.

```
SELECT trip_id, crowd_level
FROM Trips
WHERE crowd_level < (SELECT AVG(crowd_level) FROM Trips);
```

The average of all trips was calculated. Then, those below the average were filtered out.

```
UPDATE Users
SET phone = '+123456780'
WHERE user_id = 7;
```

David realized he had entered the wrong number. He updated his phone.

```
SELECT vehicle_id, license_plate, vehicle_type, current_capacity
FROM Vehicles
WHERE vehicle_type <> 'metro'
AND current_capacity < 20;
```

Vehicles other than the metro have been identified.

```
INSERT INTO Emergency (route_id, emergency_type, description, start_time, estimated_end_time)
VALUES (4, 'traffic', 'Heavy tourist traffic', NOW(), DATE_ADD(NOW(), INTERVAL 30 MINUTE));
```

Emergency notification

```
INSERT INTO Crowd_Analytics (route_id, time_slot_id, day_id, average_crowd_level, recording_date)
VALUES (4, 2, 6, 55.00, CURDATE());
```

Tourist density analytical record

```
DELETE FROM Notifications
WHERE user_id = 7 AND message_type = 'info';
```

David's test notification was deleted.

9 -) CONCLUSION

The Smart Public Transport and Transportation Assistant System represents a comprehensive solution to modern urban transportation challenges. Through careful database design, normalization, and SQL implementation, this project demonstrates:

- Practical Problem Solving : Addressing real-world transportation issues
- Technical Proficiency : Mastery of database design principles
- Scalable Architecture : Foundation for future expansion
- User-Centered Design : Focus on diverse user needs
- Data Integrity : Robust referential integrity and normalization

This system not only serves as an academic exercise but provides a blueprint for actual municipal transportation systems that could improve the daily lives of millions of commuters while optimizing city transport networks and reducing environmental impact.

10 -) REFERENCES

- Draw.io (Diagrams.net) - ER Diagram Creation Tool
- MySQL Workbench 8.0 - Database Design and Development Tool
- W3Schools SQL Tutorial - Basic to Advanced SQL Concepts
- Connolly, T., & Begg, C. (2015). Database systems: A practical approach to design, implementation, and management (6th ed.). Pearson.
- Database Normalization Explained - https://www.essentialsql.com/database-normalization/