Table of Contents

7. SQL Queries and Demonstrations

- Traveler List at a Particular Location
- Total Amount of Business a Hotel Receives in a Certain Time Frame
- List of Hotels with Available Rooms

8. Test Plan for Travel Classics

- Scope
- Testing Environment
- Test Cases
- Dependencies
- Execution Strategy
- Strategies for Risk and Mitigation
- Test Result

9. References

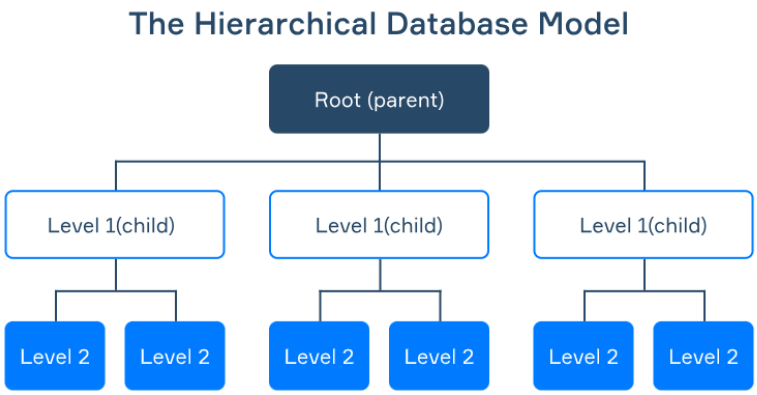
1.Database Design for Travel Classics

Introduction

A conceptual representation of data structures used to describe data, relationships between various types of data, and limitations on the data is called a data model. Because they make it easier to comprehend the different facets of organising and manipulating data within a database system, data models are crucial to database architecture.

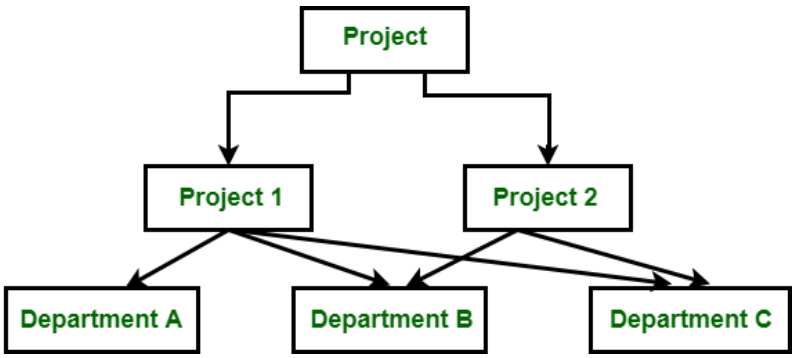
Comparison of Different Data Models

1. **Hierarchical Data Model**:
    - Data is arranged in a tree-like form with a single root and a single parent for each record in the hierarchical data model.
    - Records have one-to-many relationships with one another.
    - Example: Information Management System (IMS) database
    - Restrictions:
        - Inability to represent complex relationships with flexibility.
        - Managing many-to-many partnerships can be challenging.



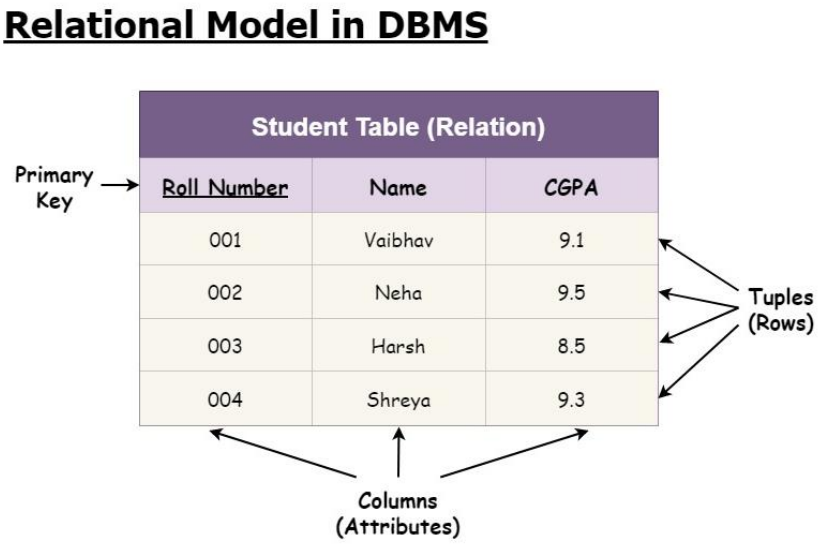
2.Network Data Model

- A group of records connected by pointers is how data is represented in the network data model.
- Multiple parent and child records may be present in a record.
- Example: CODASYL database.
- Benefits:
  - Adaptable compared to a hierarchical model.
  - Capacity to depict intricate connections.Limitations:
  - Complicated to understand and apply.
  - Inadequate standardisation.



3.: Relational Data Model

- Information is arranged into tables with rows and columns in the relational data paradigm.
- Keys are used to construct relationships between tables.
- Example: SQL databases like MySQL, PostgreSQL.
- Advantages:
  - Simplicity in design and implementation.
  - Standardized query language (SQL).
  - Support for complex queries.
- Limitations:
  - Performance issues with large-scale databases.
  - Difficulty in representing certain types of data relationships efficiently.



4.Flat file Database model

Two-dimensional data is represented on a single plane by a flat data model. Although it is not scientific in nature, the flat data model attempts to develop a standard data model. Each data element in the flat data model is represented by a two-dimensional array. Column components have equal values; row constituents have relational values. The database is shown as a single table with rows and fields in the data model. (What are the sorts of data models in DBMSs?)

Features of the Flat Model

1. Array in two dimensions

2. Absence of a structured hierarchy

3. Restricted Intricacy

Databases – Flat File Vs Relational Database

Name	Department	Boss	Phone
Smith	Sales	Britney Lurgi	9123 456
Jones	Sales	Britney Lurgi	9123 456
Lennon	Sales	Britney Lurgi	9123 456
Sade	Transport	Tom Brick	9876 543
Masoch	Transport	Tom Brick	9876 543

5.Object- Oriented Database Model

Database Management Systems (DBMS) that use object-oriented databases store data as objects that represent actual entities in the real world. Data and connections are kept together as a single object in the OODM. In OODBM, an object's fundamental attribute is its ability to build custom types. (What are the sorts of data models in DBMSs?)

In a database, objects created for a project or application are saved in their current state. By managing data as a whole object, object-oriented databases do away with the need for several tables and combine data into a single, conveniently accessible package.

Benefits of Object-Oriented Programming

Data Model:

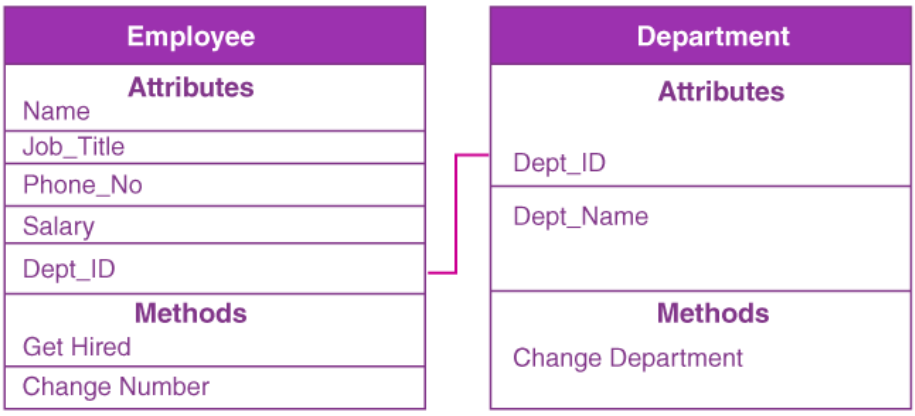
1. Code reuse is made possible by inheritance.

2. Easy to understand.

3. Because attributes can be reused, maintenance costs may be reduced; and 4. Functions can be inherited.

The object-oriented data model's drawbacks

1. Customers were hesitant to accept it because of its poor design.



Object Oriented Model

Importance of Adapting a New Data Model

The relational data model is replacing the earlier data models, like the hierarchical and network models, for a number of reasons:

1. **Simplicity and Ease of Use**:

   o Compared to hierarchical and network models, the relational data model provides a more straightforward and understandable method of representing data. Because of its simplicity, developers can comprehend and use the database more easily.

2. **Standardization**:

   o Thanks to the development of standardised query languages like SQL, the relational model has become the industry standard. Interoperability is guaranteed by this standardisation, which also makes data sharing across various platforms easier.

3. **Flexibility**:

o Relational databases are appropriate for a variety of application scenarios because they can manage a broad range of data relationships, including many-to-many, one-to-one, and one-to-many.

4. **Scalability**:

   o Relational databases have shown to be scalable, able to manage high transaction rates and substantial data quantities when the right indexing and optimisation techniques are used.

5. **Support for Complex Queries**:

   o The relational model has strong query capabilities, making it possible for users to effectively obtain and manipulate data using SQL queries.

6. **Integration with Modern Technologies**:

   o Relational databases are more applicable in today's software development environments because of their ease of integration with contemporary technologies and frameworks, such as object-relational mapping (ORM) tools and cloud platforms.

In conclusion, relational data models are preferable for contemporary database systems due to their simplicity, standardisation, flexibility, scalability, and query capabilities, even though hierarchical and network data models have their benefits.

---

2. Analysis of Different Approaches to Database Design

Creating an effective and scalable database system requires careful consideration of database design. There are several methods for creating databases, and each has benefits and drawbacks of its own. We examine and briefly discuss four popular database design methodologies below:

1. Top-Down Design Method:

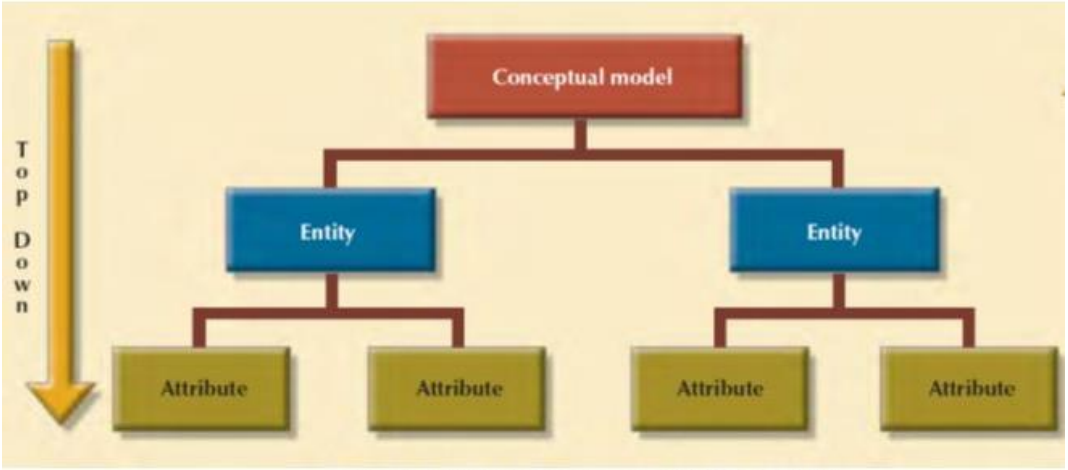
Example: Top-down design is that it begins with a high-level picture of the system and gradually disassembles it into smaller parts. Within the framework of database design, this entails breaking down the entire business needs into entities, characteristics, and relationships first.

Example: A top-down design strategy would start with identifying the primary entities for the "Travel Classics" website, which would include travellers, hotels, reservations, and destinations. The qualities of each entity would then be defined, in addition to the relationships between these entities.

Top-down Design Architecture Features:

1. High-Level Design

2. Integration and Testing



2. Bottom-Up Design Method:

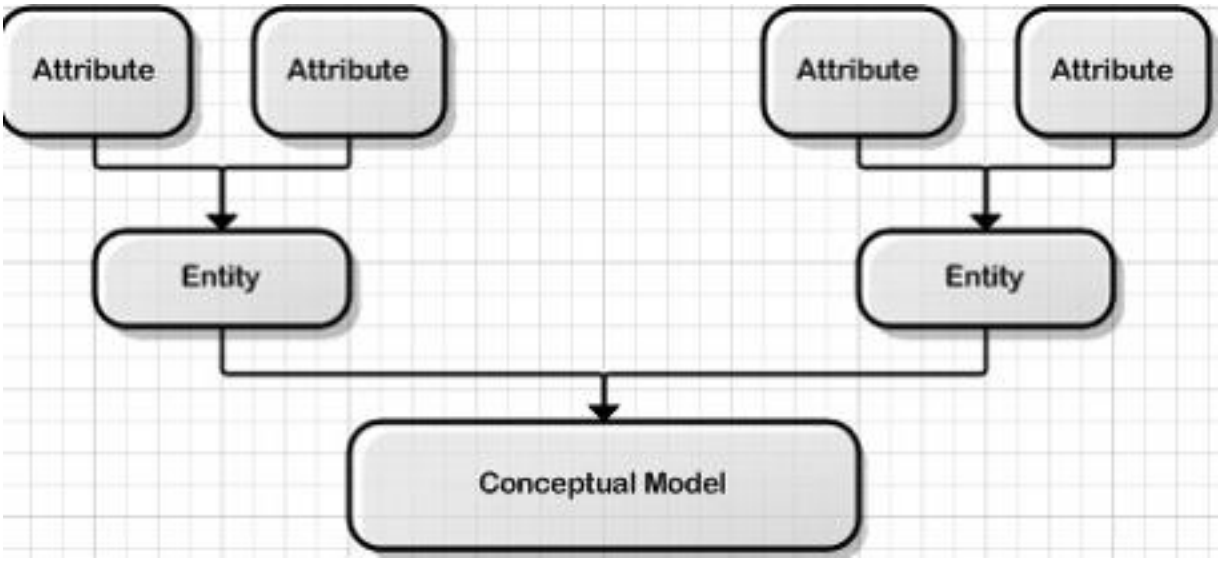
To explain, bottom-up design begins with discrete elements or entities and works its way up to a finished system. This method of database design entails locating current entities or data sources and incorporating them into a coherent database layout.

Example: When it comes to "Travel Classics," a bottom-up design strategy can begin with pre-existing traveller profiles or hotel databases, then add new features like reservation management and trip reports.

Bottom-up Approach Characteristics:

1. Creating the part

2. Testing it.



3. Centralized Design:

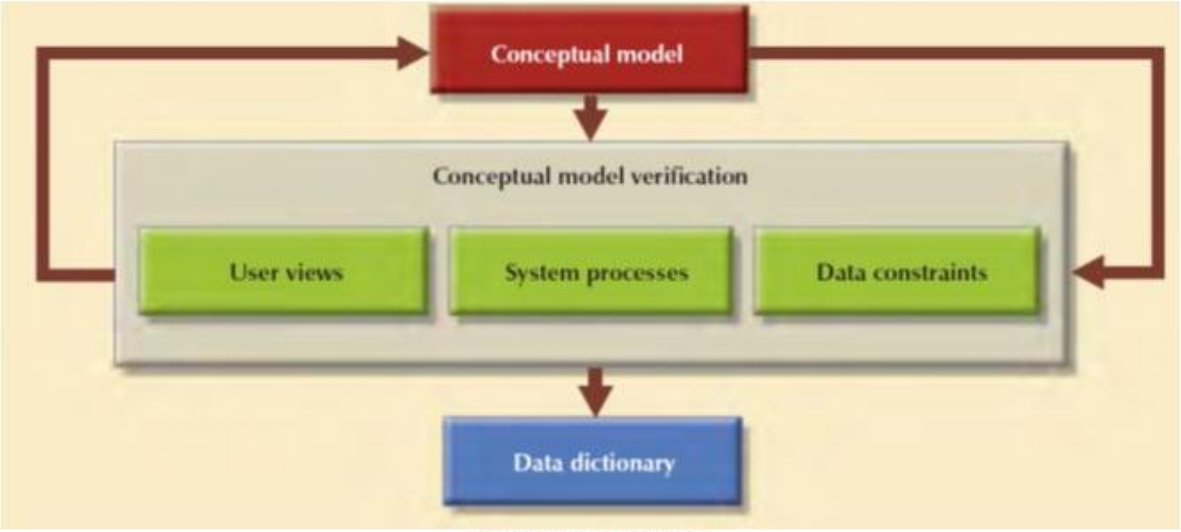
Explanation: Consolidating all database-related tasks and resources into one location or entity is known as centralised design. This method handles all aspects of data processing, retrieval, and storage through a single database server.

Example: For "Travel Classics," a centralised architecture would involve running a single database server that houses traveller, hotel, and reservation data. Via this central server, all database interactions would take place.

Centralised Design's attributes

1. One Control

2. Information Archive.



Centralized design

4. Decentralized Design:

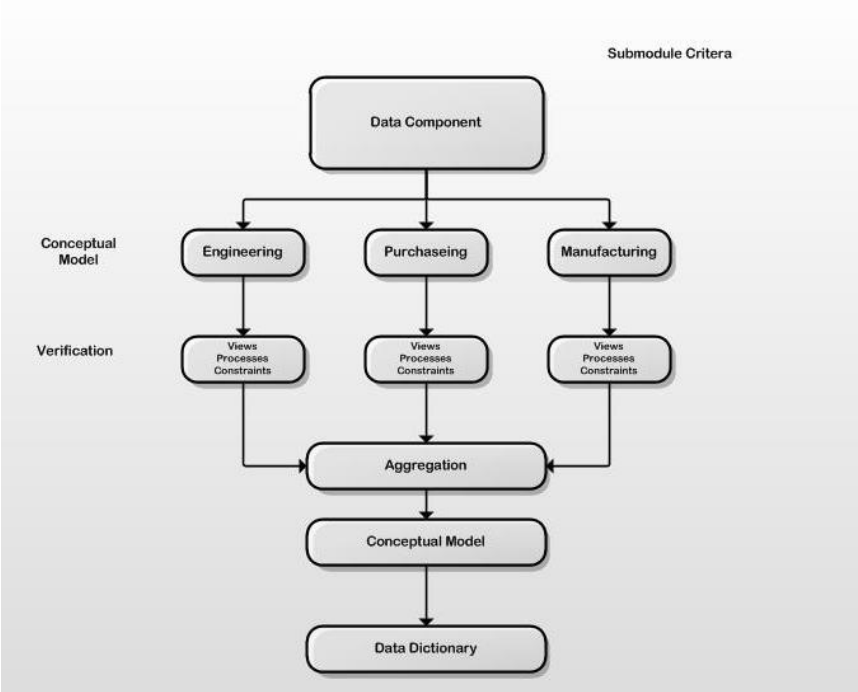
Explanation: Resources and activities related to databases are dispersed among several sites or entities through decentralised design. Data replication or synchronisation techniques are used to guarantee consistency across distributed databases, and each location or entity may have its own database server.

Example: Within "Travel Classics," a decentralised architecture could entail running distinct database servers for various nations or areas in which the business conducts business. To ensure consistency, these servers would periodically synchronise data.

Decentralized Design Characteristics:

1. Dispersed Management

2. Expandability



3. Design an Entity-Relationship Diagram (ERD)

We must determine the primary entities, their characteristics, and the connections between them in order to create an Entity-Relationship Diagram (ERD) for the "Travel Classics" scenario. This is an elaborate ERD with notations from Chen & Martin:

Entity-Relationship Diagram (ERD)

Entities:

1. **Traveler**
    - traveler_id (Primary Key)
    - name
    - address
    - email
    - country
    - reference_id (Unique)

2. **Hotel**
    - hotel_id (Primary Key)
    - name
    - country
    - city
    - street
    - registration_id (Unique)

3. **Facility**
    - facility_id (Primary Key)
    - name
    - description

4. **AccommodationOption**
    - option_id (Primary Key)
    - name
    - description

5. **Room**
    - room_id (Primary Key)
    - room_number
    - price
    - availability
    - option_id (Foreign Key)

6. **Reservation**
    - reservation_id (Primary Key)
    - traveler_id (Foreign Key)
    - hotel_id (Foreign Key)
    - room_id (Foreign Key)
    - check_in_date
    - check_out_date
    - num_guests
    - total_amount
    - reservation_date
    - payment_status

7. **TravelReport**

- report_id (Primary Key)
- traveler_id (Foreign Key)
- country
- location
- title
- description

Relationships:

1. **Traveler - Reservation (1 to Many)**:
    - Each traveler can make multiple reservations.
    - Each reservation is made by one traveler.
2. **Hotel - Reservation (1 to Many)**:
    - Each hotel can have multiple reservations.
    - Each reservation is for one hotel.
3. **Hotel - Facility (Many to Many)**:
    - Each hotel can have multiple facilities.
    - Each facility can be present in multiple hotels.
4. **Hotel - AccommodationOption (1 to Many)**:
    - Each hotel can offer multiple accommodation options.
    - Each accommodation option is offered by one hotel.
5. **AccommodationOption - Room (1 to Many)**:
    - Each accommodation option can have multiple rooms.
    - Each room belongs to one accommodation option.
6. **Traveler - TravelReport (1 to Many)**:
    - Each traveler can create multiple travel reports.
    - Each travel report is created by one traveler.

Assumptions:

• Traveler_id and hotel_id are unique identifiers for each traveller and hotel.

• A reservation_id is specific to each reservation.

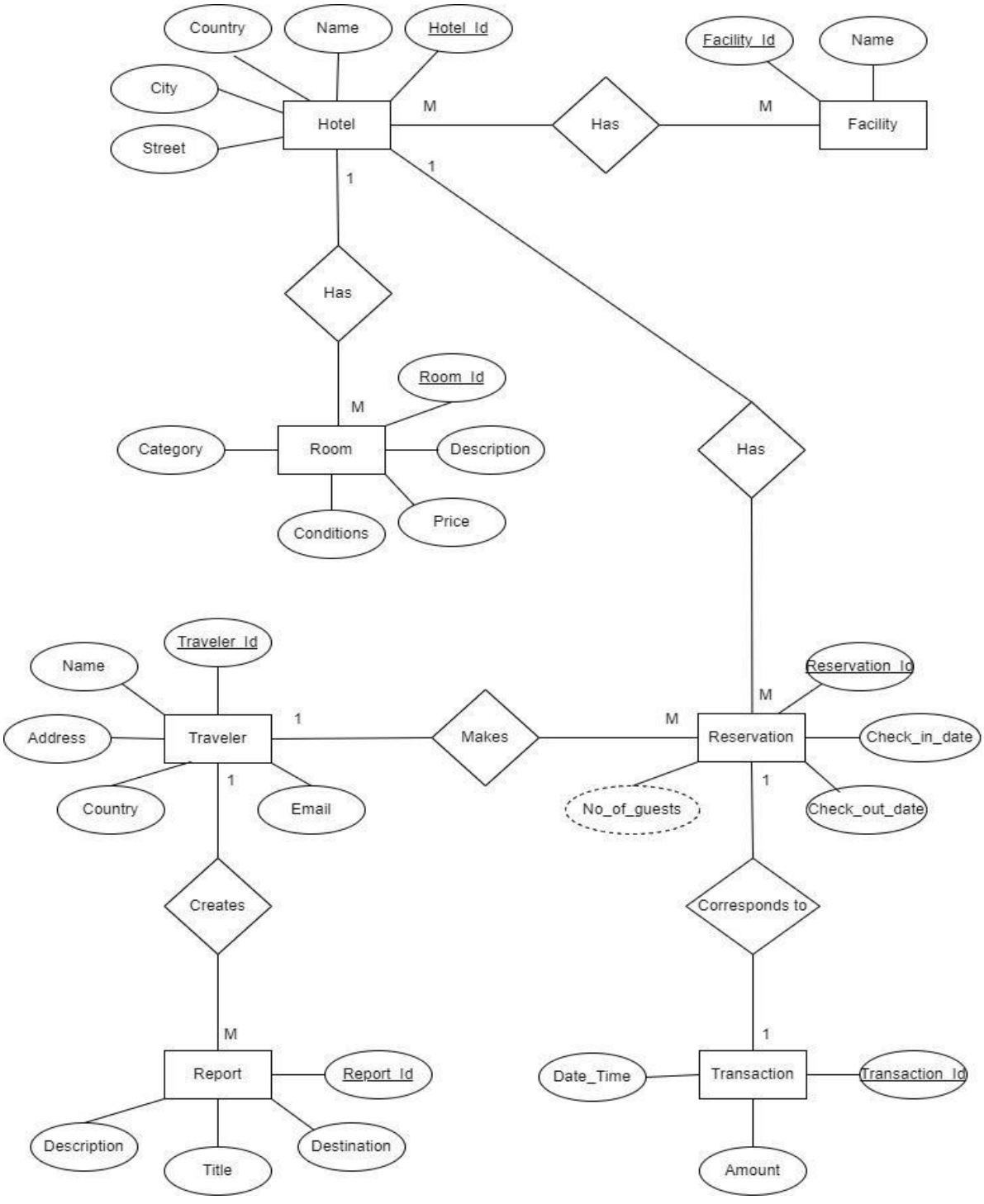
• A facility_id is exclusive to each facility.

• There is a distinct option_id for every accommodation.

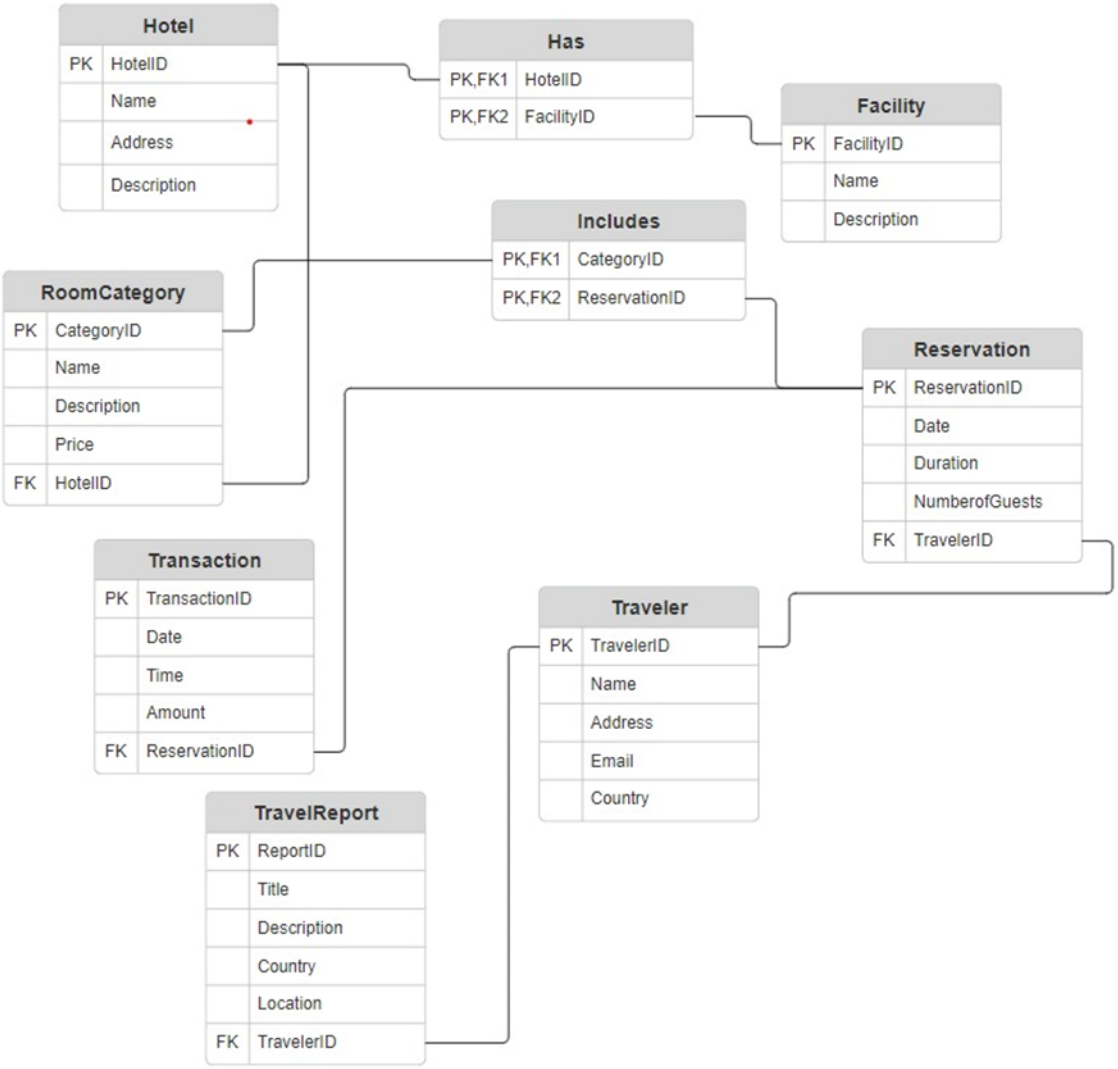
• A room_id is exclusive to each room.

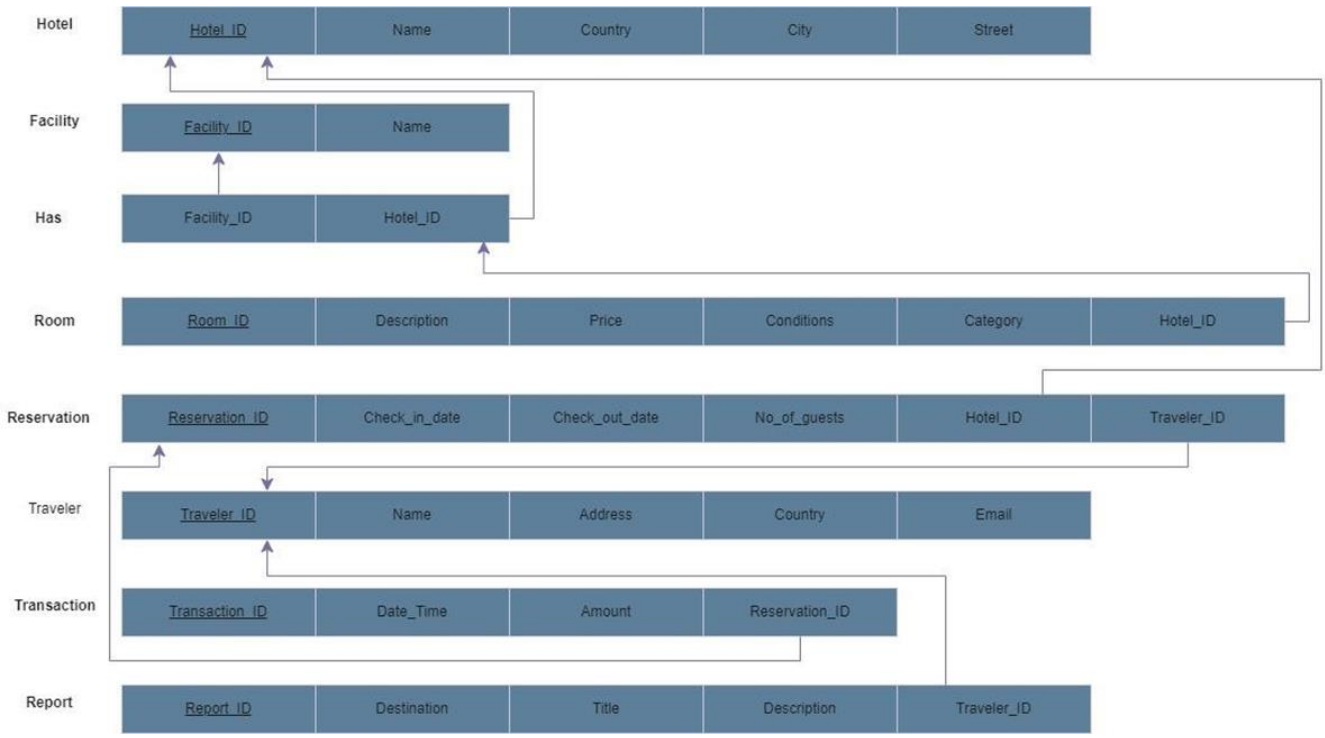
This ERD captures the relationships between the entities and their attributes, adhering to Chen & Martin notations. It provides a comprehensive overview of the database structure for the "Travel Classics".

Let's look at the entity relationship diagram with the relevant requirements for the example situation. and crucial elements that highlight a variety of traits and relationships.



4.(LO2) Draw Relational Schemas





5. The schema should be Normalize to the third normal form.

We will adhere to a methodical procedure to guarantee that dependencies are appropriately managed and data redundancy is kept to a minimum in order to normalize the database schema up to the 3rd Normal Form (3NF). The steps are as follows:

Assumptions:

1. A hotel may have more than one facility.

2. A facility may be a part of several hotels.

3. Every type of lodging has the potential to have several rooms.

4. Every room is part of a single lodging choice.

5. A passenger may make more than one reservation.

6. One tourist makes one hotel room reservation at a time.

Steps for Schema Normalization:

1. **Determine Functional Dependencies:**

Determine whether tables have functional dependencies, then remove any partial dependencies.

Functional Dependencies:

1. TRAVELER Table:

- Functional Dependencies:
- Traveler_ID → Name, Address, Email, Country
- Traveler_ID → Hotel_ID (Assuming that a visitor is limited to one hotel stay at a time)
- Traveler_ID → Reservation_ID (Presuming that a tourist is able to book several reservations)

2. HOTEL Table:

- Functional Dependencies:
- Hotel_ID → Name, Address, Facilities
- Hotel_ID → Country_ID (Presuming a hotel is situated in a single nation)

3. ROOM Table:

- Functional Dependencies:
- Room_ID → Description, Category, Condition, Price
- Room_ID → Hotel_ID (Assuming one room serves as the hotel)

4. RESERVATION Table:

- Functional Dependencies:
- Reservation_ID → Check-in Date, Check-out Date, Number of Guests
- Reservation_ID → Traveler_ID (Considering that only one traveler makes a reservation)

5. FACILITY Table:

- Functional Dependencies:
- Facility_ID → FacilityName

6. TRANSACTION Table:

- Functional Dependencies:
- Transaction_ID → Amount, Date, Time.
- Transaction_ID → Traveler_ID (Suppose that a single traveler submits a transaction)

7. REPORT Table:

- Functional Dependencies:

- Report_ID → Title, Description, Location
- Report_ID → Traveler_ID (Considering that a single traveler files numerous reports)

2. Apply 1st Normal Form (1NF):

Making sure that all of the qualities in each table are atomic—that is, indivisible and incapable of further breakdown—is the main objective of the First Normal Form (1NF). Furthermore, 1NF deals with removing repeated groupings from the tables.

1.TRAVELER Table:

Because there are no repeated groups and every attribute is atomic, the TRAVELER table already complies with 1NF. Name, Address, Email, and Country are examples of atomic properties that don't have multiple values.

2.HOTEL Table:

The HOTEL table is also in 1NF compliance. The attributes like Name, Address, and Facilities are atomic, and there are no repeating groups.

3.RESERVATION Table:

Aspects like the number of guests, check-in date, and check-out date are already atomic in the RESERVATION table. This table does not contain any repeated groups.

4.ROOM Table:

The ROOM table complies with 1NF requirements. There are no repeated groups and the Room_ID attribute is atomic.

5.FACILITY Table:

The FACILITY table satisfies the 1NF criteria, just as other tables. There are no repeated groups present, and the Facility Name property is atomic.

6.TRANSACTION Table:

The FACILITY table satisfies the 1NF criteria, just as other tables. There are no repeated groups present, and the Transaction_ID property is atomic.

7.REPORT Table:

The 1NF requirements are met by the REPORT table. There are no repeated groups and the properties Title, Description, and Location are atomic.

3. Apply 2nd Normal Form (2NF):

We must make sure that every non-key attribute in Second Normal Form depends entirely on the primary key in order for it to function. This implies that the complete primary key, not just a portion of it, should determine every characteristic in a database.

1. **TRAVELER Table:**
    - **Primary Key(s):**
        - TravelerID
    - **Attributes:**
        - Name
        - Address
        - Email
        - Country
    - **Foreign Key(s):**
        - HotelID (Referencing HOTEL Table)
        - ReservationID (Referencing RESERVATION Table)
2. **HOTEL Table:**

- **Primary Key(s):**
    - HotelID
- **Attributes:**
    - Name
    - Address
    - Facilities
- **Foreign Key(s):**
    - CountryID (Referencing COUNTRY Table)

3. **RESERVATION Table:**
    - **Primary Key(s):**
        - ReservationID
    - **Attributes:**
        - Check-in Date
        - Check-out Date
        - Number of Guests
    - **Foreign Key(s):**
        - TravelerID (Referencing TRAVELER Table)
        - HotelID (Referencing HOTEL Table)
4. **FACILITY Table:**
    - **Primary Key(s):**
        - FacilityID
    - **Attributes:**
        - FacilityName
    - **Foreign Key(s):**

- (No foreign keys in this table)

5. **REPORT Table:**
    - **Primary Key(s):**
        - ReportID
    - **Attributes:**
        - Title
        - Description
        - Location
    - **Foreign Key(s):**
        - TravelerID (Referencing TRAVELER Table)

4. Apply 3rd Normal Form (3NF):

By ensuring that none of the non-key attributes are transitively dependent on the primary key, we aim to remove transitive dependencies in Third Normal Form (3NF). Transitive dependency is the term used to describe the situation where one non-prime feature depends on another non-prime attribute.

1. **TRAVELER Table:**
    - **Primary Key(s):**
        - TravelerID
    - **Attributes**:
        - Name
        - Address
        - Email

- **Foreign Key(s):**
    - HotelID (With reference to the HOTEL Table)
    - ReservationID (Citing the RESERVATION Table)

2. **HOTEL Table:**
    - **Primary Key(s):**
        - HotelID
    - **Attributes:**
        - Name
        - Address
        - Facilities
    - **Foreign Key(s):**
        - CountryID (Citing the COUNTRY Table)
3. **RESERVATION Table:**
    - **Primary Key(s):**
        - ReservationID
    - **Attributes:**
        - Check-in Date
        - Check-out Date
        - Number of Guests
    - **Foreign Key(s):**
        - TravelerID (Citing the Traveler Table)
        - HotelID (With reference to the HOTEL Table)
4. **FACILITY Table:**
    - **Primary Key(s):**

        - FacilityID
    - **Attributes:**
        - FacilityName

5. **REPORT Table:**
    - **Primary Key(s):**
        - ReportID
    - **Attributes:**
        - Title
        - Description
        - Location
    - **Foreign Key(s):**
        - TravelerID (Citing the Traveler Table)

Explanation:

- Every non-key attribute in every table is directly reliant on the primary key.
- Every attribute is non-transitively dependent on the primary key since there are no transitive dependencies..
- This guarantees adherence to Third Normal Form (3NF), assisting in the removal of redundant information and enhancing data integrity within the database schema.

Step 5: Resulting Normalized Tables

The normalized tables that result after normalization up to the Third Normal Form

(3NF) are as follows:

1. TRAVELER Table:
    - TravelerID [PK]
    - Name
    - Address
    - Email

2. HOTEL Table:
    - HotelID [PK]
    - Name
    - Address
    - Facilities
    - CountryID [FK]

3. RESERVATION Table:
    - ReservationID [PK]
    - Check-in Date
    - Check-out Date
    - Number of Guests
    - TravelerID [FK]
    - HotelID [FK]

4. FACILITY Table:

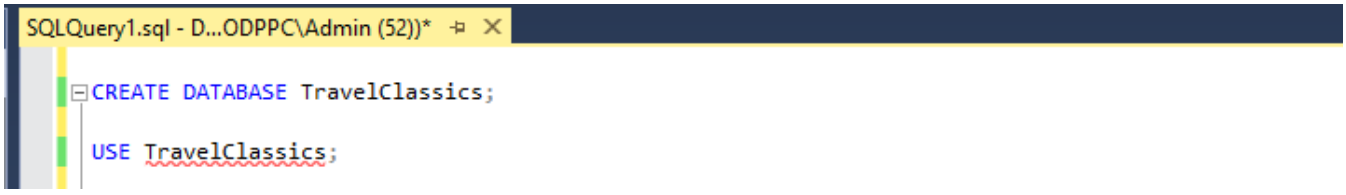
- Facility_ID [PK]
- Facility_Name

5. REPORT Table:
    - Report_ID [PK]
    - Title
    - Description
    - Location
    - Traveler_ID [FK]

Now that these tables are in Third Normal Form (3NF), all non-key characteristics are completely functionally dependent on their corresponding primary keys and there are no transitive dependencies. The normalization procedure ensures a well-structured relational database by assisting in the effective organization of the data and preventing redundancies.

06. (LO3)

Create Database and use database



Create Tables

Table Hotel

```
CREATE TABLE Hotel (
    hotel_id INT PRIMARY KEY,
    name VARCHAR(100),
    country VARCHAR(50),
    city VARCHAR(50),
    street VARCHAR(100),
    logging VARCHAR(50)
);
```

Table Facility

```
CREATE TABLE Facility (
    facility_id INT PRIMARY KEY,
    name VARCHAR(50)
);
```

Table Room

```
CREATE TABLE Room (
    room_id INT PRIMARY KEY,
    hotel_id INT,
    option_id INT,
    price DECIMAL(10, 2),
    FOREIGN KEY (hotel_id)
    REFERENCES Hotel(hotel_id)
);
```

Table Traveler

```sql
CREATE TABLE Traveler (
    traveler_id INT PRIMARY KEY,
    name VARCHAR(100),
    address VARCHAR(255),
    email VARCHAR(100),
    country VARCHAR(50)
);
```

Table Reservation

```sql
CREATE TABLE Reservation (
    reservation_id INT PRIMARY KEY,
    traveler_id INT,
    room_id INT,
    start_date DATE,
    end_date DATE,
    no_of_guests INT,
    FOREIGN KEY (traveler_id) REFERENCES Traveler(traveler_id),
    FOREIGN KEY (room_id) REFERENCES Room(room_id)
);
```

Table Transaction

```sql
CREATE TABLE Transactions (
    transaction_id INT PRIMARY KEY,
    reservation_id INT,
    amount DECIMAL(10, 2),
    date DATE,
    time TIME,
    FOREIGN KEY (reservation_id)
    REFERENCES Reservation(reservation_id)
);
```

Table Report

```sql
CREATE TABLE Report (
    report_id INT PRIMARY KEY,
    traveler_id INT,
    title VARCHAR(100),
    description VARCHAR(255),
    country VARCHAR(50),
    location VARCHAR(100),
    FOREIGN KEY (traveler_id) REFERENCES Traveler(traveler_id)
);
```

Inserting at least 15 records for each table.

Record Insertion into Hotel Table:

```sql
INSERT INTO Hotel (hotel_id, name, country, city, street, logging)
VALUES
(1, 'Grand Hotel', 'Maldives', 'Male', '123 Beachfront Ave', 'oceanfrontmaldives123'),
(2, 'Mountain View Lodge', 'Switzerland', 'Interlaken', '1 Alpenweg', 'mountainviewlodgech123'),
(3, 'Rainforest Retreat', 'Costa Rica', 'La Fortuna', '2 Rainforest Ave', 'rainforestretreatcr1'),
(4, 'Golden Sands Resort', 'Thailand', 'Phuket', '3 Paradise Road', 'goldensandsphuket'),
(5, 'Cityscape Hotel', 'Japan', 'Tokyo', '4 Skyscraper Blvd', 'cityscapehoteltokyo123'),
(6, 'Lakeside Lodge', 'New Zealand', 'Queenstown', '5 Lakeview Terrace', 'lakesidelodgenz1'),
(7, 'Desert Oasis Resort', 'UAE', 'Abu Dhabi', '6 Oasis Road', 'desertoasisabudhabi1'),
(8, 'Alpine Chalet', 'Austria', 'Innsbruck', '7 Mountainview Strasse', 'alpinechaletat123'),
(9, 'Island Paradise Hotel', 'Philippines', 'Boracay', '8 Palm Beach Road', 'islandparadiseboracay1'),
(10, 'Historic Manor', 'England', 'London', '9 Tudor Lane', 'historicmanoruk123'),
(11, 'Tropical Haven Resort', 'Fiji', 'Nadi', '10 Tropical Way', 'tropicalhavenfiji'),
(12, 'City Lights Inn', 'USA', 'Las Vegas', '11 Strip Avenue', 'citylightsinnlv123'),
(13, 'Countryside Cottage', 'Ireland', 'Killarney', '12 Green Fields Road', 'countrysidecottageie1'),
(14, 'Glacier Retreat', 'Iceland', 'Reykjavik', '13 Iceberg Street', 'glacierretreaticeland123'),
(15, 'Beachside Bungalow', 'Bahamas', 'Nassau', '14 Seaside Drive', 'beachsidebungalowbs123');
```

Record Insertion into Facility Table:

```sql
INSERT INTO Facility (facility_id, name)
VALUES
(1, 'Infinity Pool'),
(2, 'Yoga Studio'),
(3, 'Restaurant & Bar'),
(4, 'Spa'),
(5, 'Business Center'),
(6, 'Tennis Courts'),
(7, 'Outdoor Terrace'),
(8, 'Dry Cleaning Service'),
(9, 'High-Speed Internet'),
(10, 'Medical Services'),
(11, 'Valet Parking'),
(12, 'In-Room Dining'),
(13, 'Luxury Toiletries'),
(14, 'Baggage Handling'),
(15, 'Accessible Rooms');
```

Inserting Records into Room Table

```sql
INSERT INTO Room (room_id, hotel_id, option_id, price)
VALUES
(1, 1, 1,  12000.00),
(2, 1, 2,  18000.00),
(3, 1, 3,  25000.00),
(4, 2, 4,  30000.00),
(5, 2, 5,  15000.00),
(6, 2, 6,  20000.00),
(7, 3, 7,  30000.00),
(8, 3, 8,  35000.00),
(9, 3, 9,  20000.00),
(10, 4, 10,  25000.00),
(11, 4, 11,  18000.00),
(12, 4, 12,  22000.00),
(13, 5, 13,  25000.00),
(14, 5, 14,  30000.00),
(15, 5, 15,  22000.00);
```

Inserting Records into Traveler Table

```sql
INSERT INTO Traveler (traveler_id, name, address, email, country)
VALUES
(1, 'Adam Johnson', '111 River Road, City A', 'adam@example.com', 'Country A'),
(2, 'Sophie Williams', '222 Forest Lane, City B', 'sophie@example.com', 'Country B'),
(3, 'Jack Brown', '333 Mountain View, City C', 'jack@example.com', 'Country C'),
(4, 'Olivia Garcia', '444 Beachside Ave, City D', 'olivia@example.com', 'Country D'),
(5, 'Noah Martinez', '555 Sunset Blvd, City E', 'noah@example.com', 'Country E'),
(6, 'Ava Rodriguez', '666 Sunrise Road, City F', 'ava@example.com', 'Country F'),
(7, 'Ethan Wilson', '777 Lakeside Drive, City G', 'ethan@example.com', 'Country G'),
(8, 'Charlotte Taylor', '888 Meadow Lane, City H', 'charlotte@example.com', 'Country H'),
(9, 'James Smith', '999 Valley View, City I','james_smith@example.com','Country I'),
(10, 'Emma Johnson', '1010 Hilltop Avenue, City J','emma_johnson@example.com','Country J'),
(11, 'Liam Martinez', '1212 Skyline Road, City K', 'liam@example.com', 'Country K'),
(12, 'Olivia Brown', '1313 Park Place, City L', 'olivia_brown@example.com', 'Country L'),
(13, 'Lucas Wilson', '1414 Ocean View, City M', 'lucas@example.com', 'Country M'),
(14, 'Ava Taylor', '1515 Riverfront Drive, City N', 'ava_taylor@example.com', 'Country N'),
(15, 'Sophia Garcia', '1616 Lakeside Avenue, City O', 'sophia_garcia@example.com', 'Country O');
```

Inserting Records into Reservation Table:

```sql
INSERT INTO Reservation (reservation_id, traveler_id, room_id, start_date, end_date, no_of_guests)
VALUES
(1, 1, 1, '2024-05-01', '2024-05-05', 1),
(2, 2, 2, '2024-06-01', '2024-06-10', 2),
(3, 3, 3,  '2024-07-01', '2024-07-07', 1),
(4, 4, 4,  '2024-08-01', '2024-08-15', 3),
(5, 5, 5,  '2024-09-01', '2024-09-10', 2),
(6, 6, 6,  '2024-10-01', '2024-10-05', 1),
(7, 7, 7,  '2024-11-01', '2024-11-07', 4),
(8, 8, 8, '2024-12-01', '2024-12-10', 2),
(9, 9, 9, '2025-01-01', '2025-01-05', 2),
(10, 10, 10,  '2025-02-01', '2025-02-05', 1),
(11, 11, 11,  '2025-03-01', '2025-03-07', 2),
(12, 12, 12,  '2025-04-01', '2025-04-10', 4),
(13, 13, 13,  '2025-05-01', '2025-05-05', 1),
(14, 14, 14,  '2025-06-01', '2025-06-05', 3),
(15, 15, 15,  '2025-07-01', '2025-07-07', 2);
```

Inserting Records into Transaction Table:

```
INSERT INTO Transactions (transaction_id, reservation_id, amount, date, time)
VALUES
(1, 1, 10.00, '2024-05-15', '12:00:00'),
(2, 2, 15.00, '2024-06-20', '14:30:00'),
(3, 3, 20.00, '2024-07-01', '10:00:00'),
(4, 4, 25.00, '2024-08-05', '09:45:00'),
(5, 5, 30.00, '2024-09-10', '11:20:00'),
(6, 6, 35.00, '2024-10-15', '16:00:00'),
(7, 7, 40.00, '2024-11-20', '13:30:00'),
(8, 8, 45.00, '2024-12-25', '10:15:00'),
(9, 9, 50.00, '2025-01-30', '08:45:00'),
(10, 10, 55.00, '2025-03-05', '17:00:00'),
(11, 11, 60.00, '2025-04-10', '12:30:00'),
(12, 12, 65.00, '2025-05-15', '14:00:00'),
(13, 13, 70.00, '2025-06-20', '11:45:00'),
(14, 14, 75.00, '2025-07-25', '09:30:00'),
(15, 15, 80.00, '2025-08-30', '15:00:00');
```

Inserting Records into Report Table:

```
INSERT INTO Report (report_id, traveler_id, title, description, country, location)
VALUES
(1, 1, 'Toronto Travels', 'Exploring the vibrant city of Toronto', 'Canada', 'Toronto'),
(2, 2, 'Vienna Vibes', 'Immersing in the cultural richness of Vienna', 'Austria', 'Vienna'),
(3, 3, 'Tokyo Tales', 'Adventures in the bustling streets of Tokyo', 'Japan', 'Tokyo'),
(4, 4, 'London Calling', 'Discovering the historic charm of London', 'United Kingdom', 'London'),
(5, 5, 'Berlin Bound', 'Exploring the dynamic city of Berlin', 'Germany', 'Berlin'),
(6, 6, 'Seoul Escapade', 'Experiencing the modernity of Seoul', 'South Korea', 'Seoul'),
(7, 7, 'Bangkok Bliss', 'Discovering the vibrant culture of Bangkok', 'Thailand', 'Bangkok'),
(8, 8, 'Amsterdam Adventures', 'Exploring the picturesque canals of Amsterdam', 'Netherlands', 'Amsterdam'),
(9, 9, 'Havana Hideaway', 'Immersing in the colorful streets of Havana', 'Cuba', 'Havana'),
(10, 10, 'Moscow Memories', 'Exploring the historical landmarks of Moscow', 'Russia', 'Moscow'),
(11, 11, 'Singapore Sojourn', 'Discovering the modern marvels of Singapore', 'Singapore', 'Singapore'),
(12, 12, 'Dublin Delights', 'Experiencing the lively atmosphere of Dublin', 'Ireland', 'Dublin'),
(13, 13, 'Athens Adventure', 'Exploring the ancient ruins of Athens', 'Greece', 'Athens'),
(14, 14, 'Venice Voyage', 'Cruising along the canals of Venice', 'Italy', 'Venice'),
(15, 15, 'Cape Town Chronicles', 'Discovering the beauty of Cape Town', 'South Africa', 'Cape Town');
```

(LO 03)

Both the demonstration and the implementation are useful.

1. Traveler's list at a particular location.

```
SELECT *
FROM Traveler
WHERE address LIKE '%City X%';
```

Explanation:

SELECT **:** The database is instructed to retrieve every column from the Traveller table by using this query.

FROM Traveler: Names the table (the Traveller table, for example) from which the data will be retrieved.

A location such as '%City X%': This clause uses a particular criterion to choose the rows. The address LIKE '%City X%' condition determines if the substring 'City X' is present in the address column.

As a wildcard that matches any character sequence, the% sign means that %City X% will match any address that has 'City X' anywhere in the text. In SQL, pattern matching is done via the LIKE term.

2.Total amount of business a hotel receives in a certain time frame.

```sql
SELECT h.name AS hotel_name, SUM(t.amount) AS total_transactions
FROM Hotel h
JOIN Room r ON h.hotel_id = r.hotel_id
JOIN Reservation re ON r.room_id = re.room_id
JOIN Transactions t ON re.reservation_id = t.reservation_id
WHERE h.name = 'Grand Hotel'
AND re.start_date >= '2024-03-01' AND re.end_date <= '2024-03-05'
GROUP BY h.name;
```

Explanation:

The total number of transactions for the "Grand Hotel" throughout the given time period is obtained by this SQL query.

The following explains each section of the question:

a.SELECT Clause.

- "h.name AS hotel_name": Chooses the name of the hotel and uses "hotel_name" as an alias.
- "SUM(t.amount) AS total_transactions": Determines the total amount of transactions and allocates it to the appropriate transaction category.

b.Clause from which

- "FROM Hotel h" aliases the table "Hotel" to "h" and specifies it. We will collect hotel information here.

c.JOIN Clause:

- "join room r on h.hotel_id" equals "r.hotel_id": By using the 'hotel_id' column, this links the 'Hotel' and 'Room' tables. This is how rooms connected to the hotel are retrieved.

- . "JOIN Reservation re ON r.room_id" equals "re.room_id". The "room_id" field is used to link the "Room" and "Reservation" tables. Room reservations are retrieved using this method.

- "JOIN Transactions t ON re.reservation_id" would result in "t.reservation_id." The "reservation_id" field is used to link the "Transaction" and "Reservation" tables. This is how reservation-related transaction data is accessed.

d.WHERE Clause:

- WHERE 'Grand Hotel' is *h.name*: suggests that the data for the "Grand Hotel" should be the only ones studied.

- PROVIDED THAT re.start_date >= '2024-03-01'. Moreover,

re.end_date <= The range of dates for the reservation is '2024-03-05'.
The only reservations that are listed are those that start on March 1,
2024, and finish on March 5, 2024, or before.

e. Group by Clause:

- "GROUP BY h.name": Sorts the outcomes according to the hotel's name. When utilising aggregate functions like "SUM ()," this is necessary to guarantee that the computation is finished for every hotel.

To put it briefly, the query aggregates information from the Hotel, Room, Reservation, and Transactions databases and applies filters based on the hotel name and reservation dates to get the total number of transactions for the "Grand Hotel" within the selected period range.

3.A list of hotels with available rooms

```
SELECT h.name AS hotel_name, COUNT(r.room_id) AS room_count
FROM Hotel h
LEFT JOIN Room r ON h.hotel_id = r.hotel_id
GROUP BY h.name;
```

Explanation:

a.SELECT h.name AS hotel_name:

- This query obtains the name column from the Hotel table and aliases it as "hotel_name."

b.COUNT(r.room_id) AS room_count:

- The name column from the Hotel table is retrieved by this query, which aliases it as "hotel_name."

c.FROM Hotel h:

- The source table, "Hotel," is specified in this query, and it is aliased as "h."

d.LEFT JOIN:

- "Room r ON h.hotel_id" = "r.hotel_id" does a left join using
- Even in cases where the Room table has no matching rows. Nevertheless, the databases are part of the result set.

e.GROUP BY h.name:

- The 'hotel_name' column results are grouped in this way, making it possible to apply the COUNT function to each group independently and get the number of rooms for each hotel.

8.Test Plan for Travel Classics

1.The primary goal is to enhance the precision and dependability of the "Travel Classics" database.

2.Scope:

- Test table CRUD functions (Create, Read, Update, Delete).
- Verify the accuracy of the SQL query to get data..
- Verify the reliability of foreign key relationships.

3.Testing environment:

- MySQL database server.
- Data in the Travel_Classics schema has been verified.

4.Test Cases

Test Case 1: Create

- Add a new record for the traveler.
- Anticipated Outcome: The record is added to the traveler table successfully.

Test Case 2: Read

- To obtain traveler information, use TravelerID.
- Anticipated outcome: Accurate traveler data is acquired.

Test Case 3: Update

- Update the email address of a current traveller.
- Successful email address update was the anticipated result.

Test Case 4: Delete

- Delete a passenger's information.
- Anticipated result: The Traveller table record will be deleted.

Test Case 5: A list of visitors to a place

- Use a search to locate travellers in a specific location.
- Anticipated result: Accurate traveller list obtained.

Test Case 6: The sum of hotel transactions is calculated.

- Run a query to determine the total number of transactions for a specific hotel within a specified period of time.
- Anticipated Result: The accurate transaction amount is computed.

Test Case 7: Hotel list and room availability

- To get a list of hotels and the rooms they have available, run a query.
- Anticipated Outcome: An exhaustive roster of accommodations and available rooms is acquired.

5.Dependencies:

• There is a MySQL database server accessible.
• Test data should be successfully added to the Travel_Classics schema.

6.Execution Strategy:

• Carry out automated and manual testing.

• Test functionality with real-world examples.

• Use stress testing to assess how well a system performs under load.

7.Strategies for Risk and Mitigation:

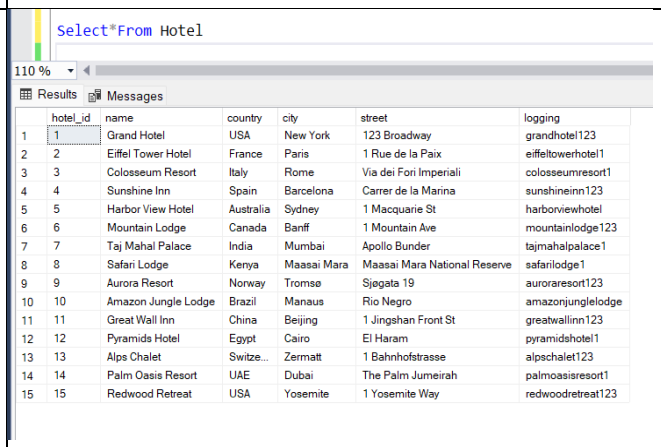
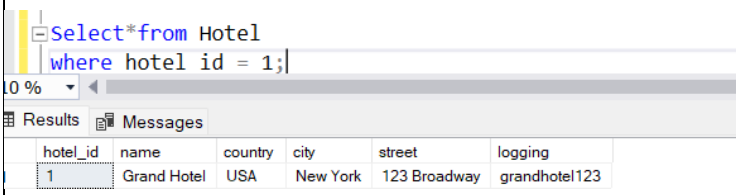
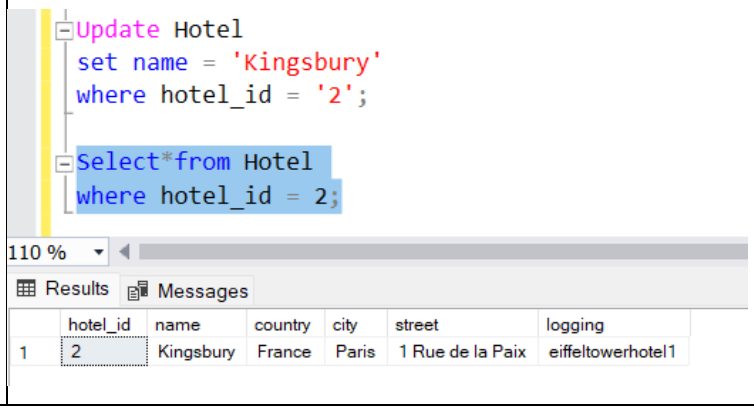
•Inaccurate or incomplete data is one source of risk.

•Mitigation: Before executing, confirm the accuracy of the test data.

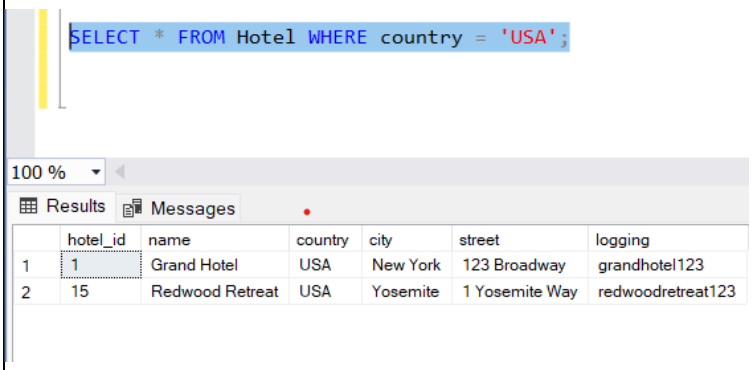
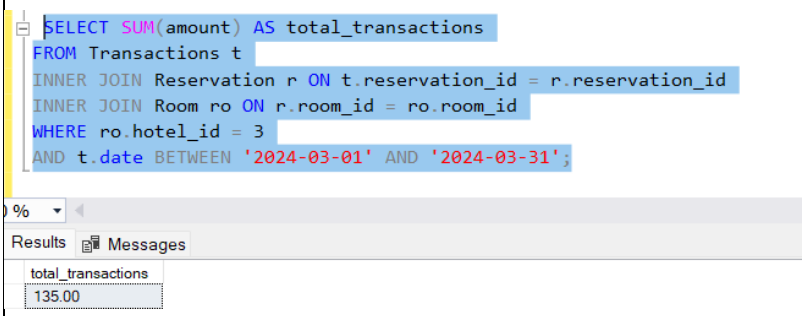
8.Conclusion:

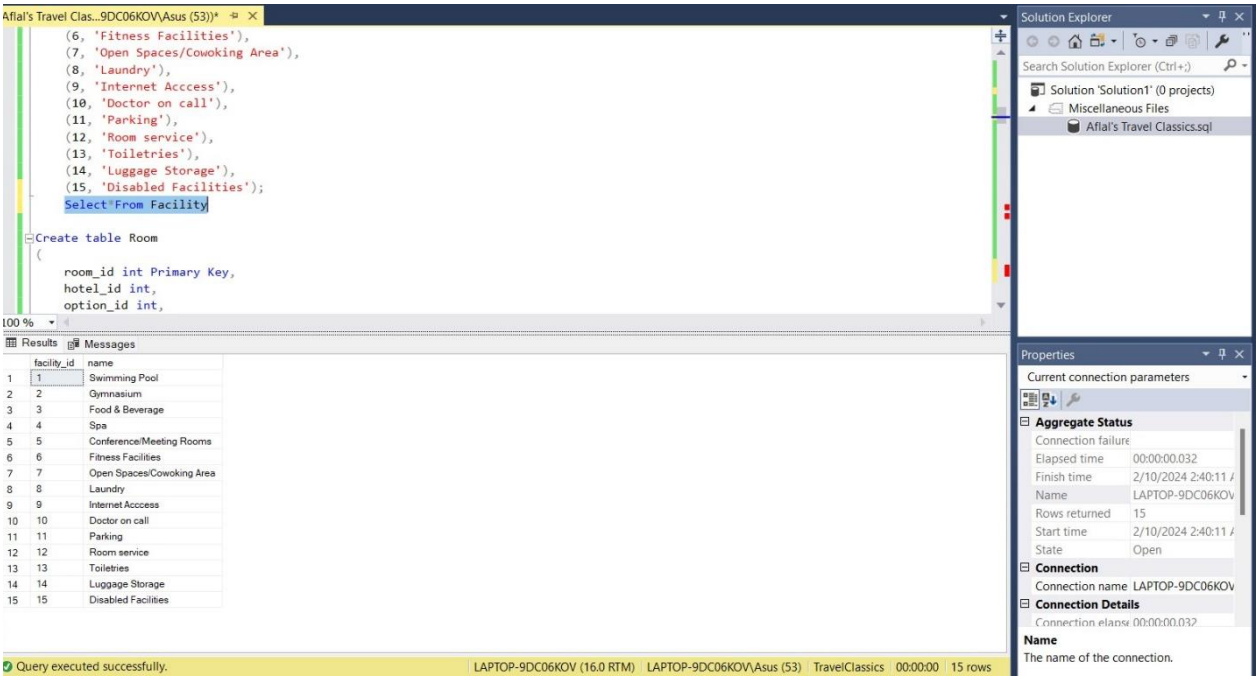
The database is dependable and prepared for deployment if every test case passes.

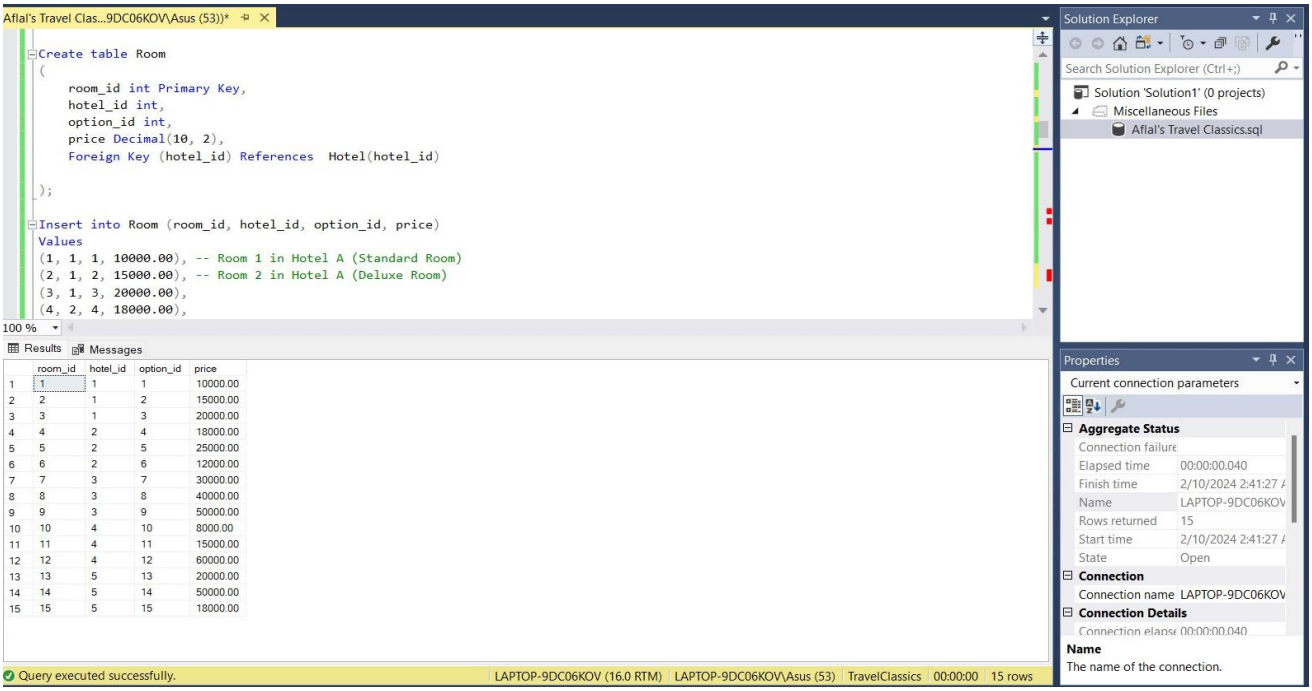
Test Result

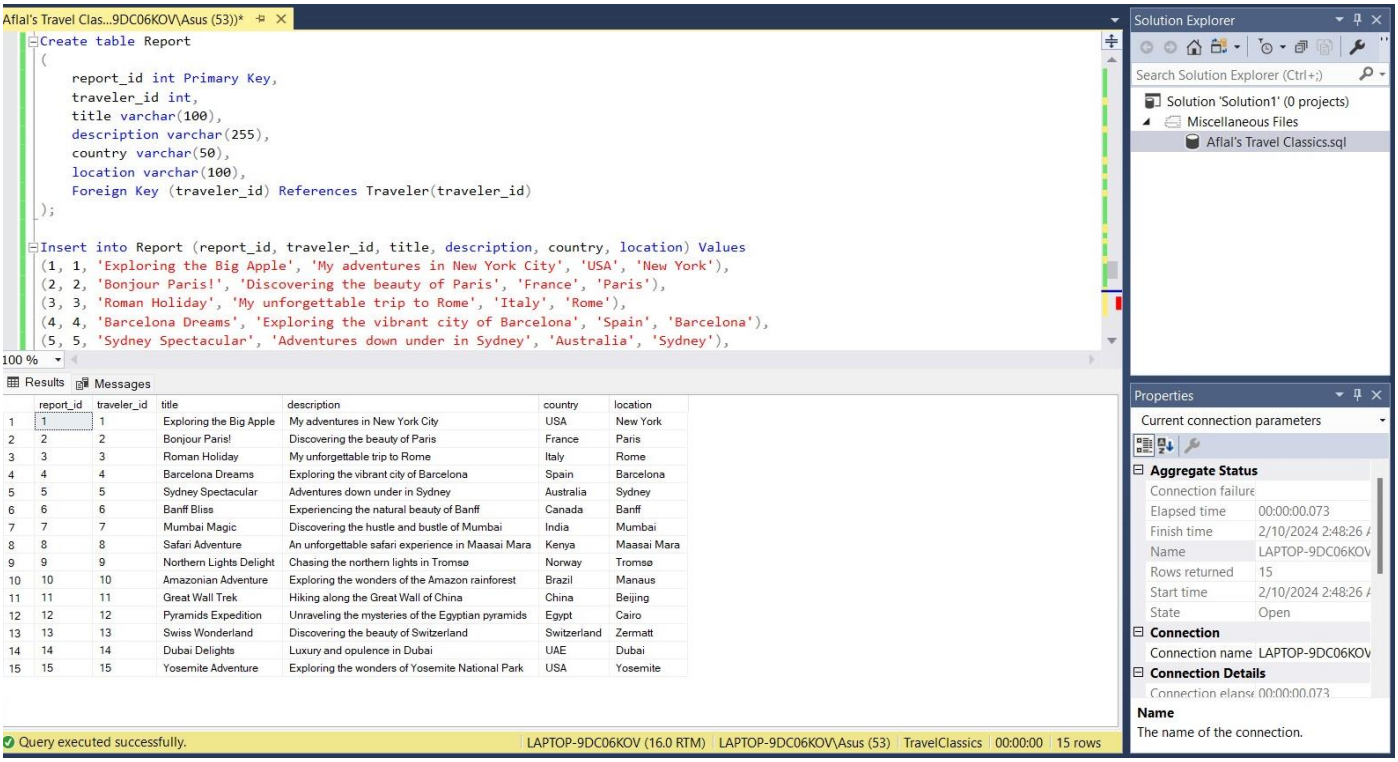
Test Case	Description	Test Data	Comment Result	Expected Result
1	Make a new traveller record by inserting it.	Enter values for the following into the Hotel: (1, "Grand Hotel," "USA," "New York," "123 Broadway," "grandhotel123");	Added to the Traveller table successfully	
2	Read: Using Hotel_id, retrieve traveller details	Hotel_id:1	The accurate hotel information is obtained.	
3	Update: Rename an already-existing hotel.	Hotel_id: ‘2’ , New Name: 'Kingsbury’	The updated name has been successfully submitted.	

4	Remove a record for a hotel	Hotel_id:'2'	The entry is removed from the hotel table.	Record is deleted from Hotel table.

5	List of hotels in the area	"USA"	The accurate hotel list for the USA has been obtained.	
6	Sum of All Transactions at a Hotel	Hotel_id:"3", Duration: '2024-03-01 to '2024-03-31'	Determined	


Aflal's Travel Clas...9DC06KOV\Asus (53))*
(6, 'Fitness Facilities'),
(7, 'Open Spaces/Cowoking Area'),
(8, 'Laundry'),
(9, 'Internet Acccess'),
(10, 'Doctor on call'),
(11, 'Parking'),
(12, 'Room service'),
(13, 'Toiletries'),
(14, 'Luggage Storage'),
(15, 'Disabled Facilities');
Select *From Facility
Create table Room
(
room_id int Primary Key,
hotel_id int,
option_id int,
100 %
Results
Messages
facility_id name
1 1 Swimming Pool
2 2 Gymnasium
3 3 Food & Beverage
4 4 Spa
5 5 Conference/Meeting Rooms
6 6 Fitness Facilities
7 7 Open Spaces/Cowoking Area
8 8 Laundry
9 9 Internet Acccess
10 10 Doctor on call
11 11 Parking
12 12 Room service
13 13 Toiletries
14 14 Luggage Storage
15 15 Disabled Facilities
Query executed successfully.
LAPTOP-9DC06KOV (16.0 RTM)
LAPTOP-9DC06KOV\Asus (53)
TravelClassics
00:00:00
15 rows
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Solution1' (0 projects)
Miscellaneous Files
Aflal's Travel Classics.sql
Properties
Current connection parameters
Aggregate Status
Connection failure
Elapsed time 00:00:00.032
Finish time 2/10/2024 2:40:11
Name LAPTOP-9DC06KOV
Rows returned 15
Start time 2/10/2024 2:40:11
State Open
Connection
Connection name LAPTOP-9DC06KOV
Connection Details
Name
The name of the connection.

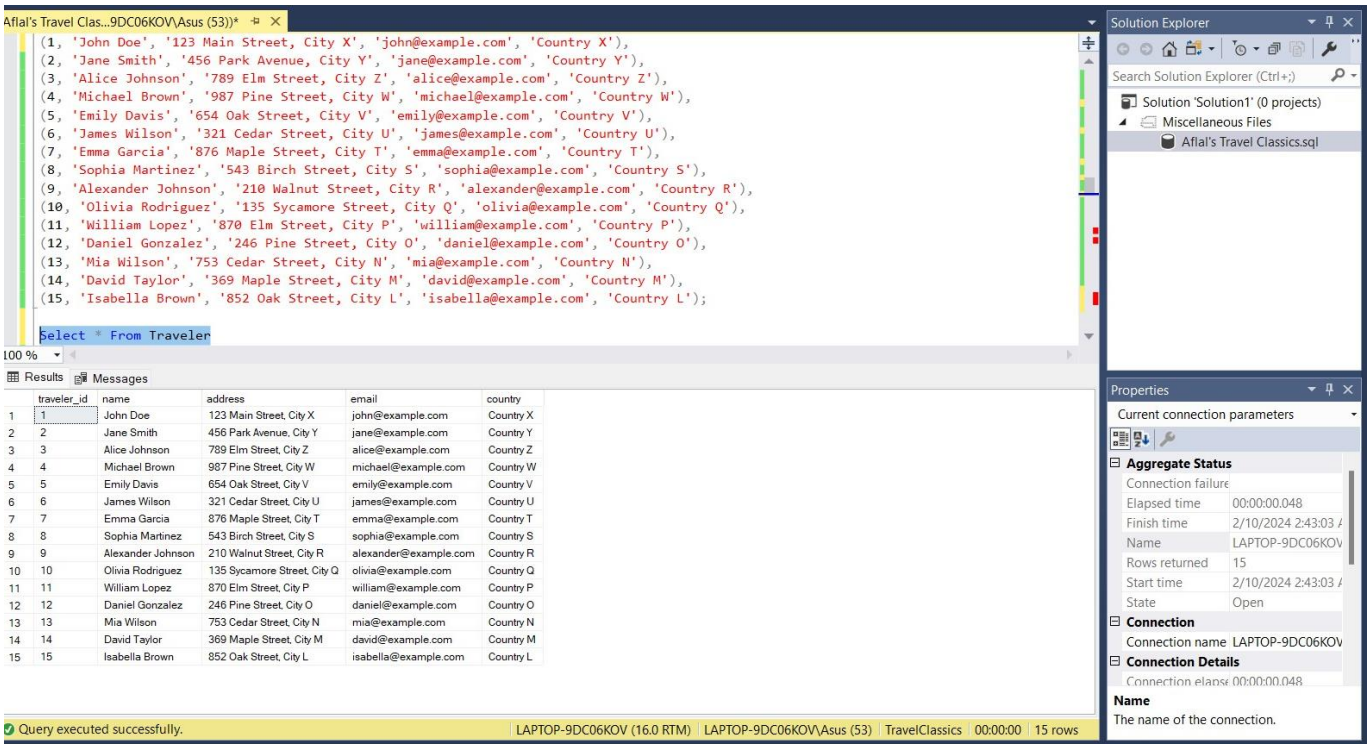

Aflal's Travel Clas...9DC06KOV\Asus (53))*
Create table Room
(
room_id int Primary Key,
hotel_id int,
option_id int,
price Decimal(10, 2),
Foreign Key (hotel_id) References Hotel(hotel_id)
);
Insert into Room (room_id, hotel_id, option_id, price)
Values
(1, 1, 1, 10000.00), -- Room 1 in Hotel A (Standard Room)
(2, 1, 2, 15000.00), -- Room 2 in Hotel A (Deluxe Room)
(3, 1, 3, 20000.00),
(4, 2, 4, 18000.00),
100 %
Results
Messages
room_id hotel_id option_id price
1 1 1 1 10000.00
2 2 1 2 15000.00
3 3 1 3 20000.00
4 4 2 4 18000.00
5 5 2 5 25000.00
6 6 2 6 12000.00
7 7 3 7 30000.00
8 8 3 8 40000.00
9 9 3 9 50000.00
10 10 4 10 8000.00
11 11 4 11 15000.00
12 12 4 12 60000.00
13 13 5 13 20000.00
14 14 5 14 50000.00
15 15 5 15 18000.00
Query executed successfully.
LAPTOP-9DC06KOV (16.0 RTM)
LAPTOP-9DC06KOV\Asus (53)
TravelClassics
00:00:00
15 rows
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Solution1' (0 projects)
Miscellaneous Files
Aflal's Travel Classics.sql
Properties
Current connection parameters
Aggregate Status
Connection failure
Elapsed time 00:00:00.040
Finish time 2/10/2024 2:41:27
Name LAPTOP-9DC06KOV
Rows returned 15
Start time 2/10/2024 2:41:27
State Open
Connection
Connection name LAPTOP-9DC06KOV
Connection Details
Name
The name of the connection.



```
Create table Report
(
    report_id int Primary Key,
    traveler_id int,
    title varchar(100),
    description varchar(255),
    country varchar(50),
    location varchar(100),
    Foreign Key (traveler_id) References Traveler(traveler_id)
);

Insert into Report (report_id, traveler_id, title, description, country, location) Values
(1, 1, 'Exploring the Big Apple', 'My adventures in New York City', 'USA', 'New York'),
(2, 2, 'Bonjour Paris!', 'Discovering the beauty of Paris', 'France', 'Paris'),
(3, 3, 'Roman Holiday', 'My unforgettable trip to Rome', 'Italy', 'Rome'),
(4, 4, 'Barcelona Dreams', 'Exploring the vibrant city of Barcelona', 'Spain', 'Barcelona'),
(5, 5, 'Sydney Spectacular', 'Adventures down under in Sydney', 'Australia', 'Sydney'),
```

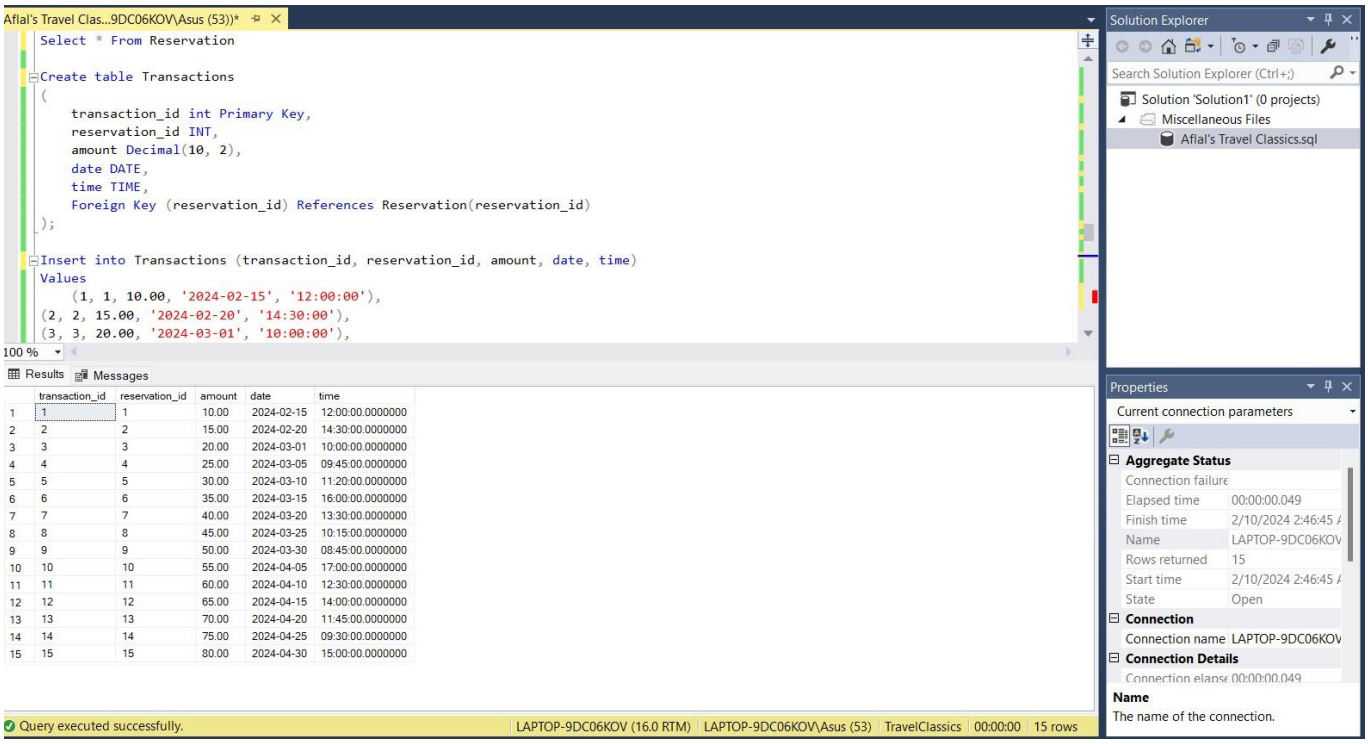
	report_id	traveler_id	title	description	country	location
1	1	1	Exploring the Big Apple	My adventures in New York City	USA	New York
2	2	2	Bonjour Paris!	Discovering the beauty of Paris	France	Paris
3	3	3	Roman Holiday	My unforgettable trip to Rome	Italy	Rome
4	4	4	Barcelona Dreams	Exploring the vibrant city of Barcelona	Spain	Barcelona
5	5	5	Sydney Spectacular	Adventures down under in Sydney	Australia	Sydney
6	6	6	Banff Bliss	Experiencing the natural beauty of Banff	Canada	Banff
7	7	7	Mumbai Magic	Discovering the hustle and bustle of Mumbai	India	Mumbai
8	8	8	Safari Adventure	An unforgettable safari experience in Maasai Mara	Kenya	Maasai Mara
9	9	9	Northern Lights Delight	Chasing the northern lights in Tromsø	Norway	Tromsø
10	10	10	Amazonian Adventure	Exploring the wonders of the Amazon rainforest	Brazil	Manaus
11	11	11	Great Wall Trek	Hiking along the Great Wall of China	China	Beijing
12	12	12	Pyramids Expedition	Unraveling the mysteries of the Egyptian pyramids	Egypt	Cairo
13	13	13	Swiss Wonderland	Discovering the beauty of Switzerland	Switzerland	Zermatt
14	14	14	Dubai Delights	Luxury and opulence in Dubai	UAE	Dubai
15	15	15	Yosemite Adventure	Exploring the wonders of Yosemite National Park	USA	Yosemite



```
(1, 'John Doe', '123 Main Street, City X', 'john@example.com', 'Country X'),
(2, 'Jane Smith', '456 Park Avenue, City Y', 'jane@example.com', 'Country Y'),
(3, 'Alice Johnson', '789 Elm Street, City Z', 'alice@example.com', 'Country Z'),
(4, 'Michael Brown', '987 Pine Street, City W', 'michael@example.com', 'Country W'),
(5, 'Emily Davis', '654 Oak Street, City V', 'emily@example.com', 'Country V'),
(6, 'James Wilson', '321 Cedar Street, City U', 'james@example.com', 'Country U'),
(7, 'Emma Garcia', '876 Maple Street, City T', 'emma@example.com', 'Country T'),
(8, 'Sophia Martinez', '543 Birch Street, City S', 'sophia@example.com', 'Country S'),
(9, 'Alexander Johnson', '210 Walnut Street, City R', 'alexander@example.com', 'Country R'),
(10, 'Olivia Rodriguez', '135 Sycamore Street, City Q', 'olivia@example.com', 'Country Q'),
(11, 'William Lopez', '870 Elm Street, City P', 'william@example.com', 'Country P'),
(12, 'Daniel Gonzalez', '246 Pine Street, City O', 'daniel@example.com', 'Country O'),
(13, 'Mia Wilson', '753 Cedar Street, City N', 'mia@example.com', 'Country N'),
(14, 'David Taylor', '369 Maple Street, City M', 'david@example.com', 'Country M'),
(15, 'Isabella Brown', '852 Oak Street, City L', 'isabella@example.com', 'Country L');

Select * From Traveler
```

	traveler_id	name	address	email	country
1	1	John Doe	123 Main Street, City X	john@example.com	Country X
2	2	Jane Smith	456 Park Avenue, City Y	jane@example.com	Country Y
3	3	Alice Johnson	789 Elm Street, City Z	alice@example.com	Country Z
4	4	Michael Brown	987 Pine Street, City W	michael@example.com	Country W
5	5	Emily Davis	654 Oak Street, City V	emily@example.com	Country V
6	6	James Wilson	321 Cedar Street, City U	james@example.com	Country U
7	7	Emma Garcia	876 Maple Street, City T	emma@example.com	Country T
8	8	Sophia Martinez	543 Birch Street, City S	sophia@example.com	Country S
9	9	Alexander Johnson	210 Walnut Street, City R	alexander@example.com	Country R
10	10	Olivia Rodriguez	135 Sycamore Street, City Q	olivia@example.com	Country Q
11	11	William Lopez	870 Elm Street, City P	william@example.com	Country P
12	12	Daniel Gonzalez	246 Pine Street, City O	daniel@example.com	Country O
13	13	Mia Wilson	753 Cedar Street, City N	mia@example.com	Country N
14	14	David Taylor	369 Maple Street, City M	david@example.com	Country M
15	15	Isabella Brown	852 Oak Street, City L	isabella@example.com	Country L

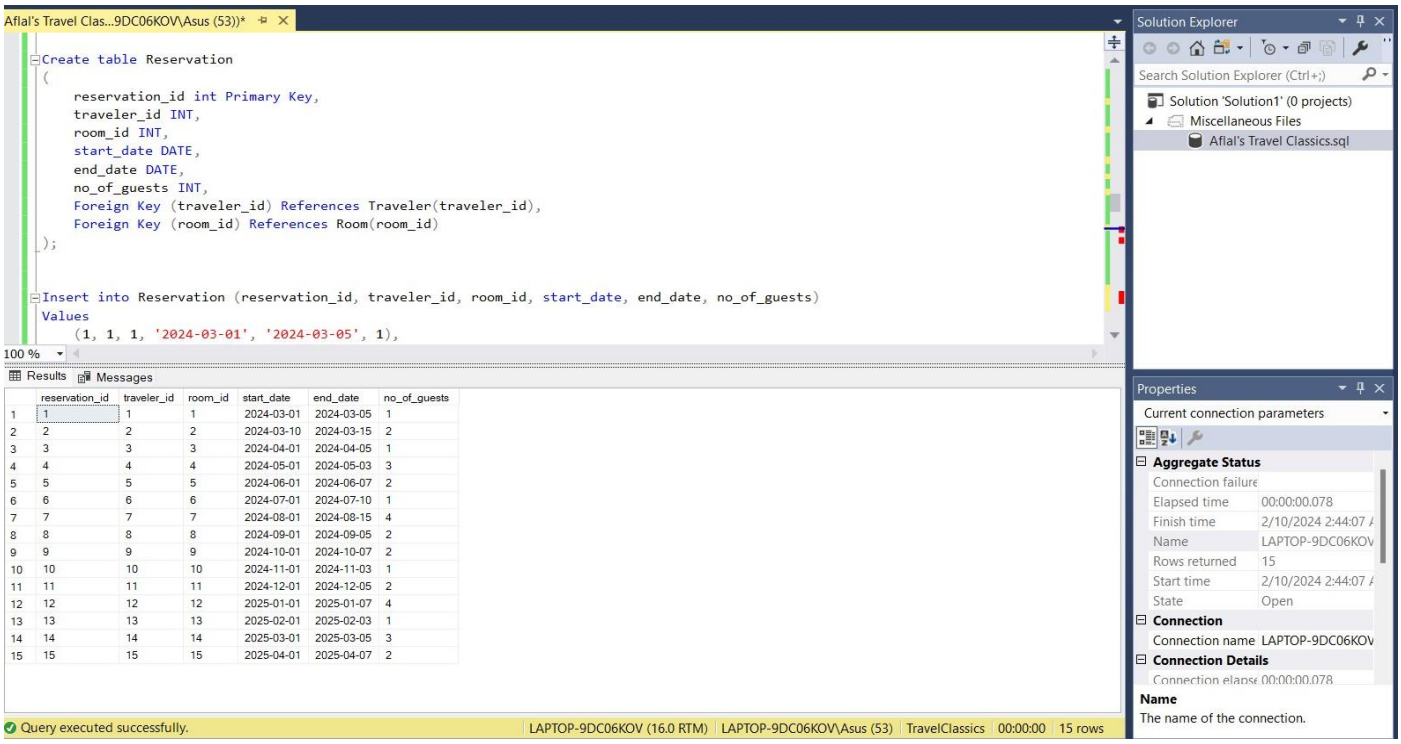


```
Select * From Reservation

Create table Transactions
(
    transaction_id int Primary Key,
    reservation_id INT,
    amount Decimal(10, 2),
    date DATE,
    time TIME,
    Foreign Key (reservation_id) References Reservation(reservation_id)
);

Insert into Transactions (transaction_id, reservation_id, amount, date, time)
Values
    (1, 1, 10.00, '2024-02-15', '12:00:00'),
(2, 2, 15.00, '2024-02-20', '14:30:00'),
(3, 3, 20.00, '2024-03-01', '10:00:00'),
```

	transaction_id	reservation_id	amount	date	time
1	1	1	10.00	2024-02-15	12:00:00.0000000
2	2	2	15.00	2024-02-20	14:30:00.0000000
3	3	3	20.00	2024-03-01	10:00:00.0000000
4	4	4	25.00	2024-03-05	09:45:00.0000000
5	5	5	30.00	2024-03-10	11:20:00.0000000
6	6	6	35.00	2024-03-15	16:00:00.0000000
7	7	7	40.00	2024-03-20	13:30:00.0000000
8	8	8	45.00	2024-03-25	10:15:00.0000000
9	9	9	50.00	2024-03-30	08:45:00.0000000
10	10	10	55.00	2024-04-05	17:00:00.0000000
11	11	11	60.00	2024-04-10	12:30:00.0000000
12	12	12	65.00	2024-04-15	14:00:00.0000000
13	13	13	70.00	2024-04-20	11:45:00.0000000
14	14	14	75.00	2024-04-25	09:30:00.0000000
15	15	15	80.00	2024-04-30	15:00:00.0000000



```
Create table Reservation
(
    reservation_id int Primary Key,
    traveler_id INT,
    room_id INT,
    start_date DATE,
    end_date DATE,
    no_of_guests INT,
    Foreign Key (traveler_id) References Traveler(traveler_id),
    Foreign Key (room_id) References Room(room_id)
);


Insert into Reservation (reservation_id, traveler_id, room_id, start_date, end_date, no_of_guests)
Values
    (1, 1, 1, '2024-03-01', '2024-03-05', 1),
```

	reservation_id	traveler_id	room_id	start_date	end_date	no_of_guests
1	1	1	1	2024-03-01	2024-03-05	1
2	2	2	2	2024-03-10	2024-03-15	2
3	3	3	3	2024-04-01	2024-04-05	1
4	4	4	4	2024-05-01	2024-05-03	3
5	5	5	5	2024-06-01	2024-06-07	2
6	6	6	6	2024-07-01	2024-07-10	1
7	7	7	7	2024-08-01	2024-08-15	4
8	8	8	8	2024-09-01	2024-09-05	2
9	9	9	9	2024-10-01	2024-10-07	2
10	10	10	10	2024-11-01	2024-11-03	1
11	11	11	11	2024-12-01	2024-12-05	2
12	12	12	12	2025-01-01	2025-01-07	4
13	13	13	13	2025-02-01	2025-02-03	1
14	14	14	14	2025-03-01	2025-03-05	3
15	15	15	15	2025-04-01	2025-04-07	2

9.References

Richard Peterson and Peterson, R. (2024) *Entity relationship (ER) diagram model with DBMS example*, *Guru99*. Available at: https://www.guru99.com/er-diagram-tutorial-dbms.html (Accessed: 10 April 2024).

Admin (2022) *Flat data model in DBMS: Gate notes*, *BYJUS*. Available at: https://byjus.com/gate/flat-data-model-in-dbms-notes/ (Accessed: 10 April 2024).

Admin (2022b) *Object-relational data model in DBMS: Gate notes*, *BYJUS*. Available at: https://byjus.com/gate/object-relational-data-model-in-dbms-notes/ (Accessed: 10 April 2024).

Dancuk, M. (2023) *What is an object-oriented database {concepts, examples, pros and cons}*, *Knowledge Base by phoenixNAP*. Available at: https://phoenixnap.com/kb/object-oriented-database (Accessed: 10 April 2024).

Doglio, F. (2022) *What is a decentralized database?*, *Medium*. Available at: https://blog.bitsrc.io/what-is-a-decentralized-database-8ed4edfdc743 (Accessed: 10 April 2024).

Gaurav, S. (2022) *Network model in DBMS*, *Scaler Topics*. Available at: https://www.scaler.com/topics/network-model-in-dbms/ (Accessed: 10 April 2024).

Data verification and data validation techniques (2023) *Intone Networks*. Available at: https://intone.com/data-verification-and-data-validation-methods/ (Accessed: 10 April 2024).

What is a relational database? (no date) *Oracle*. Available at: https://www.oracle.com/database/what-is-a-relational-database/ (Accessed: 10 April 2024).

What is data validation: Definition (no date) *Informatica*. Available at: https://www.informatica.com/services-and-training/glossary-of-terms/data-validation-definition.html (Accessed: 10 April 2024).

Top down design (no date) *Top down Design - an overview | ScienceDirect Topics*. Available at: https://www.sciencedirect.com/topics/computer-science/top-down-design (Accessed: 10 April 2024).