# Use Cases and Database Queries

## Overview

This document describes all use cases in the Airline Reservation System and the SQL queries executed for each operation. The application uses prepared statements (parameterized queries) for all database operations to prevent SQL injection attacks.

---

## Customer Use Cases

### 1. Customer Registration

**Use Case:** A new customer creates an account by providing email, name, and password.

**Queries Executed:**

1. **Check if email already exists:**

```sql
SELECT customer_email FROM Customer WHERE customer_email = %s
```

*Purpose:* Verifies the email is not already registered before creating a new account.

2. **Insert new customer:**

```sql
INSERT INTO Customer (customer_email, name, password)
VALUES (%s, %s, %s)
```

*Purpose:* Creates a new customer record with MD5-hashed password.

---

### 2. Customer Login

**Use Case:** Customer logs in using email and password.

**Query Executed:**

```sql
SELECT customer_email, name FROM Customer
WHERE customer_email = %s AND password = %s
```

*Purpose:* Authenticates customer by verifying email and MD5-hashed password match. If successful, creates a session.

---

### 3. Search Flights (Public)

**Use Case:** Anyone (logged in or not) can search for available flights by source, destination, and departure date.

**Query Executed:**

```sql
SELECT f.airline_name, f.flight_number, f.departure_airport,
       f.departure_date, f.departure_time, f.arrival_airport,
       f.arrival_date, f.arrival_time, f.base_price, f.status,
       a1.city as departure_city, a2.city as arrival_city
FROM Flight f
JOIN Airport a1 ON f.departure_airport = a1.airport_code
JOIN Airport a2 ON f.arrival_airport = a2.airport_code
WHERE CONCAT(f.departure_date, ' ', f.departure_time) > NOW()
  AND (f.departure_airport = %s OR a1.city LIKE %s)  -- if source provided
  AND (f.arrival_airport = %s OR a2.city LIKE %s)    -- if destination provided
  AND f.departure_date = %s                           -- if date provided
ORDER BY f.departure_date, f.departure_time
```

*Purpose:* Retrieves future flights matching search criteria. Supports searching by airport code or city name. Only shows flights that haven't departed yet.

---

**4. Purchase Ticket**

**Use Case:** Logged-in customer purchases a ticket for a selected flight.

**Queries Executed:**

1. **Get flight details and check availability (POST):**

```
SELECT f.*, a.num_seats,
       (SELECT COUNT(*) FROM Ticket t
        WHERE t.flight_airline = f.airline_name
        AND t.flight_number = f.flight_number
        AND t.flight_departure_date = f.departure_date
        AND t.flight_departure_time = f.departure_time) as tickets_sold
FROM Flight f
JOIN Airplane a ON f.airline_name = a.airline_name AND f.airplane_id = a.id_number
WHERE f.airline_name = %s AND f.flight_number = %s
  AND f.departure_date = %s AND f.departure_time = %s
```

*Purpose:* Retrieves flight information and calculates how many tickets have been sold. Used to verify seat availability before purchase.

2. **Get flight details for purchase form (GET):**

```
SELECT f.*, a1.city as departure_city, a2.city as arrival_city,
       a.num_seats,
       (SELECT COUNT(*) FROM Ticket t
        WHERE t.flight_airline = f.airline_name
        AND t.flight_number = f.flight_number
        AND t.flight_departure_date = f.departure_date
        AND t.flight_departure_time = f.departure_time) as tickets_sold
FROM Flight f
JOIN Airport a1 ON f.departure_airport = a1.airport_code
JOIN Airport a2 ON f.arrival_airport = a2.airport_code
JOIN Airplane a ON f.airline_name = a.airline_name AND f.airplane_id = a.id_number
WHERE f.airline_name = %s AND f.flight_number = %s
  AND f.departure_date = %s AND f.departure_time = %s
```

*Purpose:* Displays flight details and available seats on the purchase page.

3. **Insert ticket:**

```
INSERT INTO Ticket (customer_email, flight_airline, flight_number,
                   flight_departure_date, flight_departure_time,
                   card_type, card_number, card_name, card_expiry,
                   purchase_date, purchase_time)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, CURDATE(), CURTIME())
```

*Purpose:* Creates a new ticket record with payment information. Uses CURDATE() and CURTIME() to automatically set purchase timestamp.

**Business Logic:** - Verifies flight is in the future (cannot purchase past flights) - Checks available_seats = num_seats - tickets_sold - Blocks purchase if available_seats <= 0

---

## 5. View My Flights

**Use Case:** Customer views their purchased flights with optional filtering.

**Query Executed:**

```
SELECT f.airline_name, f.flight_number, f.departure_airport,
       f.departure_date, f.departure_time, f.arrival_airport,
       f.arrival_date, f.arrival_time, f.base_price, f.status,
       t.ticket_id, t.purchase_date, t.purchase_time,
       a1.city as departure_city, a2.city as arrival_city
FROM Ticket t
JOIN Flight f ON t.flight_airline = f.airline_name
    AND t.flight_number = f.flight_number
    AND t.flight_departure_date = f.departure_date
    AND t.flight_departure_time = f.departure_time
JOIN Airport a1 ON f.departure_airport = a1.airport_code
JOIN Airport a2 ON f.arrival_airport = a2.airport_code
WHERE t.customer_email = %s
  AND CONCAT(f.departure_date, ' ', f.departure_time) > NOW()  -- for future flights
  AND f.departure_date >= %s                                    -- if start_date provided
  AND f.departure_date <= %s                                    -- if end_date provided
  AND (f.departure_airport = %s OR a1.city LIKE %s)             -- if source provided
  AND (f.arrival_airport = %s OR a2.city LIKE %s)              -- if destination provided
ORDER BY f.departure_date, f.departure_time
```

*Purpose:* Retrieves all tickets purchased by the customer with associated flight details. Supports filtering by date range, source, destination, and flight type (future/past/all).

---

## 6. View and Submit Reviews

**Use Case:** Customer views past flights and submits/updates reviews with ratings and comments.

**Queries Executed:**

1. **Get past flights for review (GET):**

```
SELECT f.airline_name, f.flight_number, f.departure_airport,
       f.departure_date, f.departure_time, f.arrival_airport,
       f.arrival_date, f.arrival_time,
       a1.city as departure_city, a2.city as arrival_city,
       r.rating, r.comment
FROM Ticket t
JOIN Flight f ON t.flight_airline = f.airline_name
    AND t.flight_number = f.flight_number
    AND t.flight_departure_date = f.departure_date
    AND t.flight_departure_time = f.departure_time
JOIN Airport a1 ON f.departure_airport = a1.airport_code
JOIN Airport a2 ON f.arrival_airport = a2.airport_code
LEFT JOIN Review r ON t.customer_email = r.customer_email
    AND t.flight_airline = r.flight_airline
    AND t.flight_number = r.flight_number
    AND t.flight_departure_date = r.flight_departure_date
    AND t.flight_departure_time = r.flight_departure_time
WHERE t.customer_email = %s
  AND CONCAT(f.departure_date, ' ', f.departure_time) <= NOW()
ORDER BY f.departure_date DESC, f.departure_time DESC
```

*Purpose:* Shows all past flights the customer has tickets for, with existing reviews if any. Uses LEFT JOIN to include flights without reviews.

2. **Verify customer can review (POST):**

```
SELECT t.*, f.departure_date, f.departure_time
FROM Ticket t
JOIN Flight f ON t.flight_airline = f.airline_name
    AND t.flight_number = f.flight_number
    AND t.flight_departure_date = f.departure_date
    AND t.flight_departure_time = f.departure_time
WHERE t.customer_email = %s
  AND t.flight_airline = %s
  AND t.flight_number = %s
  AND t.flight_departure_date = %s
  AND t.flight_departure_time = %s
  AND CONCAT(f.departure_date, ' ', f.departure_time) <= NOW()
```

*Purpose:* Verifies the customer purchased the flight and it has already occurred before allowing review submission.

3. **Check if review exists (POST):**

```
SELECT * FROM Review
WHERE customer_email = %s
  AND flight_airline = %s
  AND flight_number = %s
  AND flight_departure_date = %s
  AND flight_departure_time = %s
```

*Purpose:* Determines if customer already reviewed this flight (to update vs. insert).

4. **Update existing review (POST):**

```
UPDATE Review
SET rating = %s, comment = %s
WHERE customer_email = %s
  AND flight_airline = %s
  AND flight_number = %s
  AND flight_departure_date = %s
  AND flight_departure_time = %s
```

*Purpose:* Updates an existing review if customer already reviewed the flight.

5. **Insert new review (POST):**

```
INSERT INTO Review (customer_email, flight_airline, flight_number,
                   flight_departure_date, flight_departure_time, rating, comment)
VALUES (%s, %s, %s, %s, %s, %s, %s)
```

*Purpose:* Creates a new review record for a flight the customer hasn't reviewed yet.

---

## Staff Use Cases

### 7. Staff Registration

**Use Case:** New airline staff member creates an account.

**Queries Executed:**

1. **Get list of airlines:**

```
SELECT name FROM Airline
```

*Purpose:* Populates dropdown list of available airlines for staff registration.

2. **Check if username exists:**

```
SELECT username FROM AirlineStaff WHERE username = %s
```

*Purpose:* Verifies username is not already taken.

3. **Verify airline exists:**

```
SELECT name FROM Airline WHERE name = %s
```

*Purpose:* Validates the selected airline exists in the database.

4. **Insert new staff:**

```
INSERT INTO AirlineStaff (username, password, name_first, name_last,
                         date_of_birth, airline_name, email)
VALUES (%s, %s, %s, %s, %s, %s, %s)
```

*Purpose:* Creates a new staff account with MD5-hashed password.

---

### 8. Staff Login

**Use Case:** Staff member logs in using username and password.

**Query Executed:**

```
SELECT username, name_first, name_last, airline_name
FROM AirlineStaff
WHERE username = %s AND password = %s
```

*Purpose:* Authenticates staff member and retrieves their airline affiliation. If successful, creates a session with staff privileges.

---

### 9. View Flights (Staff Home)

**Use Case:** Staff views flights for their airline (default: next 30 days).

**Query Executed:**

```
SELECT f.airline_name, f.flight_number, f.departure_airport,
       f.departure_date, f.departure_time, f.arrival_airport,
       f.arrival_date, f.arrival_time, f.base_price, f.status,
       a1.city as departure_city, a2.city as arrival_city,
       (SELECT COUNT(*) FROM Ticket t
        WHERE t.flight_airline = f.airline_name
        AND t.flight_number = f.flight_number
        AND t.flight_departure_date = f.departure_date
        AND t.flight_departure_time = f.departure_time) as tickets_sold
FROM Flight f
JOIN Airport a1 ON f.departure_airport = a1.airport_code
JOIN Airport a2 ON f.arrival_airport = a2.airport_code
WHERE f.airline_name = %s
  AND f.departure_date >= CURDATE()
  AND f.departure_date <= DATE_ADD(CURDATE(), INTERVAL 30 DAY)
ORDER BY f.departure_date, f.departure_time
```

*Purpose:* Displays upcoming flights for the staff's airline with ticket sales count. Automatically filters to next 30 days.

---

### 10. View Flights with Filtering

**Use Case:** Staff views all flights for their airline with optional filtering.

**Query Executed:**

```
SELECT f.airline_name, f.flight_number, f.departure_airport,
       f.departure_date, f.departure_time, f.arrival_airport,
       f.arrival_date, f.arrival_time, f.base_price, f.status,
       a1.city as departure_city, a2.city as arrival_city,
       (SELECT COUNT(*) FROM Ticket t
        WHERE t.flight_airline = f.airline_name
        AND t.flight_number = f.flight_number
        AND t.flight_departure_date = f.departure_date
        AND t.flight_departure_time = f.departure_time) as tickets_sold
FROM Flight f
JOIN Airport a1 ON f.departure_airport = a1.airport_code
JOIN Airport a2 ON f.arrival_airport = a2.airport_code
WHERE f.airline_name = %s
  AND CONCAT(f.departure_date, ' ', f.departure_time) > NOW()  -- if future flights
  AND f.departure_date >= %s                                    -- if start_date provided
  AND f.departure_date <= %s                                    -- if end_date provided
  AND (f.departure_airport = %s OR a1.city LIKE %s)             -- if source provided
  AND (f.arrival_airport = %s OR a2.city LIKE %s)               -- if destination provided
ORDER BY f.departure_date, f.departure_time
```

*Purpose:* Retrieves flights with filtering by date range, source, destination, and flight type (future/past/all). Only shows flights for staff's airline.

---

### 11. View Customers for a Flight

**Use Case:** Staff views all customers who purchased tickets for a specific flight.

**Queries Executed:**

1. **Get customers on flight:**

```
SELECT t.ticket_id, t.customer_email, t.purchase_date, t.purchase_time,
       c.name, c.phone_number, c.passport_number
FROM Ticket t
JOIN Customer c ON t.customer_email = c.customer_email
WHERE t.flight_airline = %s
  AND t.flight_number = %s
  AND t.flight_departure_date = %s
  AND t.flight_departure_time = %s
ORDER BY t.purchase_date DESC, t.purchase_time DESC
```

*Purpose:* Lists all customers who purchased tickets for the flight, ordered by purchase time (most recent first).

2. **Get flight details:**

```
SELECT f.*, a1.city as departure_city, a2.city as arrival_city
FROM Flight f
JOIN Airport a1 ON f.departure_airport = a1.airport_code
JOIN Airport a2 ON f.arrival_airport = a2.airport_code
```

```
WHERE f.airline_name = %s AND f.flight_number = %s
  AND f.departure_date = %s AND f.departure_time = %s
```

*Purpose:* Displays flight information at the top of the customer list page.

**Business Logic:** Verifies staff can only view customers for flights belonging to their airline.

---

### 12. Create New Flight

**Use Case:** Staff creates a new flight for their airline.

**Queries Executed:**

1. **Get airports list:**

```
SELECT airport_code, city FROM Airport
```

*Purpose:* Populates dropdown lists for departure and arrival airport selection.

2. **Get airplanes for airline:**

```
SELECT id_number FROM Airplane WHERE airline_name = %s
```

*Purpose:* Shows available airplanes that belong to the staff's airline.

3. **Verify airplane belongs to airline:**

```
SELECT id_number FROM Airplane
WHERE airline_name = %s AND id_number = %s
```

*Purpose:* Ensures the selected airplane belongs to the staff's airline before creating flight.

4. **Check if flight already exists:**

```
SELECT * FROM Flight
WHERE airline_name = %s AND flight_number = %s
  AND departure_date = %s AND departure_time = %s
```

*Purpose:* Prevents duplicate flights (same airline, number, date, and time).

5. **Insert new flight:**

```
INSERT INTO Flight (airline_name, flight_number, departure_airport,
                    departure_date, departure_time, arrival_airport,
                    arrival_date, arrival_time, base_price, status, airplane_id)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, 'on-time', %s)
```

*Purpose:* Creates a new flight record with default status 'on-time'.

---

### 13. Change Flight Status

**Use Case:** Staff updates the status of a flight (on-time or delayed).

**Query Executed:**

```
UPDATE Flight
SET status = %s
WHERE airline_name = %s AND flight_number = %s
  AND departure_date = %s AND departure_time = %s
```

*Purpose:* Updates flight status. Only allows ‘on-time’ or ‘delayed’ values (enforced by CHECK constraint).

**Business Logic:** Verifies staff can only change status for flights belonging to their airline.

---

**14. Add Airplane**

**Use Case:** Staff adds a new airplane to their airline’s fleet.

**Queries Executed:**

1. **Get all airplanes for airline:**

```
SELECT * FROM Airplane WHERE airline_name = %s ORDER BY id_number
```

*Purpose:* Displays existing airplanes for reference.

2. **Check if airplane ID exists:**

```
SELECT * FROM Airplane
WHERE airline_name = %s AND id_number = %s
```

*Purpose:* Ensures airplane ID is unique for the airline (part of composite primary key).

3. **Insert new airplane:**

```
INSERT INTO Airplane (airline_name, id_number, num_seats, manufacturer, age)
VALUES (%s, %s, %s, %s, %s)
```

*Purpose:* Adds a new airplane to the airline’s fleet.

---

**15. View Flight Ratings**

**Use Case:** Staff views average ratings for all flights of their airline.

**Query Executed:**

```
SELECT f.airline_name, f.flight_number, f.departure_airport,
       f.departure_date, f.departure_time, f.arrival_airport,
       a1.city as departure_city, a2.city as arrival_city,
       AVG(r.rating) as avg_rating,
       COUNT(r.rating) as num_reviews
FROM Flight f
JOIN Airport a1 ON f.departure_airport = a1.airport_code
JOIN Airport a2 ON f.arrival_airport = a2.airport_code
LEFT JOIN Review r ON f.airline_name = r.flight_airline
    AND f.flight_number = r.flight_number
    AND f.departure_date = r.flight_departure_date
    AND f.departure_time = r.flight_departure_time
WHERE f.airline_name = %s
GROUP BY f.airline_name, f.flight_number, f.departure_date, f.departure_time
ORDER BY f.departure_date DESC, f.departure_time DESC
```

*Purpose:* Calculates average rating and review count for each flight. Uses LEFT JOIN to include flights without reviews.

---

### 16. View Detailed Flight Reviews

**Use Case:** Staff views all individual reviews for a specific flight.

**Queries Executed:**

1. **Get flight details:**

```
SELECT f.*, a1.city as departure_city, a2.city as arrival_city
FROM Flight f
JOIN Airport a1 ON f.departure_airport = a1.airport_code
JOIN Airport a2 ON f.arrival_airport = a2.airport_code
WHERE f.airline_name = %s AND f.flight_number = %s
  AND f.departure_date = %s AND f.departure_time = %s
```

*Purpose:* Displays flight information.

2. **Get all reviews for flight:**

```
SELECT r.*, c.name as customer_name
FROM Review r
JOIN Customer c ON r.customer_email = c.customer_email
WHERE r.flight_airline = %s AND r.flight_number = %s
  AND r.flight_departure_date = %s AND r.flight_departure_time = %s
ORDER BY r.rating DESC
```

*Purpose:* Retrieves all reviews with customer names, ordered by rating (highest first).

**Business Logic:** Calculates average rating from individual reviews. Verifies staff can only view reviews for flights belonging to their airline.

---

### 17. View Sales Reports

**Use Case:** Staff generates ticket sales reports for their airline (by date range, last month, or last year).

**Queries Executed:**

1. **Last Month Report:**

```
SELECT DATE(t.purchase_date) as purchase_date, COUNT(*) as tickets_sold,
       SUM(f.base_price) as total_revenue
```

```sql
FROM Ticket t
JOIN Flight f ON t.flight_airline = f.airline_name
    AND t.flight_number = f.flight_number
    AND t.flight_departure_date = f.departure_date
    AND t.flight_departure_time = f.departure_time
WHERE t.flight_airline = %s
  AND t.purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY DATE(t.purchase_date)
ORDER BY purchase_date
```

*Purpose:* Shows daily ticket sales and revenue for the past month.

2. **Last Year Report:**

```sql
SELECT MONTH(t.purchase_date) as month, YEAR(t.purchase_date) as year,
       COUNT(*) as tickets_sold, SUM(f.base_price) as total_revenue
FROM Ticket t
JOIN Flight f ON t.flight_airline = f.airline_name
    AND t.flight_number = f.flight_number
    AND t.flight_departure_date = f.departure_date
    AND t.flight_departure_time = f.departure_time
WHERE t.flight_airline = %s
  AND t.purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY MONTH(t.purchase_date), YEAR(t.purchase_date)
ORDER BY year, month
```

*Purpose:* Shows monthly ticket sales and revenue for the past year.

3. **Date Range Report:**

```sql
SELECT DATE(t.purchase_date) as purchase_date, COUNT(*) as tickets_sold,
       SUM(f.base_price) as total_revenue
FROM Ticket t
JOIN Flight f ON t.flight_airline = f.airline_name
    AND t.flight_number = f.flight_number
    AND t.flight_departure_date = f.departure_date
    AND t.flight_departure_time = f.departure_time
WHERE t.flight_airline = %s
  AND t.purchase_date >= %s AND t.purchase_date <= %s
GROUP BY DATE(t.purchase_date)
ORDER BY purchase_date
```

*Purpose:* Shows daily ticket sales and revenue for a custom date range.

**Business Logic:** Calculates totals (total tickets sold, total revenue) from the report data. Only shows data for staff's airline.

---

## Common Query Patterns

### Subqueries for Counting Tickets

Many queries use a correlated subquery to count tickets sold:

```
(SELECT COUNT(*) FROM Ticket t
 WHERE t.flight_airline = f.airline_name
 AND t.flight_number = f.flight_number
 AND t.flight_departure_date = f.departure_date
 AND t.flight_departure_time = f.departure_time) as tickets_sold
```

*Purpose:* Calculates how many tickets have been sold for each flight without requiring a separate query.

### Date/Time Comparisons

The application uses `CONCAT(f.departure_date, ' ', f.departure_time) > NOW()` to compare flight departure datetime with current time, ensuring only future flights are shown in searches and purchases.

### JOIN Operations

Most queries use JOINs to combine related tables: - **Flight + Airport:** To get city names for departure/arrival airports - **Ticket + Flight:** To get flight details for purchased tickets - **Ticket + Customer:** To get customer information for staff views - **Review + Customer:** To get customer names in reviews - **Flight + Airplane:** To get seat capacity information

### Security Features

- All queries use prepared statements (parameterized queries) with `%s` placeholders
- User input is never directly concatenated into SQL strings
- Session-based authentication ensures users can only access their own data or their airline's data
- Staff can only view/modify flights for their own airline