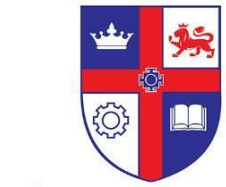


CC5051NI - Databases

60% Individual Coursework

2025-26 Autumn

Credit: 15 Semester Long Module

Student Name: Abiram Bhatta

London Met ID: 24046558

College ID: NP01AI4A240144

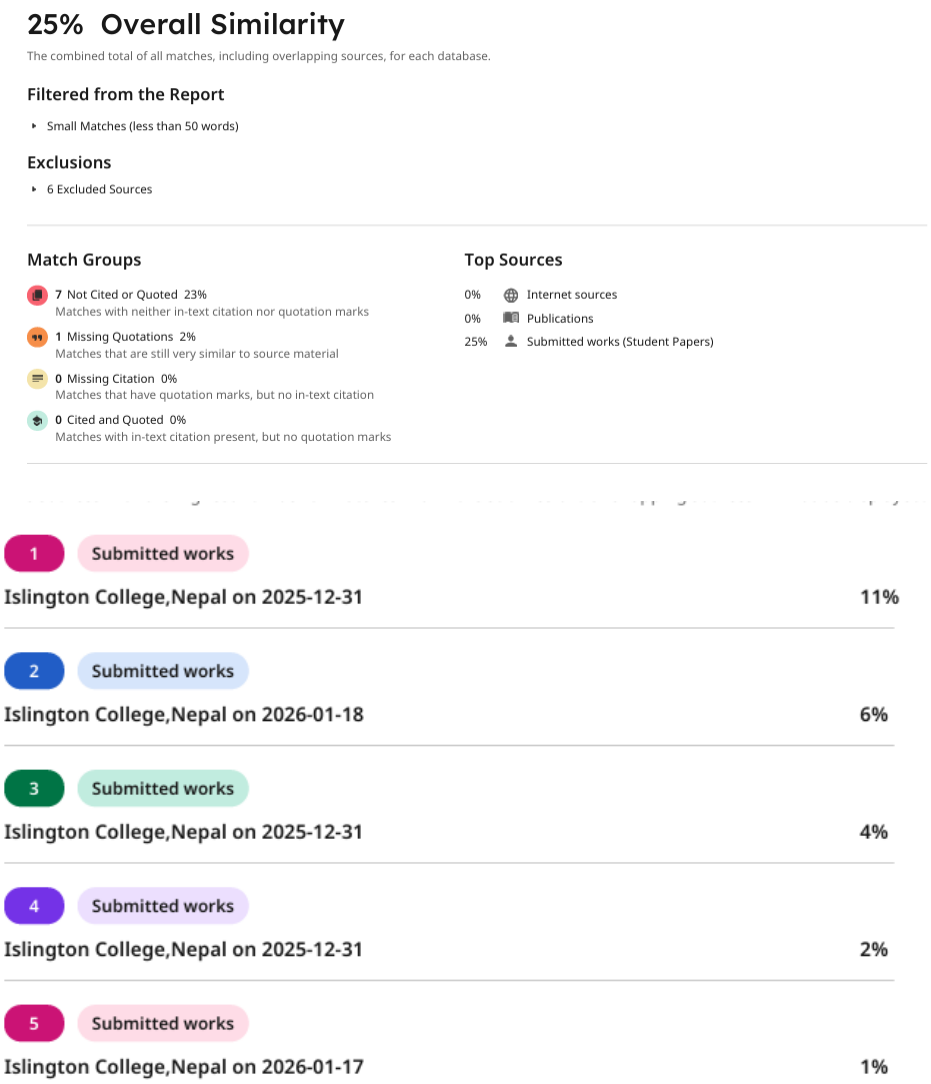
Assignment Due Date: Sunday, January 18, 2026

Assignment Submission Date: Sunday, January 18, 2026

Word Count: 3855

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Similarity Report



s

Table OF Contents

Table of Figures

Table of Tables

PART 1: INTRODUCTION

a) Introduction of the business and its forte

SajiloYatra is a newly established intercity travelling service provider in Nepal that was started with the idea of changing the transportation sector of the key centres including Kathmandu, Pokhara, Chitwan and Butwal. Inspired by the successful transportation systems that are built with the aspects of connectivity and reliability, SajiloYatra will fill the gap between the large urban regions and the surrounding regio`ns. The mission of the organization is to offer safe, comfortable, and punctual travels using modern technology to ease operations.

Organized fleet management and customer-focused service provision is the strength of SajiloYatra. In contrast with the traditional companies of transport where the problem of decentralization of management is frequent, SajiloYatra is more oriented towards the systematic route planning and distribution of resources. The company attaches importance to punctuality, safety of the passengers, and transparency, not trying to remove the typical pain points of the industry that include schedule ambiguity and overcrowding.

Figure 2: Sajilo Yatra Bus

Figure 1: Sajilo Yatra Logo

Mr Kiran Thapa the founder of Sajilo Yatra, plans on implementing a "Modern Online Booking and Management System" that will solve the existing issues in the industry, including poor booking systems and centralized management. The strong point of the database system of SajiloYatra is that it can effectively track and regulate the intricate relations between passengers, journeys, buses, and payments. This system will guarantee successful and convenient traveling experience as it will keep the precise records of routes, schedules, and staff assignments, and the problems such as overbooking, and unavailability of operations will be resolved eventually.

SajiloYatra's database system is essential because the current intercity transportation sector in Nepal suffers from inefficiencies, overbookings, and poor management. By implementing a modern online booking and management system, the company can streamline operations, track resources effectively, and ensure passengers experience safe, timely, and comfortable journeys. The system not only addresses existing industry problems but also enhances transparency,

reliability, and overall customer satisfaction, making it a necessary solution for the modern transport challenges.

b. Description of Current Business Activities and Operations

SajiloYatra is a company, which has a wide range of operations in the field of transportation and logistics at the moment.
Existing Business Operations:

- Intercity Bus Services: This service offers planned transport between large cities in Nepal based on a special fleet.
- Related Transportation Services: It provides services to book additional transport services other than bus services like flight tickets or rail ticket services through formed partnerships to further offer itself as a complete travel services provider.
- Vehicle on Hire: Providing the services of hiring buses during large groups or occasions.
- Parcel and Courier Logistics: The usage of bus network to deliver the small parcels and goods in-between the city terminals.
- Planned New Service: The main element of the expansion is a particular addition of the "Modern Online Booking and Management System" into the intercity bus service aimed at the further digitalization of the passenger journey.

System Requirement Operations: The new system should be able to effectively track and handle the key data interactions as follows:

- Trips and Routes: This is a management of departure times, arrival times and the route assignments.
- Passengers and Bookings: To avoid overbooking, passengers, types of particular booking classes
  and history of travels are recorded.
- Financial Transactions: Dealing with the ticket generation and accommodating payments, including instances where tickets are generated without payment being verified first.
- Resource Management: The availability of buses and their condition of maintenance as well as the assignment of staff to particular trips.

c. Business Rules

- One route can have multiple trips scheduled on it.
- Each trip must be associated with exactly one route.

- A trip is operated by exactly one bus.
- A bus can operate multiple trips over time.

- A trip can have multiple staff members assigned to it.
- A staff member can be assigned to multiple trips over time.

- A trip can have multiple passengers traveling on it.
- A passenger can travel on multiple trips.

- A passenger can make multiple bookings.
- Each booking is made by only one passenger.

- A route must have one or more buses assigned to it.
- A bus can be assigned to multiple routes over time.

- Each booking is for only one trip.
- A trip can have multiple bookings.

- A booking can generate one ticket.
- Each ticket is linked to only one booking.

- A ticket can have zero or one payment associated with it.
- Each payment is made for only one ticket.

- A passenger can have multiple preferences recorded.
- Each preference belongs to only one passenger.

- Each booking must have a confirmed status before a ticket is generated.
- Tickets are issued only for confirmed bookings.

- Payment may be recorded after ticket issuance.
- A ticket can exist without immediate payment.

- Booking class type must be one of Regular, Premium, or VIP.
- Each booking is assigned to one class type.

- Payment method must be one of Cash, Card, or Online.

- Each payment uses one payment method.

- Bus status must be one of Available, Maintenance, or Unavailable.
- Each bus has one status indicating its operational state at a time.

- Trip status must be one of Scheduled, Completed, or Cancelled.
- Each trip has one status indicating its current state.

d. Assumptions

- Each bus has a fixed seating capacity that determines the maximum number of passengers per trip.
- Passenger preferences are optional and may not be recorded for all passengers.
- Phone_Number is included as an attribute for Passenger to enable contact for booking confirmations and notifications.
- Each booking generates exactly one ticket.
- Booking class types are limited to Regular, Premium, and VIP.
- Staff roles include drivers and conductors, distinguished by a role attribute.
- A bus may be unavailable due to maintenance or operational reasons, tracked by an availability status attribute.
- Routes are defined by start city and end city attributes for operational clarity.
- Trip schedules include departure date, departure time, and arrival time.
- At least one driver and one conductor must be assigned to a bus for it to operate a trip.
- Payment methods include cash, card, and online payment options.
- Ticket fare is calculated based on route distance and booking class type.
- Each ticket is uniquely identified and includes fare and issue date information.

PART 2: Initial ERD

Table 1: Passenger Table

S. No.	Attribute Name	Data Type	Constraint
1	Passenger_ID	Number	Primary Key
2	Name	Character	
3	DOB	Date	
4	Phone_Number	Character	

5	Email	Character	
6	Address	Character	

Table 2: Booking Table

S. No.	Attribute Name	Data Type	Constraint
1	Booking_ID	Number	Primary Key
2	Class_Type	Character	
3	Booking_Date	Date	
4	Status	Character	

Table 3: Trip Table

S. No.	Attribute Name	Data Type	Constraint
1	Trip_ID	Number	Primary Key
2	Departure_Time	Character	
3	Arrival_Time	Character	
4	Date	Date	
5	Route_Name	Character	
6	Start_City	Character	
7	End_City	Character	

b. Entity Relationship Diagram

The initial ERD includes the entities Passenger, Booking and Trip with relationships, using Crow's foot Notation.

Figure 3: Initial ERD

PART 3: Normalization

Normalization is a systematic process to organize data in a database, reducing redundancy and anomalies. It involves applying normal forms based on functional dependencies.

a. Un-Normalized Form (UNF)

In UNF, all data is in a single relation with repeating groups, leading to anomalies such as redundancy, update anomalies, and insertion/deletion issues. The UNF is structured around Trip as the base entity, with repeating groups for passengers and staff assignments.

UNF: Trip(Trip_ID, Departure_Time, Arrival_Time, Date, Route_ID, Route_Name, Start_City, End_City, Distance, Bus_ID, Bus_Model, Bus_Capacity, Bus_Status, Registration_Number, {Staff_ID, Staff_Name, Role, Staff_Phone}, {Booking_ID, Passenger_ID, Passenger_Name, DOB, Phone_Number, Email, Address, LuggageRequirement, TravelNotification, Class_Type, Booking_Date, Status, Ticket_ID, Fare, Issue_Date, Payment_ID, Amount, Payment_Date, Method})

b. First Normal Form (1NF)

A relation is in 1NF if all its attributes contain only atomic values, and there are no repeating groups or multi-valued attributes. Each column holds a single value per row, and each row is unique.

Steps to First Normal Form (1NF)

To achieve 1NF, repeating groups should be removed by creating separate relations and carrying the primary key forward. Ensuring atomic values and no multi-valued attributes.

1. **Identify repeating groups:**
    - Staff details linked to trip
    - Booking details
2. **Remove repeating groups by creating new tables:**
    - Extracting Trip_Staff into separate table
    - Extracting Booking into separate table

3. **Add foreign keys where needed and focus on atomicity.**
   1NF Relations:
   - Trip (**Trip_ID**, Departure_Time, Arrival_Time, Date, Route_ID, Route_Name, Start_City, End_City, Distance, Bus_ID, Bus_Model, Bus_Capacity, Bus_Status, Registration_Number)

   - Trif_Staff (**Staff_ID, Trip_ID**, Staff_Name, Role, Staff_Phone)

   - Booking (**Booking_ID**, Trip_ID, Passenger_ID, Passenger_Name, DOB, Phone_Number, Email, Address, Luggage_Requirement, Travel_Notification, Class_Type, Booking_Date, Status, Ticket_ID, Booking_ID, Fare, Issue_Date, Payment_ID, Ticket_ID, Amount, Payment_Date, Method)

c. Second Normal Form (2NF)

Steps to Second Normal Form (2NF)

To achieve 2NF, tables with composite primary keys must be identified and check for partial dependencies. If any non-key attribute depends on only part of the composite key, it should be separated into a new relation.

1. Trip table: Route attributes (Route_Name, Start_City, End_City, Distance) depend on Route_ID, not Trip_ID.
Route_ID → Route_Name, Start_City, End_City, Distance

Bus attributes (Bus_Model, Bus_Capacity, Bus_Status, Registration_Number) depend on Bus_ID, not Trip_ID.
Bus_ID → Bus_Model, Bus_Capacity, Bus_Status, Registration_Number

2. Staff table: Staff attributes (Staff_Name, Role, Staff_Phone) depend only on Staff_ID (partial dependency).
Partial dependency: Staff_ID → Staff_Name, Role, Staff_Phone

3. Booking table: Passenger attributes (Passenger_Name, DOB, Phone_Number, Email, Address) depend on Passenger_ID, not Booking_ID.

Passenger_ID → Passenger_Name, DOB, Phone_Number, Email, Address

2NF Relations:

• Trip (**Trip_ID**, Route_ID, Bus_ID, Departure_Time, Arrival_Time, Date)

• Route (**Route_ID**, Route_Name, Start_City, End_City, Distance)

• Bus (**Bus_ID**, Bus_Model, Bus_Capacity, Bus_Status, Registration_Number)

• Staff (**Staff_ID**, Staff_Name, Role, Staff_Phone)

• Trip_Staff (**Trip_ID, Staff_ID**)

• Passenger (**Passenger_ID**, Passenger_Name, DOB, Phone_Number, Email, Address, Luggage_Requirement, Travel_Notification)

• Booking (**Booking_ID**, Trip_ID, Passenger_ID, Class_Type, Booking_Date, Status)

• Ticket (**Ticket_ID**, Booking_ID, Fare, Issue_Date)

• Payment (**Payment_ID**, Ticket_ID, Amount, Payment_Date, Method)

d. Third Normal Form (3NF)

A relation is in 3NF if it is in 2NF and no non-key attribute is transitively dependent on the primary key. This means eliminating transitive dependencies where a non-key attribute depends on another non-key attribute.

Steps to Third Normal Form (3NF)

1. Removal of Transitive dependency:

- Booking_ID → Ticket_ID → Payment attributes
- Payment attributes depend on Payment_ID, not on Booking_ID
- Payment remains in a separate relation

2. Bridge entities needs to be added for many to many realtions:

Route to Bus is a many-to-many relationship and requires RouteBus bridge table (as each route has one or more buses assigned to it)

Trip to Staff is a many-to-many relationship (handled by Trip_Staff)

3. Final verification of atomic attributes:

- All attributes are atomic and depend only on their primary keys

- Passenger (**Passenger_ID**, Passenger_Name, DOB, Phone_Number, Email, Address, Luggage_Requirement, Travel_Notification)
- Booking (**Booking_ID**, Passenger_ID, Trip_ID, Class_Type, Booking_Date, Status)
- Ticket (**Ticket_ID**, Booking_ID, Fare, Issue_Date)
- Payment (**Payment_ID**, Ticket_ID, Amount, Payment_Date, Method)
- Trip (**Trip_ID**, Route_ID, Bus_ID, Departure_Time, Arrival_Time, Date)
- Route (**Route_ID**, Route_Name, Start_City, End_City, Distance)
- Bus (**Bus_ID**, Bus_Model, Capacity, Status, Registration_Number)
- RouteBus (**Route_ID, Bus_ID**)
- Staff (**Staff_ID**, Staff_Name, Role, Staff_Phone)
- Trip_Staff (**Trip_ID, Staff_ID**)

Functional Dependencies for 3NF Verification:

- Passenger: **Passenger_ID** → Passenger_Name, DOB, Phone_Number, Email, Address, Luggage_Requirement, Travel_Notification
- Booking: **Booking_ID** → Passenger_ID, Trip_ID, Class_Type, Booking_Date, Status
- Ticket: **Ticket_ID** → Booking_ID, Fare, Issue_Date

- Payment: **Payment_ID** → Ticket_ID, Amount, Payment_Date, Method
- Trip: **Trip_ID** → Route_ID, Bus_ID, Departure_Time, Arrival_Time, Date
- Route: **Route_ID** → Route_Name, Start_City, End_City, Distance
- Bus: **Bus_ID** → Bus_Model, Capacity, Status, Registration_Number
- RouteBus: **Route_ID, Bus_ID** → (no non-key attributes)
- Staff: **Staff_ID** → Staff_Name, Role, Staff_Phone
- Trip_Staff: **Trip_ID, Staff_ID** → (no non-key attributes)

Part 4: Data Dictionary and Final ERD

a. Identification of Entities and Attributes

Table 1: Passenger

S.No.	Attribute Name	Data Type	Size	Constraints
1	Passenger_ID	NUMBER	10	Primary Key
2	Name	CHARACTER	50	Not Null
3	DOB	DATE		Not Null
4	Phone_Number	CHARACTER	15	Not Null
5	Email	CHARACTER	50	Unique
6	Address	CHARACTER	100	
7	Luggage_Requirement	CHARACTER	100	
8	Travel_Notification	CHARACTER	100	

Table 2: Booking

S.No.	Attribute Name	Data Type	Size	Constraints
1	Booking_ID	NUMBER	10	Primary Key
2	Passenger_ID	NUMBER	10	Foreign Key (PASSENGER), Not Null
3	Trip_ID	NUMBER	10	Foreign Key (TRIP), Not Null
4	Class_Type	CHARACTER	20	Check (Regular, Premium, VIP)
5	Booking_Date	DATE	-	Not Null
6	Status	CHARACTER	20	Check (Confirmed, Pending, Cancelled)

Table 3: Ticket

S.No.	Attribute Name	Data Type	Size	Constraints
1	Ticket_ID	NUMBER	10	Primary Key
2	Booking_ID	NUMBER	10	Foreign Key (BOOKING), Unique, Not Null
3	Fare	NUMBER	10	Not Null
4	Issue_Date	DATE	-	Not Null

Table 4: Payment

S.No.	Attribute Name	Data Type	Size	Constraints
1	Payment_ID	NUMBER	10	Primary Key
2	Ticket_ID	NUMBER	10	Foreign Key (TICKET), Unique, Not Null
3	Amount	NUMBER	10	Not Null
4	Date	DATE		Not Null

5	Method	CHARACTER	20	Check (Cash, Card, Online)

Table 5: Trip

S.No.	Attribute Name	Data Type	Size	Constraints
1	Trip_ID	NUMBER	10	Primary Key
2	Route_ID	NUMBER	10	Foreign Key (ROUTE), Not Null
3	Departure_Time	CHARACTER	10	Not Null
4	Arrival_Time	CHARACTER	10	Not Null
5	Date	DATE	-	Not Null
6	Bus_ID	NUMBER	10	Foreign Key (BUS), Not Null

Table 6: Route

S.No.	Attribute Name	Data Type	Size	Constraints
1	Route_ID	NUMBER	10	Primary Key
2	Name	CHARACTER	50	Not Null
3	Start_City	CHARACTER	30	Not Null
4	End_City	CHARACTER	30	Not Null
5	Distance	NUMBER	10	Not Null

Table 7: Bus

S.No.	Attribute Name	Data Type	Size	Constraints
1	Bus_ID	NUMBER	10	Primary Key
2	Model	CHARACTER	30	Not Null
3	Capacity	NUMBER	5	Not Null
4	Status	CHARACTER	20	Check (Available, Maintenance, Unavailable)
5	Registration_Number	CHARACTER	20	Unique, Not Null

Table 8: Staff

S.No.	Attribute Name	Data Type	Size	Constraints
1	Staff_ID	NUMBER	10	Primary Key
2	Name	CHARACTER	50	Not Null
3	Role	CHARACTER	20	Check (Driver, Conductor)
4	Phone_Number	CHARACTER	15	Not Null

Table 9: Route_Bus

S.No.	Attribute Name	Data Type	Size	Constraints
1	Route_ID	NUMBER	10	COMPOSITE KEY
2	Bus_ID	NUMBER	10	

Table 10: Trip_Staff

S.No.	Attribute Name	Data Type	Size	Constraints
1	Trip_ID	NUMBER	10	COMPOSITE KEY
2	Staff_ID	NUMBER	10	

b. Entity Relationship Diagram

The Final ERD represents the fully normalized database structure for the SajiloYatra Modern Online Booking and Management System. This diagram illustrates all entities derived from the Third Normal Form (3NF) normalization process, along with their relationships, cardinality, and modality.

The ERD consists of 10 entities: PASSENGER, BOOKING, TICKET, PAYMENT, TRIP, ROUTE, BUS, STAFF, ROUTE_BUS, and TRIP_STAFF. These entities are interconnected through various relationships that reflect the business rules and operational requirements of the intercity travel service.

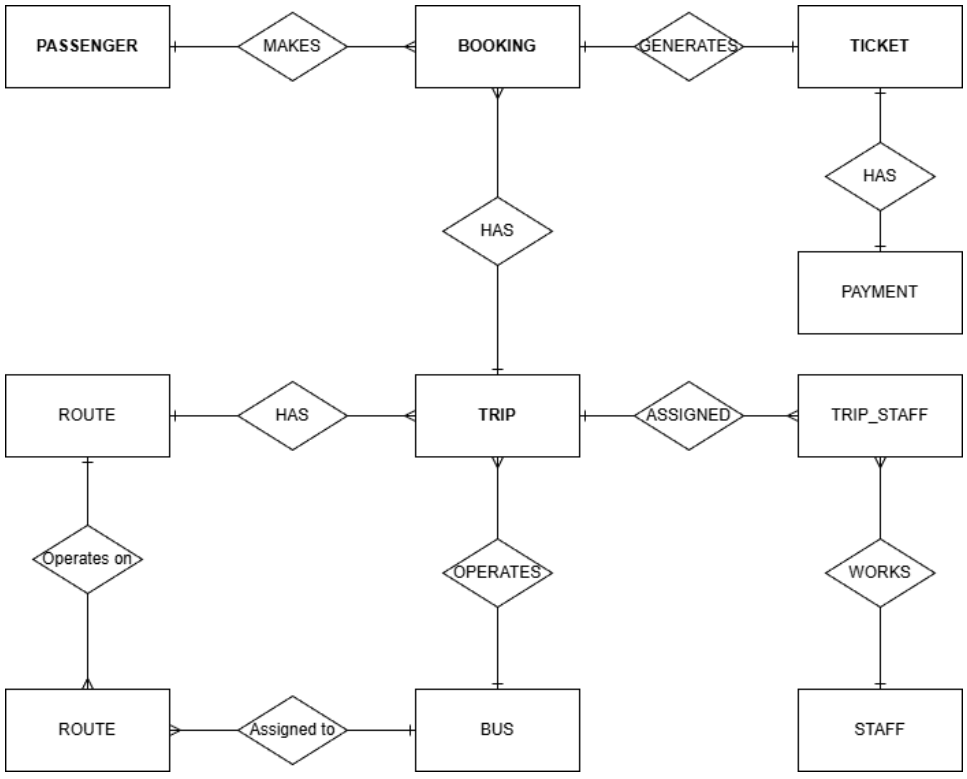


Figure 4: Final ERD

Part 5 Implementation

a. Create Commands

Passenger Table

```
SQL> CREATE TABLE PASSENGER (Passenger_ID NUMBER(10) PRIMARY KEY, Name VARCH
AR2(50) NOT NULL, DOB DATE NOT NULL, Phone_Number VARCHAR2(15) NOT NULL, Ema
il VARCHAR2(50) UNIQUE, Address VARCHAR2(100), Luggage_Requirement VARCHAR2(
100), Travel_Notification VARCHAR2(100));

Table created.
```

Figure 5: Creating Table Passenger

Route Table

```
SQL> CREATE TABLE ROUTE (Route_ID NUMBER(10) PRIMARY KEY, Name VARCHAR2(50)
NOT NULL, Start_City VARCHAR2(30) NOT NULL, End_City VARCHAR2(30) NOT NULL,
Distance NUMBER(10) NOT NULL);

Table created.
```

Figure 6: Creating Table Route

Bus Table

```
SQL> CREATE TABLE BUS (Bus_ID NUMBER(10) PRIMARY KEY, Model VARCHAR2(30) NOT
 NULL, Capacity NUMBER(5) NOT NULL, Status VARCHAR2(20) CHECK (Status IN ('A
vailable', 'Maintenance', 'Unavailable')), Registration_Number VARCHAR2(20)
UNIQUE NOT NULL);

Table created.
```

Figure 7: Creating Table Bus

Staff Table

```
SQL> CREATE TABLE STAFF (Staff_ID NUMBER(10) PRIMARY KEY, Name VARCHAR2(50)
NOT NULL, Role VARCHAR2(20) CHECK (Role IN ('Driver', 'Conductor')), Phone_N
umber VARCHAR2(15) NOT NULL);

Table created.
```

Figure 8: Creating Table Staff

Trip Table

```
SQL> CREATE TABLE TRIP (Trip_ID NUMBER(10) PRIMARY KEY, Route_ID NUMBER(10)
NOT NULL, Bus_ID NUMBER(10) NOT NULL, Departure_Time VARCHAR2(10) NOT NULL,
Arrival_Time VARCHAR2(10) NOT NULL, Date_Trip DATE NOT NULL, CONSTRAINT fk_t
rip_route FOREIGN KEY (Route_ID) REFERENCES ROUTE(Route_ID), CONSTRAINT fk_t
rip_bus FOREIGN KEY (Bus_ID) REFERENCES BUS(Bus_ID));

Table created.
```

Figure 9: Creating Table Trip

Booking Table

```
SQL> CREATE TABLE BOOKING (Booking_ID NUMBER(10) PRIMARY KEY, Passenger_ID
 NUMBER(10) NOT NULL, Trip_ID NUMBER(10) NOT NULL, Class_Type VARCHAR2(20)
 CHECK (Class_Type IN ('Regular', 'Premium', 'VIP')), Booking_Date DATE NO
T NULL, Status VARCHAR2(20) CHECK (Status IN ('Confirmed', 'Pending', 'Can
celled')), CONSTRAINT fk_booking_passenger FOREIGN KEY (Passenger_ID) REFE
RENCES PASSENGER(Passenger_ID), CONSTRAINT fk_booking_trip FOREIGN KEY (Tr
ip_ID) REFERENCES TRIP(Trip_ID));

Table created.
```

Figure 10: Creating Table Booking

Ticket Table

```
SQL> CREATE TABLE TICKET (Ticket_ID NUMBER(10) PRIMARY KEY, Booking_ID NUMBER
(10) UNIQUE NOT NULL, Fare NUMBER(10) NOT NULL, Issue_Date DATE NOT NULL, CON
STRAINT         fk_ticket_booking FOREIGN KEY (Booking_ID) REFERENCES BOOKING
(Booking_ID));

Table created.
```

Figure 11: Creating Table Ticket

Payment Table

```
SQL> CREATE TABLE PAYMENT (Payment_ID NUMBER(10) PRIMARY KEY, Ticket_ID NUMBE
R(10) UNIQUE NOT NULL, Amount NUMBER(10) NOT NULL, Date_Payment DATE NOT NULL
, Method VARCHAR2(20) CHECK (Method IN ('Cash', 'Card', 'Online')), CONSTRAIN
T fk_payment_ticket FOREIGN KEY (Ticket_ID) REFERENCES TICKET(Ticket_ID));

Table created.
```

Figure 12: Creating Table Payment

Route_Bus Table

```
SQL> CREATE TABLE ROUTE_BUS (Route_ID NUMBER(10), Bus_ID NUMBER(10), PRIMARY
 KEY (Route_ID, Bus_ID), CONSTRAINT fk_routebus_route FOREIGN KEY (Route_ID)
 REFERENCES ROUTE(Route_ID), CONSTRAINT fk_routebus_bus FOREIGN KEY (Bus_ID)
 REFERENCES BUS(Bus_ID));

Table created.
```

Figure 13: Creating Table Route_Bus

Trip_Staff Table

```
     CREATE TABLE TRIP_STAFF (Trip_ID NUMBER(10), Staff_ID NUMBER(10), PRIMARY KEY (
Trip_ID, Staff_ID), CONSTRAINT fk_tripstaff_trip FOREIGN KEY (Trip_ID) REFERENCES TR
IP(Trip_ID), CONSTRAINT fk_tripstaff_staff FOREIGN KEY (Staff_ID) REFERENCES STAFF(S
taff_ID));

Table created.
```

Figure 14: Creating Table Trip_Staff

b. Insert Commands

Passenger Values

```
SQL> INSERT ALL
  2    INTO PASSENGER VALUES (1, 'Ramesh Adhikari', TO_DATE('1990-05-15', 'Y
YYY-MM-DD'), '9841234567', 'ramesh.adhikari@email.com', 'Kathmandu, Nepal',
'One large suitcase', 'SMS notifications enabled')
  3    INTO PASSENGER VALUES (2, 'Sunita Karki', TO_DATE('1985-08-22', 'YYYY
-MM-DD'), '9851234568', 'sunita.karki@email.com', 'Pokhara, Nepal', 'Two sma
ll bags', 'Email notifications enabled')
  4    INTO PASSENGER VALUES (3, 'Bikash Gurung', TO_DATE('1995-03-10', 'YYY
Y-MM-DD'), '9861234569', 'bikash.gurung@email.com', 'Chitwan, Nepal', NULL,
NULL)
  5    INTO PASSENGER VALUES (4, 'Anjali Shrestha', TO_DATE('1988-11-30', 'Y
YYY-MM-DD'), '9871234570', 'anjali.shrestha@email.com', 'Butwal, Nepal', 'On
e backpack', 'SMS and Email enabled')
  6    INTO PASSENGER VALUES (5, 'Prakash Tamang', TO_DATE('1992-07-18', 'YY
YY-MM-DD'), '9881234571', 'prakash.tamang@email.com', 'Kathmandu, Nepal', NU
LL, 'SMS notifications enabled')
  7    INTO PASSENGER VALUES (6, 'Kopila Rai', TO_DATE('1998-01-25', 'YYYY-M
M-DD'), '9891234572', 'kopila.rai@email.com', 'Pokhara, Nepal', 'One medium
suitcase', NULL)
  8    INTO PASSENGER VALUES (7, 'Dipak Thapa', TO_DATE('1987-09-12', 'YYYY-
MM-DD'), '9801234573', 'dipak.thapa@email.com', NULL, 'Two large bags', 'Ema
il notifications enabled')
  9  SELECT * FROM dual;

7 rows created.
```

Figure 15: Inserting into Passenger

Route Values

```
SQL> INSERT ALL
  2    INTO ROUTE VALUES (101, 'Kathmandu-Pokhara Express', 'Kathmandu', 'Po
khara', 200)
  3    INTO ROUTE VALUES (102, 'Pokhara-Chitwan Direct', 'Pokhara', 'Chitwan
', 150)
  4    INTO ROUTE VALUES (103, 'Kathmandu-Chitwan Highway', 'Kathmandu', 'Ch
itwan', 180)
  5    INTO ROUTE VALUES (104, 'Chitwan-Butwal Route', 'Chitwan', 'Butwal',
120)
  6    INTO ROUTE VALUES (105, 'Kathmandu-Butwal Express', 'Kathmandu', 'But
wal', 280)
  7    INTO ROUTE VALUES (106, 'Pokhara-Butwal Direct', 'Pokhara', 'Butwal',
 160)
  8    INTO ROUTE VALUES (107, 'Butwal-Kathmandu Return', 'Butwal', 'Kathman
du', 280)
  9  SELECT * FROM dual;

7 rows created.
```

Figure 16: Inserting into Route

Bus Values

```
SQL> INSERT ALL
  2    INTO BUS VALUES (201, 'Volvo B11R', 45, 'Available', 'BA-1-PA-1234')
  3    INTO BUS VALUES (202, 'Tata Starbus', 40, 'Available', 'BA-2-KHA-5678
')
  4    INTO BUS VALUES (203, 'Ashok Leyland', 50, 'Maintenance', 'BA-3-GA-90
12')
  5    INTO BUS VALUES (204, 'Mahindra Tourister', 35, 'Available', 'BA-1-CH
A-3456')
  6    INTO BUS VALUES (205, 'Scania K360', 48, 'Available', 'BA-2-JA-7890')

  7    INTO BUS VALUES (206, 'Mercedes-Benz', 42, 'Unavailable', 'BA-3-TA-23
45')
  8    INTO BUS VALUES (207, 'Hino RK8', 38, 'Available', 'BA-1-THA-6789')
  9  SELECT * FROM dual;

7 rows created.
```

Figure 17: Inserting into Bus

Staff Values

```
SQL> INSERT ALL
  2    INTO STAFF VALUES (301, 'Keshav Poudel', 'Driver', '9841111111')
  3    INTO STAFF VALUES (302, 'Binod Basnet', 'Conductor', '9851111112')
  4    INTO STAFF VALUES (303, 'Nabin Khadka', 'Driver', '9861111113')
  5    INTO STAFF VALUES (304, 'Suman Bhattarai', 'Conductor', '9871111114')

  6    INTO STAFF VALUES (305, 'Rajan Limbu', 'Driver', '9881111115')
  7    INTO STAFF VALUES (306, 'Mohan Chhetri', 'Conductor', '9891111116')
  8    INTO STAFF VALUES (307, 'Arjun Magar', 'Driver', '9801111117')
  9  SELECT * FROM dual;

7 rows created.
```

Figure 18: Inserting into Staff

Trip Values

```
SQL> INSERT ALL
  2    INTO TRIP VALUES (401, 101, 201, '06:00 AM', '12:00 PM', TO_DATE('202
4-12-25', 'YYYY-MM-DD'))
  3    INTO TRIP VALUES (402, 102, 202, '07:00 AM', '11:30 AM', TO_DATE('202
4-12-25', 'YYYY-MM-DD'))
  4    INTO TRIP VALUES (403, 103, 204, '08:00 AM', '01:00 PM', TO_DATE('202
4-12-26', 'YYYY-MM-DD'))
  5    INTO TRIP VALUES (404, 104, 205, '09:00 AM', '12:30 PM', TO_DATE('202
4-12-26', 'YYYY-MM-DD'))
  6    INTO TRIP VALUES (405, 105, 207, '06:30 AM', '02:00 PM', TO_DATE('202
4-12-27', 'YYYY-MM-DD'))
  7    INTO TRIP VALUES (406, 106, 201, '10:00 AM', '03:00 PM', TO_DATE('202
4-12-27', 'YYYY-MM-DD'))
  8    INTO TRIP VALUES (407, 107, 202, '07:30 AM', '03:00 PM', TO_DATE('202
4-12-28', 'YYYY-MM-DD'))
  9  SELECT * FROM dual;

7 rows created.
```

Figure 19: Inserting into Trip

Booking Values

```
SQL> INSERT ALL
  2    INTO BOOKING VALUES (501, 1, 401, 'Premium', TO_DATE('2024-12-20', 'Y
YYY-MM-DD'), 'Confirmed')
  3    INTO BOOKING VALUES (502, 2, 402, 'Regular', TO_DATE('2024-12-21', 'Y
YYY-MM-DD'), 'Confirmed')
  4    INTO BOOKING VALUES (503, 3, 403, 'VIP', TO_DATE('2024-12-22', 'YYYY-
MM-DD'), 'Confirmed')
  5    INTO BOOKING VALUES (504, 4, 404, 'Regular', TO_DATE('2024-12-23', 'Y
YYY-MM-DD'), 'Pending')
  6    INTO BOOKING VALUES (505, 5, 405, 'Premium', TO_DATE('2024-12-23', 'Y
YYY-MM-DD'), 'Confirmed')
  7    INTO BOOKING VALUES (506, 6, 406, 'VIP', TO_DATE('2024-12-24', 'YYYY-
MM-DD'), 'Confirmed')
  8    INTO BOOKING VALUES (507, 7, 407, 'Regular', TO_DATE('2024-12-24', 'Y
YYY-MM-DD'), 'Cancelled')
  9  SELECT * FROM dual;

7 rows created.
```

Figure 20: Inserting into Booking

Ticket Values

```
SQL> INSERT ALL
  2    INTO TICKET VALUES (601, 501, 1500, TO_DATE('2024-12-20', 'YYYY-MM-DD
'))
  3    INTO TICKET VALUES (602, 502, 800, TO_DATE('2024-12-21', 'YYYY-MM-DD'
))
  4    INTO TICKET VALUES (603, 503, 2000, TO_DATE('2024-12-22', 'YYYY-MM-DD
'))
  5    INTO TICKET VALUES (604, 504, 700, TO_DATE('2024-12-23', 'YYYY-MM-DD'
))
  6    INTO TICKET VALUES (605, 505, 1800, TO_DATE('2024-12-23', 'YYYY-MM-DD
'))
  7    INTO TICKET VALUES (606, 506, 2200, TO_DATE('2024-12-24', 'YYYY-MM-DD
'))
  8    INTO TICKET VALUES (607, 507, 900, TO_DATE('2024-12-24', 'YYYY-MM-DD'
))
  9  SELECT * FROM dual;

7 rows created.
```

Figure 21: Inserting into Ticket

Payment Values

```
SQL> INSERT ALL
  2    INTO PAYMENT VALUES (701, 601, 1500, TO_DATE('2024-12-20', 'YYYY-MM-D
D'), 'Online')
  3    INTO PAYMENT VALUES (702, 602, 800, TO_DATE('2024-12-21', 'YYYY-MM-DD
'), 'Cash')
  4    INTO PAYMENT VALUES (703, 603, 2000, TO_DATE('2024-12-22', 'YYYY-MM-D
D'), 'Card')
  5    INTO PAYMENT VALUES (704, 605, 1800, TO_DATE('2024-12-23', 'YYYY-MM-D
D'), 'Online')
  6    INTO PAYMENT VALUES (705, 606, 2200, TO_DATE('2024-12-24', 'YYYY-MM-D
D'), 'Card')
  7    INTO PAYMENT VALUES (706, 607, 900, TO_DATE('2024-12-24', 'YYYY-MM-DD
'), 'Cash')
  8    INTO PAYMENT VALUES (707, 604, 700, TO_DATE('2024-12-25', 'YYYY-MM-DD
'), 'Online')
  9  SELECT * FROM dual;

7 rows created.
```

Figure 22: Inserting into Payment

Routes_Bus Values

```
SQL> INSERT ALL
  2    INTO ROUTE_BUS VALUES (101, 201)
  3    INTO ROUTE_BUS VALUES (102, 202)
  4    INTO ROUTE_BUS VALUES (103, 204)
  5    INTO ROUTE_BUS VALUES (104, 205)
  6    INTO ROUTE_BUS VALUES (105, 207)
  7    INTO ROUTE_BUS VALUES (106, 201)
  8    INTO ROUTE_BUS VALUES (107, 202)
  9  SELECT * FROM dual;

7 rows created.
```

Figure 23: Inserting into Route_Bus

Trip_Staff Values

```
SQL> INSERT ALL
  2    INTO TRIP_STAFF VALUES (401, 301)
  3    INTO TRIP_STAFF VALUES (401, 302)
  4    INTO TRIP_STAFF VALUES (402, 303)
  5    INTO TRIP_STAFF VALUES (402, 304)
  6    INTO TRIP_STAFF VALUES (403, 305)
  7    INTO TRIP_STAFF VALUES (403, 306)
  8    INTO TRIP_STAFF VALUES (404, 307)
  9    INTO TRIP_STAFF VALUES (405, 301)
 10    INTO TRIP_STAFF VALUES (405, 302)
 11  SELECT * FROM dual;

9 rows created.
```

Figure 24: Inserting into Trip_Staff

c. Select Commands

Select Passenger

```
SQL> SELECT * FROM PASSENGER ORDER BY Passenger_ID;

PASSENGER_ID NAME           DOB       PHONE_NUMBER    EMAIL                    ADDRESS                  LUGGAGE_REQUIREMENT  TRAVEL_NOTIFICATION
------------ -------------- --------- --------------- ------------------------ ------------------------ -------------------- --------------------
           1 Ramesh Adhikari 15-MAY-90 9841234567      ramesh.adhikari@email.com Kathmandu, Nepal         One large suitcase   SMS notifications en
                                                                                                                            abled

           2 Sunita Karki    22-AUG-85 9851234568      sunita.karki@email.com    Pokhara, Nepal           Two small bags       Email notifications
                                                                                                                            enabled

           3 Bikash Gurung   10-MAR-95 9861234569      bikash.gurung@email.com   Chitwan, Nepal
           4 Anjali Shrestha 30-NOV-88 9871234570      anjali.shrestha@email.com Butwal, Nepal            One backpack         SMS and Email enable
                                                                                                                            d

           5 Prakash Tamang  18-JUL-92 9881234571      prakash.tamang@email.com  Kathmandu, Nepal                              SMS notifications en
                                                                                                                            abled

           6 Kopila Rai      25-JAN-98 9891234572      kopila.rai@email.com      Pokhara, Nepal           One medium suitcase
           7 Dipak Thapa     12-SEP-87 9801234573      dipak.thapa@email.com                              Two large bags       Email notifications
                                                                                                                            enabled

7 rows selected.
```

Figure 25: Select from Passenger

Select Route

```
SQL> SELECT * FROM ROUTE ORDER BY Route_ID;

  ROUTE_ID NAME            START_CITY                     END_CITY                       DISTANCE
---------- --------------- ------------------------------ ------------------------------ ----------
       101 Kathmandu-Pokha Kathmandu                      Pokhara                               200
           ra Express

       102 Pokhara-Chitwan Pokhara                        Chitwan                               150
            Direct

       103 Kathmandu-Chitw Kathmandu                      Chitwan                               180
           an Highway

       104 Chitwan-Butwal  Chitwan                        Butwal                                120
           Route

       105 Kathmandu-Butwa Kathmandu                      Butwal                                280
           l Express

       106 Pokhara-Butwal  Pokhara                        Butwal                                160
           Direct

       107 Butwal-Kathmand Butwal                         Kathmandu                             280
           u Return


7 rows selected.
```

Figure 26: Select from Route

Select Bus

```
SQL> SELECT * FROM BUS ORDER BY Bus_ID;

    BUS_ID MODEL                          CAPACITY STATUS             REGISTRATION_NUMBER
---------- ------------------------------ ---------- -------------------- --------------------
       201 Volvo B11R                             45 Available            BA-1-PA-1234
       202 Tata Starbus                           40 Available            BA-2-KHA-5678
       203 Ashok Leyland                          50 Maintenance          BA-3-GA-9012
       204 Mahindra Tourister                     35 Available            BA-1-CHA-3456
       205 Scania K360                            48 Available            BA-2-JA-7890
       206 Mercedes-Benz                          42 Unavailable          BA-3-TA-2345
       207 Hino RK8                               38 Available            BA-1-THA-6789

7 rows selected.
```

Figure 27: Select from Bus

Select Staff

```
SQL> SELECT * FROM STAFF ORDER BY Staff_ID;

  STAFF_ID NAME            ROLE               PHONE_NUMBER
---------- --------------- ------------------ ---------------
       301 Keshav Poudel   Driver             9841111111
       302 Binod Basnet    Conductor          9851111112
       303 Nabin Khadka    Driver             9861111113
       304 Suman Bhattarai Conductor          9871111114
       305 Rajan Limbu     Driver             9881111115
       306 Mohan Chhetri   Conductor          9891111116
       307 Arjun Magar     Driver             9801111117

7 rows selected.
```

Figure 28: Select from Staff

Select Trip

```
SQL> SELECT * FROM TRIP ORDER BY Trip_ID;

   TRIP_ID   ROUTE_ID     BUS_ID DEPARTURE_ ARRIVAL_TI DATE_TRIP
---------- ---------- ---------- ---------- ---------- ---------
       401        101        201 06:00 AM   12:00 PM   25-DEC-24
       402        102        202 07:00 AM   11:30 AM   25-DEC-24
       403        103        204 08:00 AM   01:00 PM   26-DEC-24
       404        104        205 09:00 AM   12:30 PM   26-DEC-24
       405        105        207 06:30 AM   02:00 PM   27-DEC-24
       406        106        201 10:00 AM   03:00 PM   27-DEC-24
       407        107        202 07:30 AM   03:00 PM   28-DEC-24

7 rows selected.
```

Figure 29: Select from Trip

Select Booking

```
SQL> SELECT * FROM BOOKING ORDER BY Booking_ID;

BOOKING_ID PASSENGER_ID    TRIP_ID CLASS_TYPE          BOOKING_D STATUS
---------- ------------ ---------- ------------------- --------- --------------------
       501            1        401 Premium             20-DEC-24 Confirmed
       502            2        402 Regular             21-DEC-24 Confirmed
       503            3        403 VIP                 22-DEC-24 Confirmed
       504            4        404 Regular             23-DEC-24 Pending
       505            5        405 Premium             23-DEC-24 Confirmed
       506            6        406 VIP                 24-DEC-24 Confirmed
       507            7        407 Regular             24-DEC-24 Cancelled

7 rows selected.
```

Figure 30: Select from Booking

Select Ticket

```
SQL> SELECT * FROM TICKET ORDER BY Ticket_ID;

 TICKET_ID BOOKING_ID       FARE ISSUE_DAT
---------- ---------- ---------- ---------
       601        501       1500 20-DEC-24
       602        502        800 21-DEC-24
       603        503       2000 22-DEC-24
       604        504        700 23-DEC-24
       605        505       1800 23-DEC-24
       606        506       2200 24-DEC-24
       607        507        900 24-DEC-24

7 rows selected.
```

Figure 31: Select from Ticket

Select Payment

```
SQL> SELECT * FROM PAYMENT ORDER BY Payment_ID;

PAYMENT_ID  TICKET_ID     AMOUNT DATE_PAYM METHOD
---------- ---------- ---------- --------- --------------------
       701        601       1500 20-DEC-24 Online
       702        602        800 21-DEC-24 Cash
       703        603       2000 22-DEC-24 Card
       704        605       1800 23-DEC-24 Online
       705        606       2200 24-DEC-24 Card
       706        607        900 24-DEC-24 Cash
       707        604        700 25-DEC-24 Online

7 rows selected.
```

Figure 32: Select from Payment

Select Route_Bus

```
SQL> SELECT * FROM ROUTE_BUS;

  ROUTE_ID     BUS_ID
---------- ----------
       101        201
       102        202
       103        204
       104        205
       105        207
       106        201
       107        202

7 rows selected.
```

Figure 33: Select from Route _Bus

Select Trip_Staff

```
SQL> SELECT * FROM TRIP_STAFF;

   TRIP_ID   STAFF_ID
---------- ----------
       401        301
       401        302
       402        303
       402        304
       403        305
       403        306
       404        307
       405        301
       405        302

9 rows selected.
```

Figure 34: Select from Trip_Staff

Part 6. Queries

a. Information query

1. List all routes along with the total number of trips scheduled on each route.

```
SQL> SELECT r.Name AS Route_Name, COUNT(t.Trip_ID) AS Total_Trips FROM ROUTE r LEFT JOIN
 TRIP t ON r.Route_ID = t.Route_ID GROUP BY r.Name ORDER BY Total_Trips DESC;

ROUTE_NAME                     TOTAL_TRIPS
------------------------------ -----------
Pokhara-Chitwan Direct                   1
Kathmandu-Chitwan Highway                1
Chitwan-Butwal Route                     1
Pokhara-Butwal Direct                    1
Butwal-Kathmandu Return                  1
Kathmandu-Pokhara Express                1
Kathmandu-Butwal Express                 1
```

Figure 35: Information Querry 1

2. List all trips departing from “Pokhara” with their departure time and assigned route name.

```
SQL> SELECT t.Trip_ID, t.Departure_Time, r.Name AS Route_Name FROM TRIP t JOIN ROUTE r
ON t.Route_ID = r.Route_ID WHERE r.Start_City = 'Pokhara' ORDER BY t.Departure_Time;

    TRIP_ID DEPARTURE_ ROUTE_NAME
---------- --------- --------------------------------
       402 07:00 AM   Pokhara-Chitwan Direct
       406 10:00 AM   Pokhara-Butwal Direct
```

Figure 36: Information Query 2

3. List the names of passengers whose names start with 'A', along with the total number of trips they have booked.

```
SQL> SELECT p.Name, COUNT(b.Booking_ID) AS Total_Trips FROM PASSENGER p JOIN BOOKING
b ON p.Passenger_ID = b.Passenger_ID GROUP BY p.Name HAVING p.Name LIKE 'A%' ORDER BY
 Total_Trips DESC;

NAME                           TOTAL_TRIPS
------------------------------ -----------
Anjali Shrestha                          1
```

Figure 37: Information Query 3

4. List all passengers who have booked more than 2 trips in December 2025.

```
SQL> SELECT p.Name, COUNT(b.Booking_ID) AS Trips_In_Dec FROM PASSENGER p JOIN BOOKING
b ON p.Passenger_ID = b.Passenger_ID JOIN TRIP t ON b.Trip_ID = t.Trip_ID WHERE t.Date
_Trip BETWEEN TO_DATE('2025-12-01','YYYY-MM-DD') AND TO_DATE('2025-12-31','YYYY-MM-DD'
) GROUP BY p.Name HAVING COUNT(b.Booking_ID) > 2 ORDER BY Trips_In_Dec DESC;

NAME                           TRIPS_IN_DEC
------------------------------ ------------
Ramesh Adhikari                           3
```

Figure 38: Information Query 4

5. List all tickets issued in January 2025 with passenger names, trip IDs, and payment status (‘Paid’ or ‘Unpaid’)

```
SQL> SELECT p.Name, b.Trip_ID, CASE WHEN pay.Payment_ID IS NULL THEN 'Unpaid' ELSE 'Paid'
END AS Payment_Status FROM TICKET t JOIN BOOKING b ON t.Booking_ID = b.Booking_ID JOIN PAS
SENGER p ON b.Passenger_ID = p.Passenger_ID LEFT JOIN PAYMENT pay ON t.Ticket_ID = pay.Tic
ket_ID WHERE t.Issue_Date BETWEEN TO_DATE('2025-01-01','YYYY-MM-DD') AND TO_DATE('2025-01-
31','YYYY-MM-DD') ORDER BY t.Issue_Date;

NAME                           TRIP_ID PAYMEN
------------------------------ ---------- ------
Ramesh Adhikari                       401 Paid
Ramesh Adhikari                       402 Paid
Ramesh Adhikari                       403 Paid
Anjali Shrestha                       407 Paid
Bikash Gurung                         405 Paid
Bikash Gurung                         406 Paid
Sunita Karki                          404 Paid
```

Figure 39: Information Query 5

b Transaction query

1. Find the trip with the latest departure time.

```
SQL> SELECT * FROM (SELECT * FROM TRIP ORDER BY Departure_Time DESC) WHERE ROWNUM = 1;

    TRIP_ID    ROUTE_ID      BUS_ID DEPARTURE_ ARRIVAL_TI DATE_TRIP
---------- ---------- ---------- ---------- ---------- ---------
       406         106         201 10:00 AM   03:00 PM   25-DEC-25
```

Figure 40: Transaction Query 1

2. List the top three passengers who have booked the most trips.

```
SQL> SELECT * FROM (SELECT p.Name, COUNT(b.Booking_ID) AS Total_Bookings FROM PASSENGER
p JOIN BOOKING b ON p.Passenger_ID = b.Passenger_ID GROUP BY p.Name ORDER BY Total_Booki
ngs DESC) WHERE ROWNUM <= 3;

NAME                          TOTAL_BOOKINGS
----------------------------- --------------
Ramesh Adhikari                            3
Bikash Gurung                              2
Anjali Shrestha                            1
```

Figure 41: Transaction Query 2

3. List all passengers who have made payments greater than the average payment amount, along with the total amount they have paid.

```
SQL> SELECT p.Name, SUM(pay.Amount) AS Total_Paid FROM PASSENGER p JOIN BOOKING b ON
p.Passenger_ID = b.Passenger_ID JOIN TICKET t ON b.Booking_ID = t.Booking_ID JOIN PAY
MENT pay ON t.Ticket_ID = pay.Ticket_ID GROUP BY p.Name HAVING SUM(pay.Amount) > (SEL
ECT AVG(Amount) FROM PAYMENT) ORDER BY Total_Paid DESC;

NAME                          TOTAL_PAID
----------------------------- ----------
Ramesh Adhikari                     4300
Bikash Gurung                       4000
```

Figure 42: Transaction Query 3

4. Calculate total revenue generated per route along with the number of trips operated on each route.

```
SQL> SELECT r.Name AS Route_Name, COUNT(t.Trip_ID) AS Total_Trips, NVL(SUM(pay.Amount
),0) AS Total_Revenue FROM ROUTE r JOIN TRIP t ON r.Route_ID = t.Route_ID JOIN BOOKIN
G b ON t.Trip_ID = b.Trip_ID JOIN TICKET ti ON b.Booking_ID = ti.Booking_ID LEFT JOIN
 PAYMENT pay ON ti.Ticket_ID = pay.Ticket_ID GROUP BY r.Name ORDER BY Total_Revenue D
ESC;

ROUTE_NAME                     TOTAL_TRIPS TOTAL_REVENUE
------------------------------ ----------- -------------
Pokhara-Butwal Direct                    1          2200
Kathmandu-Chitwan Highway                1          2000
Kathmandu-Butwal Express                 1          1800
Kathmandu-Pokhara Express                1          1500
Butwal-Kathmandu Return                  1           900
Pokhara-Chitwan Direct                   1           800
Chitwan-Butwal Route                     1           700
```

Figure 43: Transaction Query 4

5. List all trips where the number of booked tickets is less than 50% of the bus capacity. **(Assume bus capacity is available in Bus table linked through Route.)**

```
SQL> SELECT t.Trip_ID, r.Name AS Route_Name, b.Capacity, COUNT(bo.Booking_ID) AS Tick
ets_Booked FROM TRIP t JOIN ROUTE r ON t.Route_ID = r.Route_ID JOIN BUS b ON t.Bus_ID
 = b.Bus_ID LEFT JOIN BOOKING bo ON t.Trip_ID = bo.Trip_ID GROUP BY t.Trip_ID, r.Name
, b.Capacity HAVING COUNT(bo.Booking_ID) < 0.5 * b.Capacity;

    TRIP_ID ROUTE_NAME                         CAPACITY TICKETS_BOOKED
----------- ---------------------------------- ---------- --------------
        402 Pokhara-Chitwan Direct                     40              1
        401 Kathmandu-Pokhara Express                  45              1
        404 Chitwan-Butwal Route                       48              1
        403 Kathmandu-Chitwan Highway                  35              1
        406 Pokhara-Butwal Direct                      45              1
        407 Butwal-Kathmandu Return                    40              1
        405 Kathmandu-Butwal Express                   38              1
```

Figure 44: Transaction Query 5

Part 7. Critical Evaluation

a. Critical Analysis of Module, Its Usage, and Correlation with other subjects.

The database module adopted to SajiloYatra illustrates the use of the concept of relational database to address real business issues, in this case, in the travel industry at intercity. The system allows the management of both simple and complicated processes like scheduling trips or booking passengers and tracking payments, etc. by involving structured Passengers, Routes, Buses, Staff, Trips, Bookings, Tickets, Payments, and their relation to each other.

This module brings out the significance of normalization (1NF, 2NF, 3NF) in order to minimize redundancy and provide data integrity. The use of foreign keys, composite keys and constraints impose proper relationship between entities as one being the relationship between Tickets and Bookings and the other between Trips and Bookings. It also shows how to manage transactions as evidenced in the payment and issuing tickets.

The module incorporates ideas of other courses, such as:

- Software Engineering: Design A database schema is created based on business rules and requirements.
- Programming/DBMS Writing SQL commands, creating constraints, and handling relational data.
- Data Analysis: Allowing queries to create useful business information, such as revenue per route or passenger booking trends.
- Information Systems: Utilization of system design principles in order to facilitate operational performance and user experience.

Overall, the module has unified broad theoretical knowledge on databases with practical purposes, which highlights the significance of relational databases to business operations.

b. Critical Reflection of Coursework.

The SajiloYatra coursework offered a practical scenario, which assisted in acquiring skills in database design, normalization and the SQL query writing skills. The systematic process of identifying business rules, table development, data entry and sophisticated querying is a reflection of the real world projects.

Strengths:

- Business rules and assumptions were clearly identified.
- Introduction of correct relational tables and constraints and normalization.
- Ability to generate meaningful queries for business decisions

Challenges:

- Dealing with consistency of data between unique key constraints and foreign key constraints, particularly implementing the update and add test-data processes.
- To normalize the database following the proper steps and making sure its gives correct result and redundancy is minimized.
- Assuring queries to give the desired results using very few sample data, then inserts and updates are to be planned well.
-

Learning Outcomes:

- Better knowledge of normalization and relational database design.
- Practical understanding of Oracle SQL including joins, aggregate functions, and CASE statements.
- An understanding of the role of a database in business processes and decision making in transport management.

To sum up, this coursework has enhanced theoretical and practical skills that prove the importance of databases in the management of complex and real-life systems such as SajiloYatra.

References

Codecademy Team, 2018. *What is Normalization in DBMS? Explained with Examples.* [Online]
Available at: https://www.codecademy.com/article/normalization-in-dbms
[Accessed 16th December 2025].

Khaptad Yatayat, 2024. *About Us.* [Online]
Available at: https://khaptadyatayat.com.np/about-us
[Accessed 16th December 2025].

LucidCharts, 2020. *er-diagrams.* [Online]
Available at: https://www.lucidchart.com/pages/er-diagrams
[Accessed 16th December 2025].