Final Project: Transportation Management System

Name : Rahul Hiteshkumar Prajapati

Master of Professional Studies in Analytics

ITC 6000 - Database Management Systems

Professor : Karen Johnson

27th March, 2024

Introduction

Transportation plays a pivotal role across various industries, facilitating the efficient movement of goods and people. A transportation management system(TMS) acts as a vital logistics platform, leveraging modern technologies to execute business plans and streamline the physical transportation of goods.

This comprehensive transportation management solution addresses several critical challenges encountered by companies in managing their transportation operations. Firstly, it caters to the intricate logistics landscape by offering streamlined processes and automated tools for route optimization, enabling businesses to manage multiple variables efficiently. Secondly, it meets customer expectations for prompt deliveries, transparent communication, and exceptional service which are imperative for sustaining business success. Lastly, by centralizing and organizing data related to vehicles, customer deliveries, routes, schedules, and payment records, the TMS ensures data accuracy, integrity and security to logistics companies.

Designed with a focus on end-users such as administrative staff, dispatchers, drivers, and customer service representatives, the TMS offers functionalities like route optimization, vehicle tracking, schedule management, and customer interaction tools. As for the app's cost model, it can vary between free and paid versions. In the free model, the app is accessible at no cost to users, potentially attracting a wider user base and generating revenue through alternative means such as advertisements, sponsorships, or partnership. Conversely, the paid version entails a one-time purchase fee for accessing premium features not available in the free version, offering users enhanced functionalities for a fee.

Personally motivated by the challenges faced in managing my family's transportation business, the developer aims to address these issues by creating an efficient and cost-effective transportation management app.

Business Analysis

There are several end-users or personas who will get the benefits by using this TMS, how this TMS will they use are also defined below with a list of end users.

1. **Administrative Staffs:** These individuals are responsible for managing the overall settings and configurations of the TMS. They have access to the vehicle's records, routes, schedules, drivers, and user permissions within the system. Administrative staff use the TMS to set up and configure parameters such as vehicle capacities, driver assignments, and operating hours.
2. **Dispatchers:** Dispatchers play a crucial role in coordinating transportation activities and ensuring the timely delivery of goods. They use the TMS to assign routes, schedules, and tasks to drivers based on factors such as delivery locations, vehicle capacities and time constraints. They communicate directly with drivers through the TMS to provide instructions, address issues, and ensure smooth transportation operations.
3. **Drivers:** They are the main or primary users of this TMS as they rely on the application to access their assigned tasks, routes and schedules. They use the TMS to view detailed information about delivery locations, including addresses, delivery instructions provided by the dispatchers. They also use the TMS to log mileage, track fuel consumption, and maintenance records.
4. **Customer Service Representatives:** They interact with customers to address inquiries and resolve issues related to deliveries. They use the TMS to track orders, monitor shipment statuses, and provide updates to customers about their deliveries. They leverage the TMS to access delivery details, including shipment dates, estimated arrival times, and proof of delivery documentation. They also use the system to manage payment records and process refunds.
5. **Managers and Executives:** Managers and executives utilize the TMS to monitor key performance indicators(KPIs) and make strategic decisions to optimize transportation operations. They use the TMS to access reports and analytics dashboards that provide insights into factors such as delivery performance, vehicle utilization, and operational efficiency. They also use the TMS to communicate with stakeholders, set business objectives, and track progress towards organizational goals.

Business Rules

- **VEHICLES** are associated with **DRIVERS** who operate them.
- Each **ROUTE** has a corresponding **SCHEDULE** for transportation services.
- **CUSTOMERS** place **ORDERS** for their transportation services of goods.
- **MAINTENANCE RECORDS**, **FUEL CONSUMPTION RECORDS** and **INSURANCE** details are linked to the specific **VEHICLES**.
- **PAYMENT RECORDS** are associated with **CUSTOMER** orders.

Table Design and Analysis

In this TMS, a total of 10 tables are there to build the entire transportation management system. I have listed those tables below with their attributes.

1. **VEHICLE:**
   -> vehicle_id (primary key) : Unique identifier for all the vehicles.
   -> make: Name of the manufacturers of vehicles.
   -> model: Name of the model of vehicles.
   -> number_of_tyres: Number of tyres in a particular vehicle.
   -> year: Manufacturing year of vehicles.
   -> capacity: Total capacity of vehicles (in Tons)

2. **DRIVER:**
   -> driver_id (primary key) : Unique identifier for all the drivers.
   -> vehicle_id (foreign key): Unique identifier for all the vehicles.
   -> driver_fname: First name of all drivers.
   -> driver_lname: Last name of all drivers.
   -> license_number: License number of all the drivers.
   -> contact_information: Contact number of all the drivers

3. **ROUTE:**
   -> route_id (primary key) : Unique identifier for all the routes.
   -> driver_id (foreign key) : Unique identifier for all the drivers.
   -> starting_point: Name of the starting point or pick up point of the goods.
   -> destination: Name of the destination or delivery point of the goods.
   -> distance: Total distance between starting point and destination (in Miles).
   -> estimated_time: Total estimated times to deliver the goods (in Days).

4. **SCHEDULE:**
   -> schedule_id (primary key) : Unique identifier for all the schedules.
   -> route_id (Foreign Key): Unique identifier for all the routes.
   -> departure_time: Time of departure.
   -> arrival_time: Time of arrival.

5. **CUSTOMER:**
   -> customer_id (primary key) : Unique identifier for all the customers.
   -> customer_fname: First name of all the customers.
   -> customer_lname: Last name of all the customers.
   -> contact_information: Contact number of all the customers.
   -> address: detailed address of all the customers.
   -> email: Email id of all the customers.

6. **ORDER_TABLE:**
   -> order_id (primary key) : Unique identifier for all the orders.
   -> customer_id (foreign key) : Unique identifier for all the customers.
   -> payment_record_id (foreign key) : Unique identifier for all the payments.
   -> delivery_address: Detailed delivery address.
   -> delivery_time: Time of the delivery.
   -> delivery_date: Date of the delivery.

7. **MAINTENANCE_RECORD:**
   -> maintenance_record_id (primary key) : Unique identifier for all the maintenance records.
   -> vehicle_id (foreign key) : Unique identifier for all the vehicles.
   -> maintenance_type: Type of the maintenance (e.g. regular service, repair).
   -> date: Date of the maintenance.
   -> description: detailed description of the maintenance.
   -> cost: Total cost of the maintenance (in CAD).

8. **FUEL_CONSUMPTION:**
   -> fuel_transaction_id (primary key) : Unique identifier for all the fuel transactions.
   -> vehicle_id (foreign key) : Unique identifier for all the vehicles.
   -> date: Date of refilling the fuel in vehicles.
   -> fuel_type: Type of the fuels.
   -> quantity: Total quantity of the fuel (in Liters).
   -> cost_per_unit: Unit cost of fuel per liter.
   -> total_cost: Total cost of fuels.

9. **INSURANCE_DETAIL:**

-> insurance_policy_id (primary key) : Unique identifier for all the insurance policies.

-> vehicle_id (foreign key) : Unique identifier for all the vehicles.

-> insurance_company: Name of the insurance providers.

-> policy_number: Policy numbers.

-> coverage_details: Detailed description of coverage.

-> expiry_date: Expiry date of the insurance.

10. **PAYMENT_RECORD:**

-> payment_record_id (primary key) : Unique identifier for all the payment records.

-> order_id (foreign key) : Unique identifier for all the orders.

-> customer_id (foreign key) : Unique identifier for all the customers.

-> payment_date: Date of the payment made.

-> payment_method: Name of the payment mode.

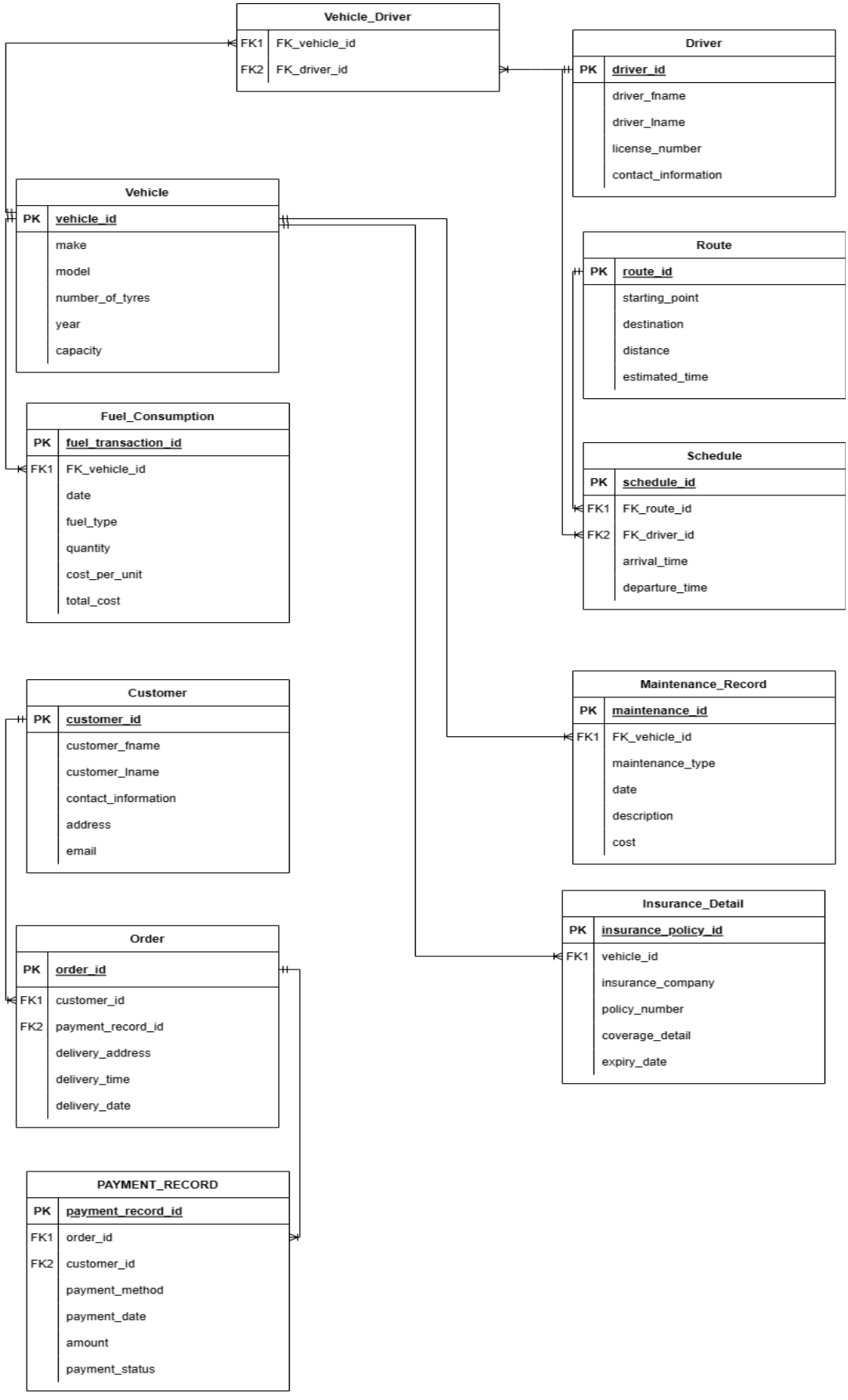
-> amount: Total number of amounts.

-> payment_status: Status of the payment.

ER Diagram

Link : ERD_of_Transportation_Management_System

Note: In the ERD, I have put the FK suffix in the foreign key so that we can easily identify them in ERD. Also, I have mentioned the relation between all the tables.


Vehicle_Driver
FK1 FK_vehicle_id
FK2 FK_driver_id
Driver
PK driver_id
driver_fname
driver_lname
license_number
contact_information
Vehicle
PK vehicle_id
make
model
number_of_tyres
year
capacity
Route
PK route_id
starting_point
destination
distance
estimated_time
Fuel_Consumption
PK fuel_transaction_id
FK1 FK_vehicle_id
date
fuel_type
quantity
cost_per_unit
total_cost
Schedule
PK schedule_id
FK1 FK_route_id
FK2 FK_driver_id
arrival_time
departure_time
Customer
PK customer_id
customer_fname
customer_lname
contact_information
address
email
Maintenance_Record
PK maintenance_id
FK1 FK_vehicle_id
maintenance_type
date
description
cost
Insurance_Detail
PK insurance_policy_id
FK1 vehicle_id
insurance_company
policy_number
coverage_detail
expiry_date
Order
PK order_id
FK1 customer_id
FK2 payment_record_id
delivery_address
delivery_time
delivery_date
PAYMENT_RECORD
PK payment_record_id
FK1 order_id
FK2 customer_id
payment_method
payment_date
amount
payment_status

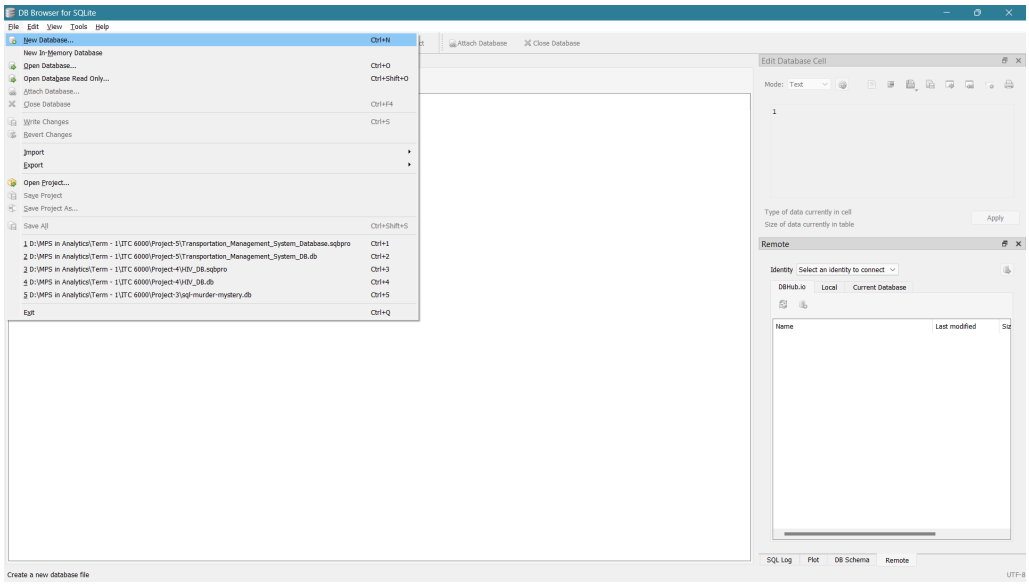
Relationships

1. Drivers to Vehicle_Driver (One-to-Many)
2. Vehicle to Vehicle_Driver (One-to-Many)
3. Schedule to Driver (One-to-Many)
4. Schedule to Route (One-to-Many)
5. Order to Customer (One-to-Many)
6. Maintenance_Records to Vehicle (One-to-Many)
7. Fuel_Consumption to Vehicle (One-to-Many)
8. Insurance_Detail to Vehicle (One-to-Many)
9. Payment_Record to Customer (One-to-Many)
10. Payment_Record to Order (One-to-Many)

Database Implementation

Now it's time to implement all the things that we have discussed and seen earlier in this report. We are going to make a database, create tables and insert records into those tables, for this we will use a DB browser.

1. Create a new Database by the name of “Transportation_Management_System_DB”.
   Figure 1



After creating the database, we are going to create the tables which are listed above in the Table design and analysis section using the CREATE TABLE command. Also we are going to insert records or data into these tables using the INSERT command.

Let’s look at the one example of how to create tables in a DB browser and insert the records into the tables.

Note: Here, I have provided the example of the Fuel_Consumption table. Because it covers how to create tables and insert records. Also how to define primary keys and foreign keys. Rest of the tables can be generated and insert data using these commands only.

Figure 2

Create a Fuel_Consumption Table and Insert the data into it using SQL commands.

```sql
CREATE TABLE Fuel_Consumption(
    fuel_transaction_id INT AUTO INCREMENT PRIMARY KEY,
    FK_driver_id INT,
    fill_date date NOT NULL,
    fuel_type VARCHAR(100) NOT NULL,
    quantity INT NOT NULL,
    cost_per_unit INT NOT NULL,
    total_cost INT NOT NULL,
    FOREIGN KEY (FK_driver_id) REFERENCES Driver(driver_id)
);
```

```sql
INSERT INTO Fuel_Consumption (fuel_transaction_id, FK_driver_id, fill_date, fuel_type, quantity, cost_per_unit, total_cost)
VALUES
(1, 1, '2023-01-01', 'Diesel', 50, 2, 100),
(2, 2, '2023-01-02', 'Gasoline', 40, 3, 120),
(3, 3, '2023-01-03', 'Diesel', 60, 2.5, 150),
(4, 4, '2023-01-04', 'Gasoline', 45, 3.2, 144),
(5, 5, '2023-01-05', 'Diesel', 55, 2.3, 126.5),
(6, 6, '2023-01-06', 'Gasoline', 50, 3.1, 155),
(7, 7, '2023-01-07', 'Diesel', 65, 2.6, 169),
(8, 8, '2023-01-08', 'Gasoline', 55, 3.3, 181.5),
(9, 9, '2023-01-09', 'Diesel', 70, 2.8, 196),
(10, 10, '2023-01-10', 'Gasoline', 60, 3.5, 210);
```

Analytics, Reports and Metrics

Now, we are going to perform SQL queries on our TMS database. We'll retrieve some useful information by joining two or more tables.

1. Retrieve list of drivers along with their assigned vehicles

Figure 3

```
-- 1. Retrieve list of drivers along with their assigned vehicles --

SELECT d.driver_fname, d.driver_lname, v.make, v.model
FROM Driver d
INNER JOIN Vehicle_Driver vd ON d.driver_id = vd.FK_driver_id
INNER JOIN Vehicle v ON vd.FK_vehicle_id = v.vehicle_id;
```

	driver_fname	driver_lname	make	model
1	John	Doe	Toyota	Camry
2	William	Jackson	Ford	F-150
3	Jessica	Taylor	Honda	Civic
4	Emily	Brown	Chevrolet	Silverado
5	Michael	Johnson	Nissan	Altima
6	Sarah	Williams	BMW	X5
7	Jane	Smith	Mercedes-Benz	E-Class
8	David	Martinez	Audi	A4
9	Chris	Anderson	Hyundai	Elantra
10	Samantha	Davis	Kia	Optima

Here, we have retrieved this information from two tables Vehicle and Driver. In the above figure 3 we can see the driver's first and last name along with their assigned vehicle company name and its model name.

2. Total Fuel consumption by each driver

Figure 4

```
-- 2. Total Fuel Consumption by each driver --

SELECT d.driver_fname, d.driver_lname, SUM(fc.quantity) AS total_fuel_consumed
FROM Driver d
INNER JOIN Fuel_Consumption fc ON d.driver_id = fc.FK_driver_id
GROUP BY d.driver_id;
```

	driver_fname	driver_lname	total_fuel_consumed
1	John	Doe	50
2	Jane	Smith	40
3	Michael	Johnson	60
4	Emily	Brown	45
5	Chris	Anderson	55
6	Sarah	Williams	50
7	David	Martinez	65
8	Jessica	Taylor	55
9	William	Jackson	70
10	Samantha	Davis	60

3. Total amount paid by each customer, this we can know by joining the customer table with payment record table. Refer to figure 5.

Figure 5

```
-- 3. Total amount paid by each customer --

SELECT c.customer_id, c.customer_fname, c.customer_lname, SUM(pr.amount) AS total_amount_paid
FROM Customer c
JOIN Payment_Record pr ON c.customer_id = pr.FK_customer_id
GROUP BY c.customer_id, c.customer_fname, c.customer_lname;
```

	customer_id	customer_fname	customer_lname	total_amount_paid
1	1	John	Doe	100
2	2	Jane	Smith	150
3	3	Michael	Johnson	200
4	4	Emily	Brown	250
5	5	David	Wilson	300
6	6	Sarah	Davis	350
7	7	Robert	Martinez	400
8	8	Jessica	Thompson	450
9	9	William	Lee	500
10	10	Elizabeth	Garcia	550

4. Retrieve all maintenance records for vehicles with capacity greater than 4 tons. By executing this query we can know the maintenance record for the vehicle which has capacity to carry more than 4 tons. Here we can know the date of maintenance , total cost, driver associated with a particular vehicle and description of the maintenance. Refer to figure 6.

Figure 6

```
-- 4. Retrieve all maintenance records for vehicles with capacity greater than 4 --

SELECT m.*, v.make, v.model
FROM Maintenance_Record m
JOIN Vehicle_Driver vd ON m.FK_driver_id = vd.FK_driver_id
JOIN Vehicle v ON vd.FK_vehicle_id = v.vehicle_id
WHERE v.capacity > 4;
```

	maintenance_id	FK_driver_id	maintenance_date	description	cost	make	model
1	1	1	2023-01-05	Regular service	100	Toyota	Camry
2	8	8	2023-02-10	Regular service	130	Honda	Civic
3	3	3	2023-01-15	Brake replacement	150	Nissan	Altima
4	6	6	2023-01-30	Wheel alignment	180	BMW	X5
5	2	2	2023-01-10	Oil change	80	Mercedes-Benz	E-Class
6	7	7	2023-02-05	Oil change	160	Audi	A4
7	5	5	2023-01-25	Engine check	200	Hyundai	Elantra
8	10	10	2023-02-20	Regular service	110	Kia	Optima

5. What would be the average cost of fuel consumption, this we can know by joining the driver table with fuel_consumption table. Refer to Figure 7.

Figure 7

```
-- 5. Average Cost of fuel consumption --

SELECT d.driver_id, d.driver_fname, d.driver_lname, AVG(fc.total_cost) AS avg_fuel_cost
FROM Driver d
JOIN Fuel_Consumption fc ON d.driver_id = fc.FK_driver_id
GROUP BY d.driver_id, d.driver_fname, d.driver_lname;
```

driver_id	driver_fname	driver_lname	avg_fuel_cost
1	John	Doe	100.0
2	Jane	Smith	120.0
3	Michael	Johnson	150.0
4	Emily	Brown	144.0
5	Chris	Anderson	126.5
6	Sarah	Williams	155.0
7	David	Martinez	169.0
8	Jessica	Taylor	181.5
9	William	Jackson	196.0
10	Samantha	Davis	210.0

6. Now, We want to know what is the longest route and how much time it takes and who was the driver of that vehicle. Here, we have joined three tables together, route, driver and schedule. Refer to Figure 8.

Figure 8

```
284  -- 6. Routes with longest estimated time and their corresponding drives --
285
286  SELECT r.route_id, r.starting_point, r.destination, r.estimated_time, d.driver_fname, d.driver_lname
287  FROM Route r
288  JOIN Schedule s ON r.route_id = s.FK_route_id
289  JOIN Driver d ON s.FK_driver_id = d.driver_id
290  WHERE r.estimated_time = (SELECT MAX(estimated_time) FROM Route);
```

	route_id	starting_point	destination	estimated_time	driver_fname	driver_lname
1	6	City F	City G	250	Sarah	Williams

As we can see from figure 8 that there is only one route from City F to City G that takes around 250 minutes and Sarah Williams was the driver for that route.

7. At last we want to find out how many customers have pending payment status and who are those customers. Refer to figure 9.

Figure 9

```
292    -- 7. List of customers with pending payment status --
293
294    SELECT c.customer_id, c.customer_fname, c.customer_lname, p.payment_status
295    FROM Customer c
296    JOIN Order_Table o ON c.customer_id = o.FK_customer_id
297    JOIN Payment_Record p ON o.order_id = p.FK_order_id
298    WHERE p.payment_status = 'Pending';
299
```

	customer_id	customer_fname	customer_lname	payment_status
1	2	Jane	Smith	Pending
2	4	Emily	Brown	Pending
3	5	David	Wilson	Pending
4	9	William	Lee	Pending

Security Concerns

The data in these tables contains various sensitive information that raises security and privacy concerns. For instance, the **Customer** table stores the personal details such as names, contact information, address and email ids. This information could be exploited if accessed by unauthorized parties, leading to potential privacy violations and risks such as identity theft or spamming.

Similarly, in the **Driver** table we have stored the license number of each driver and their contact information. While the **Insurance_detail** table includes the policy number and name of the company that provides the insurance. Moreover, The **order_table** contains delivery addresses and contact details, which if compromised could jeopardize the security and privacy of customers' personal information.

To prevent these concerns, robust security measures such as access controls, encryption, and authentication mechanisms should be implemented to safeguard the data. Regular security audits and monitoring procedures should be in place to detect and prevent unauthorized access or data breaches.

Architecture

1. **Client/Server Architecture Solution:** In our transportation management system (TMS) project, various end-users like administrative staff, dispatchers, drivers, and customer service representatives engage with the system through intuitive interfaces. These interfaces may take the form of web-based portals accessible via browsers or mobile applications installed on smartphones or tablets. Our database resides on central servers, managing incoming requests and delivering pertinent information. Utilizing a client/server architecture ensures centralized administration, scalability, and enhanced security. This separation of user interface design from the backend logic facilitates streamlined maintenance and updates.

2. **Cloud-Based Hosting:** Cloud-based hosting entails deploying and overseeing the TMS application and data on remote servers offered by cloud service providers like AWS, Azure, or GCP. These platforms furnish dependable infrastructure services encompassing virtual servers, storage, networking, and enhanced security features within the cloud milieu for our TMS application. Opting for cloud-based hosting offers several advantages, including scalability to manage fluctuating workloads, adaptability to adjust resources as needed, accessibility from any location, and cost-effectiveness through pay-as-you-go models. In contrast, local storage lacks these functionalities, necessitating additional security measures and making database migration challenging.

3. **Storage Requirements:** Storage requirements for the TMS hinge on factors like data volume, update frequency, and data types such as text, image, or video. Given our need to store vehicle, driver, maintenance, schedule, route, delivery, and payment record information, the TMS demands a substantial storage capacity. Sufficient storage capacity guarantees seamless handling of large data volumes without performance interruptions or resource limitations. Employing efficient data storage techniques like indexing, compression, and partitioning optimizes storage usage and improves performance.

Project Wrap-up and Future Consideration

Throughout my journey in developing the transportation management system, I've acquired valuable insights and skills. Firstly, I've gained proficiency in business analysis, learning to conceptualize the various user personas and envision the use cases for the TMS. Secondly, I've mastered the art of formulating Entity-Relationship Diagrams (ERDs), understanding the intricacies of entities, attributes, primary keys, and foreign keys, and adeptly creating tables in database browsers while efficiently inserting records. Additionally, I've navigated the complexities of establishing relationships among diverse entities, ensuring seamless data management.

Furthermore, I've delved into the construction of the system's architecture, refining it to enhance functionality and scalability. This process has honed my ability to design robust frameworks that effectively cater to the system's requirements. Lastly, I've come to appreciate the paramount importance of security considerations post-database creation, recognizing the critical need to safeguard sensitive information from potential threats. Overall, this journey has equipped me with a comprehensive skill set and a deeper understanding of database development processes, positioning me to tackle complex challenges in future projects with confidence and expertise.

In the foreseeable future, I envision scaling up my Transportation Management System (TMS) to accommodate a broader user base and enhance its accessibility and usability. This expansion will involve incorporating advanced functionalities aimed at optimizing transportation operations and delivering an unparalleled user experience. Among the key enhancements on the roadmap are the implementation of real-time vehicle tracking capabilities, sophisticated route optimization algorithms, electronic proof of delivery mechanisms, and comprehensive performance analytics tools.

References

DB Browser for SQLite. (n.d.),(February 21st, 2024), https://sqlitebrowser.org/dl/

Rahul H. Prajapati, (February 25th, 2024), Prajapati_Module1_Final_Project.pdf, Database Management System, Northeastern University

Rahul H. Prajapati, (March 3rd, 2024), Prajapati_Milestone2_BusinessRules_ERD.pdf, Database Management System, Northeastern University

Database Academy, Database Components, https://database.academy/database-tutorials/database-architecture/

Database Academy, Database Approach, https://database.academy/database-tutorials/database-architecture/