NICE (Nature International Cruise Excellence)

CS-GY 6083 - B, Fall 2024

Submission Date- 12/08/24

BY

Giorgi Merabishvili(GM3386),

Dinesh Sathunuri(DS7675),

Shreya Reddy(SN3871)

TABLE OF CONTENT

1. Executive Summary: 2
2. E-R Model Screenshot: 3
3. Relational Model Screenshot: 4
4. Assumptions for Database Design: 5
1. Entity Definitions: 5
2. Relationships: 5
3. Data Integrity Constraints: 5
4. Booking Constraints: 6
5. Payment Structure: 6
6. Package and Age Constraints: 6
7. Operational Constraints: 6
8. Non-Functional Requirements: 6
9. Data Consistency: 6
5. Software, Programming Language, and Database Details: 7
6. DDL CODE: 8
7. Constraints, Triggers and Procedures: 14
8. DML Code with Sample Data: 16
9. List of Tables: 35
10. Record Counts: 36
11. Web Application Outputs: 37
13. Project Reflections: 46
14. SQL Queries: 47
Table joins with at least 3 tables in join 47
Multi-row subquery 51
Correlated subquery 53
SET operator query 55

1. Executive Summary:

Nature International Cruise Excellence (NICE) is a leading cruise line offering premium travel experiences across 35 ports in the USA, Canada, Mexico, and the Caribbean. With a focus on luxury and comfort, NICE provides a variety of staterooms, from spacious suites to compact accommodations, ensuring a suitable option for every guest. Each ship is thoughtfully divided into four sections: Bow (Forward), Stern (Aft), Port Side (Left), and Starboard Side (Right), optimizing navigation and space utilization for guests.

NICE enhances the cruising experience with diverse dining options, including specialty restaurants featuring Italian, Mexican, Japanese, and Chinese cuisines. Guests can also enjoy 24/7 service at the Round Clock Café or unwind with beverages at various bars. Entertainment and recreational activities are abundant, with theaters, pools, fitness centers, and relaxation spaces catering to all age groups.

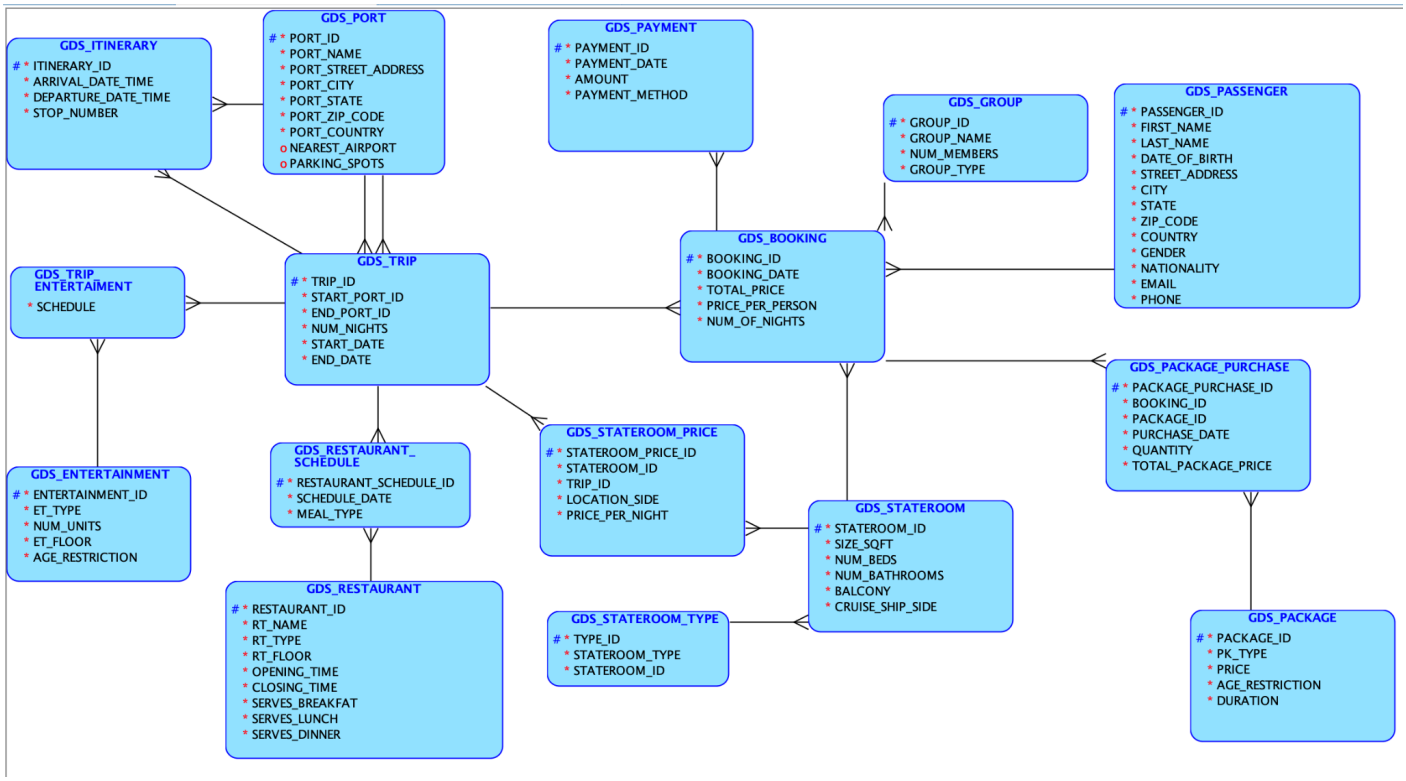
The cruise line's flexible pricing structure, which includes tailored packages for dining, beverages, and internet access, ensures a personalized experience for each traveler. With its commitment to quality service and customer satisfaction, NICE continues to set the standard for excellence in cruising.

The database for the NICE cruise management system is structured using **relational database design principles**. It organizes data into multiple related tables, ensuring normalization, which minimizes redundancy and ensures data integrity.

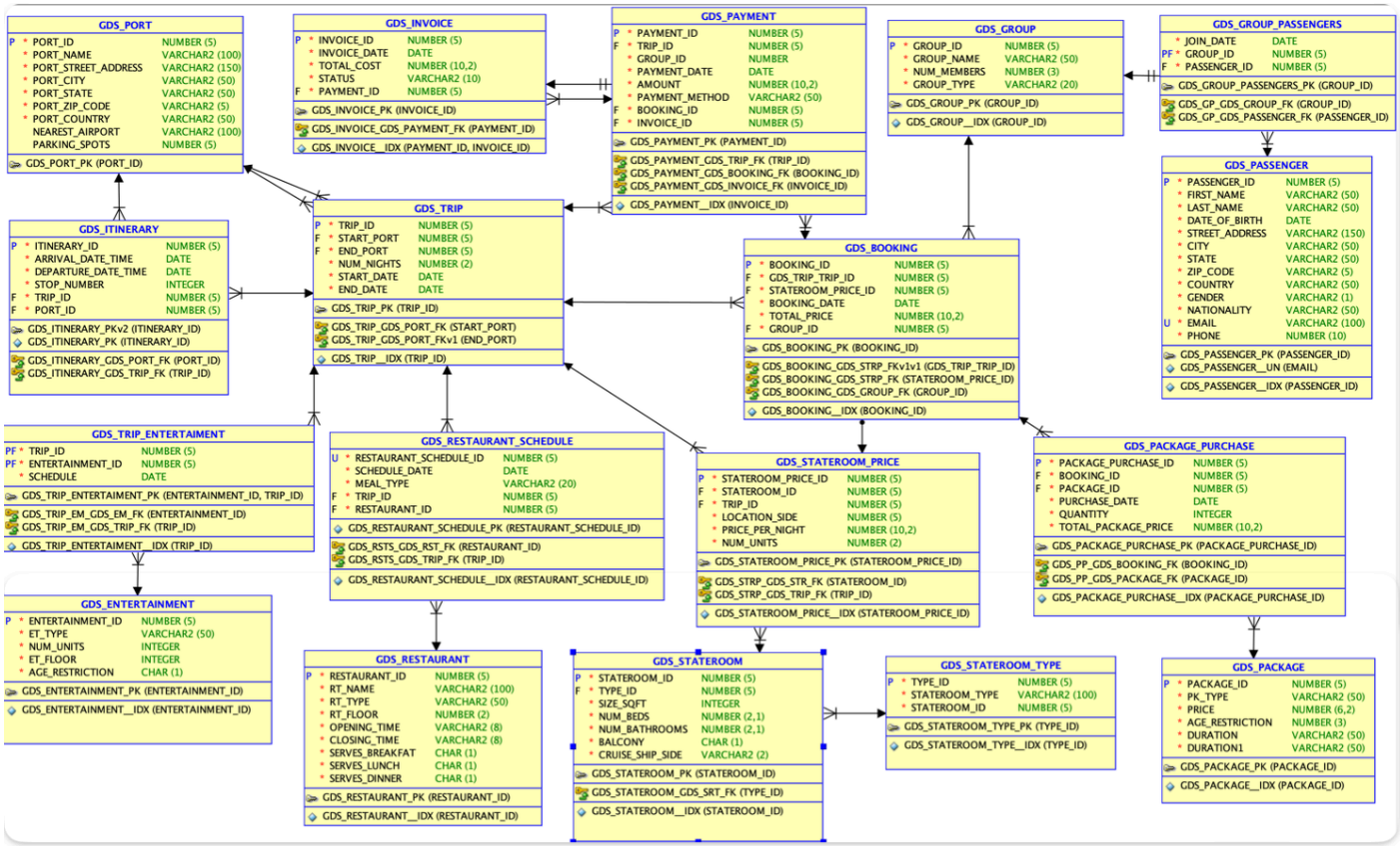
The schema is tailored for a cruise management system, capable of handling bookings, passenger management, and trip planning. It supports both **administrative tasks** (e.g., updating schedules) and **customer interactions** (e.g., making reservations).

Foreign key constraints ensure that relationships between tables are maintained. For efficiency, **Indexes and normalization** reduce storage space and improve query performance. New features or data points can be added with minimal changes to the schema. For security, measures like **triggers and constraints** reduce the risk of inconsistent or malicious data entry.

2. E-R Model Screenshot:



3. Relational Model Screenshot:



4. Assumptions for Database Design:

Entity Definitions:

- Each booking is linked to a unique passenger and can optionally be associated with a group.
- Passengers can have multiple bookings over time but are required to provide detailed personal information for each new booking.
- Every stateroom is associated with a specific type, which dictates its attributes such as size and amenities.
- Trips have distinct start and end ports, and they are scheduled for a fixed number of nights with fixed dates.

Relationships:

- A group can consist of multiple passengers, but each passenger can belong to only one group per booking.
- The itinerary for each trip consists of a defined sequence of stops at different ports.
- Each trip can involve multiple packages, entertainment options, and restaurant schedules.

Data Integrity Constraints:

- The primary keys for all entities within the database are defined as auto-incrementing numbers, guaranteeing the uniqueness of each record
- All foreign key constraints are assumed to maintain referential integrity between related entities, and cascading updates/deletes may not be implemented to prevent accidental data loss.
- All fields marked as NOT NULL are essential for the complete and accurate recording of data, preventing the creation of incomplete records.
- Unlike other countries that are involved, Canada uses 6 digit zip codes, thus zip code constraint is 6 digits.
- State fields can be used for provinces for Canadian addresses since instead of states, Canada has 10 provinces.

Booking Constraints:

- Each booking must have an associated trip and stateroom, and therefore cannot be created without this information.
- Bookings must comply with defined constraints:
    i. The price per person for bookings must be a non-negative value (i.e., price_per_person >= 0).
    ii. The number of nights for each booking must be greater than zero (i.e., num_of_nights > 0).
- Packages purchased can only be referenced against existing bookings, ensuring that only valid combinations exist.

Payment Structure:

- Each payment is unique to a booking and is required to confirm participation in a trip, ensuring that no booking is active without a corresponding payment.
- The payment method in the payment records must reflect valid transaction types recognized by the system.
- Payments must also meet certain constraints, such as the payment amount being non-negative (i.e., amount >= 0).

Package and Age Constraints:

- Each package must have a non-negative age restriction age_restriction >= 0), which will be enforced during the booking process.

Operational Constraints:

- Entertainment options are scheduled in conjunction with trips, and passengers may select applicable entertainment based on trip schedules.
- Each restaurant has a defined operating schedule (opening and closing times) that dictates availability to passengers.

Non-Functional Requirements:

- The system is designed to accommodate multiple concurrent bookings, allowing for scalability during peak seasons with a high volume of guests.

Data Consistency:

- CHECK constraints will be used to enforce business rules, such as ensuring that the price_per_night in the stateroom prices is always a positive number and that num_beds and num_bathrooms are not negative.

- Unique constraints will ensure that no two passengers in the system can have the same email address.

5. Software, Programming Language, and Database Details:

For the development of the web-based user interface, the following technologies were utilized:

1. Frontend Framework:
    - **React** was used to build the user interface. This JavaScript-based library offers a dynamic and responsive design, ensuring an intuitive and user-friendly experience for both customers and employees.

2. Backend Framework:
    - **Spring Boot** was implemented for backend development. This powerful framework simplifies the creation of RESTful APIs and ensures a secure and efficient connection between the web interface and the database.

3. Database:
    - **MySQL** was chosen as the relational database management system. Its reliability, scalability, and compatibility with Spring Boot make it ideal for managing the structured data required for the application.

This combination of technologies enabled the creation of a robust, scalable, and secure web application that meets business needs effectively.

6. DDL CODE:

```
-- Creating tables for MySQL

CREATE TABLE gds_booking (
   booking_id        INT NOT NULL,
   trip_id           INT NOT NULL,
   stateroom_price_id INT NOT NULL,
   booking_date      DATE NOT NULL,
   total_price       DECIMAL(10, 2) NOT NULL,
   group_id          INT NOT NULL,
   PRIMARY KEY (booking_id)
);

COMMENT ON COLUMN gds_booking.booking_id IS 'Unique identifier for each booking';
COMMENT ON COLUMN gds_booking.booking_date IS 'Booking DATE';
COMMENT ON COLUMN gds_booking.total_price IS 'TOTAL PRICE OF THE BOOKING';

CREATE INDEX gds_booking_idx ON gds_booking (booking_id);

CREATE TABLE gds_entertainment (
   entertainment_id INT NOT NULL,
   et_type          VARCHAR(50) NOT NULL,
   num_units        INT NOT NULL,
   et_floor         INT NOT NULL,
   age_restriction  INT NOT NULL,
   PRIMARY KEY (entertainment_id)
);

COMMENT ON COLUMN gds_entertainment.entertainment_id IS 'Unique identifier for each
entertainment type';
COMMENT ON COLUMN gds_entertainment.et_type IS 'Type of entertainment (e.g., Theater,
Pool)';
COMMENT ON COLUMN gds_entertainment.num_units IS 'Number of units available';
COMMENT ON COLUMN gds_entertainment.et_floor IS 'Floor location of the entertainment';

CREATE INDEX gds_entertainment_idx ON gds_entertainment (entertainment_id);

CREATE TABLE gds_group (
   group_id    INT NOT NULL,
   group_name  VARCHAR(50) NOT NULL,
   num_members INT NOT NULL,
```

```
    group_type  VARCHAR(20) NOT NULL,
    PRIMARY KEY (group_id)
);

COMMENT ON COLUMN gds_group.group_id IS 'Unique identifier for each group';
COMMENT ON COLUMN gds_group.group_name IS 'Name of the group (e.g., Family,
Friends)';
COMMENT ON COLUMN gds_group.num_members IS 'Number of members in the group';
COMMENT ON COLUMN gds_group.group_type IS 'LIKE FAMILY, FRIENDS,
INDIVIDUAL';

CREATE INDEX gds_group_idx ON gds_group (group_id);

CREATE TABLE gds_group_passengers (
    join_date    DATE NOT NULL,
    group_id     INT NOT NULL,
    passenger_id INT NOT NULL,
    PRIMARY KEY (group_id)
);

COMMENT ON COLUMN gds_group_passengers.join_date IS 'The Group Join Date';

CREATE TABLE gds_invoice (
    invoice_id   INT NOT NULL,
    invoice_date DATE NOT NULL,
    total_cost   DECIMAL(10, 2) NOT NULL,
    status       VARCHAR(10) NOT NULL,
    payment_id   INT NOT NULL,
    PRIMARY KEY (invoice_id)
);

COMMENT ON COLUMN gds_invoice.status IS 'E.g., "Paid", "Pending", "Overdue"';

CREATE UNIQUE INDEX gds_invoice_idx ON gds_invoice (payment_id, invoice_id);

CREATE TABLE gds_itinerary (
    itinerary_id        INT NOT NULL,
    arrival_date_time   DATETIME NOT NULL,
    departure_date_time DATETIME NOT NULL,
    stop_number         INT NOT NULL,
    trip_id             INT NOT NULL,
    port_id             INT NOT NULL,
    PRIMARY KEY (itinerary_id)
);
```

```
COMMENT ON COLUMN gds_itinerary.arrival_date_time IS 'Arrival of the Itinerary';
COMMENT ON COLUMN gds_itinerary.departure_date_time IS 'Departure of Itinerary';
COMMENT ON COLUMN gds_itinerary.stop_number IS 'Order of the stop in the itinerary';

CREATE TABLE gds_package (
   package_id      INT NOT NULL,
   pk_type         VARCHAR(50) NOT NULL,
   price           DECIMAL(6, 2) NOT NULL,
   age_restriction INT NOT NULL,
   duration        VARCHAR(50) NOT NULL,
   duration1       VARCHAR(50) NOT NULL,
   PRIMARY KEY (package_id)
);

COMMENT ON COLUMN gds_package.package_id IS 'Unique identifier for each package';
COMMENT ON COLUMN gds_package.pk_type IS 'Type of package (entire trip, per night)';
COMMENT ON COLUMN gds_package.price IS 'Price of the package per night or per trip';
COMMENT ON COLUMN gds_package.age_restriction IS 'Age restriction for the package, if
any';
COMMENT ON COLUMN gds_package.duration IS 'like pernight, entire trip';
COMMENT ON COLUMN gds_package.duration1 IS 'Duration';

CREATE INDEX gds_package_idx ON gds_package (package_id);

CREATE TABLE gds_package_purchase (
   package_purchase_id INT NOT NULL,
   booking_id          INT NOT NULL,
   package_id          INT NOT NULL,
   purchase_date       DATE NOT NULL,
   quantity            INT NOT NULL,
   total_package_price DECIMAL(10, 2) NOT NULL,
   PRIMARY KEY (package_purchase_id)
);

COMMENT ON COLUMN gds_package_purchase.package_purchase_id IS 'Unique identifier
for each package purchase.';
COMMENT ON COLUMN gds_package_purchase.purchase_date IS 'Date of the package
purchase';
COMMENT ON COLUMN gds_package_purchase.quantity IS 'Number of units of the package
purchased.';
COMMENT ON COLUMN gds_package_purchase.total_package_price IS 'Total price for the
package purchase.';

CREATE INDEX gds_package_purchase_idx ON gds_package_purchase
(package_purchase_id);
```

```
CREATE TABLE gds_passenger (
   passenger_id   INT NOT NULL,
   first_name     VARCHAR(50) NOT NULL,
   last_name      VARCHAR(50) NOT NULL,
   date_of_birth  DATE NOT NULL,
   street_address VARCHAR(150) NOT NULL,
   city           VARCHAR(50) NOT NULL,
   state          VARCHAR(50) NOT NULL,
   zip_code       VARCHAR(5) NOT NULL,
   country        VARCHAR(50) NOT NULL,
   gender         CHAR(1) NOT NULL,
   nationality    VARCHAR(50) NOT NULL,
   email          VARCHAR(100) NOT NULL,
   phone          VARCHAR(10) NOT NULL,
   PRIMARY KEY (passenger_id),
   UNIQUE (email)
);

COMMENT ON COLUMN gds_passenger.passenger_id IS 'Unique identifier for each
passenger';
COMMENT ON COLUMN gds_passenger.first_name IS 'Passenger"s first name';
COMMENT ON COLUMN gds_passenger.last_name IS 'Passenger"s last name';
COMMENT ON COLUMN gds_passenger.date_of_birth IS 'DATE OF BIRTH';
COMMENT ON COLUMN gds_passenger.street_address IS 'Passenger’s street address';
COMMENT ON COLUMN gds_passenger.zip_code IS 'Postal code for passenger"s address';
COMMENT ON COLUMN gds_passenger.country IS 'Country of the passenger’s address';
COMMENT ON COLUMN gds_passenger.gender IS 'Passenger"s gender (M/F/O)';
COMMENT ON COLUMN gds_passenger.nationality IS 'Passenger"s nationality';
COMMENT ON COLUMN gds_passenger.email IS 'Passenger"s email address, must be unique';
COMMENT ON COLUMN gds_passenger.phone IS 'Passenger"s contact phone number';

CREATE INDEX gds_passenger_idx ON gds_passenger (passenger_id);

CREATE TABLE gds_payment (
   payment_id     INT NOT NULL,
   trip_id        INT NOT NULL,
   group_id       INT NOT NULL,
   payment_date   DATE NOT NULL,
   amount         DECIMAL(10, 2) NOT NULL,
   payment_method VARCHAR(50) NOT NULL,
   booking_id     INT NOT NULL,
   invoice_id     INT NOT NULL,
   PRIMARY KEY (payment_id)
);
```

```
COMMENT ON COLUMN gds_payment.payment_id IS 'Unique identifier for each payment';
COMMENT ON COLUMN gds_payment.trip_id IS 'Identifier for the associated trip';
COMMENT ON COLUMN gds_payment.group_id IS 'Identifier for the associated group';
COMMENT ON COLUMN gds_payment.payment_date IS 'Date when the payment was made';
COMMENT ON COLUMN gds_payment.amount IS 'Payment amount in USD';
COMMENT ON COLUMN gds_payment.payment_method IS 'Method of payment (e.g., Credit
Card, Bank)';

CREATE UNIQUE INDEX gds_payment_idx ON gds_payment (invoice_id);

CREATE TABLE gds_port (
   port_id            INT NOT NULL,
   port_name          VARCHAR(100) NOT NULL,
   port_street_address VARCHAR(150) NOT NULL,
   port_city          VARCHAR(50) NOT NULL,
   port_state         VARCHAR(50) NOT NULL,
   port_zip_code      VARCHAR(5) NOT NULL,
   port_country       VARCHAR(50) NOT NULL,
   nearest_airport    VARCHAR(100),
   parking_spots      INT,
   PRIMARY KEY (port_id)
);

COMMENT ON COLUMN gds_port.port_id IS 'Unique identifier for each port';
COMMENT ON COLUMN gds_port.port_name IS 'Name of the port';
COMMENT ON COLUMN gds_port.port_street_address IS 'Street address of the port';
COMMENT ON COLUMN gds_port.port_city IS 'City where the port is located';
COMMENT ON COLUMN gds_port.port_state IS 'State or province where the port is located';
COMMENT ON COLUMN gds_port.port_zip_code IS 'Postal code of the port’s location';
COMMENT ON COLUMN gds_port.port_country IS 'Country where the port is located';
COMMENT ON COLUMN gds_port.nearest_airport IS 'Name of the nearest airport to the port';
COMMENT ON COLUMN gds_port.parking_spots IS 'Number of parking spots available at the
port';

CREATE TABLE gds_restaurant (
   restaurant_id   INT NOT NULL,
   rt_name         VARCHAR(100) NOT NULL,
   rt_type         VARCHAR(50) NOT NULL,
   rt_floor        INT NOT NULL,
   opening_time    TIME NOT NULL,
   closing_time    TIME NOT NULL,
   serves_breakfast BOOLEAN NOT NULL,
   serves_lunch    BOOLEAN NOT NULL,
   serves_dinner   BOOLEAN NOT NULL,
```

```
    PRIMARY KEY (restaurant_id)
);

COMMENT ON COLUMN gds_restaurant.rt_name IS 'Name of the Restaurant';
COMMENT ON COLUMN gds_restaurant.rt_type IS 'Type of Restaurant';
COMMENT ON COLUMN gds_restaurant.rt_floor IS 'Floor Number';
COMMENT ON COLUMN gds_restaurant.opening_time IS 'Opening Time';
COMMENT ON COLUMN gds_restaurant.closing_time IS 'Closing Time';
COMMENT ON COLUMN gds_restaurant.serves_breakfast IS 'Breakfast Served?';
COMMENT ON COLUMN gds_restaurant.serves_lunch IS 'Lunch Served?';
COMMENT ON COLUMN gds_restaurant.serves_dinner IS 'Dinner Served?';

CREATE TABLE gds_stateroom (
    stateroom_id INT NOT NULL,
    room_type    VARCHAR(50) NOT NULL,
    price        DECIMAL(10, 2) NOT NULL,
    size         VARCHAR(50) NOT NULL,
    capacity     INT NOT NULL,
    PRIMARY KEY (stateroom_id)
);

COMMENT ON COLUMN gds_stateroom.stateroom_id IS 'Unique identifier for each
stateroom';
COMMENT ON COLUMN gds_stateroom.room_type IS 'Type of stateroom (e.g., Oceanview,
Suite)';
COMMENT ON COLUMN gds_stateroom.price IS 'Price per night for the stateroom';
COMMENT ON COLUMN gds_stateroom.size IS 'Size of the stateroom (e.g., 200 sqft)';
COMMENT ON COLUMN gds_stateroom.capacity IS 'Number of passengers that can occupy
the stateroom';

CREATE TABLE gds_trip (
    trip_id          INT NOT NULL,
    start_date       DATE NOT NULL,
    end_date         DATE NOT NULL,
    start_port_id    INT NOT NULL,
    end_port_id      INT NOT NULL,
    nights           INT NOT NULL,
    PRIMARY KEY (trip_id)
);

COMMENT ON COLUMN gds_trip.trip_id IS 'Unique identifier for each trip';
COMMENT ON COLUMN gds_trip.start_date IS 'Start date of the trip';
COMMENT ON COLUMN gds_trip.end_date IS 'End date of the trip';
COMMENT ON COLUMN gds_trip.start_port_id IS 'Starting port for the trip';
COMMENT ON COLUMN gds_trip.end_port_id IS 'Ending port for the trip';
```

```
COMMENT ON COLUMN gds_trip.nights IS 'Number of nights for the trip';

CREATE INDEX gds_trip_idx ON gds_trip (trip_id);
```

7. Constraints, Triggers and Procedures:

Constraints are critical to maintaining the integrity and consistency of the database. The provided code includes several types of constraints:

1. **Primary Key Constraints**
   - Ensure that each record in the table has a unique identifier.

     Example:
     ```
     ALTER TABLE gds_booking ADD CONSTRAINT gds_booking_pk PRIMARY KEY (booking_id);
     ```

   This ensures `booking_id` is unique in the `gds_booking` table.

2. **Foreign Key Constraints**
   - Maintain referential integrity by ensuring values in one table reference valid values in another table.

   Example:
   ```
   ALTER TABLE gds_booking

       ADD CONSTRAINT gds_booking_trip_fk FOREIGN KEY (trip_id)

           REFERENCES gds_trip (trip_id);
   ```

   This ensures that `trip_id` in `gds_booking` exists as a valid `trip_id` in the `gds_trip` table.

3. **Unique Constraints**
   - Ensure that certain columns in a table must have unique values.

   Example:
   ```
   ALTER TABLE gds_passenger ADD CONSTRAINT gds_passenger__un UNIQUE (email);
   ```

   This ensures no two passengers can have the same email address.

4. **Not Null Constraints**
   - Ensure that specific columns cannot have `NULL` values.

   Example:
   ```
   payment_date DATE NOT NULL,
   amount DECIMAL(10, 2) NOT NULL,
   ```

   Columns like `payment_date` and `amount` must always have a value in the

`gds_payment` table.

Constraints: Primary Key, Foreign Key, Unique, Not Null, and Indexes were included.

Triggers: The reduce_stateroom_quantity_after_booking trigger ensures real-time updates to stateroom availability by automatically decreasing the num_units in the gds_stateroom_price table after a booking is made. This prevents overbooking and keeps availability consistent.

Procedure: The processPaymentAndInvoice stored procedure handles payments and invoice updates efficiently. It inserts payment details into the gds_payment table and updates the gds_invoice table to reflect the amount paid, centralizing payment logic and ensuring accuracy.

Indexes on foreign key columns (e.g., start_port, trip_id) optimize JOIN operations, while frequently searched columns (e.g., email, start_date) are indexed to speed up lookups. Composite indexes (e.g., booking_date and total_cost) further improve performance for complex queries. Together, these features enhance application efficiency, reliability, and scalability.

Benefits of These Features

1. **Automation**:
    - The trigger ensures data consistency without requiring manual updates after bookings.
2. **Reusability**:
    - The stored procedure encapsulates payment logic, reducing redundancy and potential errors in the application code.
3. **Performance**:
    - Indexes significantly enhance query performance, especially for complex queries, JOINs, and frequently accessed columns.
4. **Scalability**:
    - These optimizations ensure the application can handle increasing volumes of data efficiently.

By integrating these features, the application becomes more reliable, faster, and easier to maintain, ensuring smooth operations and a better user experience.

8. DML Code with Sample Data:

```
INSERT INTO gds_stateroom_type (stateroom_type)
VALUES
    ('The Haven Suite'),
    ('Club Balcony Suite'),
    ('Family Large Balcony'),
    ('Family Balcony'),
    ('Oceanview Window'),
    ('Inside Stateroom'),
    ('Studio Stateroom');
-- Insert Stateroom Details with updated cruise_ship_side
-- Insert Stateroom Details with all cruise ship sides and stateroom types
INSERT INTO gds_stateroom (type_id, size_sqft, num_beds, num_bathrooms, balcony, cruise_ship_side)
VALUES
    -- The Haven Suite
    (8, 1000, 6, 3, 2, 'BO'),  -- Bow
    (8, 1000, 6, 3, 2, 'ST'),  -- Stern
    (8, 1000, 6, 3, 2, 'PS'),  -- Port Side
    (8, 1000, 6, 3, 2, 'SB'),  -- Starboard Side

    -- Club Balcony Suite
    (9, 800, 4, 2, 2, 'BO'),   -- Bow
    (9, 800, 4, 2, 2, 'ST'),   -- Stern
    (9, 800, 4, 2, 2, 'PS'),   -- Port Side
    (9, 800, 4, 2, 2, 'SB'),   -- Starboard Side
```

```
-- Family Large Balcony
(10, 600, 4, 2, 1, 'BO'),   -- Bow
(10, 600, 4, 2, 1, 'ST'),   -- Stern
(10, 600, 4, 2, 1, 'PS'),   -- Port Side
(10, 600, 4, 2, 1, 'SB'),   -- Starboard Side

-- Family Balcony
(11, 400, 4, 1.5, 1, 'BO'), -- Bow
(11, 400, 4, 1.5, 1, 'ST'), -- Stern
(11, 400, 4, 1.5, 1, 'PS'), -- Port Side
(11, 400, 4, 1.5, 1, 'SB'), -- Starboard Side

-- Oceanview Window
(12, 300, 2, 1, 0, 'BO'),   -- Bow
(12, 300, 2, 1, 0, 'ST'),   -- Stern
(12, 300, 2, 1, 0, 'PS'),   -- Port Side
(12, 300, 2, 1, 0, 'SB'),   -- Starboard Side

-- Inside Stateroom
(13, 200, 2, 1, 0, 'BO'),   -- Bow
(13, 200, 2, 1, 0, 'ST'),   -- Stern
(13, 200, 2, 1, 0, 'PS'),   -- Port Side
(13, 200, 2, 1, 0, 'SB'),   -- Starboard Side

-- Studio Stateroom
(14, 150, 1, 1, 0, 'BO'),   -- Bow
(14, 150, 1, 1, 0, 'ST'),   -- Stern
```

(14, 150, 1, 1, 0, 'PS'), -- Port Side

(14, 150, 1, 1, 0, 'SB'); -- Starboard Side

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (1, 'Port of Miami', '123 Ocean Drive', 'Miami', 'FL', '33131', 'USA', 'Miami International Airport', 200);

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (2, 'Port of Los Angeles', '456 Harbor Blvd', 'Los Angeles', 'CA', '90012', 'USA', 'LAX International Airport', 150);

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (3, 'Port of New York', '789 Liberty St', 'New York', 'NY', '10005', 'USA', 'JFK International Airport', 100);

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (4, 'Port of Galveston', '101 Cruise Terminal', 'Galveston', 'TX', '77550', 'USA', 'Houston Airport', 75);

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (5, 'Port of Seattle', '121 Waterfront', 'Seattle', 'WA', '98101', 'USA', 'Seattle-Tacoma Airport', 80);

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (6, 'Port of San Francisco', '222 Pier St', 'San Francisco', 'CA', '94107', 'USA', 'SFO International Airport', 60);

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (7, 'Port of Vancouver', '333 Cruise Rd', 'Vancouver', 'BC', 'V5K0A', 'Canada', 'YVR International Airport', 90);

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (8, 'Port of Boston', '444 Harbor Ave', 'Boston', 'MA', '02108', 'USA', 'BOS International Airport', 50);

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (9, 'Port of New Orleans', '555 Canal St', 'New Orleans', 'LA', '70112', 'USA', 'MSY International Airport', 120);

INSERT INTO gds_port (port_id, port_name, port_street_address, port_city, port_state, port_zip_code, country, nearest_airport, parking_spots)

VALUES (10, 'Port of San Diego', '666 Harbor Dr', 'San Diego', 'CA', '92101', 'USA', 'SAN International Airport', 100);

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (1, 1, 2, 7, '2025-06-01', '2025-06-08');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (2, 2, 3, 5, '2025-06-09', '2025-06-14');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (3, 3, 4, 3, '2025-06-15', '2025-06-18');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (4, 4, 1, 6, '2025-06-19', '2025-06-25');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (5, 5, 1, 4, '2025-06-26', '2025-06-30');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (6, 6, 7, 10, '2025-07-01', '2025-07-11');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (7, 7, 8, 8, '2025-07-12', '2025-07-20');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (8, 8, 9, 5, '2025-07-21', '2025-07-26');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (9, 9, 10, 3, '2025-07-27', '2025-07-30');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

```
VALUES (10, 10, 1, 6, '2025-08-01', '2025-08-07');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (11, 1, 3, 7, '2025-08-10', '2025-08-17');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (12, 1, 4, 10, '2025-08-18', '2025-08-28');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (13, 5, 6, 5, '2025-08-29', '2025-09-03');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (14, 2, 8, 8, '2025-09-04', '2025-09-12');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (15, 4, 1, 6, '2025-09-13', '2025-09-19');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (16, 3, 5, 7, '2025-09-20', '2025-09-27');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (17, 6, 7, 5, '2025-09-28', '2025-10-02');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (18, 2, 5, 9, '2025-10-03', '2025-10-12');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (19, 8, 9, 4, '2025-10-13', '2025-10-16');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (20, 10, 3, 5, '2025-10-17', '2025-10-21');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (21, 4, 1, 7, '2025-10-22', '2025-10-29');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (22, 7, 2, 11, '2025-10-30', '2025-11-11');
INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)
VALUES (23, 3, 4, 6, '2025-11-12', '2025-11-18');
```

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (24, 5, 6, 8, '2025-11-19', '2025-11-27');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (25, 9, 2, 7, '2025-12-01', '2025-12-08');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (26, 10, 1, 6, '2025-12-09', '2025-12-15');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (27, 7, 3, 6, '2025-12-16', '2025-12-22');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (28, 6, 5, 8, '2025-12-23', '2025-12-30');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (29, 4, 1, 5, '2025-12-31', '2026-01-05');

INSERT INTO gds_trip (trip_id, start_port, end_port, num_nights, start_date, end_date)

VALUES (30, 1, 2, 9, '2026-01-06', '2026-01-15');

INSERT INTO gds_passenger (passenger_id, first_name, last_name, date_of_birth, street_address, zip_code, country, gender, nationality, email, phone)

VALUES (1, 'John', 'Doe', TO_DATE('1985-05-15', 'YYYY-MM-DD'), '123 Main St', '33131', 'USA', 'M', 'American', 'john.doe@example.com', 1234567890);

INSERT INTO gds_passenger

(passenger_id, first_name, last_name, date_of_birth, street_address, city, state, zip_code, country, gender, nationality, email, phone)

VALUES

(2, 'Jane', 'Smith', '1990-07-22', '456 Elm St', 'New York', 'NY', '10005', 'USA', 'F', 'American', 'jane.smith@example.com', 2345678901),

(3, 'Mike', 'Johnson', '1982-09-02', '789 Pine St', 'Los Angeles', 'CA', '90012', 'USA', 'M', 'American', 'mike.johnson@example.com', 3456789012),

(4, 'Sarah', 'Brown', '1995-11-18', '321 Oak St', 'Houston', 'TX', '77001', 'USA', 'F', 'American', 'sarah.brown@example.com', 4567890123),

(5, 'Chris', 'Lee', '1982-01-10', '654 Maple St', 'Seattle', 'WA', '98101', 'USA', 'M', 'American', 'chris.lee@example.com', 5678901234),

(6, 'Katie', 'Wilson', '1993-07-16', '987 Cedar St', 'Miami', 'FL', '33130', 'USA', 'F', 'American', 'katie.wilson@example.com', 6789012345),

(7, 'Tom', 'Garcia', '1989-09-05', '654 Birch St', 'Orlando', 'FL', '32801', 'USA', 'M', 'American', 'tom.garcia@example.com', 7890123456),

(8, 'Emma', 'Martinez', '1990-12-12', '321 Palm St', 'Chicago', 'IL', '60601', 'USA', 'F', 'American', 'emma.martinez@example.com', 8901234567),

(9, 'David', 'Hernandez', '1980-03-10', '222 Walnut St', 'San Francisco', 'CA', '94101', 'USA', 'M', 'American', 'david.hernandez@example.com', 9012345678),

(10, 'Linda', 'Nguyen', '1992-04-28', '999 Fir St', 'Dallas', 'TX', '75201', 'USA', 'F', 'American', 'linda.nguyen@example.com', 1123456789),

(11, 'James', 'Lee', '1986-11-22', '888 Spruce St', 'Austin', 'TX', '73301', 'USA', 'M', 'American', 'james.lee@example.com', 1234567890),

(12, 'Samantha', 'Clark', '1988-06-03', '777 Cypress St', 'Phoenix', 'AZ', '85001', 'USA', 'F', 'American', 'samantha.clark@example.com', 2345678901),

(13, 'Robert', 'Lewis', '1981-02-14', '666 Hemlock St', 'Houston', 'TX', '77001', 'USA', 'M', 'American', 'robert.lewis@example.com', 3456789012),

(14, 'Emily', 'Walker', '1985-09-19', '555 Willow St', 'Jacksonville', 'FL', '32201', 'USA', 'F', 'American', 'emily.walker@example.com', 4567890123),

(15, 'Matthew', 'Hall', '1994-03-25', '444 Cherry St', 'Philadelphia', 'PA', '19101', 'USA', 'M', 'American', 'matthew.hall@example.com', 5678901234),

(16, 'Ashley', 'Allen', '1992-05-20', '333 Maple St', 'Columbus', 'OH', '43215', 'USA', 'F', 'American', 'ashley.allen@example.com', 6789012345),

(17, 'Brian', 'Young', '1989-08-12', '222 Poplar St', 'San Antonio', 'TX', '78201', 'USA', 'M', 'American', 'brian.young@example.com', 7890123456),

(18, 'Megan', 'King', '1996-10-05', '111 Palm St', 'San Diego', 'CA', '92101', 'USA', 'F', 'American', 'megan.king@example.com', 8901234567),

(19, 'Daniel', 'Scott', '1984-12-14', '222 Birch St', 'San Jose', 'CA', '95101', 'USA', 'M', 'American', 'daniel.scott@example.com', 9012345678),

(20, 'Nicole', 'Green', '1992-04-02', '111 Walnut St', 'Nashville', 'TN', '37201', 'USA', 'F', 'American', 'nicole.green@example.com', 1234567890),

(21, 'Tyler', 'Adams', '1995-06-18', '222 Fir St', 'Charlotte', 'NC', '28201', 'USA', 'M', 'American', 'tyler.adams@example.com', 2345678901),

(22, 'Hannah', 'Baker', '1990-01-09', '333 Pine St', 'Detroit', 'MI', '48201', 'USA', 'F', 'American', 'hannah.baker@example.com', 3456789012),

(23, 'Jason', 'Nelson', '1988-05-29', '444 Apple St', 'Baltimore', 'MD', '21201', 'USA', 'M', 'American',

'jason.nelson@example.com', 4567890123),

(24, 'Laura', 'Carter', '1983-08-21', '555 Cherry St', 'Indianapolis', 'IN', '46201', 'USA', 'F', 'American', 'laura.carter@example.com', 5678901234),

(25, 'Henry', 'Mitchell', '1993-03-30', '666 Cedar St', 'Fort Worth', 'TX', '76101', 'USA', 'M', 'American', 'henry.mitchell@example.com', 6789012345),

(26, 'Diana', 'Perez', '1982-07-15', '777 Maple St', 'El Paso', 'TX', '79901', 'USA', 'F', 'American', 'diana.perez@example.com', 7890123456),

(27, 'Tyler', 'Rogers', '1994-09-10', '888 Fir St', 'Boston', 'MA', '02101', 'USA', 'M', 'American', 'tyler.rogers@example.com', 8901234567),

(28, 'Ruth', 'Reed', '1979-12-30', '999 Walnut St', 'Milwaukee', 'WI', '53201', 'USA', 'F', 'American', 'ruth.reed@example.com', 9012345678),

(29, 'Caleb', 'Ward', '1987-02-03', '111 Pine St', 'Tucson', 'AZ', '85701', 'USA', 'M', 'American', 'caleb.ward@example.com', 1234567890),

(30, 'Ella', 'Cook', '1991-10-25', '222 Maple St', 'Fresno', 'CA', '93701', 'USA', 'F', 'American', 'ella.cook@example.com', 2345678901);

INSERT INTO gds_group (group_name, num_members) VALUES

('Adventure Seekers', 5),

('Beach Lovers', 6),

('Family Cruise', 7),

('Senior Travelers', 4),

('Young Professionals', 8);

INSERT INTO gds_passenger_group (passenger_id, group_id, join_date) VALUES

(1, 11, '2024-01-15'), -- Dinesh in Adventure Seekers

(2, 11, '2024-01-15'), -- Jane in Adventure Seekers

(3, 11, '2024-01-15'), -- Mike in Adventure Seekers

(4, 11, '2024-01-15'), -- Sarah in Adventure Seekers

(5, 11, '2024-01-15'), -- Chris in Adventure Seekers

(6, 12, '2024-01-15'), -- Katie in Beach Lovers

(7, 12, '2024-01-15'), -- Tom in Beach Lovers

```
(8, 12, '2024-01-15'),   -- Emma in Beach Lovers
(9, 12, '2024-01-15'),   -- David in Beach Lovers
(10, 12, '2024-01-15'),  -- Linda in Beach Lovers
(11, 12, '2024-01-15'),  -- James in Beach Lovers

(12, 13, '2024-01-15'),  -- Samantha in Family Cruise
(13, 13, '2024-01-15'),  -- Robert in Family Cruise
(14, 13, '2024-01-15'),  -- Emily in Family Cruise
(15, 13, '2024-01-15'),  -- Matthew in Family Cruise
(16, 13, '2024-01-15'),  -- Ashley in Family Cruise
(17, 13, '2024-01-15'),  -- Brian in Family Cruise
(18, 13, '2024-01-15'),  -- Megan in Family Cruise

(19, 14, '2024-01-15'),  -- Daniel in Senior Travelers
(20, 14, '2024-01-15'),  -- Nicole in Senior Travelers
(21, 14, '2024-01-15'),  -- Tyler in Senior Travelers
(22, 14, '2024-01-15'),  -- Hannah in Senior Travelers

(23, 15, '2024-01-15'),  -- Jason in Young Professionals
(24, 15, '2024-01-15'),  -- Laura in Young Professionals
(25, 15, '2024-01-15'),  -- Henry in Young Professionals
(26, 15, '2024-01-15'),  -- Diana in Young Professionals
(27, 15, '2024-01-15'),  -- Tyler in Young Professionals
(28, 15, '2024-01-15'),  -- Ruth in Young Professionals
(29, 15, '2024-01-15'),  -- Caleb in Young Professionals
(30, 15, '2024-01-15');  -- Ella in Young Professionals
```

```
INSERT INTO gds_booking (trip_id, stateroom_price_id, booking_date, total_cost, group_id) VALUES
    (1, 5, '2024-02-01', 5000.00, 11),   -- Adventure Seekers, Trip 1, Stateroom 8 Forward
    (1, 6, '2024-02-01', 4000.00, 12),   -- Beach Lovers, Trip 1, Stateroom 9 Forward
    (1, 7, '2024-02-01', 3000.00, 13),   -- Family Cruise, Trip 1, Stateroom 10 Forward
    (1, 8, '2024-02-01', 2000.00, 14),   -- Senior Travelers, Trip 1, Stateroom 11 Forward
    (1, 9, '2024-02-01', 1500.00, 15),   -- Young Professionals, Trip 1, Stateroom 12 Forward

    (2, 114, '2024-03-15', 5250.00, 11),   -- Adventure Seekers, Trip 2, Stateroom 8 Forward
    (2, 115, '2024-03-15', 4250.00, 12),   -- Beach Lovers, Trip 2, Stateroom 9 Forward
    (2, 116, '2024-03-15', 3250.00, 13),   -- Family Cruise, Trip 2, Stateroom 10 Forward
    (2, 117, '2024-03-15', 2100.00, 14),   -- Senior Travelers, Trip 2, Stateroom 11 Forward
    (2, 118, '2024-03-15', 1550.00, 15);   -- Young Professionals, Trip 2, Stateroom 12 Forward
;
INSERT INTO gds_package (pk_type, price, age_restriction, duration)
VALUES
('Water and Non-Alcoholic', 40.00, 0, 'Per person per night'),
('Unlimited Bar', 80.00, 21, 'Per person per night'),
('Internet 200 minutes, 100 GB', 150.00, 0, 'Per person for entire trip'),
('Unlimited internet', 250.00, 0, 'Per person for entire trip'),
('Specialty dining (Italian, La-carte, Mexican, Japanese, Chinese)', 60.00, 0, 'Per person per night');

INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date,
quantity, total_package_price)
VALUES (1, 1, 1, TO_DATE('2024-01-01', 'YYYY-MM-DD'), 1, 1500.00);
INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date,
quantity, total_package_price)
VALUES (2, 2, 2, TO_DATE('2024-01-02', 'YYYY-MM-DD'), 1, 2000.00);
```

INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date, quantity, total_package_price)

VALUES (3, 3, 3, TO_DATE('2024-01-03', 'YYYY-MM-DD'), 1, 2500.00);

INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date, quantity, total_package_price)

VALUES (4, 4, 4, TO_DATE('2024-01-04', 'YYYY-MM-DD'), 2, 6000.00);

INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date, quantity, total_package_price)

VALUES (5, 5, 5, TO_DATE('2024-01-05', 'YYYY-MM-DD'), 1, 850.00);

INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date, quantity, total_package_price)

VALUES (6, 6, 6, TO_DATE('2024-01-06', 'YYYY-MM-DD'), 1, 1800.00);

INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date, quantity, total_package_price)

VALUES (7, 7, 7, TO_DATE('2024-01-07', 'YYYY-MM-DD'), 2, 2400.00);

INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date, quantity, total_package_price)

VALUES (8, 8, 8, TO_DATE('2024-01-08', 'YYYY-MM-DD'), 1, 1700.00);

INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date, quantity, total_package_price)

VALUES (9, 9, 9, TO_DATE('2024-01-09', 'YYYY-MM-DD'), 1, 5000.00);

INSERT INTO gds_package_purchase (package_purchase_id, booking_id, package_id, purchase_date, quantity, total_package_price)

VALUES (10, 10, 10, TO_DATE('2024-01-10', 'YYYY-MM-DD'), 3, 1800.00);

INSERT INTO gds_payment (payment_id, trip_id, group_id, payment_date, amount, payment_method, booking_id)

VALUES (1, 1, 1, TO_DATE('2024-01-01', 'YYYY-MM-DD'), 700.00, 'Credit Card', 1);

INSERT INTO gds_payment (trip_id, group_id, payment_date, amount, payment_method, booking_id) VALUES

(1, 11, '2024-02-01', 5000.00, 'Credit Card', 15),

(1, 12, '2024-02-01', 4000.00, 'Bank Transfer', 16),

(1, 13, '2024-02-01', 3000.00, 'PayPal', 17),

```
    (1, 14, '2024-02-01', 2000.00, 'Debit Card', 18),
    (1, 15, '2024-02-01', 1500.00, 'Credit Card', 19),

    (2, 11, '2024-03-15', 5250.00, 'Credit Card', 20),
    (2, 12, '2024-03-15', 4250.00, 'Bank Transfer', 21),
    (2, 13, '2024-03-15', 3250.00, 'PayPal', 22),
    (2, 14, '2024-03-15', 2100.00, 'Debit Card', 23),
    (2, 15, '2024-03-15', 1550.00, 'Credit Card', 24);
INSERT INTO gds_entertainment (entertainment_id, et_type, num_units, et_floor, age_restriction)
VALUES (1, 'Theater', 100, 3, 0);
INSERT INTO gds_entertainment (entertainment_id, et_type, num_units, et_floor, age_restriction)
VALUES (2, 'Pool', 50, 1, 0);
INSERT INTO gds_entertainment (entertainment_id, et_type, num_units, et_floor, age_restriction)
VALUES (3, 'Casino', 30, 5, 21);
INSERT INTO gds_entertainment (entertainment_id, et_type, num_units, et_floor, age_restriction)
VALUES (4, 'Spa', 20, 2, 18);
INSERT INTO gds_entertainment (entertainment_id, et_type, num_units, et_floor, age_restriction)
VALUES (5, 'Kids Club', 40, 1, 0);
INSERT INTO gds_entertainment (entertainment_id, et_type, num_units, et_floor, age_restriction)
VALUES (6, 'Live Music', 60, 4, 0);
INSERT INTO gds_entertainment (entertainment_id, et_type, num_units, et_floor, age_restriction)
VALUES (7, 'Art Gallery', 15, 2, 0);
INSERT INTO gds_entertainment (entertainment_id, et_type, num_units, et_floor, age_restriction)
VALUES (8, 'Rock Climbing', 10, 3, 12);

-- Insert into gds_trip_entertainment
INSERT INTO gds_trip_entertainment (trip_id, entertainment_id, schedule)
```

```
VALUES (1, 1, '2024-06-01');

INSERT INTO gds_trip_entertainment (trip_id, entertainment_id, schedule)

VALUES (1, 2, '2024-06-02');

INSERT INTO gds_trip_entertainment (trip_id, entertainment_id, schedule)

VALUES (2, 3, '2024-06-10');

INSERT INTO gds_trip_entertainment (trip_id, entertainment_id, schedule)

VALUES (2, 4, '2024-06-11');

INSERT INTO gds_trip_entertainment (trip_id, entertainment_id, schedule)

VALUES (3, 5, '2024-06-16');

INSERT INTO gds_trip_entertainment (trip_id, entertainment_id, schedule)

VALUES (3, 6, '2024-06-17');

INSERT INTO gds_trip_entertainment (trip_id, entertainment_id, schedule)

VALUES (4, 7, '2024-06-20');

INSERT INTO gds_trip_entertainment (trip_id, entertainment_id, schedule)

VALUES (4, 8, '2024-06-21');


-- Insert into gds_restaurant

INSERT INTO gds_restaurant (restaurant_id, rt_name, rt_type, rt_floor, opening_time, closing_time,
serves_breakfast, serves_lunch, serves_dinner)

VALUES (1, 'Ocean Grill', 'Seafood', 3,

    '2024-01-01 07:00:00',

    '2024-01-01 22:00:00', 1, 1, 1);

INSERT INTO gds_restaurant (restaurant_id, rt_name, rt_type, rt_floor, opening_time, closing_time,
serves_breakfast, serves_lunch, serves_dinner)

VALUES (2, 'Spicy Thai Bistro', 'Thai', 2,

    '2024-01-01 10:00:00',

    '2024-01-01 23:00:00', 0, 1, 1);

INSERT INTO gds_restaurant (restaurant_id, rt_name, rt_type, rt_floor, opening_time, closing_time,
serves_breakfast, serves_lunch, serves_dinner)
```

VALUES (3, 'Pasta Palace', 'Italian', 1,

'2024-01-01 09:00:00',

'2024-01-01 21:00:00', 0, 1, 1);

INSERT INTO gds_restaurant (restaurant_id, rt_name, rt_type, rt_floor, opening_time, closing_time, serves_breakfast, serves_lunch, serves_dinner)

VALUES (4, 'Vegan Delights', 'Vegan', 4,

'2024-01-01 08:00:00',

'2024-01-01 20:00:00', 1, 1, 0);

INSERT INTO gds_restaurant (restaurant_id, rt_name, rt_type, rt_floor, opening_time, closing_time, serves_breakfast, serves_lunch, serves_dinner)

VALUES (5, 'Burger Haven', 'American', 3,

'2024-01-01 11:00:00',

'2024-01-01 22:00:00', 0, 1, 1);

INSERT INTO gds_restaurant (restaurant_id, rt_name, rt_type, rt_floor, opening_time, closing_time, serves_breakfast, serves_lunch, serves_dinner)

VALUES (6, 'Ming Wok Chinese', 'Chinese', 5,

'2024-01-01 12:00:00',

'2024-01-01 20:00:00', 0, 1, 1);

INSERT INTO gds_restaurant (restaurant_id, rt_name, rt_type, rt_floor, opening_time, closing_time, serves_breakfast, serves_lunch, serves_dinner)

VALUES (7, 'Round Clock Café', 'Café', 10,

'2024-01-01 00:00:00',

'2024-01-01 23:59:59', 1, 1, 1);

INSERT INTO gds_restaurant (restaurant_id, rt_name, rt_type, rt_floor, opening_time, closing_time, serves_breakfast, serves_lunch, serves_dinner)

VALUES (8, 'Pool Bar', 'Bar', 10,

'2024-01-01 10:00:00',

'2024-01-01 22:00:00', 0, 0, 0);

INSERT INTO gds_restaurant (restaurant_id, rt_name, type, rt_floor, opening_time, closing_time, serves_breakfast, serves_lunch, serves_dinner)

```
VALUES (9, 'Stout Bar', 'Bar', 7,
    '2024-01-01 10:00:00',
    '2024-01-01 22:00:00', 0, 0, 0);

-- Insert into gds_restaurant_schedule
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-01', 'Breakfast', 1, 1);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-01', 'Lunch', 1, 2);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-01', 'Dinner', 1, 3);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-02', 'Breakfast', 1, 1);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-02', 'Lunch', 1, 2);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-02', 'Dinner', 1, 3);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-09', 'Breakfast', 2, 1);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-09', 'Lunch', 2, 4);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-09', 'Dinner', 2, 3);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-01', 'Lunch', 1, 6);
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)
VALUES ('2024-06-01', 'Dinner', 1, 6);
```

```
INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)

VALUES ('2024-06-01', 'Breakfast', 1, 7);

INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)

VALUES ('2024-06-01', 'Lunch', 1, 7);

INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)

VALUES ('2024-06-01', 'Dinner', 1, 7);

INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)

VALUES ('2024-06-01', 'Beverages', 1, 8);

INSERT INTO gds_restaurant_schedule (schedule_date, meal_type, trip_id, restaurant_id)

VALUES ('2024-06-01', 'Beverages', 1, 9);


-- Insert into gds_itinerary

INSERT INTO gds_itinerary (itinerary_id, trip_id, port_id, arrival_date_time, departure_date_time,
stop_number)

VALUES (1, 1, 1, '2024-06-01 08:00:00', '2024-06-01 17:00:00', 1);

INSERT INTO gds_itinerary (itinerary_id, trip_id, port_id, arrival_date_time, departure_date_time,
stop_number)

VALUES (2, 1, 2, '2024-06-02 08:00:00', '2024-06-02 17:00:00', 2);

INSERT INTO gds_itinerary (itinerary_id, trip_id, port_id, arrival_date_time, departure_date_time,
stop_number)

VALUES (3, 1, 3, '2024-06-03 09:00:00', '2024-06-03 16:00:00', 3);

INSERT INTO gds_itinerary (itinerary_id, trip_id, port_id, arrival_date_time, departure_date_time,
stop_number)

VALUES (4, 2, 4, '2024-06-10 10:00:00', '2024-06-10 19:00:00', 1);

INSERT INTO gds_itinerary (itinerary_id, trip_id, port_id, arrival_date_time, departure_date_time,
stop_number)

VALUES (5, 2, 5, '2024-06-11 10:00:00', '2024-06-11 19:00:00', 2);

INSERT INTO gds_itinerary (itinerary_id, trip_id, port_id, arrival_date_time, departure_date_time,
stop_number)

VALUES (6, 3, 6, '2024-06-16 10:00:00', '2024-06-16 19:00:00', 1);
```

```
INSERT INTO gds_itinerary (itinerary_id, trip_id, port_id, arrival_date_time, departure_date_time, stop_number)

VALUES (7, 4, 7, '2024-06-20 10:00:00', '2024-06-20 19:00:00', 1);

INSERT INTO gds_itinerary (itinerary_id, trip_id, port_id, arrival_date_time, departure_date_time, stop_number)

VALUES (8, 5, 8, '2024-06-25 11:00:00', '2024-06-25 20:00:00', 1);

INSERT INTO gds_itinerary (itinerary_id, trip_id, port_id, arrival_date_time, departure_date_time, stop_number)

VALUES (9, 6, 9, '2024-07-01 09:00:00', '2024-07-01 15:00:00', 1);


INSERT INTO gds_stateroom_price (stateroom_id, trip_id, location_side, price_per_night, num_units)

VALUES

    (39, 1, 'Forward', 1000.00, 6),  -- Stateroom 8, Bow, Trip 1

    (40, 1, 'After', 1000.00, 6),   -- Stateroom 8, Stern, Trip 1

    (41, 1, 'Left', 1000.00, 6),    -- Stateroom 8, Port Side, Trip 1

    (42, 1, 'Right', 1000.00, 6),   -- Stateroom 8, Starboard Side, Trip 1

    (43, 1, 'Forward', 800.00, 4),   -- Stateroom 9, Bow, Trip 1

    (44, 1, 'After', 800.00, 4),    -- Stateroom 9, Stern, Trip 1

    (45, 1, 'Left', 800.00, 4),     -- Stateroom 9, Port Side, Trip 1

    (46, 1, 'Right', 800.00, 4),    -- Stateroom 9, Starboard Side, Trip 1

    (47, 1, 'Forward', 600.00, 4),  -- Stateroom 10, Bow, Trip 1

    (48, 1, 'After', 600.00, 4),   -- Stateroom 10, Stern, Trip 1

    (49, 1, 'Left', 600.00, 4),    -- Stateroom 10, Port Side, Trip 1

    (50, 1, 'Right', 600.00, 4),   -- Stateroom 10, Starboard Side, Trip 1

    (51, 1, 'Forward', 400.00, 4),  -- Stateroom 11, Bow, Trip 1

    (52, 1, 'After', 400.00, 4),   -- Stateroom 11, Stern, Trip 1

    (53, 1, 'Left', 400.00, 4),    -- Stateroom 11, Port Side, Trip 1

    (54, 1, 'Right', 400.00, 4),   -- Stateroom 11, Starboard Side, Trip 1

    (55, 1, 'Forward', 300.00, 2),  -- Stateroom 12, Bow, Trip 1
```

```
    (56, 1, 'After', 300.00, 2),   -- Stateroom 12, Stern, Trip 1
    (57, 1, 'Left', 300.00, 2),    -- Stateroom 12, Port Side, Trip 1
    (58, 1, 'Right', 300.00, 2),   -- Stateroom 12, Starboard Side, Trip 1
    (59, 1, 'Forward', 200.00, 2),  -- Stateroom 13, Bow, Trip 1
    (60, 1, 'After', 200.00, 2),   -- Stateroom 13, Stern, Trip 1
    (61, 1, 'Left', 200.00, 2),    -- Stateroom 13, Port Side, Trip 1
    (62, 1, 'Right', 200.00, 2),   -- Stateroom 13, Starboard Side, Trip 1
    (63, 1, 'Forward', 150.00, 1),  -- Stateroom 14, Bow, Trip 1
    (64, 1, 'After', 150.00, 1),   -- Stateroom 14, Stern, Trip 1
    (65, 1, 'Left', 150.00, 1),    -- Stateroom 14, Port Side, Trip 1
    (66, 1, 'Right', 150.00, 1);   -- Stateroom 14, Starboard Side, Trip 1

-- Insert Stateroom Price for Trip 2
INSERT INTO gds_stateroom_price (stateroom_id, trip_id, location_side, price_per_night, num_units)
VALUES
    (39, 2, 'Forward', 1050.00, 6),  -- Stateroom 8, Bow, Trip 2
    (40, 2, 'After', 1050.00, 6),   -- Stateroom 8, Stern, Trip 2
    (41, 2, 'Left', 1050.00, 6),    -- Stateroom 8, Port Side, Trip 2
    (42, 2, 'Right', 1050.00, 6),   -- Stateroom 8, Starboard Side, Trip 2
    (43, 2, 'Forward', 850.00, 4),   -- Stateroom 9, Bow, Trip 2
    (44, 2, 'After', 850.00, 4),    -- Stateroom 9, Stern, Trip 2
    (45, 2, 'Left', 850.00, 4),     -- Stateroom 9, Port Side, Trip 2
    (46, 2, 'Right', 850.00, 4),    -- Stateroom 9, Starboard Side, Trip 2
    (47, 2, 'Forward', 650.00, 4),  -- Stateroom 10, Bow, Trip 2
    (48, 2, 'After', 650.00, 4),   -- Stateroom 10, Stern, Trip 2
    (49, 2, 'Left', 650.00, 4),    -- Stateroom 10, Port Side, Trip 2
    (50, 2, 'Right', 650.00, 4),   -- Stateroom 10, Starboard Side, Trip 2
```

(51, 2, 'Forward', 420.00, 4), -- Stateroom 11, Bow, Trip 2

(52, 2, 'After', 420.00, 4), -- Stateroom 11, Stern, Trip 2

(53, 2, 'Left', 420.00, 4), -- Stateroom 11, Port Side, Trip 2

(54, 2, 'Right', 420.00, 4), -- Stateroom 11, Starboard Side, Trip 2

(55, 2, 'Forward', 310.00, 2), -- Stateroom 12, Bow, Trip 2

(56, 2, 'After', 310.00, 2), -- Stateroom 12, Stern, Trip 2

(57, 2, 'Left', 310.00, 2), -- Stateroom 12, Port Side, Trip 2

(58, 2, 'Right', 310.00, 2), -- Stateroom 12, Starboard Side, Trip 2

(59, 2, 'Forward', 210.00, 2), -- Stateroom 13, Bow, Trip 2

(60, 2, 'After', 210.00, 2), -- Stateroom 13, Stern, Trip 2

(61, 2, 'Left', 210.00, 2), -- Stateroom 13, Port Side, Trip 2

(62, 2, 'Right', 210.00, 2), -- Stateroom 13, Starboard Side, Trip 2

(63, 2, 'Forward', 160.00, 1), -- Stateroom 14, Bow, Trip 2

(64, 2, 'After', 160.00, 1), -- Stateroom 14, Stern, Trip 2

(65, 2, 'Left', 160.00, 1), -- Stateroom 14, Port Side, Trip 2

(66, 2, 'Right', 160.00, 1); -- Stateroom 14, Starboard Side, Trip 2

9. List of Tables:

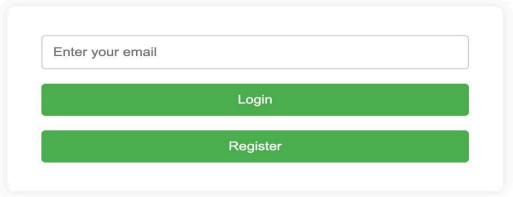
	gds_tables
	gds_booking
	gds_entertainment
	gds_group
	gds_invoice
	gds_itinerary
	gds_package
	gds_package_purchase
	gds_passenger
	gds_passenger_group
	gds_payment
	gds_port
	gds_restaurant
	gds_restaurant_schedule
	gds_stateroom
	gds_stateroom_price
	gds_stateroom_type
	gds_trip
	gds_trip_entertainment

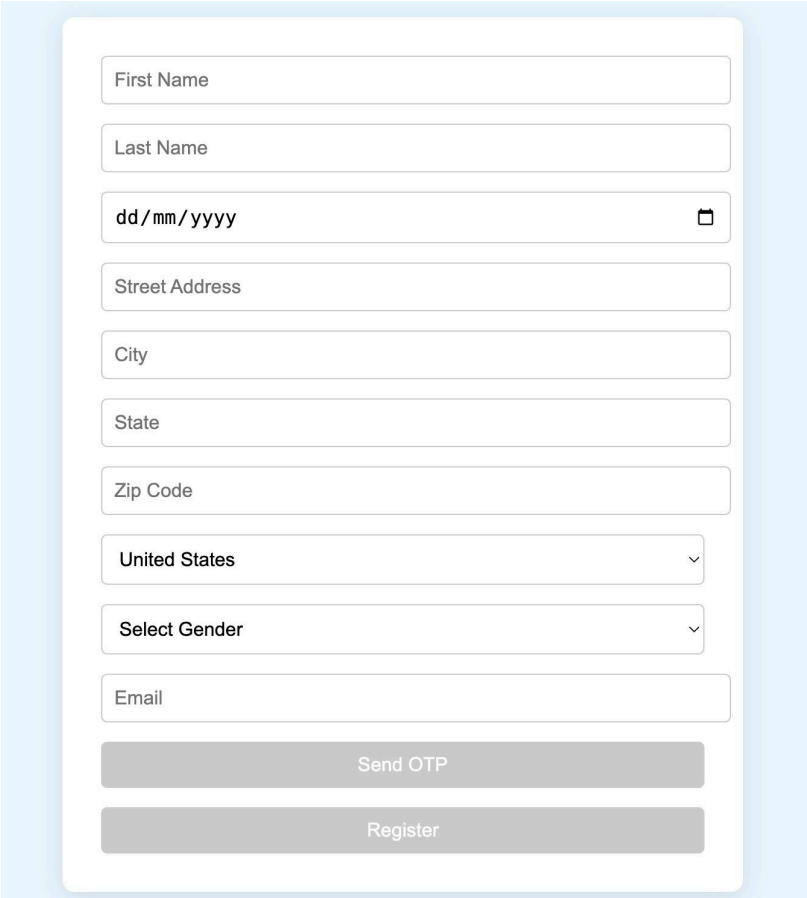
10. Record Counts:

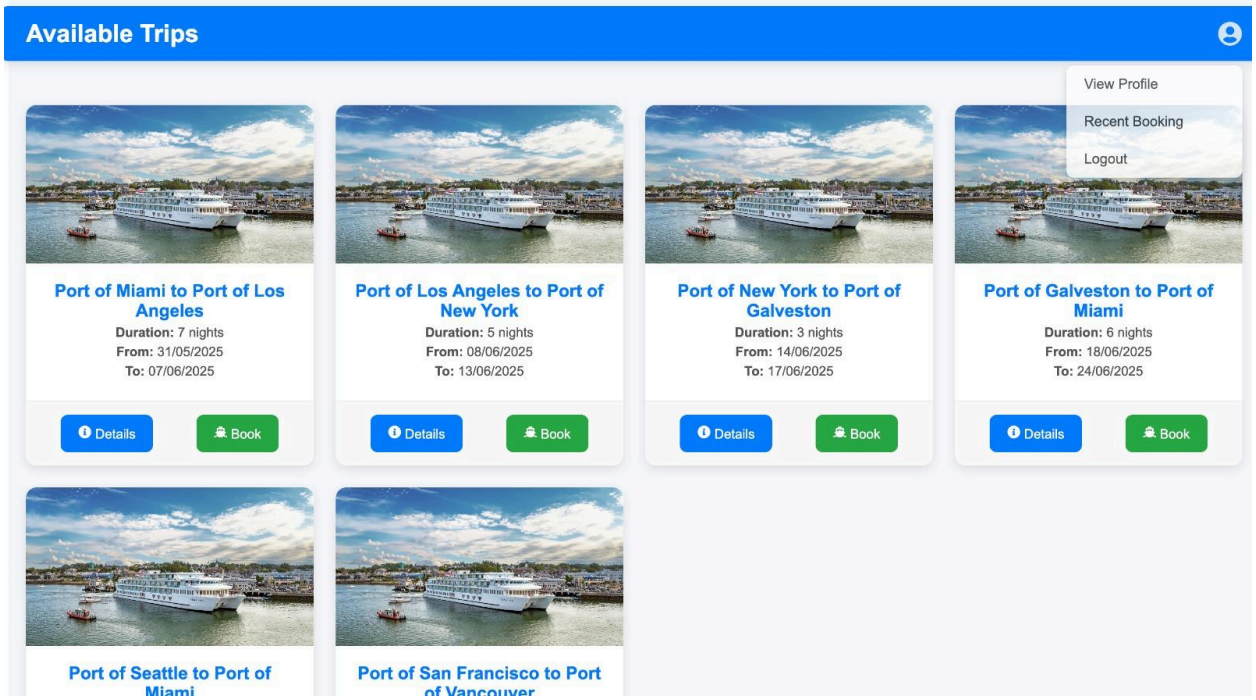
	table_name	row_count
	gds_stateroom_type	7
	gds_group	7
	gds_passenger	30
	gds_port	10
	gds_trip	31
	gds_stateroom	28
	gds_entertainment	9
	gds_restaurant	10
	gds_booking	12
	gds_itinerary	9
	gds_package	5
	gds_package_purchase	5
	gds_restaurant_schedule	17
	gds_stateroom_price	56
	gds_trip_entertainment	8
	gds_payment	12
	gds_invoice	12

11. Web Application Outputs:

For Users:




First Name
Last Name
dd/mm/yyyy
Street Address
City
State
Zip Code
United States
Select Gender
Email
Send OTP
Register


Available Trips
View Profile
Recent Booking
Logout
Port of Miami to Port of Los Angeles
Duration: 7 nights
From: 31/05/2025
To: 07/06/2025
Details
Book
Port of Los Angeles to Port of New York
Duration: 5 nights
From: 08/06/2025
To: 13/06/2025
Details
Book
Port of New York to Port of Galveston
Duration: 3 nights
From: 14/06/2025
To: 17/06/2025
Details
Book
Port of Galveston to Port of Miami
Duration: 6 nights
From: 18/06/2025
To: 24/06/2025
Details
Book
Port of Seattle to Port of Miami
Port of San Francisco to Port of Vancouver

Available Staterooms

Select a Stateroom

The Haven Suite
Size: 1000 sqft
Beds: 6
Bathrooms: 3
Balcony: 2
Cruise Ship Side: BO
Location Side: Forward
Price per Night: $1000

The Haven Suite
Size: 1000 sqft
Beds: 6
Bathrooms: 3
Balcony: 2
Cruise Ship Side: ST
Location Side: After
Price per Night: $1000

The Haven Suite
Size: 1000 sqft
Beds: 6
Bathrooms: 3
Balcony: 2
Cruise Ship Side: PS
Location Side: Left
Price per Night: $1000

The Haven Suite
Size: 1000 sqft
Beds: 6
Bathrooms: 3
Balcony: 2
Cruise Ship Side: SB
Location Side: Right
Price per Night: $1000

Club Balcony Suite
Size: 800 sqft
Beds: 4
Bathrooms: 2
Balcony: 2
Cruise Ship Side: BO
Location Side: Forward
Price per Night: $800

Club Balcony Suite
Size: 800 sqft
Beds: 4
Bathrooms: 2
Balcony: 2
Cruise Ship Side: ST
Location Side: After
Price per Night: $800

Club Balcony Suite
Size: 800 sqft
Beds: 4
Bathrooms: 2
Balcony: 2
Cruise Ship Side: PS
Location Side: Left
Price per Night: $800

Club Balcony Suite
Size: 800 sqft
Beds: 4
Bathrooms: 2
Balcony: 2
Cruise Ship Side: SB
Location Side: Right
Price per Night: $800

Available Packages

Water and Non-Alcoholic
Price: $40 USD
Duration: Per person per night days
- 0 +

Unlimited Bar
Price: $80 USD
Duration: Per person per night days
- 1 +

Internet 200 minutes, 100 GB
Price: $150 USD
Duration: Per person for entire trip days
- 1 +

Unlimited internet
Price: $250 USD
Duration: Per person for entire trip days
- 1 +

Specialty dining (Italian, La-carte, Mexican, Japanese, Chinese)
Price: $60 USD
Duration: Per person per night days
- 0 +

User Details

User Information

First Name: Dinesh
Last Name: Sathunuri
Email: dsathunuri@gmail.com
Date of Birth: 1999-08-29
Street Address: 179 Clendenny Avenue
City: Jersey City
State: New Jersey
Zip Code: 07304
Country: United States
Gender: M
Nationality: Indian
Phone: 9298237775

Edit

Total Cost

Stateroom	$1050.00
Unlimited Bar	$80.00
Internet 200 minutes, 100 GB	$150.00
Unlimited internet	$250.00
Total Cost	$1530.00

Payment Method

Credit Card

Card Number

Enter your card number

Expiry Date

MM/YY

CVV

Enter CVV

Edit Group | Edit Packages & Stateroom | Finalize Booking | Cancel

Booking Summary

Trip Details

From: Port of Miami
To: Port of Los Angeles
Duration: 7 nights
Start Date: 31/05/2025
End Date: 07/06/2025

Group Name

Group Name: Single

Group Members

Dinesh Sathunuri dsathunuri@gmail.com

Selected Stateroom

Type: Studio Stateroom
Price per Night: $150

Selected Packages

Unlimited Bar - $80 (Quantity: 1)
Internet 200 minutes, 100 GB - $150 (Quantity: 1)
Unlimited internet - $250 (Quantity: 1)

Past Bookings

Go to Home

Booking ID: 2

Download Booking Details PDF | Download Invoice PDF

- Water and Non-Alcoholic (Qty: 1) - $40 per person per night
- Unlimited Bar (Qty: 1) - $80 per person per night

- **Dinesh Sathunuri** - dsathunuri@gmail.com
- **Ella Cook** - ella.cook@example.com

- Oceanview Window - 300 sqft, 2 beds, 1 bathroom(s)

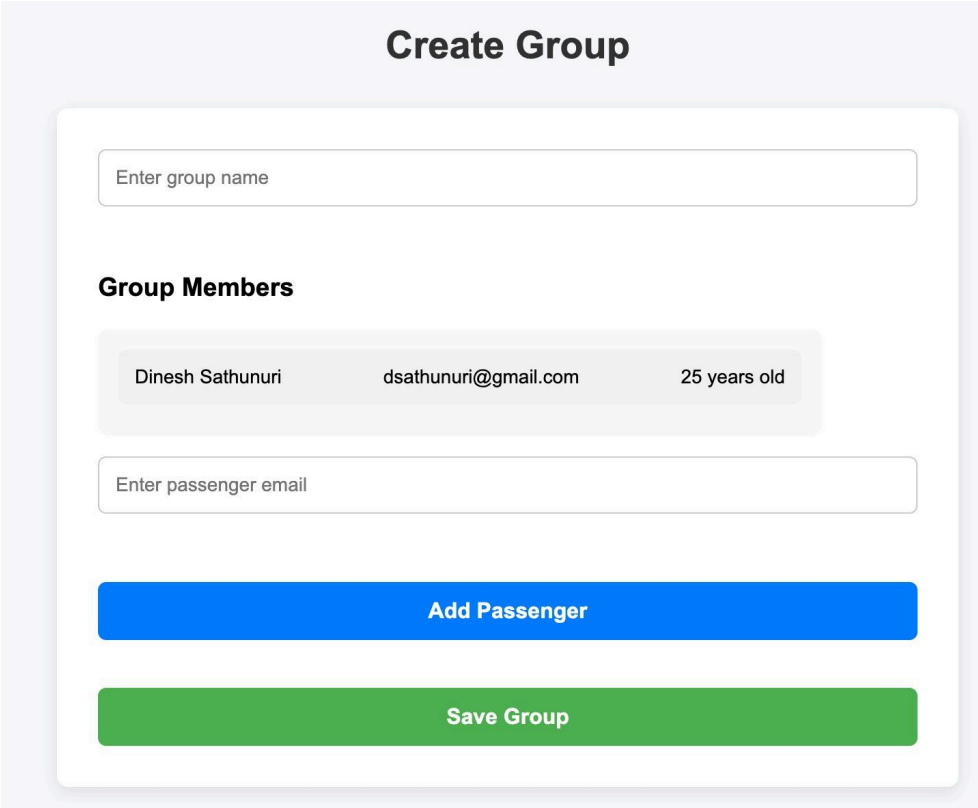
Booking ID: 4

Download Booking Details PDF | Download Invoice PDF

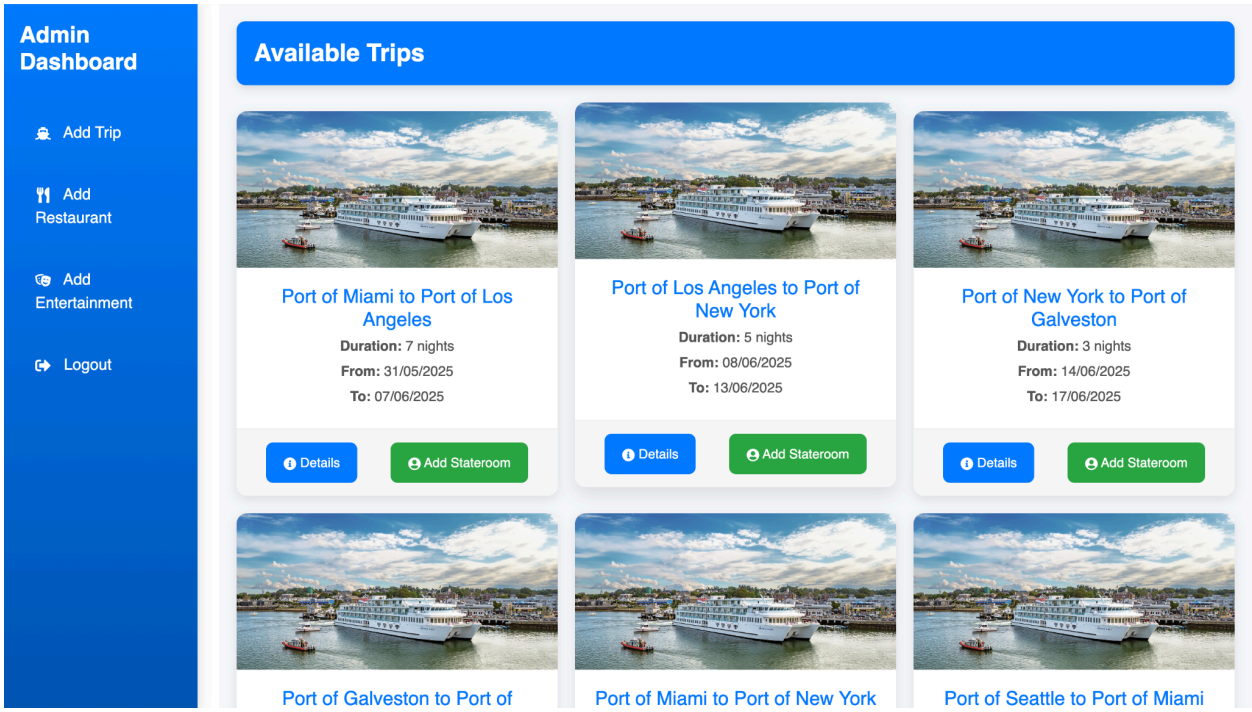
- Unlimited Bar (Qty: 1) - $80 per person per night
- Internet 200 minutes, 100 GB (Qty: 1) - $150 per person per night
- Unlimited internet (Qty: 1) - $250 per person per night

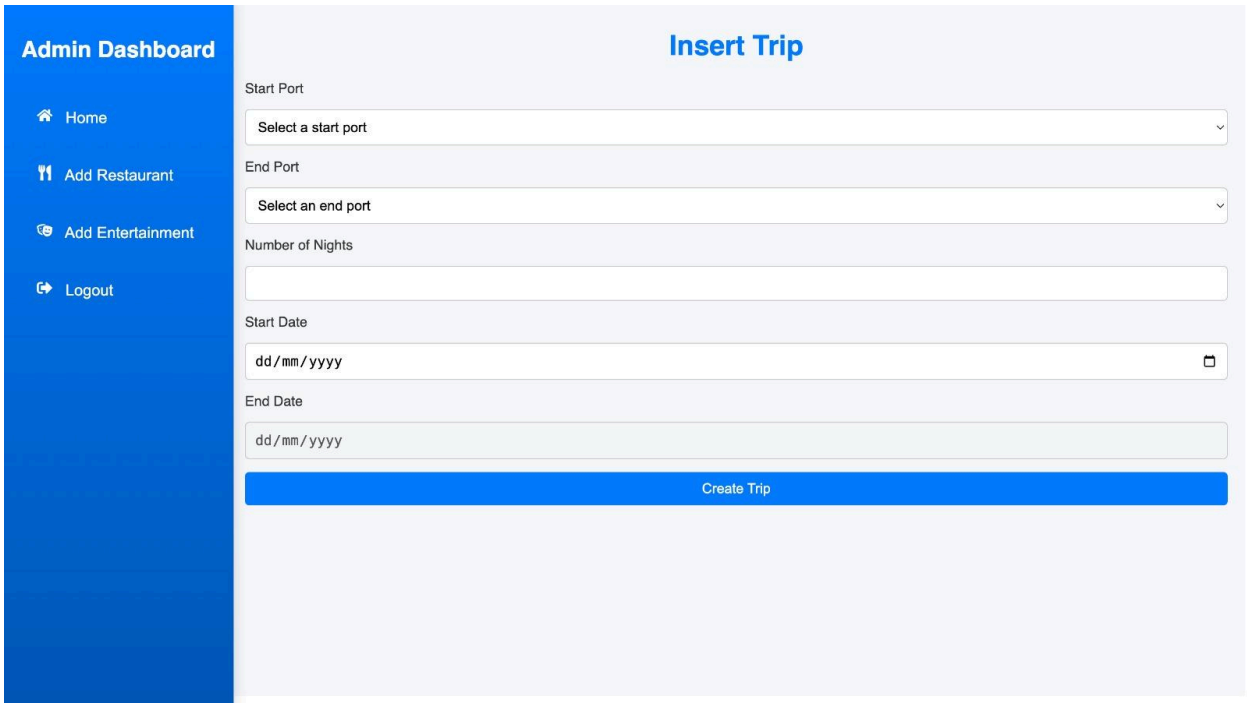
- **Dinesh Sathunuri** - dsathunuri@gmail.com

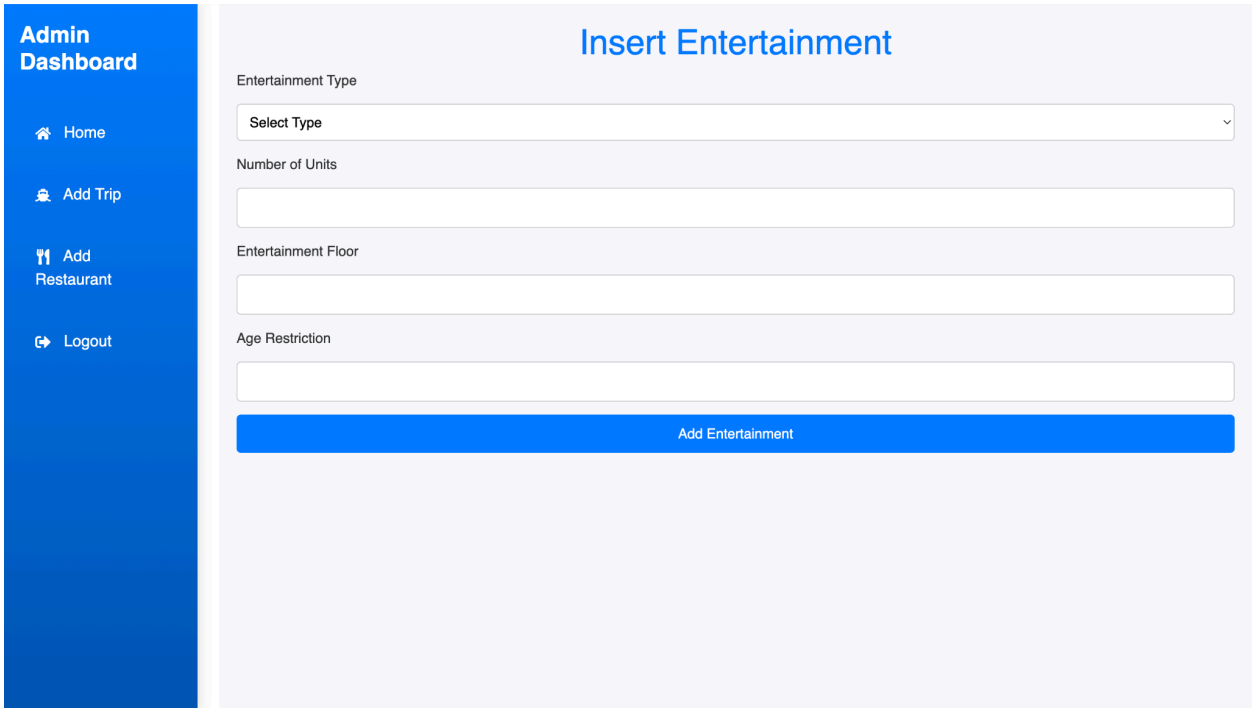
- Studio Stateroom - 150 sqft, 1 beds, 1 bathroom(s)


Create Group
Enter group name
Group Members
Dinesh Sathunuri
dsathunuri@gmail.com
25 years old
Enter passenger email
Add Passenger
Save Group

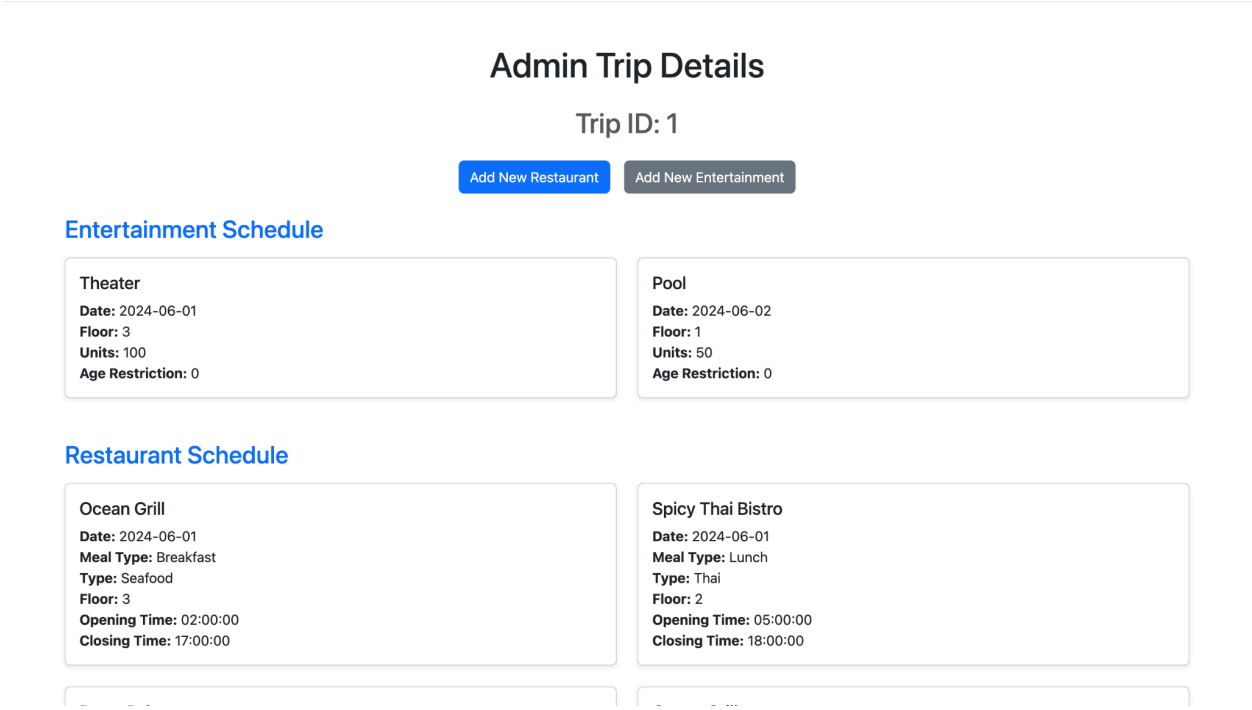
For Admin:


Admin Dashboard
Add Trip
Add Restaurant
Add Entertainment
Logout
Available Trips
Port of Miami to Port of Los Angeles
Duration: 7 nights
From: 31/05/2025
To: 07/06/2025
Details
Add Stateroom
Port of Los Angeles to Port of New York
Duration: 5 nights
From: 08/06/2025
To: 13/06/2025
Details
Add Stateroom
Port of New York to Port of Galveston
Duration: 3 nights
From: 14/06/2025
To: 17/06/2025
Details
Add Stateroom
Port of Galveston to Port of
Port of Miami to Port of New York
Port of Seattle to Port of Miami


Admin Dashboard
Home
Add Restaurant
Add Entertainment
Logout
Insert Trip
Start Port
Select a start port
End Port
Select an end port
Number of Nights
Start Date
dd/mm/yyyy
End Date
dd/mm/yyyy
Create Trip


Admin Dashboard
Home
Add Trip
Add Restaurant
Logout
Insert Entertainment
Entertainment Type
Select Type
Number of Units
Entertainment Floor
Age Restriction
Add Entertainment

Admin Dashboard
Home
Add Trip
Add Entertainment
Logout
Insert Restaurant
Restaurant Name
Restaurant Type
Restaurant Floor
Opening Time
--:-- --
Closing Time
--:-- --
Serves Breakfast
No
Serves Lunch
No
Serves Dinner


Admin Trip Details
Trip ID: 1
Add New Restaurant
Add New Entertainment
Entertainment Schedule
Theater
Date: 2024-06-01
Floor: 3
Units: 100
Age Restriction: 0
Pool
Date: 2024-06-02
Floor: 1
Units: 50
Age Restriction: 0
Restaurant Schedule
Ocean Grill
Date: 2024-06-01
Meal Type: Breakfast
Type: Seafood
Floor: 3
Opening Time: 02:00:00
Closing Time: 17:00:00
Spicy Thai Bistro
Date: 2024-06-01
Meal Type: Lunch
Type: Thai
Floor: 2
Opening Time: 05:00:00
Closing Time: 18:00:00

12. Security Features:

In our web application, we implemented several security measures to ensure data integrity and reliability. One key approach was using the @Transactional annotation in the booking creation process. This ensured all database operations—such as inserting groups, creating bookings, adding package purchases, assigning passengers, and processing payments—were executed as a single atomic transaction. If any step, like payment processing, failed, the entire transaction was rolled back automatically, preventing partial updates and maintaining data consistency.

For user authentication, we adopted an OTP-based method instead of traditional passwords. This approach enhances security by requiring a dynamically generated One-Time Password (OTP) for user login, which is sent to their registered email or phone. This eliminates risks associated with password theft, such as brute force attacks or credential reuse, and ensures secure and time-sensitive access to the system.

To prevent deadlock scenarios in database operations, we implemented the FOR UPDATE WAIT 15 SQL statement. This technique allows transactions to wait for up to 15 seconds to acquire a lock on a resource before timing out. It helps manage high-concurrency situations by reducing the chances of deadlocks while ensuring smooth operation under load.

In addition to the previously mentioned security measures, we also implemented session monitoring through cookies. Upon user login, a session cookie is created, which tracks the user's active session across requests. This ensures that only authenticated users can access sensitive parts of the application. The session is terminated once the user logs out, and the cookie is cleared, enhancing session security and preventing unauthorized access after the user has logged out. This approach helps maintain user authentication integrity throughout their interaction with the application.

In addition to the other security measures, we also leveraged JDBC Template to prevent SQL injection. By using parameterized queries with JDBC Template, we ensure that user inputs are handled safely, avoiding the risks associated with SQL injection attacks. This approach sanitizes inputs, making it harder for attackers to manipulate queries and access or alter sensitive data.

These measures significantly improved the security and reliability of our application, ensuring a safe and consistent user experience while addressing common system vulnerabilities.

13. Project Reflections:

One of the key lessons learned during the project was the importance of effective planning and time management. Breaking tasks into smaller milestones and prioritizing them helped in managing the workload efficiently, though some unexpected delays highlighted areas for improvement.

The project provided an opportunity to enhance technical skills, especially in database design, backend development with Spring Boot, and frontend integration using React. Implementing these technologies in a coordinated manner was both challenging and rewarding.

Collaborating remotely with the team underscored the importance of clear communication and using collaborative tools like Git,intellij and Visual Studio. Regular updates and meetings ensured alignment, though time zone differences occasionally caused delays.

Security was a significant focus, and implementing measures like transactional workflows, OTP-based authentication, and deadlock prevention provided valuable insights into building secure systems. These measures improved both the application's reliability and user trust.

Overall, the project highlighted the importance of adaptability, continuous learning, and teamwork in overcoming constraints and delivering a robust solution.

14. SQL Queries:

Table joins with at least 3 tables in join.

```
SELECT
  gs.stateroom_id,
  gst.stateroom_type AS type,
  gs.size_sqft,
  gs.num_beds,
  gs.num_bathrooms,
  gs.balcony,
  gs.cruise_ship_side,
  gsp.stateroom_price_id,
  gsp.trip_id,
  gsp.location_side,
  gsp.price_per_night,
  gsp.num_units
FROM
  gds_stateroom gs
JOIN
  gds_stateroom_type gst
  ON gs.type_id = gst.type_id
JOIN
  gds_stateroom_price gsp
  ON gs.stateroom_id = gsp.stateroom_id
WHERE
  gsp.trip_id = ?;
```

stateroom_id	type	size_sqft	num_beds	num_bathroo...	balcony	cruise_ship_si...	stateroom_price_id	trip_id	location_side	price_per_ni...	nu
39	The Haven Suite	1000	6	3	2	BO	1	1	Forward	1000.00	6
40	The Haven Suite	1000	6	3	2	ST	2	1	After	1000.00	6
41	The Haven Suite	1000	6	3	2	PS	3	1	Left	1000.00	6
42	The Haven Suite	1000	6	3	2	SB	4	1	Right	1000.00	6
43	Club Balcony Suite	800	4	2	2	BO	5	1	Forward	800.00	3
44	Club Balcony Suite	800	4	2	2	ST	6	1	After	800.00	3
45	Club Balcony Suite	800	4	2	2	PS	7	1	Left	800.00	3
46	Club Balcony Suite	800	4	2	2	SB	8	1	Right	800.00	3
47	Family Large Balcony	600	4	2	1	BO	9	1	Forward	600.00	3
48	Family Large Balcony	600	4	2	1	ST	10	1	After	600.00	4
49	Family Large Balcony	600	4	2	1	PS	11	1	Left	600.00	4
50	Family Large Balcony	600	4	2	1	SB	12	1	Right	600.00	4
51	Family Balcony	400	4	2	1	BO	13	1	Forward	400.00	4
52	Family Balcony	400	4	2	1	ST	14	1	After	400.00	4
53	Family Balcony	400	4	2	1	PS	15	1	Left	400.00	4
54	Family Balcony	400	4	2	1	SB	16	1	Right	400.00	4
55	Oceanview Window	300	2	1	0	BO	17	1	Forward	300.00	2
56	Oceanview Window	300	2	1	0	ST	18	1	After	300.00	2

This query is designed to retrieve detailed information about the available staterooms for a

specific trip, allowing the business to identify which staterooms meet the needs of passengers who have booked or are considering booking a trip. The data retrieved includes the stateroom's characteristics, such as size, number of beds and bathrooms, balcony availability, and pricing details for each stateroom.

The business purpose of this query is to:

- Help customers choose suitable staterooms based on their preferences for type, size, and amenities (e.g., balcony, number of beds, bathrooms).
- Enable pricing analysis by providing details on the price per night for each stateroom, allowing the business to manage room pricing effectively and optimize occupancy.
- Support the reservation process, ensuring that passengers can be offered available rooms that fit their preferences and budget for a given trip.

```
SELECT
  gds_booking.booking_id,
  gds_booking.trip_id,
  gds_booking.stateroom_price_id,
  gds_booking.booking_date,
  gds_booking.total_cost,
  gds_group.group_id,
  gds_group.group_name,
  gds_group.num_members,
  gds_passenger.passenger_id,
  gds_passenger.first_name,
  gds_passenger.last_name,
  gds_passenger.email,
  gds_passenger.phone
FROM
  gds_booking
INNER JOIN
  gds_group ON gds_booking.group_id = gds_group.group_id
INNER JOIN
  gds_passenger_group ON gds_group.group_id = gds_passenger_group.group_id
INNER JOIN
```

```
  gds_passenger ON gds_passenger_group.passenger_id = gds_passenger.passenger_id
WHERE
  gds_group.group_id IN (
    SELECT gds_group.group_id
    FROM gds_passenger_group
    INNER JOIN gds_passenger ON gds_passenger_group.passenger_id =
gds_passenger.passenger_id
    WHERE gds_passenger.email = ?
  )
ORDER BY gds_booking.booking_id;
```

booking_id	trip_id	stateroom_price_id	booking_date	total_cost	group_id	group_name	num_members	passe
2	1	18	2024-12-04	2220.00	8	Family	2	1
2	1	18	2024-12-04	2220.00	8	Family	2	30
4	2	138	2024-12-07	1280.00	10	Single	1	1
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	1
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	2
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	3
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	4
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	5
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	1
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	2
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	3
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	4
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	5
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	6
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	7
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	8
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	9
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	10

The purpose of this query is to retrieve comprehensive booking details based on a specific passenger's email ID. This allows the business to quickly identify and display all relevant bookings associated with a given passenger and their group. The query combines information about bookings, groups, and passengers, offering a detailed view of the reservations.

Specific Objectives:

1. Customer Support: Enables customer service teams to quickly fetch booking information for a passenger based on their email address, facilitating issue resolution or updates.
2. Booking Insights: Provides a holistic view of bookings, including associated group details and contact information, to manage customer relationships effectively.
3. Audit and Follow-Up: Assists in verifying the bookings and following up with group members regarding their reservations, payments, or travel plans.

```
SELECT
  gs.stateroom_id,
  gst.stateroom_type AS type,
  gs.size_sqft,
  gs.num_beds,
  gs.num_bathrooms,
  gs.balcony,
  gs.cruise_ship_side,
  gsp.stateroom_price_id,
  gsp.location_side,
  gsp.price_per_night,
  gsp.num_units
FROM
  gds_stateroom gs
JOIN
  gds_stateroom_type gst
  ON gs.type_id = gst.type_id
JOIN
  gds_stateroom_price gsp
  ON gs.stateroom_id = gsp.stateroom_id
WHERE
  gsp.stateroom_price_id = ?
  AND EXISTS (
    SELECT 1
    FROM gds_stateroom_price gsp2
    WHERE gsp2.stateroom_id = gs.stateroom_id
    AND gsp2.stateroom_price_id = gsp.stateroom_price_id
  )
```

stateroom_id	type	size_sqft	num_beds	num_bathroo...	balcony	cruise_ship_si...	stateroom_price_id	location_side	price_per_ni...	num_units
56	Oceanview Window	300	2	1	0	ST	18	After	300.00	2

The stateroom_price_id is stored as part of the booking process, and this query fetches related stateroom details to confirm and display the stateroom booked. This retrieval process is critical for ensuring that the correct stateroom details, including pricing and amenities, are tied to the

booking.

Business Objectives

1. Verification: Ensures that the stateroom_price_id corresponds to valid data in the database, avoiding invalid bookings or data mismatches.
2. Customer Display: Retrieves and presents the stateroom details (e.g., type, size, price) for the customer when viewing their booking.
3. Audit and Reporting: Provides a consistent link between bookings and the staterooms reserved, ensuring accurate reporting and auditing of reservations.
4. Booking Updates: Facilitates changes or reviews of the booked stateroom if necessary, by pulling all relevant details dynamically.

Multi-row subquery

```
SELECT
  gds_booking.booking_id,
  gds_booking.trip_id,
  gds_booking.stateroom_price_id,
  gds_booking.booking_date,
  gds_booking.total_cost,
  gds_group.group_id,
  gds_group.group_name,
  gds_group.num_members,
  gds_passenger.passenger_id,
  gds_passenger.first_name,
  gds_passenger.last_name,
  gds_passenger.email,
  gds_passenger.phone
FROM
  gds_booking
INNER JOIN
  gds_group ON gds_booking.group_id = gds_group.group_id
INNER JOIN
  gds_passenger_group ON gds_group.group_id = gds_passenger_group.group_id
INNER JOIN
```

```
  gds_passenger ON gds_passenger_group.passenger_id = gds_passenger.passenger_id
WHERE
  gds_group.group_id IN (
    SELECT gds_group.group_id
    FROM gds_passenger_group
    INNER JOIN gds_passenger ON gds_passenger_group.passenger_id =
gds_passenger.passenger_id
    WHERE gds_passenger.email = ?
  )
ORDER BY gds_booking.booking_id;
```

booking_id	trip_id	stateroom_price_id	booking_date	total_cost	group_id	group_name	num_members	passe
2	1	18	2024-12-04	2220.00	8	Family	2	1
2	1	18	2024-12-04	2220.00	8	Family	2	30
4	2	138	2024-12-07	1280.00	10	Single	1	1
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	1
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	2
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	3
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	4
15	1	5	2024-02-01	5000.00	11	Adventure Seekers	5	5
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	1
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	2
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	3
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	4
20	2	114	2024-03-15	5250.00	11	Adventure Seekers	5	5
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	6
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	7
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	8
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	9
16	1	6	2024-02-01	4000.00	12	Beach Lovers	6	10

The purpose of this query is to retrieve comprehensive booking details based on a specific passenger's email ID. This allows the business to quickly identify and display all relevant bookings associated with a given passenger and their group. The query combines information about bookings, groups, and passengers, offering a detailed view of the reservations.

Specific Objectives:

4. Customer Support: Enables customer service teams to quickly fetch booking information for a passenger based on their email address, facilitating issue resolution or updates.
5. Booking Insights: Provides a holistic view of bookings, including associated group details and contact information, to manage customer relationships effectively.
6. Audit and Follow-Up: Assists in verifying the bookings and following up with group members regarding their reservations, payments, or travel plans.

Correlated subquery.

```
SELECT
  r.restaurant_id,
  r.rt_name,
  r.rt_type,
  r.rt_floor,
  r.opening_time,
  r.closing_time
FROM
  gds_restaurant r
WHERE
  EXISTS (
    SELECT 1
    FROM gds_restaurant_schedule rs
    WHERE rs.restaurant_id = r.restaurant_id
    AND rs.trip_id = ?)
```

Result Grid Filter Rows: Search Edit: Export/Import:

restaurant_id	rt_name	rt_type	rt_floor	opening_time	closing_time
1	Ocean Grill	Seafood	3	2024-01-01 07:00:00	2024-01-01 22:00:00
2	Spicy Thai Bistro	Thai	2	2024-01-01 10:00:00	2024-01-01 23:00:00
3	Pasta Palace	Italian	1	2024-01-01 09:00:00	2024-01-01 21:00:00
6	Ming Wok Chinese	Chinese	5	2024-01-01 12:00:00	2024-01-01 20:00:00
7	Round Clock Café	Café	10	2024-01-01 00:00:00	2024-01-01 23:59:59
8	Pool Bar	Bar	10	2024-01-01 10:00:00	2024-01-01 22:00:00
9	Stout Bar	Bar	7	2024-01-01 10:00:00	2024-01-01 22:00:00
10	Biryani Pot	Indian	4	1970-01-01 12:00:00	1970-01-01 20:00:00
NULL	NULL	NULL	NULL	NULL	NULL

Business Purpose:

The purpose of this query is to retrieve the details of **restaurants available for a specific trip**. It identifies the type, location, and operating hours of restaurants that are associated with the given trip_id. This helps in understanding what dining options are available to customers during a particular trip, aiding in customer planning and enhancing the booking experience.

Use Case:

- **Customer/Passenger**: A user can view all restaurants available for a trip, including the type (fine dining, casual, etc.), operating hours, and location, to plan their meals accordingly.
- **Business Insight**: For trip management, it helps operators understand which restaurants

are scheduled for which trips and adjust their staffing and supplies accordingly.

```
SELECT
  e.entertainment_id,
  e.et_type,
  e.num_units,
  e.et_floor,
  e.age_restriction
FROM
  gds_entertainment e
WHERE
  EXISTS (
    SELECT 1
    FROM gds_trip_entertainment te
    WHERE te.entertainment_id = e.entertainment_id
    AND te.trip_id = ?
  );
```

Result Grid Filter Rows: Search

entertainment...	et_type	num_units	et_floor	age_restricti...
1	Theater	100	3	0
2	Pool	50	1	0
NULL	NULL	NULL	NULL	NULL

Business Purpose:

This query retrieves a list of available entertainment options for a specific trip. It provides details about the type, number of units, floor location, and any age restrictions associated with each entertainment option. This information helps customers view what types of entertainment are available during their trip, ensuring they can choose based on their preferences and needs, enhancing the overall customer experience by informing them of on-board activities and

entertainment during the trip.

SET operator query.

```
SELECT t.trip_id, p1.port_name AS start_port, p2.port_name AS end_port,
    t.num_nights, t.start_date, t.end_date
FROM gds_trip t
JOIN gds_port p1 ON t.start_port = p1.port_id
JOIN gds_port p2 ON t.end_port = p2.port_id
WHERE t.start_date = CURRENT_DATE
UNION
SELECT t.trip_id, p1.port_name AS start_port, p2.port_name AS end_port,
    t.num_nights, t.start_date, t.end_date
FROM gds_trip t
JOIN gds_port p1 ON t.start_port = p1.port_id
JOIN gds_port p2 ON t.end_port = p2.port_id
WHERE t.start_date BETWEEN CURRENT_DATE AND DATE_ADD(CURRENT_DATE, INTERVAL
7 MONTH)
```

Result Grid Filter Rows: Search Export:

trip_id	start_port	end_port	num_nights	start_date	end_date
1	Port of Miami	Port of Los Angeles	7	2025-06-01	2025-06-08
2	Port of Los Angeles	Port of New York	5	2025-06-09	2025-06-14
3	Port of New York	Port of Galveston	3	2025-06-15	2025-06-18
4	Port of Galveston	Port of Miami	6	2025-06-19	2025-06-25
32	Port of Miami	Port of New York	7	2025-06-20	2025-06-27
5	Port of Seattle	Port of Miami	4	2025-06-26	2025-06-30
6	Port of San Francisco	Port of Vancouver	10	2025-07-01	2025-07-11

Business Purpose

Objective

The purpose of the query is to display available trips for booking:

1. Trips Starting Today: For users looking to book last-minute trips.

2. Trips Scheduled in the Next 7 Months: For users planning their travel in advance.

Use Case

This query supports a trip availability search feature where users can:

- Browse trips starting today or within the next seven months.
- Select and proceed with booking directly from the available options.

Key Business Benefits

1. Comprehensive Search Results: Users can see all trips available for booking, catering to various preferences and timelines.
2. Improved Booking Experience: Users can explore options across a wide timeframe (today to seven months from now).
3. Enhanced User Engagement: Encourages more bookings by making all upcoming trips visible.

Q5) Query with in-line view or WITH clause

```
WITH trip_info AS (
  SELECT
    t.trip_id,
    p1.port_name AS start_port,
    p2.port_name AS end_port
  FROM
    gds_trip t
  JOIN
    gds_port p1 ON t.start_port = p1.port_id
  JOIN
    gds_port p2 ON t.end_port = p2.port_id
  WHERE
    t.trip_id = 1
)
SELECT
  trip_info.trip_id,
  trip_info.start_port,
```

```
    trip_info.end_port
FROM
    trip_info;
```

trip_id	start_port	end_port
1	Port of Miami	Port of Los Angeles

Business Purpose:

This query is intended to display the start port and end port for a specific trip, identified by its trip_id.

This is helpful for:

- Customer Information: Showing the departure and arrival ports of a trip to potential customers, allowing them to make informed booking decisions.
- Operational Insights: Assisting cruise line operations by identifying the specific start and end points of a trip for logistical coordination.
- Trip Planning: Enabling travel agents or passengers to easily view the start and end ports for their selected journey.

Q6) TOP-N/BOTTOM-N query

```
SELECT
    p.package_id,
    p.pk_type,
    SUM(pp.total_package_price) AS total_sales
FROM
    gds_package_purchase pp
JOIN
    gds_package p ON pp.package_id = p.package_id
GROUP BY
    p.package_id, p.pk_type
```

```
ORDER BY
   total_sales DESC
LIMIT 5;
```

package_id	pk_type	total_sales
24	Unlimited internet	250.00
22	Unlimited Bar	160.00
23	Internet 200 minutes, 100 GB	150.00
21	Water and Non-Alcoholic	40.00

The purpose of this query is to identify the top 5 best-selling packages based on the total sales in the system. This helps the business analyze which packages are the most popular among customers. By examining the top-selling packages, the company can make informed decisions about marketing, inventory management, and pricing strategies. Additionally, understanding the most popular packages allows the business to optimize offerings and focus on high-demand products, which can lead to increased profitability and customer satisfaction.

This query is valuable for:

1. **Sales Strategy**: Focusing marketing efforts on the most popular packages to increase sales.
2. **Pricing Adjustments**: Identifying high-demand packages to assess whether price adjustments can be made for profit optimization.
3. **Inventory Planning**: Ensuring sufficient availability of top-selling packages to meet customer demand.
4. **Customer Preferences**: Gaining insight into customer preferences and designing future packages based on trends.