UNIVERSITY OF WOLVERHAMPTON

7C1022: Database System and Security

School of Mathematics and Computer Science Faculty of Science and Engineering

Case Study Review Report

- **Njinju ZIlefac Fogap - 2439344**
- **Chukwuebuka Emmanuel Owoh - 2328893**
- **Patrick Nnaemeka Nwonah - 2437973**

DATE: 1/04/2025

TABLE OF CONTENTS

PART A:

Part 1 (LO1, LO2) is about the design of your database.

1. Introduction to the Chosen Scenario (Transport Management System)

Scenario Overview

In Cameroon and many parts of Africa, travel agencies still rely on paper-based systems for recording client information, bookings, and payments. This system is inefficient, prone to errors, and lacks scalability. To address these challenges, a Transport Management System (TMS) is proposed. This system will serve as a centralized database and web application that allows users to book and pay for transport services from different agencies using various payment methods.

Key Features

- **Agency Management:** Agencies can register, manage schedules, and track bookings.
- **Digital Client Management:** Secure storage and management of customer details.
- **Online Booking System:** Users can book tickets via a web platform.
- **Reporting & Analytics:** Data-driven insights into bookings, revenue, and customer behavior.

By implementing this solution, travel agencies can modernize their operations, improve efficiency, and offer a seamless travel experience.

Informal Description:

The "Transport Management System" must manage numerous interactions. Agencies can register with administrators for each agency, allowing them to list various buses along with their departure and arrival locations, accompanied by comprehensive schedules for each trip. Users or customers have the option to create a profile, book specific buses, and make payments through various methods, all governed by a payment status. Once the payment status indicates completion, the transaction is approved, and journey details including route, arrival, and departure times are sent to clients, along with additional information about the bus. In the future, customers will have the capability to select specific seats on the bus.

Information Needs:

- Customer details (Userid, Name, Password, Email, Address, Payment methods, Registration date,Phone Number).
- Admin details (AdminID, Name, Password, Email, Role, Email).
- Agency details (AgencyID, Name, Location, Status, Contact information, Registration Number).
- Booking (BookingID, Status (confirmed/pending/cancelled).
- Payment (PaymentID, Timestamp, Transaction Status, TransactionID, Amount, Payment method).
- Bus (BusID, Model, Status, Capacity, License plate, Last Maintenance Date).
- Schedule (ScheduleID, Status, Price, departure, Available seats).
- Route (RouteID, DistanceKM, Arrival location, Departure location, Duration, Price).

2. Business Rules for the Transport Management System

The business rules define how different entities interact in the system:

1. **User Registration & Booking Rules:**

- A user must register before making a booking.
- A user can have multiple bookings.
- Each booking must be linked to an existing route and schedule.

2. **Payment Rules:**

- Each booking requires a valid payment transaction.
- Payments must be verified before a booking is confirmed.
- Multiple payment methods should be supported (e.g., mobile money, credit card).

3. **Agency and Bus Management Rules:**

- An agency can register multiple buses.
- Each bus belongs to one agency and must have a valid schedule.
- Agencies are responsible for updating their schedules.

4. **Route and Schedule Rules:**

- Each route consists of a departure and arrival location.

A bus must be assigned to a valid schedule before it can be booked.

These rules ensure data integrity and consistency in the system.

3. Entity Relationship (ER) Diagram

Bellow you will find and architecture diagram that represents the ER diagram for this scenario:

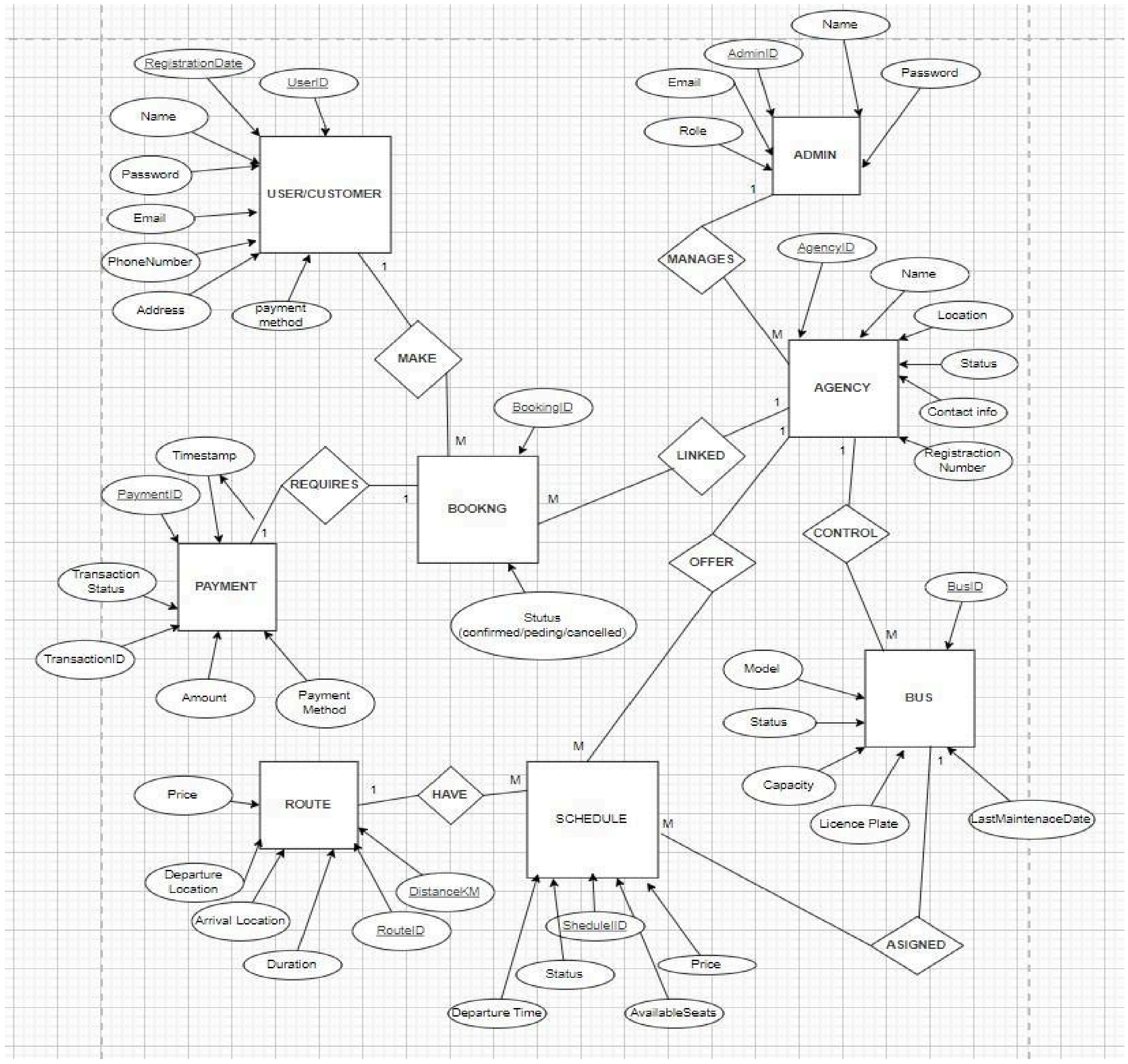


fig: Architecture of Transport Management System.

After coming out with the architecture of this scenario we now implemented it on Oracle SQL Developer Data Modeler, below you will see the logical model and the Relational model.

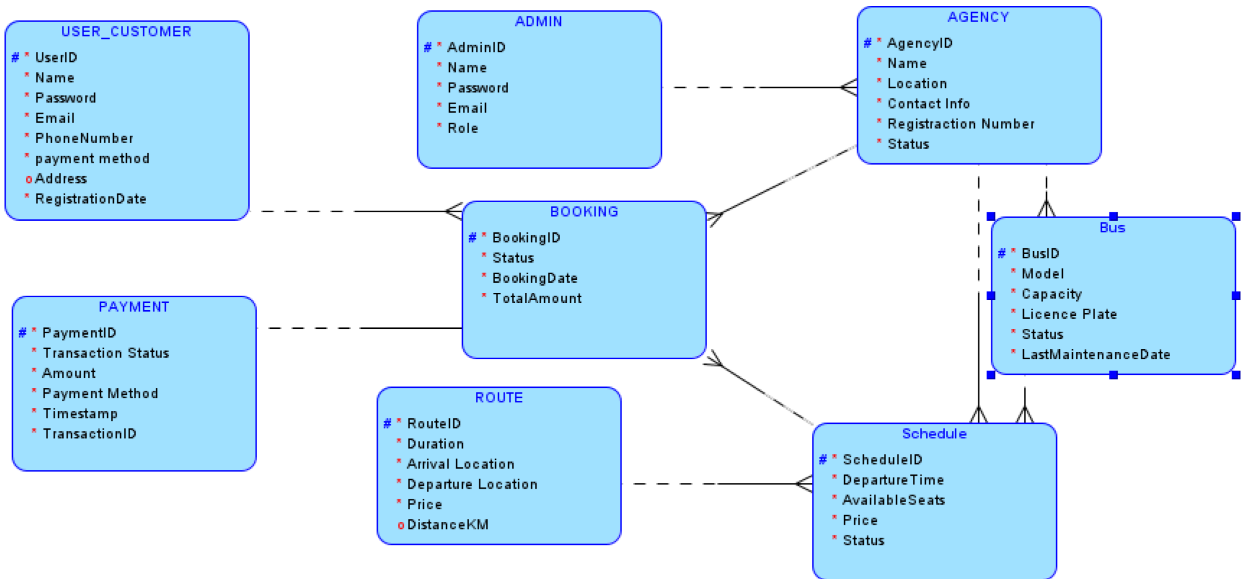


Fig: Logical Model of Transport Management System

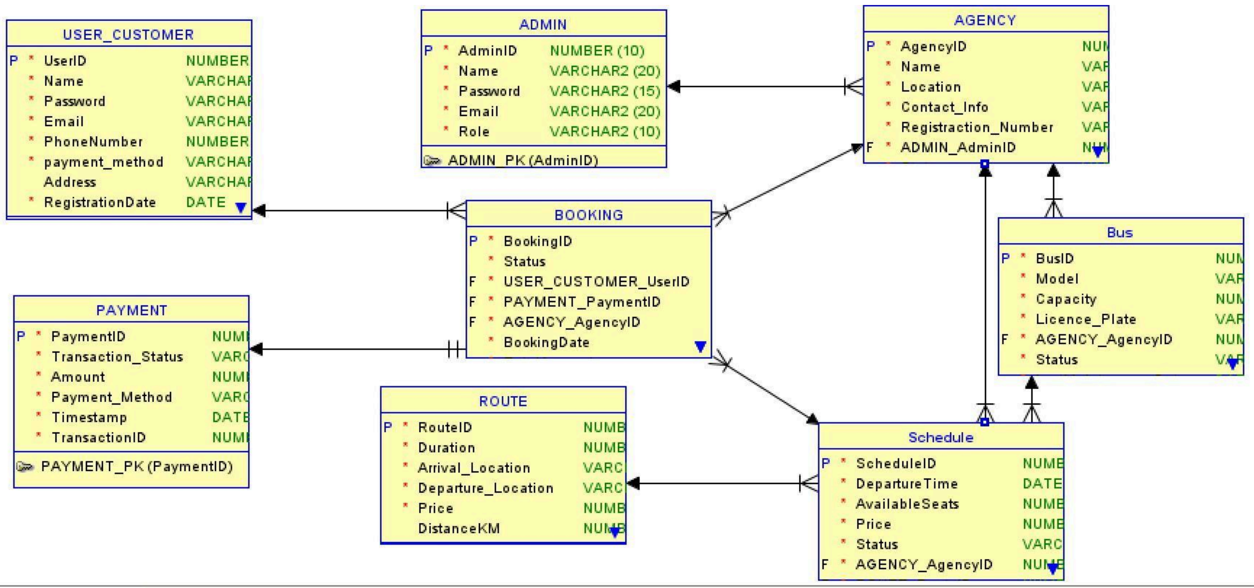


Fig: Relational model of Transport Management System

The ER diagram for this transport management system includes eight key entities:

1. **Admin:** Manages the system, agencies, and customer support.
2. **User_Customer:** Registers users who book and make payments.
3. **Agency:** Manages buses and schedules.
4. **Bus:** Represents transport vehicles assigned to routes.

5. **Route:** Defines travel routes with departure and arrival locations.
6. **Schedule:** Connects users to available routes and agencies.
7. **Booking:** Captures user bookings.
8. **Payment:** Tracks transactions and verifies payment status.

Relationships between every entity.

- **User → Booking (1:M):** One user can have multiple bookings.
- **Agency → Bus (1:M):** One agency owns multiple buses.
- **Agency → Schedule (1:M):** One agency can offer multiple schedules.
- **Route → Schedule (1:M):** One route can have multiple schedules.
- **Bus → Schedule (1:M):** One bus can be assigned to multiple schedules.
- **Schedule → Booking (1:M):** One schedule can have multiple bookings.
- **Booking → Payment (1:1):** Each booking has one payment.
- **Admin → Agency (1:M):** One admin manages multiple agencies.

The ER model is implemented using **Oracle SQL Developer Data Modeler**, ensuring efficient data representation.

4. Normalization Process

The database is structured using **normalization** to reduce redundancy and improve efficiency.

Normalization Steps

1. **First Normal Form (1NF)**:

- Ensures all attributes have atomic values.
- Each entity has a primary key.

2. **Second Normal Form (2NF)**:

- Removes partial dependencies (each attribute depends on the whole primary key).
- Example: The **Schedule** table was created to separate bus, route, and agency details.

3. **Third Normal Form (3NF)**:

- Eliminates transitive dependencies (non-key attributes depending on other non-key attributes).
- Example: The Payment table only stores payment-related details, separate from booking.

Example using an unnormalized table (UNF):

BookingID	UserID	UserName	UserEmail	UserPhone	PaymentID	PaymentStatus	Amount	PaymentMethod	TransactionID	AgencyID	AgencyNa me	RouteID	Departure_Arrival	ScheduleID	DepartureTime	BusID	BusModel	AvailableSeats
201	1	Alice	alice@email.com	6789012345, 6789011111	3001	Successful	45000	Credit Card	9234561	601	Prime Expr ess	801	Douala-Yaoundé	901	2025-03-10 08:00	1001	Toyota Coaster	10
202	2	Bob	bob@email.com	6789023456, 6789022222	3002	Failed	32000	Mobile Money	8234562	602	Global Transit	802	Bafoussam-Yaoundé	902	2025-03-12 14:00	1002	Mercedes Sprinter	12

TABLE: UNNORMALIZED TABLE

The above table is in an unnormalized form, this is due to the existence of repeating groups, non-atomic values, partial and transitive dependencies. So, we need to transform it into 1NF, 2NF and 3NF.

First Normal Form TABLE:

BookingID	UserID	UserName	UserEmail	UserPhone	PaymentID	PaymentStatus	Amount	PaymentMethod	TransactionID	AgencyID	AgencyName	RouteID	Departure_Location	Arrival_Location	ScheduleID	DepartureTime	BusID	BusModel	AvailableSeats
201	1	Alice	alice@email.com	6789012345	3001	Successful	45000	Credit Card	9234561	601	Prime Express	801	Douala	Yaoundé	901	2025-03-10 08:00	1001	Toyota Coaster	10
201	1	Alice	alice@email.com	6789011111	3001	Successful	45000	Credit Card	9234561	601	Prime Express	801	Douala	Yaoundé	901	2025-03-10 08:00	1001	Toyota Coaster	10
202	2	Bob	bob@email.com	6789023456	3002	Failed	32000	Mobile Money	8234562	602	Global Transit	802	Bafoussam	Yaoundé	902	2025-03-12 14:00	1002	Mercedes Sprinter	12
202	2	Bob	bob@email.com	6789022222	3002	Failed	32000	Mobile Money	8234562	602	Global Transit	802	Bafoussam	Yaoundé	902	2025-03-12 14:00	1002	Mercedes Sprinter	12

From the above 1NF, we notice the following:

- Atomic values only, that is multiple phone numbers are separated into rows.
- Each field stores only one value, hence limiting redundancy.
- Unique rows.

Second Normal Form:

In this form we intend to report all partial dependencies from the 1NF, ensuring that every non-key column must depend on the entire primary key and breaking tables into smaller and logical groups while making sure the table is in 1NF.

1. **User table :** (PK -> UserID , attribute -> UserName, UserEmail)

UserID	**UserPhone**	**UserName**	**UserEmai**
1	6789012345	Alice	alice@email.com
1	6789011111	Alice	alice@email.com
2	6789023456	Bob	bob@email.com
2	6789022222	Bob	bob@email.com

2. **Agency Table :** (PK-> AgencyID , attribute -> AgencyName)

AgencyID	**AgencyName**
601	Prime Express
602	Global Transit

3. **Route Table :** (PK -> RouteID, attribute -> Departure_location, Arrival_Location)

RouteID	**Departure_Location**	**Arrival_Location**
801	Douala	Yaoundé
802	Bafoussam	Yaoundé

4. **Bus Table**: (PK -> BusID, Attribute -> BusModel, AvailableSeats)

BusID	**BusModel**	**AvailableSeats**
1001	Toyota Coaster	10
1002	Mercedes Sprinter	12

5. **Booking Table** : (PK ->BookingID , Attribute -> UserID, PaymentID, PaymentStatus, Amount, PaymentMethod, TransactionID, AgencyID, RouteID, ScheduleID, BusID)

BookingI	**UserID**	**PaymentI**	**PaymentS**	**Payment**	**Transacti**	**AgencyID**	**RouteID**	**ScheduleI**	**BusID**

D		D	tatus	Method	onID			D	
201	1	3001	Successfu l	Credit Card	9234561	601	801	901	1001
202	2	3002	Failed	Mobile Money	8234562	602	802	902	1002

6. **UserAgencyRouteBusAgencyBookingTable** :

UserID	AgencyID	RouteID	BusID	BookingID	Amount
1	601	801	1001	201	45000
2	602	802	1002	202	32000

Third Normal Form:

To achieve the 3rd Normal Form (3NF), it is essential to eliminate transitive dependencies, meaning that all non-key attributes must depend solely on the primary key without any intermediary. Most of the tables mentioned above are already in 3NF, as their attributes are directly linked to the primary key. However, in the UserTable, both UserName and UserEmail are dependent on UserID rather than UserPhone. This situation represents a transitive dependency since UserPhone is included in the primary key but does not influence the values of UserName or UserEmail. Therefore, it is necessary to further divide the user table.

1. **UserTable**:

UserID	UserName	UserEmail
1	Alice	alice@email.com
2	Bob	bob@email.com

2. **UserPhone Table** :

UserID	UserPhone
1	6789012345
1	6789011111
2	6789023456
2	6789022222

3. **Agency Table :** (PK-> AgencyID , attribute -> AgencyName)

AgencyID	AgencyName
601	Prime Express
602	Global Transit

4. **Route Table :** (PK -> RouteID, attribute -> Departure_location, Arrival_Location)

RouteID	Departure_Location	Arrival_Location
801	Douala	Yaoundé
802	Bafoussam	Yaoundé

5. **Bus Table** : (PK -> BusID , Attribute -> BusModel, AvailableSeats)

BusID	BusModel	AvailableSeats
1001	Toyota Coaster	10
1002	Mercedes Sprinter	12

6. **Booking Table** : (PK ->BookingID , Attribute -> UserID, PaymentID, PaymentStatus, Amount, PaymentMethod, TransactionID, AgencyID, RouteID, ScheduleID, BusID)

BookingID	UserID	PaymentID	PaymentStatus	Payment Method	TransactionID	AgencyID	RouteID	ScheduleID	BusID
201	1	3001	Successful	Credit Card	9234561	601	801	901	1001
202	2	3002	Failed	Mobile Money	8234562	602	802	902	1002

7. **UserAgencyRouteBusAgencyBookingTable** :

UserID	AgencyID	RouteID	BusID	BookingID	Amount
1	601	801	1001	201	45000
2	602	802	1002	202	32000

The result schema above consists of 7 tables, each table now satisfies the requirements of 3NF, with no partial and transitive dependencies

PART B:

The second part (LO1, LO3) is about implementing the database and legal and ethical aspects of data governance and security related to your scenario.

Database Implementation:

A. SQL Statements to Create and Populate Tables

Here are the SQL statements for table creation and data insertion.

Create Tables

Generated by Oracle SQL Developer Data Modeler 24.3.1.351.0831

- **at: 2025-02-28 12:08:32 CET**
- **site: Oracle Database 11g**
- **type: Oracle Database 11g**

▪ predefined type, no DDL - MDSYS.SDO_GEOMETRY

▪ predefined type, no DDL -

```
XMLTYPE CREATE TABLE ADMIN

   (

    AdminID NUMBER (10) NOT NULL,

    Name     VARCHAR2 (20) NOT NULL,

    Password VARCHAR2 (15) NOT NULL,

    Email    VARCHAR2 (20) NOT NULL,

    Role     VARCHAR2 (10) NOT NULL

   )

;

ALTER TABLE ADMIN

   ADD CONSTRAINT ADMIN_PK PRIMARY KEY (AdminID);

CREATE TABLE AGENCY

   (

    AgencyID             NUMBER (10) NOT NULL,
    Name                  VARCHAR2 (20) NOT NULL,

    Location              VARCHAR2 (20) NOT NULL,

    Contact_Info          VARCHAR2 (20) NOT NULL,

    Registraction_Number VARCHAR2 (30) NOT

    NULL, ADMIN_AdminID              NUMBER (10)

    NOT NULL, Status  VARCHAR2 (10) NOT NULL

   )

;

ALTER TABLE AGENCY

ADD CONSTRAINT AGENCY_PK PRIMARY KEY (AgencyID);

CREATE TABLE BOOKING

   (

    BookingID             NUMBER (20) NOT NULL,

    Status                VARCHAR2 (25) NOT NULL,

    USER_CUSTOMER_UserID NUMBER (10) NOT NULL,

    PAYMENT_PaymentID    NUMBER (30) NOT NULL,

    AGENCY_AgencyID      NUMBER (10) NOT NULL,

    BookingDate           DATE NOT NULL,

    TotalAmount           NUMBER (20) NOT NULL,

    Schedule_ScheduleID NUMBER (30) NOT NULL

   )

;

CREATE UNIQUE INDEX BOOKING_IDX ON BOOKING

   (

    PAYMENT_PaymentID ASC

   )

;

ALTER TABLE BOOKING

   ADD CONSTRAINT BOOKING_PK PRIMARY KEY (BookingID);
```

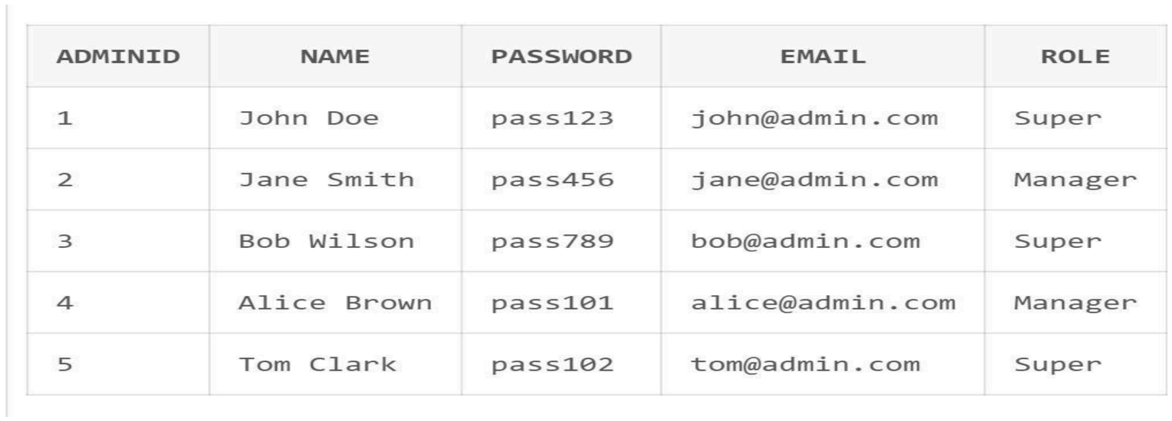
```
CREATE TABLE Bus

(

     BusID                NUMBER (20) NOT NULL,

     Model                VARCHAR2 (20) NOT NULL,

     Capacity             NUMBER (20) NOT NULL,

     Licence_Plate        VARCHAR2 (20) NOT NULL,

     AGENCY_AgencyID      NUMBER (10) NOT NULL,

     Status               VARCHAR2 (10) NOT NULL,

     LastMaintenanceDate DATE NOT NULL

    )

;

ALTER TABLE Bus

    ADD CONSTRAINT Bus_PK PRIMARY KEY (BusID);

CREATE TABLE PAYMENT

    (

     PaymentID            NUMBER (30) NOT NULL,

     Transaction_Status VARCHAR2 (10) NOT

     NULL, Amount    NUMBER (30,5) NOT NULL,

     Payment_Method     VARCHAR2 (20) NOT NULL,

     Timestamp            DATE NOT NULL,

     TransactionID        NUMBER (25) NOT NULL

    )

;

ALTER TABLE PAYMENT

    ADD CONSTRAINT PAYMENT_PK PRIMARY KEY ( PaymentID ) ;

CREATE TABLE ROUTE

    (

     RouteID              NUMBER (30) NOT NULL,

     Duration             NUMBER (20) NOT NULL,
     Arrival_Location VARCHAR2 (20) NOT NULL,
     Departure_Location VARCHAR2 (20) NOT NULL,
     Price              NUMBER (20) NOT NULL,
     DistanceKM         NUMBER (10)
    )
;
ALTER TABLE ROUTE
    ADD CONSTRAINT ROUTE_PK PRIMARY KEY ( RouteID ) ;
CREATE TABLE Schedule
    (
     ScheduleID      NUMBER (30) NOT NULL,
     DepartureTime DATE NOT NULL,
     AvailableSeats NUMBER (20) NOT NULL,
     Price           NUMBER (20,3) NOT NULL,
     Status          VARCHAR2 (20) NOT NULL,
     AGENCY_AgencyID NUMBER (10) NOT
     NULL, ROUTE_RouteID NUMBER (30) NOT
     NULL, Bus_BusID
     NUMBER (20) NOT NULL
    )
;
ALTER TABLE Schedule
    ADD CONSTRAINT Schedule_PK PRIMARY KEY (ScheduleID);
CREATE TABLE USER_CUSTOMER
    (
     UserID          NUMBER (10) NOT NULL,
     Name             VARCHAR2 (20) NOT NULL,
     Password         VARCHAR2 (20) NOT NULL,
     Email           VARCHAR2 (15) NOT NULL,
     PhoneNumber      NUMBER (15) NOT NULL,
     payment_method VARCHAR2 (20) NOT NULL,
     Address          VARCHAR2 (20),
     RegistrationDate DATE NOT NULL
    )
;
ALTER TABLE USER_CUSTOMER
    ADD CONSTRAINT USER_CUSTOMER_PK PRIMARY KEY (UserID);
ALTER TABLE AGENCY
    ADD CONSTRAINT AGENCY_ADMIN_FK FOREIGN KEY
    (
     ADMIN_AdminID
    )
    REFERENCES
    ADMIN (
     AdminID
    )
;
ALTER TABLE BOOKING
    ADD CONSTRAINT BOOKING_AGENCY_FK FOREIGN KEY
    (
     AGENCY_AgencyID
    )
    REFERENCES AGENCY
    (
     AgencyID
    )
;
ALTER TABLE BOOKING
    ADD CONSTRAINT BOOKING_PAYMENT_FK FOREIGN KEY
    (
     PAYMENT_PaymentID
    )
    REFERENCES PAYMENT
    (
     PaymentID
    )
;
ALTER TABLE BOOKING
    ADD CONSTRAINT BOOKING_Schedule_FK FOREIGN KEY
    (
     Schedule_ScheduleID
    )
    REFERENCES Schedule
    (
     ScheduleID
    )
;
ALTER TABLE BOOKING
    ADD CONSTRAINT BOOKING_USER_CUSTOMER_FK FOREIGN
    KEY (
     USER_CUSTOMER_UserID
    )
    REFERENCES
    USER_CUSTOMER (
     UserID
    )
;
ALTER TABLE Bus
```

```
    ADD CONSTRAINT Bus_AGENCY_FK FOREIGN KEY
    (
     AGENCY_AgencyID
    )
    REFERENCES AGENCY
    (
     AgencyID
    )
;
ALTER TABLE Schedule
    ADD CONSTRAINT Schedule_AGENCY_FK FOREIGN
    KEY (
     AGENCY_AgencyID
    )
    REFERENCES AGENCY
    (
     AgencyID
    )
;
ALTER TABLE Schedule
    ADD CONSTRAINT Schedule_Bus_FK FOREIGN KEY
    (
     Bus_BusID
    )
    REFERENCES
    Bus (
     BusID
    )
;
```

```
ALTER TABLE Schedule
    ADD CONSTRAINT Schedule_ROUTE_FK FOREIGN
    KEY (
     ROUTE_RouteID
    )
    REFERENCES ROUTE
    (
     RouteID
    )
;
-- Oracle SQL Developer Data Modeler Summary Report:
-- CREATE TABLE                            8
-- CREATE INDEX                            1
-- ALTER TABLE                            17
-- CREATE VIEW                             0
-- ALTER VIEW                              0
-- CREATE PACKAGE                          0
-- CREATE PACKAGE BODY                     0
-- CREATE PROCEDURE                        0
-- CREATE FUNCTION                         0
-- CREATE TRIGGER                          0
-- ALTER TRIGGER                           0
-- CREATE COLLECTION TYPE                  0
-- CREATE STRUCTURED TYPE                  0
-- CREATE STRUCTURED TYPE BODY             0
-- CREATE CLUSTER                          0
-- CREATE CONTEXT                          0
-- CREATE DATABASE                         0
-- CREATE DIMENSION                        0
-- CREATE DIRECTORY                  0
-- CREATE DISK GROUP                 0
-- CREATE ROLE                       0
-- CREATE ROLLBACK SEGMENT           0
-- CREATE SEQUENCE                   0
-- CREATE MATERIALIZED VIEW          0
-- CREATE MATERIALIZED VIEW LOG      0
-- CREATE SYNONYM                    0
-- CREATE TABLESPACE                 0
-- CREATE USER                       0
-- DROP TABLESPACE                   0
-- DROP DATABASE                     0
-- REDACTION POLICY                  0
-- ORDS DROP SCHEMA                  0
-- ORDS ENABLE SCHEMA                0
-- ORDS ENABLE OBJECT                0
-- ERRORS                            0
-- WARNINGS                          0
```

Populate Tables

```
-- Insert into ADMIN
INSERT INTO ADMIN VALUES (1, 'John Doe', 'pass123', 'john@admin.com', 'Super');
INSERT INTO ADMIN VALUES (2, 'Jane Smith', 'pass456', 'jane@admin.com', 'Manager');
INSERT INTO ADMIN VALUES (3, 'Bob Wilson', 'pass789', 'bob@admin.com', 'Super');
INSERT INTO ADMIN VALUES (4, 'Alice Brown', 'pass101', 'alice@admin.com', 'Manager');
INSERT INTO ADMIN VALUES (5, 'Tom Clark', 'pass102', 'tom@admin.com', 'Super');
```



ADMINID	NAME	PASSWORD	EMAIL	ROLE
1	John Doe	pass123	john@admin.com	Super
2	Jane Smith	pass456	jane@admin.com	Manager
3	Bob Wilson	pass789	bob@admin.com	Super
4	Alice Brown	pass101	alice@admin.com	Manager
5	Tom Clark	pass102	tom@admin.com	Super

Fig: Table ADMIN

-- Insert into AGENCY

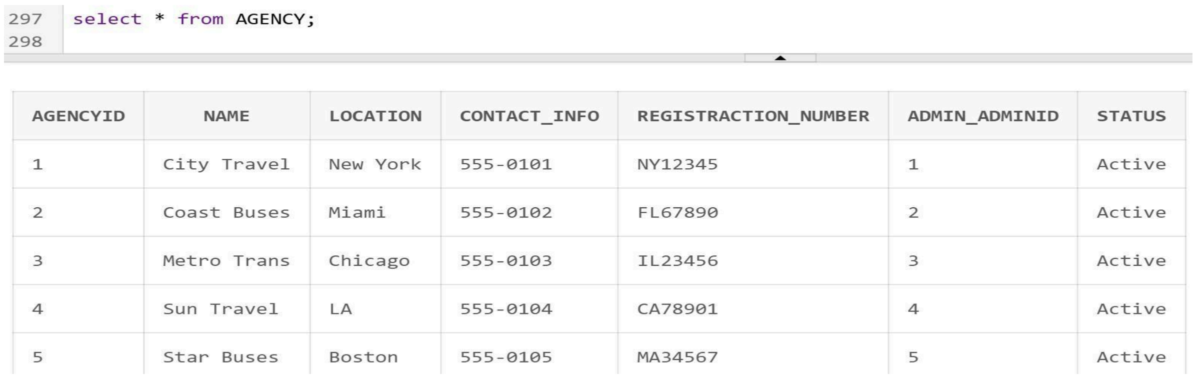
INSERT INTO AGENCY VALUES (1, 'City Travel', 'New York', '555-0101', 'NY12345', 1, 'Active');

INSERT INTO AGENCY VALUES (2, 'Coast Buses', 'Miami', '555-0102', 'FL67890', 2, 'Active');

INSERT INTO AGENCY VALUES (3, 'Metro Trans', 'Chicago', '555-0103', 'IL23456', 3, 'Active');

INSERT INTO AGENCY VALUES (4, 'Sun Travel', 'LA', '555-0104', 'CA78901', 4, 'Active');

INSERT INTO AGENCY VALUES (5, 'Star Buses', 'Boston', '555-0105', 'MA34567', 5, 'Active');



```
select * from AGENCY;
```

AGENCYID	NAME	LOCATION	CONTACT_INFO	REGISTRACTION_NUMBER	ADMIN_ADMINID	STATUS
1	City Travel	New York	555-0101	NY12345	1	Active
2	Coast Buses	Miami	555-0102	FL67890	2	Active
3	Metro Trans	Chicago	555-0103	IL23456	3	Active
4	Sun Travel	LA	555-0104	CA78901	4	Active
5	Star Buses	Boston	555-0105	MA34567	5	Active

Fig: Table AGENCY

- Insert into USER_CUSTOMER

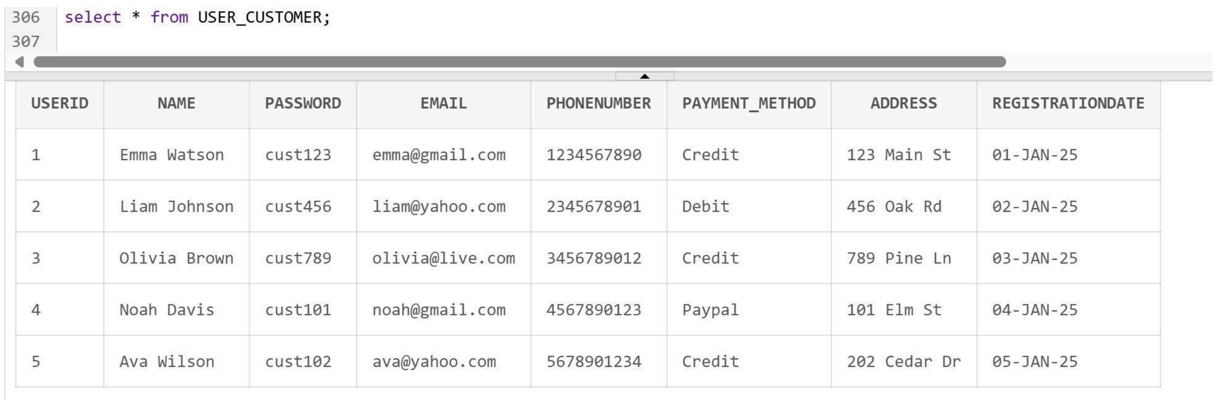
INSERT INTO USER_CUSTOMER VALUES (1, 'Emma Watson', 'cust123', 'emma@gmail.com', 1234567890, 'Credit', '123 Main St', TO_DATE ('2025-01-01', 'YYYY-MM-DD'));

INSERT INTO USER_CUSTOMER VALUES (2, 'Liam Johnson', 'cust456', 'liam@yahoo.com', 2345678901, 'Debit', '456 Oak Rd', TO_DATE('2025-01-02', 'YYYY-MM-DD'));

INSERT INTO USER_CUSTOMER VALUES (3, 'Olivia Brown', 'cust789', 'olivia@live.com', 3456789012, 'Credit', '789 Pine Ln', TO_DATE('2025-01-03', 'YYYY-MM-DD'));

INSERT INTO USER_CUSTOMER VALUES (4, 'Noah Davis', 'cust101', 'noah@gmail.com', 4567890123, 'Paypal', '101 Elm St', TO_DATE('2025-01-04', 'YYYY-MM-DD'));

INSERT INTO USER_CUSTOMER VALUES (5, 'Ava Wilson', 'cust102', 'ava@yahoo.com', 5678901234, 'Credit', '202 Cedar Dr', TO_DATE('2025-01-05', 'YYYY-MM-DD'));



```
select * from USER_CUSTOMER;
```

USERID	NAME	PASSWORD	EMAIL	PHONENUMBER	PAYMENT_METHOD	ADDRESS	REGISTRATIONDATE
1	Emma Watson	cust123	emma@gmail.com	1234567890	Credit	123 Main St	01-JAN-25
2	Liam Johnson	cust456	liam@yahoo.com	2345678901	Debit	456 Oak Rd	02-JAN-25
3	Olivia Brown	cust789	olivia@live.com	3456789012	Credit	789 Pine Ln	03-JAN-25
4	Noah Davis	cust101	noah@gmail.com	4567890123	Paypal	101 Elm St	04-JAN-25
5	Ava Wilson	cust102	ava@yahoo.com	5678901234	Credit	202 Cedar Dr	05-JAN-25

Fig: Table User_Customer

- Insert into ROUTE

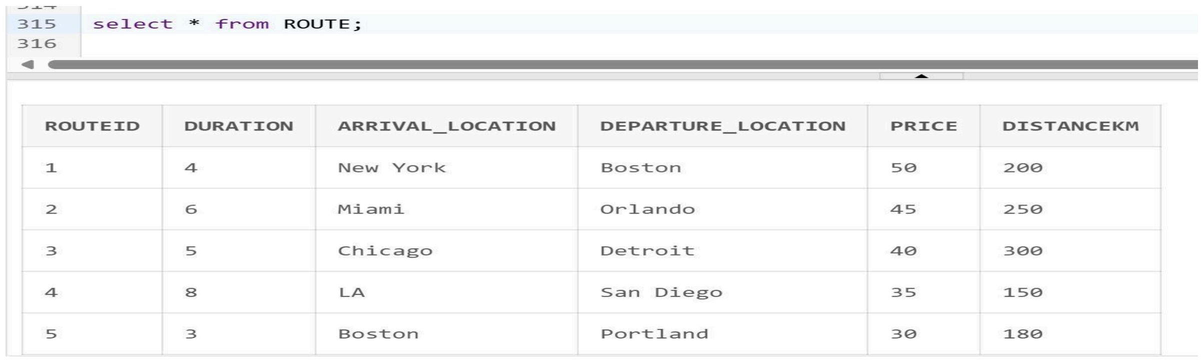
▪ INSERT INTO ROUTE VALUES (1, 4, 'New York', 'Boston', 50, 200);

▪ INSERT INTO ROUTE VALUES (2, 6, 'Miami', 'Orlando', 45, 250);

▪ INSERT INTO ROUTE VALUES (3, 5, 'Chicago', 'Detroit', 40, 300);

▪ INSERT INTO ROUTE VALUES (4, 8, 'LA', 'San Diego', 35, 150);

▪ INSERT INTO ROUTE VALUES (5, 3, 'Boston', 'Portland', 30, 180);



```
select * from ROUTE;
```

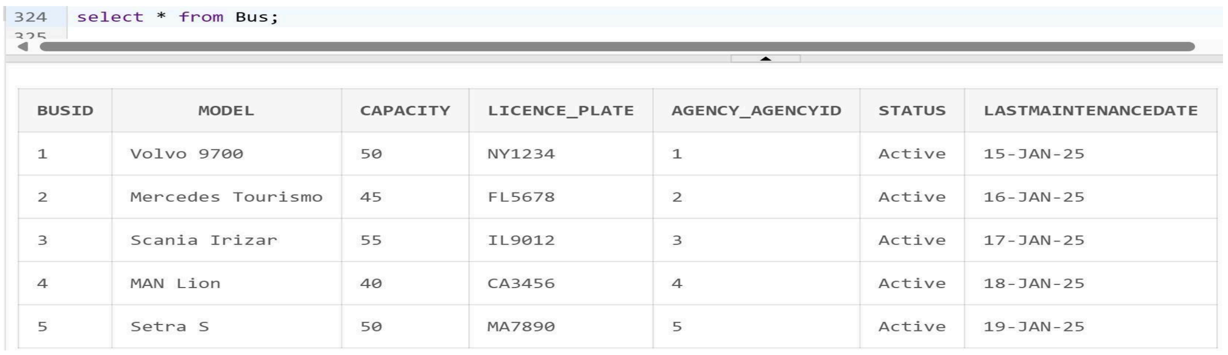
ROUTEID	DURATION	ARRIVAL_LOCATION	DEPARTURE_LOCATION	PRICE	DISTANCEKM
1	4	New York	Boston	50	200
2	6	Miami	Orlando	45	250
3	5	Chicago	Detroit	40	300
4	8	LA	San Diego	35	150
5	3	Boston	Portland	30	180

Fig: Table for ROUTE

- Insert into Bus

INSERT INTO Bus VALUES (1, 'Volvo 9700', 50, 'NY1234', 1, 'Active', TO_DATE ('2025-01-15', 'YYYY-MM-DD'));

INSERT INTO Bus VALUES (2, 'Mercedes Tourismo', 45, 'FL5678', 2, 'Active', TO_DATE ('2025-01-16', 'YYYY-MM-DD'));
INSERT INTO Bus VALUES (3, 'Scania Irizar', 55, 'IL9012', 3, 'Active', TO_DATE ('2025-01-17', 'YYYY-MM-DD'));
INSERT INTO Bus VALUES (4, 'MAN Lion', 40, 'CA3456', 4, 'Active', TO_DATE ('2025-01-18', 'YYYY-MM-DD'));
INSERT INTO Bus VALUES (5, 'Setra S', 50, 'MA7890', 5, 'Active', TO_DATE ('2025-01-19', 'YYYY-MM-DD'));



```
select * from Bus;
```

BUSID	MODEL	CAPACITY	LICENCE_PLATE	AGENCY_AGENCYID	STATUS	LASTMAINTENANCEDATE
1	Volvo 9700	50	NY1234	1	Active	15-JAN-25
2	Mercedes Tourismo	45	FL5678	2	Active	16-JAN-25
3	Scania Irizar	55	IL9012	3	Active	17-JAN-25
4	MAN Lion	40	CA3456	4	Active	18-JAN-25
5	Setra S	50	MA7890	5	Active	19-JAN-25

Fig: Table for BUS.

- Insert into Schedule

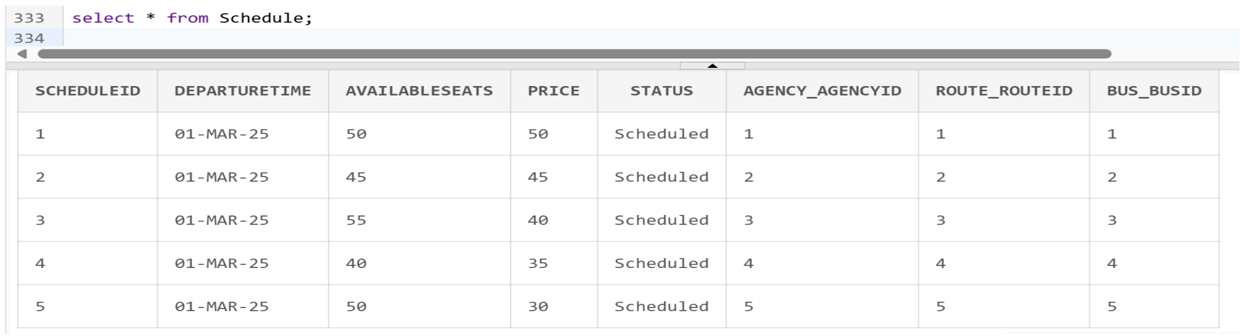
INSERT INTO Schedule VALUES (1, TO_DATE ('2025-03-01 08:00', 'YYYY-MM-DD HH24:MI'), 50, 50.000, 'Scheduled', 1, 1, 1);

INSERT INTO Schedule VALUES (2, TO_DATE ('2025-03-01 09:00', 'YYYY-MM-DD HH24:MI'), 45, 45.000, 'Scheduled', 2, 2, 2);

INSERT INTO Schedule VALUES (3, TO_DATE ('2025-03-01 10:00', 'YYYY-MM-DD HH24:MI'), 55, 40.000, 'Scheduled', 3, 3, 3);

INSERT INTO Schedule VALUES (4, TO_DATE ('2025-03-01 11:00', 'YYYY-MM-DD HH24:MI'), 40, 35.000, 'Scheduled', 4, 4, 4);

INSERT INTO Schedule VALUES (5, TO_DATE ('2025-03-01 12:00', 'YYYY-MM-DD HH24:MI'), 50, 30.000, 'Scheduled', 5, 5, 5);



```
select * from Schedule;
```

SCHEDULEID	DEPARTURETIME	AVAILABLESEATS	PRICE	STATUS	AGENCY_AGENCYID	ROUTE_ROUTEID	BUS_BUSID
1	01-MAR-25	50	50	Scheduled	1	1	1
2	01-MAR-25	45	45	Scheduled	2	2	2
3	01-MAR-25	55	40	Scheduled	3	3	3
4	01-MAR-25	40	35	Scheduled	4	4	4
5	01-MAR-25	50	30	Scheduled	5	5	5

fig: table for SCHEDULE

- Insert into PAYMENT

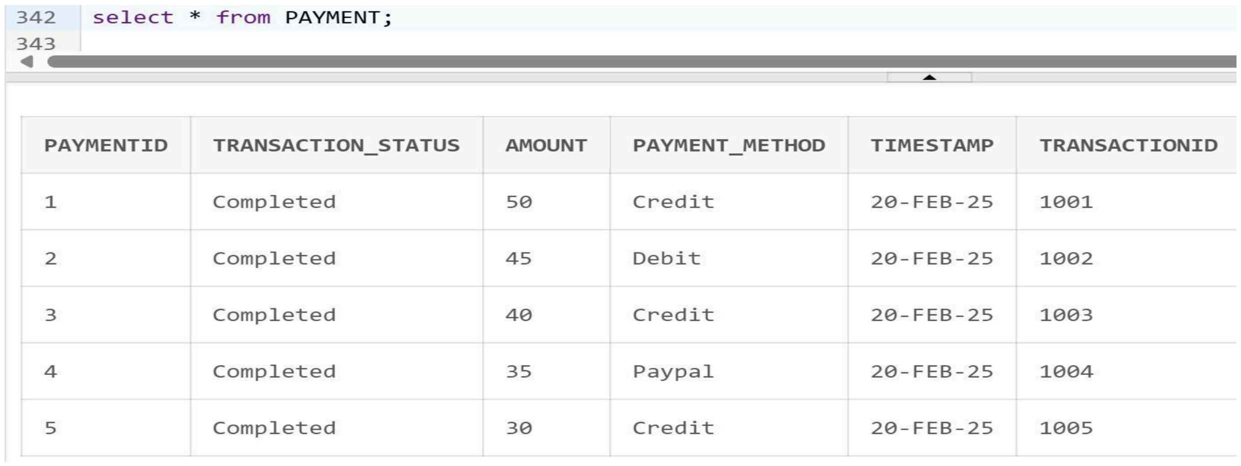
INSERT INTO PAYMENT VALUES (1, 'Completed', 50.00000, 'Credit', TO_DATE ('2025-02-20', 'YYYY-MM-DD'), 1001);

INSERT INTO PAYMENT VALUES (2, 'Completed', 45.00000, 'Debit', TO_DATE ('2025-02-20', 'YYYY-MM-DD'), 1002);

INSERT INTO PAYMENT VALUES (3, 'Completed', 40.00000, 'Credit', TO_DATE('2025-02-20', 'YYYY-MM-DD'), 1003);

INSERT INTO PAYMENT VALUES (4, 'Completed', 35.00000, 'Paypal', TO_DATE('2025-02-20', 'YYYY-MM-DD'), 1004);

INSERT INTO PAYMENT VALUES (5, 'Completed', 30.00000, 'Credit', TO_DATE('2025-02-20', 'YYYY-MM-DD'), 1005);



```
select * from PAYMENT;
```

PAYMENTID	TRANSACTION_STATUS	AMOUNT	PAYMENT_METHOD	TIMESTAMP	TRANSACTIONID
1	Completed	50	Credit	20-FEB-25	1001
2	Completed	45	Debit	20-FEB-25	1002
3	Completed	40	Credit	20-FEB-25	1003
4	Completed	35	Paypal	20-FEB-25	1004
5	Completed	30	Credit	20-FEB-25	1005

Fig: Table for PAYMENT

- Insert into BOOKING

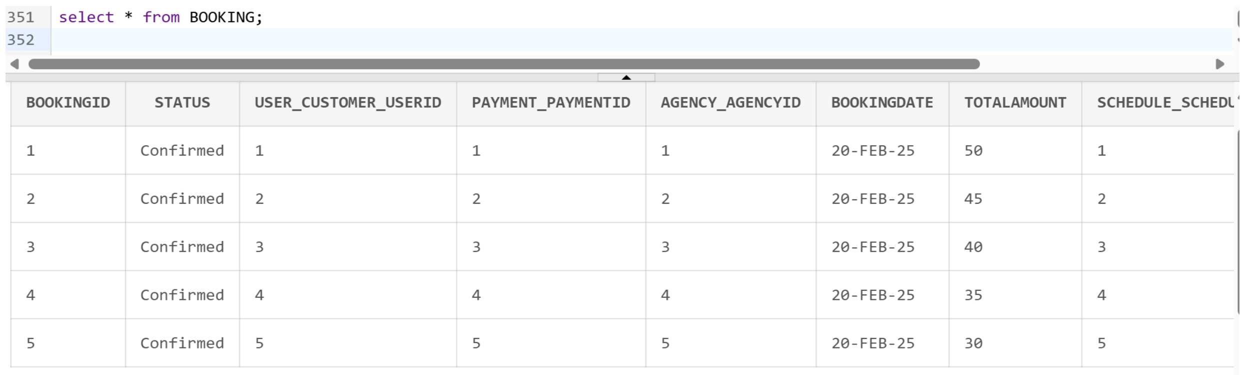
INSERT INTO BOOKING VALUES (1, 'Confirmed', 1, 1, 1, TO_DATE ('2025-02-20', 'YYYY-MM-DD'), 50, 1);

INSERT INTO BOOKING VALUES (2, 'Confirmed', 2, 2, 2, TO_DATE ('2025-02-20', 'YYYY-MM-DD'), 45, 2);

INSERT INTO BOOKING VALUES (3, 'Confirmed', 3, 3, 3, TO_DATE ('2025-02-20', 'YYYY-MM-DD'), 40, 3);

INSERT INTO BOOKING VALUES (4, 'Confirmed', 4, 4, 4, TO_DATE ('2025-02-20', 'YYYY-MM-DD'), 35, 4);

INSERT INTO BOOKING VALUES (5, 'Confirmed', 5, 5, 5, TO_DATE ('2025-02-20', 'YYYY-MM-DD'), 30, 5);



```
select * from BOOKING;
```

BOOKINGID	STATUS	USER_CUSTOMER_USERID	PAYMENT_PAYMENTID	AGENCY_AGENCYID	BOOKINGDATE	TOTALAMOUNT	SCHEDULE_SCHEDU
1	Confirmed	1	1	1	20-FEB-25	50	1
2	Confirmed	2	2	2	20-FEB-25	45	2
3	Confirmed	3	3	3	20-FEB-25	40	3
4	Confirmed	4	4	4	20-FEB-25	35	4
5	Confirmed	5	5	5	20-FEB-25	30	5

Fig: Table for BOOKING

B. SQL Queries to Retrieve Data

At least one query must contain a **JOIN operation**.

1. **Revenue Analysis per Agency:**

This query calculates the total revenue generated by each travel agency.

SELECT

A. Name AS Agency_Name,

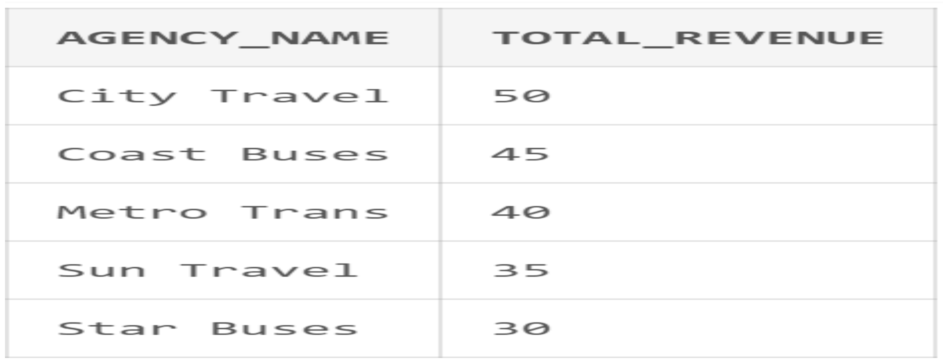
SUM(B.TotalAmount) AS Total_Revenue

FROM BOOKING B

JOIN AGENCY A ON B.AGENCY_AgencyID = A.AgencyID

GROUP BY A.Name

ORDER BY Total_Revenue DESC;



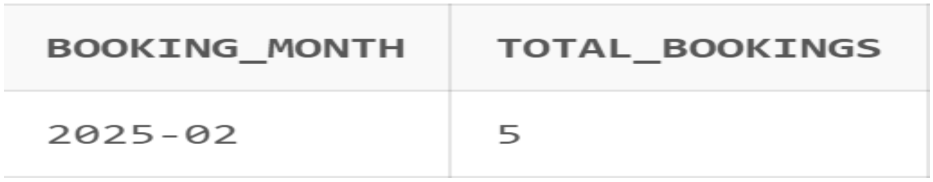
AGENCY_NAME	TOTAL_REVENUE
City Travel	50
Coast Buses	45
Metro Trans	40
Sun Travel	35
Star Buses	30

FIG: Total Revenue per Agency

2. **Peak Booking Period Analysis:**

This query finds the busiest month in terms of the number of bookings.

```
SELECT
    TO_CHAR(B.BookingDate, 'YYYY-MM') AS Booking_Month,
    COUNT(B.BookingID) AS Total_Bookings
FROM BOOKING B
GROUP BY TO_CHAR(B.BookingDate, 'YYYY-MM')
ORDER BY Total_Bookings DESC
FETCH FIRST 1 ROW ONLY;
```



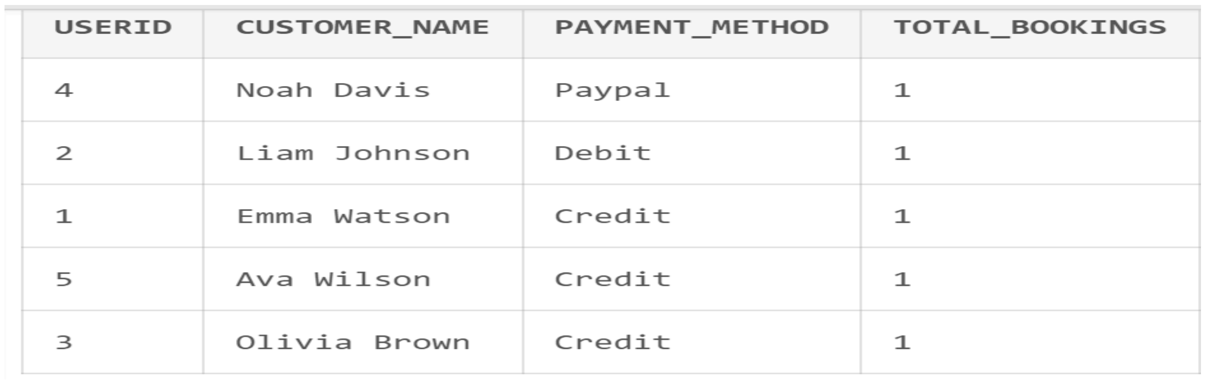
BOOKING_MONTH	TOTAL_BOOKINGS
2025-02	5

FIG: Busiest month in terms of the number of bookings.

3. **Customer Booking History & Preferred Payment Method:**

This query retrieves a list of customers along with their total number of bookings and preferred payment methods.

```
SELECT
▪   U.UserID,
▪   U.Name AS Customer_Name,
▪   U.payment_method,
▪       COUNT(B.BookingID) AS
Total_Bookings FROM USER_CUSTOMER U
LEFT JOIN BOOKING B ON U.UserID = B.USER_CUSTOMER_UserID
GROUP BY U.UserID, U.Name, U.payment_method
ORDER BY Total_Bookings DESC;
```



USERID	CUSTOMER_NAME	PAYMENT_METHOD	TOTAL_BOOKINGS
4	Noah Davis	Paypal	1
2	Liam Johnson	Debit	1
1	Emma Watson	Credit	1
5	Ava Wilson	Credit	1
3	Olivia Brown	Credit	1

Fig: Customer History

DATA GOVERNANCE IN THE TRANSPORT MANAGEMENT SYSTEM IN CAMEROON

Data governance in the transport management system in Cameroon will enhance the reliability, accuracy, and validity of our data. The emphasis here will be on using security measures to deny any unauthorized access and control in order to maintain data accuracy and consistency.

DATA SECURITY AND ETHICAL PRINCIPLES OF THIS PROJECT

In order to avoid unauthorized access, breaches, and cyber threats, there are some security measures that must be put in place and these measures are:

a. RIGHT TO CONTROL AND ACCESS DATA

- **Role-Based Access Control**: This allows only the authorized persons like the admins and those that issue tickets to have access and make changes to data.
- **Multi-Factor authentication**: Those making use of the service should be allowed to provide various means of authentication apart from their names to enable them to log in.
- **Encoding of users' information**: Identification numbers should be changed into a pattern that cannot be read easily through the use of private keys whereby only those with the decoding key can have access to data.

b. DATA ENCRYPTION

- **At Rest**: Keeps encoded booking information, commuters' details, and payments.
- **In Transit**: This makes use of Secure Sockets Layer(SSL) and Transport Layer Security (TLS) to protect the interface between service users and the system

c. AUDIT LOGGING AND MONITORING

- This helps to assess all the database entries and changes, and discovers dubious attempts.
- It makes use of the Intrusion Detection System (IDS) to keep an eye on improper access to the database.
- This will help to reveal who tampered with the database.

d. **INFORMATION BACKUP**

- To prevent passengers' information from being lost, there must be a regular backup of data which might be every week or every hour.
- For data to always be available, duplicates of it must be provided.
- There must be a disaster retrieval plan of action in a situation where there is cyber intrusion or system hacking.

e. **GUIDE AGAINST CYBER INTRUSION OR SYSTEM HACKING**

- In order to prevent cyber-attacks in the database of the transport system, firewalls and anti-malware devices must be made available.
- Distributed Denial of Service (DDos) must be made available to mitigate service disruptions.
- Frequent security check to discover and amend any data that is prone to attack

INTEGRITY AND DATA GOVERNANCE OF THIS PROJECT

The concept of integrity emphasizes the accuracy and consistency of data unless it is altered by a user who is authorized to have access to it.

- **REFERENTIAL INTEGRITY**
  - **Links relating to data**: Every booking should reference a particular line of travel, commuter, and remittance.
  - **Prevention of records that do not have any association in the database**: This simply means that no cancelled or expunged schedule or time should have a ticket.

- **PREVENTION OF PROHIBITED ALTERATIONS** :
- Data should be designed in such a way that it can only be read to ward-off any unwanted modification.
- Stamps and digital signatures should be made available in order to monitor changes.

CONCURRENCY CONTROL IN THE TRANSPORT MANAGEMENT SYSTEM IN CAMEROON

Concurrency control is an intricate issue that requires special concern not only in online transport booking but also in every aspect of transactions that are carried out on the internet. Concurrency control in online bookings and loggings in the Cameroon transport management system is very important to prevent data inconsistencies, double bookings, and unauthorized access. Here are several ways to ensure concurrency control:

Use of ACID Properties: In order to enable logging operations to remain consistent, ACID properties (Atomicity, consistency, isolation, and, Durability) must be applied

- **ATOMICITY**: Atomicity will be very crucial in this project. Under this principle, a transaction is either fully completed or not executed at all. Here are the reasons why Atomicity will be very important in the online booking in the transport management system in Cameroon:

    1. **Prevention of Partial Bookings**: when a user books a ticket online, there might be many operations taking place at the same time which may include checking of available seats, deducting payment, assigning a seat, and generating a ticket. So assuming any of the operations fails, Atomicity will ensure the entire transaction is rolled back. It will also prevent a situation where a seat is successfully booked without successful payment.
    2. **Data Integrity**: Atomicity will prevent the occurrence of power loss or server crash which will leave the database in this transport management system project in an inconsistent state. For example, showing that a seat has been booked, meanwhile no payment was received. Atomicity will ensure that these anomalies do not occur.
    3. **Prevention of double booking**: In a situation where different users tend to book the same seat at the same time, Atomicity will be used to ensure that once a seat is booked, it is immediately updated in the database hence, cannot be allocated to another person.
    4. An excellent booking procedure will minimize booking errors in the system. A user whose booking fails, can retry without dealing with unnecessary deductions and omission confirmations.

- **CONSISTENCY**: consistency is another important fundamental of ACID that enables a database to be valid both before and after transactions. In the online booking in this project, consistency of data will be very important. The following are various ways consistency can help maintain a healthy database:

    1. **Prevention of invalid or corrupt data**: consistency will ensure that only valid data are stored for instance, when a booking is made, the number of available seats should decrease, or in the event of ticket cancellation, the seat should be made available for another person. However, if an invalid attempt violates the booking rule (e.g., booking a seat on a non-existent bus), the system should nullify the transaction in order to maintain consistency.
    2. **Prevention of overbooking**: in a situation where there are plenty number of people trying to book the last available few seats at the same time, the system must ensure that no more seats are allocated than available, failure to do this, the system might allow more bookings than actual seats or users may get booking confirmations, but might be informed later that their seats are unavailable. Therefore, once a seat is booked, it must be updated immediately across all system components to avoid overbooking.
    3. **Maintenance of Accurate Payment and Ticketing Records**: A single booking by a service user might trigger other activities like deducting the fare from the user's account, assigning a seat, storing the passenger details, and generating a booking confirmation, however, the system must ensure that payment deduction is not successful if a ticket is not issued or ticket should not be generated without a successful payment. Furthermore, the system must be consistent in order to avoid issues like payment deduction without ticket issuance, or tickets being generated without payment.
    4. **Ensuring Data Integrity across Multiple Services**: Transport management system often interacts with other services like payment gateways, seat allocation systems, loyalty programs, and third-party booking platforms. If one of these services updates the database but fails to recognize the change, it can cause conflicts. Therefore, the system must ensure that all connected services reflect the correct booking status, avoiding disputes and discrepancies.

- **ISOLATION**: This is used to prevent concurrent transactions interfering with one another; hence, it does not allow data inconsistency and race conditions. In the online seat booking, isolation will play a crucial role when handling multiple users that might want to book their tickets simultaneously. The followings shows the relevance of isolation in this project:

- **Prohibition of more than one booking**: In a situation whereby different service users try to book the exact seat at a time, the system should be designed in such a way that the bookings should be one after the other. A seat should not be made available to two different users and it must follow a proper sequence. If there is no isolation, different people might be allocated the same seat and this can lead to overbooking and conflicts.

- **Prevention of dirty reads**: The system should be designed in a way that only the committed data should be saved or recorded in the database.

- **Reduces Phantom Read**: This happens if a transaction gets varieties of results for the same query as a result of another transaction being added up for the time being. Furthermore, Isolation in this system will enable all the bookings to be done independently which will help to avoid multiple bookings. With the use of good isolation methods in this transportation system in Cameroon, there will be accuracy, veracity, and the consistency of our data, and passengers will find it easy to make their bookings.

- **DURABILITY**: Durability as a very important tool, will enable the system to record and keep data safe so that they will not be lost. Hence, information about confirmed bookings, tickets, and payments are kept intact. However, the following are the importance of durability in this project:

- **Permanent recording of successful transactions**
  When a user's ticket booking becomes successful, durability will make sure that:

1. The seat he is allocated to is permanently recorded in the database

2. Information about payments and remittances are recorded and will not be lost.
3. A situation of system rebooting, resetting, and restarting, transaction confirmation must remain accessible.
4. Failure to apply isolation in this project, system mishap can bring about loss of bookings which will generate a lot of complaints and loss of money for the service user.

- **Prevention of loss of data when there is system breakdown** :
  When the server breaks down or power fails right after a ticket is booked, durability will:

1. Enable the database to keep the booking information.
2. Will enable the system to retrieve and proceed with the last successful bookings.
3. Commuters will not lose the seats allocated to them as a result of glitches.

- **Assures a risk-free process of payment**.
  Durability will ensure that:

1. Immediately a payment is made and it shows successful, the record will be immutable
2. Passengers are not at risk of being charged for duplicate bookings if the system restarts.
3. When there is a glitch, transaction logs will help to settle conflicts.
   Durability, therefore, must be properly utilized otherwise, passengers will constantly lose their money without getting any successful booking which will have an adverse effect on the reputation of the project.

- **Maintenance of data integrity in every system resetting**

  In this broad transport management project, effort will be made to ensure that

1. Every booking/or transaction that is successful before a system glitch is still valid and in a perfect condition.
2. The number of seats that are available are still correct even after the system has been restarted.
3. Booking report and information remain intact.

- **Tickets' recovery and validation**

Passengers should:

1. At any given time, recover their tickets.
2. At the onboarding point, be able to present their tickets for a check.
3. Retrieve their booking history even after many months.

IMPLEMENTATION OF DURABILITY IN THE ONLINE TICKET PURCHASE OF THIS TRANSPORT PROJECT

1. Write-Ahead Logging (WAL): This is a measure that is taken to ensure that before a transaction is recorded in the database, it should have been written to a log.
2. Duplication and backup: duplicate copies of information and booking transactions should be kept in a variety of servers so that data will not be lost.
3. Storage of data in the cloud: Information should be stored and accessed through the internet.
4. Committed transactions: Only successful transactions should be recorded permanently.

Conclusion

The Transport Management System provides a scalable and efficient solution for managing travel bookings in Cameroon. With a structured database design, normalization, and security considerations, the system ensures smooth operation while maintaining data integrity, security, and ethical standards.

References

Security in Databases

1. Connolly, T. and Begg, C. (2015) Database Systems: A Practical Approach to Design, Implementation, and Management. 6th edn. Harlow: Pearson Education.
2. Bertino, E. and Sandhu, R. (2005) 'Database Security—Concepts, Approaches, and Challenges', IEEE Transactions on Dependable and Secure Computing, 2(1), pp. 2-19. doi:10.1109/TDSC.2005.9.
3. Kim, H. and Solomon, M. (2018) Fundamentals of Information Systems Security. 3rd edn. Burlington: Jones & Bartlett Learning.
4. Stallings, W. (2020) Cryptography and Network Security: Principles and Practice. 8th edn. Upper Saddle River: Pearson Education.

Data Integrity in Databases

5. Elmasri, R. and Navathe, S. (2020) Fundamentals of Database Systems. 7th edn. Harlow: Pearson Education.
6. Codd, E. F. (1970) 'A Relational Model of Data for Large Shared Data Banks', Communications of the ACM, 13(6), pp. 377-387. doi:10.1145/362384.362685.
7. Date, C. J. (2019) *Database Design and Relational Theory: Normal Forms and All That Jazz.* 2nd edn. Sebastopol: O'Reilly Media.

Ethics in Data Management and AI

8. Floridi, L. (2013) *The Ethics of Information.* Oxford: Oxford University Press.
9. Solove, D. J. (2020) *The Digital Person: Technology and Privacy in the Information Age.* New York: NYU Press.
10. European Commission (2018) *General Data Protection Regulation (GDPR): Official Journal of the European Union.* Available at: https://gdpr-info.eu (Accessed: 5 March 2025).
11. Tavani, H. (2021) *Ethics and Technology: Controversies, Questions, and Strategies for Ethical Computing.* 6th edn. Hoboken: Wiley.
12. Boddington, P. (2017) *Towards a Code of Ethics for Artificial Intelligence.* Cham: Springer.