Barcelona School of Economics

Airline Database Modeling

DSDM

Hannes Schiemann | Simon Vellin

October 31, 2024

Contents

1 Conceptual Model

Figure 1 reflects the conceptual design behind our proposed Airline database.

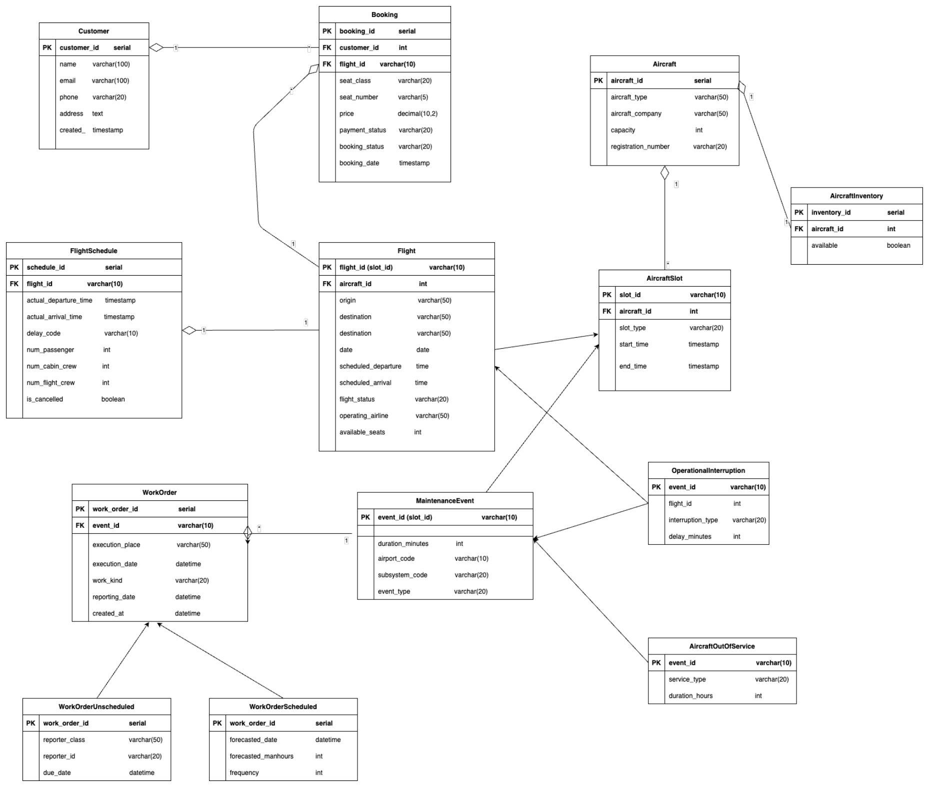

Figure 1: Airline Database ULM Diagram

(a) Customer Relationships

- Customer → Booking: Each customer can have multiple bookings, but bookings cannot exist without a customer (aggregation - $\diamond$).

(b) Booking Relationships

- Booking → Customer: Each booking is associated with one customer (aggregation - $\diamond$).
- Booking → Flight: Each booking is linked to a specific flight, but flights can exist without bookings (aggregation - $\diamond$).

(c) Flight Relationships

- Flight → Booking: Each flight can have multiple bookings (aggregation - ◇).
- Flight → FlightSchedule: Each flight can have multiple schedules (e.g., reschedules), but flight schedules cannot exist without a flight (aggregation - ◇).
- Flight → Aircraft: Each flight is associated with one aircraft, and flights cannot exist without an assigned aircraft (aggregation - ◇).

(d) AircraftSlot Relationships

- AircraftSlot → Flight / MaintenanceEvent: AircraftSlot is the parent class of Flight and Maintenance Event. Each slot can either be assigned to either a flight or a maintenance event. Each flight or maintenance event is an aircraft slot.

(e) FlightSchedule Relationships

- FlightSchedule → Flight: Each schedule entry is linked to a specific flight (aggregation - ◇).

(f) Aircraft Relationships

- Aircraft → AircraftInventory: Each aircraft has one inventory status and inventory records are associated with an existing aircraft (aggregation - ◇).
- Aircraft → AircraftSlot: Each aircraft can have multiple slots assigned for flights or maintenance events, each slot must have one and only one specific aircraft (aggregation - ◇).

(g) MaintenanceEvent Relationships

- MaintenanceEvent → WorkOrder: Each maintenance event can have multiple work orders, but work orders cannot exist without a maintenance event (aggregation - ◇).
- MaintenanceEvent → AircraftOutOfService / OperationalInterruption: MaintenanceEvent is the parent class of AicraftOutOfService and OperationalInterruption. Each MaintenanceEvent is either an AircraftOutOfService or OperationalInterruptionan.

(h) OperationalInterruption

- OperationalInterruption → Flight: An OperationalInterruption affects one specific Flight

(i) WorkOrder Relationships

- WorkOrder → MaintenanceEvent: Each work order is linked to a specific maintenance event (aggregation - ◇).
- WorkOrder → WorkOrderScheduled / WorkOrderUnscheduled: WorkOrder is a parent class of WorkOrderUnscheduled and WorkOrderScheduled
- WorkOrder → WorkOrderUnscheduled: Each unscheduled work order is a type of work order and depends on it for existence

2 SQL Statements

The SQL script in the corresponding Appendix (Listing 1) translates the conceptual model into relational tables. Key relationships are defined, with `PRIMARY KEY` and `FOREIGN KEY` constraints to enforce referential integrity.

3 Normalization and Functional Dependencies

The database tables have been designed with normalization standards to ensure data integrity and minimize redundancy. Each table is analyzed below to identify its highest normal form, based on the functional dependencies derived from the ER diagram.

(a) Customer Table

The `customer` table contains atomic attributes, all fully dependent on the primary key, `customer_id`. Functional dependencies such as `customer_id` → `name`, `email`, `phone`, and `address` confirm that this table satisfies Boyce-Codd Normal Form (BCNF).

(b) Flight Table

BCNF: all non-key attributes (origin_airport, destination_airport, scheduled_departure) are fully functionally dependent on the primary key `flight_id`, with no partial or transitive dependencies.

(c) Booking Table

3NF: all attributes, such as `booking_status` and `payment_status`, are fully dependent on the composite key (`flight_id`, `customer_id`) but do not satisfy BCNF due to the composite dependency.

(d) Aircraft Table

BCNF: all non-key attributes (aircraft_type, capacity, registration_number) are fully functionally dependent on the primary key `aircraft_id`.

(e) AircraftSlot

BCNF: all non-trivial functional dependencies have `SlotId`, the primary key, as their determinant, with no transitive dependencies among non-key attributes.

(f) Maintenance Event Table

BCNF: all non-key attributes (event_type, duration) are fully functionally dependent on the primary key `event_id`.

(g) AircraftOutOfService

BCNF: all non-key attributes (service_type, duration, duration_unit) are fully functionally dependent on the primary key `event_id`.

(h) OperationalInterruption

BCNF: all non-key attributes (flight_id, interruption_type, delay_minutes) are fully functionally dependent on the primary key `event_id`.

(i) Summary of Normal Forms

All tables adhere to at least Third Normal Form (3NF), eliminating partial dependencies and ensuring that each attribute is functionally dependent on the primary key. Where applicable, tables satisfy Boyce-Codd Normal Form (BCNF), minimizing redundancy and reinforcing data integrity within the database schema.

4 Random Data Generation

The SQL script in the corresponding Appendix (Listing 2) provides random data entry across tables for testing and analysis.

5 Relational Algebra - Functional Requirements

(a) Retrieve Available Flights

Filter flights based on origin, destination, departure and available seats:

$$R = Flight(Origin =' Origin', Destination =' Destination', ScheduledDeparture =' Departure', AvailableS$$

This table contains all the information available in Flight table for these specific flights.

(b) Retrieve Customer's Bookings with Flight and Payment Status

1. Select Booking for specific customer:

$$A = Booking(CustomerId =' CustomerId')$$

2. Join A with matching flights

$$B = A(FlightId = FlightId)Flight$$

3. Join B with customer to also get customer specific information.

$$C = B(CustomerId = CustomerId)$$

Table C now contains all the booking information of the specific bookings plus the flight details of these bookings plus the customer specific information. Now you could select for the features of interest such as customer name, flight status, booking status, seat number etc.

6 Extra Queries

(a) Get canceled flights, corresponding ticket price and customer

1. Get canceled Flights by selecting is_canceled column of FlightSchedule where argument is TRUE

$$A = FlightSchedule[IsCanceled =' TRUE']$$

2. Match flight_id with booking

$$B = Booking(FlightId = A)$$

3. Projection B to only get price, customer and flight_id

$$C = B[FlightId, CustomerId, Price]$$

(b) Retrieve Delayed Flights at destination JFK

1. Select Flights with destination JFK and filter for scheduled arrival and flight id

$$A = Flight(destination =' JFK')B = A[ScheduledArrival, FlightId]$$

2. Projection of FlightSchedule for actual_arrival_time and flight_id

$$C = FlightSchedule[ActualArrivalTime, FlightId]$$

3. Join B and C on matching flight_id

$$D = B(FlightId = FlightId)C$$

4. Filter for delayed flights

$$E = D(ArrivalTime > ScheduledArrival)$$

7 Appendix

2) Commented SQL Statements

```sql
-- Drop the Customer table if it exists, including any
    dependencies (CASCADE)
DROP TABLE IF EXISTS Customer CASCADE;
CREATE TABLE Customer (
    -- Primary key with SERIAL auto-increment for unique
    customer IDs
    customer_id SERIAL PRIMARY KEY,
    -- Customer name, email (unique and required), phone,
    and address fields
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    address TEXT,
    -- Automatically set creation timestamp
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Drop the Aircraft table if it exists
DROP TABLE IF EXISTS Aircraft CASCADE;
CREATE TABLE Aircraft (
    -- Unique identifier for each aircraft with SERIAL for
    auto-incrementing IDs
    aircraft_id SERIAL PRIMARY KEY,
    -- Aircraft details: type, company, capacity, and unique
     registration number
    aircraft_type VARCHAR(50),
    aircraft_company VARCHAR(50),
    capacity INT,
    -- Unique registration number for each aircraft
    registration_number VARCHAR(20) UNIQUE
);

-- Drop the aircraft_slot table if it exists
DROP TABLE IF EXISTS aircraft_slot CASCADE;
CREATE TABLE aircraft_slot (
    -- Primary key for each slot, using VARCHAR for
    flexibility in naming conventions
    slot_id VARCHAR(10) PRIMARY KEY,
    -- Foreign key referencing the Aircraft table, ensuring
    each slot is tied to an aircraft
    aircraft_id INTEGER NOT NULL REFERENCES aircraft(
    aircraft_id),
    -- Timestamps to specify start and end times for the
    slot
    start_time TIMESTAMP NOT NULL,
    end_time TIMESTAMP NOT NULL,
    -- Type of slot, constrained to either 'Flight' or 'Maintenance' for consistent categorization
    slot_type VARCHAR(20) NOT NULL CHECK (slot_type IN ('Flight', 'Maintenance')),
    -- Timestamp for when the slot was created, defaulting to the current time
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

-- Drop the Flight table if it exists
DROP TABLE IF EXISTS Flight CASCADE;
CREATE TABLE Flight (
    -- Primary key referencing aircraft_slot to inherit slot_id for each flight
    flight_id VARCHAR(10) PRIMARY KEY REFERENCES aircraft_slot (slot_id),
    -- Origin and destination locations for the flight
    origin VARCHAR(50) NOT NULL,
    destination VARCHAR(50) NOT NULL,
    -- Date of the flight
    date DATE NOT NULL,
    -- Status of the flight, constrained to specific values
    flight_status VARCHAR(20) CHECK (flight_status IN ('Scheduled', 'Delayed', 'Cancelled')) DEFAULT 'Scheduled',
    -- Operating airline and the number of available seats, constrained to non-negative values
    operating_airline VARCHAR(50),
    available_seats INT CHECK (available_seats >= 0)
);

-- Drop the Booking table if it exists
DROP TABLE IF EXISTS Booking CASCADE;
CREATE TABLE Booking (
    -- Primary key with SERIAL for unique booking IDs
    booking_id SERIAL PRIMARY KEY,
    -- Foreign key linking to the customer making the booking
    customer_id INT REFERENCES Customer(customer_id),
    -- Foreign key linking to the specific flight
    flight_id VARCHAR(10) NOT NULL REFERENCES Flight (flight_id),
    -- Details of the booking: seat class, seat number, price, and statuses
    seat_class VARCHAR(20),
    seat_number VARCHAR(5),
    price DECIMAL(10, 2),
    -- Payment status and booking status constrained to specific options
    payment_status VARCHAR(20) CHECK (payment_status IN ('Paid', 'Pending', 'Cancelled')) DEFAULT 'Pending',
    booking_status VARCHAR(20) CHECK (booking_status IN ('Active', 'Cancelled')) DEFAULT 'Active',
    -- Timestamp of the booking creation, defaulting to current time
    booking_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    -- Unique constraint to prevent duplicate active bookings for the same seat on a flight
    UNIQUE (flight_id, seat_number, booking_status)
);

-- Drop the maintenance_event table if it exists
DROP TABLE IF EXISTS maintenance_event CASCADE;
CREATE TABLE maintenance_event (
    -- Primary key for maintenance events, referencing the aircraft_slot table for event ID
    event_id VARCHAR(10) PRIMARY KEY REFERENCES aircraft_slot (slot_id),
    -- Airport and subsystem codes, with constraints to standardize event types
    airport_code VARCHAR(5) NOT NULL,
    subsystem_code VARCHAR(20) NOT NULL,
    -- Event type constrained to 'AOS' or 'OI'
    event_type VARCHAR(3) NOT NULL CHECK (event_type IN ('AOS' ,'OI'))
);

-- Drop the operational_interruption table if it exists
DROP TABLE IF EXISTS operational_interruption CASCADE;
CREATE TABLE operational_interruption (
    -- Primary key referencing a maintenance event
    event_id VARCHAR(10) PRIMARY KEY REFERENCES maintenance_event(event_id),
    -- Foreign key linking to the flight impacted by the interruption
    flight_id VARCHAR(10) NOT NULL REFERENCES flight (flight_id),
    -- Type of interruption, constrained to specific options
    interruption_type VARCHAR(20) NOT NULL CHECK (interruption_type IN ('DELAY', 'SAFETY')),
    -- Delay duration in minutes, constrained to be non-negative
    delay_minutes INTEGER CHECK (delay_minutes >= 0)
);

-- Drop the aircraft_out_of_service table if it exists
DROP TABLE IF EXISTS aircraft_out_of_service CASCADE;
CREATE TABLE aircraft_out_of_service (
    -- Primary key referencing a maintenance event for each out-of-service event
    event_id VARCHAR(10) PRIMARY KEY REFERENCES maintenance_event (event_id),
    -- Service type, constrained to either 'MAINTENANCE' or 'REVISION'
    service_type VARCHAR(20) NOT NULL CHECK (service_type IN ('MAINTENANCE', 'REVISION')),
    -- Duration of service with constraints based on the type of service
    duration INTEGER NOT NULL CHECK (
        (service_type = 'MAINTENANCE' AND duration BETWEEN 1 AND 24) OR
        (service_type = 'REVISION' AND duration BETWEEN 1 AND 30)
    ),
    -- Duration unit, constrained based on the service type
    duration_unit VARCHAR(10) NOT NULL CHECK (
        (service_type = 'MAINTENANCE' AND duration_unit = 'HOURS') OR
        (service_type = 'REVISION' AND duration_unit = 'DAYS')
    )
);

-- Drop the work_order table if it exists
DROP TABLE IF EXISTS work_order CASCADE;
CREATE TABLE work_order (
    -- Primary key with SERIAL for unique work order IDs
    work_order_id SERIAL PRIMARY KEY,
    -- Foreign key linking to the maintenance event related to this work order
    event_id VARCHAR(10) NOT NULL REFERENCES maintenance_event (event_id),
    -- Location and date of work execution, with details on the type of work
    execution_place VARCHAR(100) NOT NULL,
    execution_date TIMESTAMP NOT NULL,
    -- Type of work (scheduled or unscheduled) with constraints
    work_kind VARCHAR(20) NOT NULL CHECK (work_kind IN ('Scheduled', 'Unscheduled')),
    -- Duration in hours and a timestamp for when the order was created
    duration_hours INT,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

-- Drop the work_order_scheduled table if it exists
DROP TABLE IF EXISTS work_order_scheduled CASCADE;
```

```sql
CREATE TABLE work_order_scheduled (
    -- Primary key referencing the work order
    work_order_id INTEGER PRIMARY KEY REFERENCES work_order
   (work_order_id),
    -- Forecasted date and man-hours required for the
   scheduled work
    forecasted_date TIMESTAMP NOT NULL,
    forecasted_manhours INTEGER NOT NULL CHECK (
   forecasted_manhours > 0),
    -- Frequency of scheduled work (e.g., in days or cycles)
    frequency INTEGER
);

-- Drop the work_order_unscheduled table if it exists
DROP TABLE IF EXISTS work_order_unscheduled CASCADE;
CREATE TABLE work_order_unscheduled (
    -- Primary key referencing the work order
    work_order_id INTEGER PRIMARY KEY REFERENCES work_order
   (work_order_id),
    -- Class and ID of the reporter for unscheduled work,
   plus a due date
    reporter_class VARCHAR(50) NOT NULL,
    reporter_id VARCHAR(50) NOT NULL,
    due_date TIMESTAMP NOT NULL
);

-- Drop the Flight_schedule table if it exists
DROP TABLE IF EXISTS Flight_schedule CASCADE;
CREATE TABLE Flight_schedule (
    -- Primary key with SERIAL for unique schedule IDs
    schedule_id SERIAL PRIMARY KEY,
    -- Foreign key linking to the flight being scheduled
    flight_id VARCHAR(10) REFERENCES Flight(flight_id),
    -- Actual departure and arrival times
    actual_departure_time TIMESTAMP,
    actual_arrival_time TIMESTAMP,
    -- Delay code and the number of passengers and crew
   members
    delay_code VARCHAR(10),
    num_passengers INT,
    num_cabin_crew INT,
    num_flight_crew INT,
    -- Boolean flag for cancellation status
    is_cancelled BOOLEAN DEFAULT FALSE
);

-- Drop the Aircraft_inventory table if it exists
DROP TABLE IF EXISTS Aircraft_inventory CASCADE;
CREATE TABLE Aircraft_inventory (
    -- Primary key with SERIAL for unique inventory IDs
    inventory_id SERIAL PRIMARY KEY,
    -- Foreign key linking to the aircraft in the inventory
    aircraft_id INT REFERENCES Aircraft(aircraft_id),
    -- Boolean flag indicating if the aircraft is available
    available BOOLEAN DEFAULT TRUE
);
```

Listing 1: Commented SQL Statements

4) Random Data Generation

```sql
INSERT INTO Customer (name, email, phone, address) VALUES
('John Doe', 'john.doe@example.com', '123-456-7890', '123 Elm Street'),
('Jane Smith', 'jane.smith@example.com', '234-567-8901', '456 Oak Street'),
('Michael Johnson', 'm.johnson@example.com', '345-678-9012', '789 Pine Street'),
('Emily Davis', 'emily.davis@example.com', '456-789-0123', '135 Maple Avenue'),
('David Wilson', 'david.wilson@example.com', '567-890-1234', '246 Cedar Road'),
('Sarah Brown', 'sarah.brown@example.com', '678-901-2345', '357 Spruce Drive'),
('Chris Miller', 'chris.miller@example.com', '789-012-3456', '468 Birch Boulevard'),
('Amanda Taylor', 'amanda.taylor@example.com', '890-123-4567', '579 Walnut Lane'),
('Daniel Moore', 'daniel.moore@example.com', '901-234-5678', '680 Cherry Street'),
('Jessica Anderson', 'j.anderson@example.com', '012-345-6789', '791 Fir Avenue');

INSERT INTO Aircraft (aircraft_id,aircraft_type, aircraft_company, capacity, registration_number) VALUES
(1,'Boeing 737', 'Boeing', 180, 'B737-001'),
(2,'Airbus A320', 'Airbus', 160, 'A320-002'),
(3,'Boeing 747', 'Boeing', 366, 'B747-003'),
(4,'Embraer E190', 'Embraer', 100, 'E190-004'),
(5,'Airbus A330', 'Airbus', 250, 'A330-005'),
(6,'Boeing 787', 'Boeing', 296, 'B787-006'),
(7,'Bombardier CRJ700', 'Bombardier', 78, 'CRJ700-007'),
(8,'Airbus A380', 'Airbus', 500, 'A380-008'),
(9,'Boeing 757', 'Boeing', 239, 'B757-009'),
(10,'Airbus A340', 'Airbus', 375, 'A340-010'),
(11,'Boeing 737', 'Boeing', 180, 'B737-101'),
(12,'Airbus A320', 'Airbus', 160, 'A320-102'),
(13,'Boeing 747', 'Boeing', 366, 'B747-103'),
(14,'Embraer E190', 'Embraer', 100, 'E190-104'),
(15,'Airbus A330', 'Airbus', 250, 'A330-105'),
(16,'Boeing 787', 'Boeing', 296, 'B787-106'),
(17,'Bombardier CRJ700', 'Bombardier', 78, 'CRJ700-107'),
(18,'Airbus A380', 'Airbus', 500, 'A380-108'),
(20,'Boeing 757', 'Boeing', 239, 'B757-109');


INSERT INTO Aircraft_slot (slot_id, aircraft_id, slot_type,
    start_time, end_time) VALUES
('FL001',1, 'Flight', '2024-12-01 07:00:00', '2024-12-01
    17:00:00'),
('FL002',2, 'Flight', '2024-12-01 09:00:00', '2024-12-02
    18:00:00'),
('FL003' ,3, 'Flight', '2024-12-02 09:00:00', '2024-12-02
    18:00:00'),
('FL004' ,4, 'Flight', '2024-12-02 09:00:00', '2024-12-02
    18:00:00'),
('FL005' ,5, 'Flight', '2024-12-03 09:00:00', '2024-12-02
    18:00:00'),
('FL006' ,6, 'Flight', '2024-12-03 09:00:00', '2024-12-02
    18:00:00'),
('FL007' ,7, 'Flight', '2024-12-04 09:00:00', '2024-12-02
    18:00:00'),
('FL008' ,8, 'Flight', '2024-12-04 09:00:00', '2024-12-02
    18:00:00'),
('FL009' ,9, 'Flight', '2024-12-05 09:00:00', '2024-12-02
    18:00:00'),
('FL010' ,10, 'Flight', '2024-12-05 09:00:00', '2024-12-02
    18:00:00'),
('A01' ,11, 'Maintenance', '2024-12-05 09:00:00', '
    2024-12-02 18:00:00'),
('A02' ,12, 'Maintenance', '2024-12-05 09:00:00', '
    2024-12-02 18:00:00'),
('A03' ,13, 'Maintenance', '2024-12-05 09:00:00', '
    2024-12-02 18:00:00'),
('A04' ,14, 'Maintenance', '2024-12-05 09:00:00', '
    2024-12-02 18:00:00'),
('OI1' ,15, 'Maintenance', '2024-12-05 09:00:00', '
    2024-12-02 18:00:00'),
('OI2' ,16, 'Maintenance', '2024-12-05 09:00:00', '
    2024-12-02 18:00:00'),
('OI3' ,17, 'Maintenance', '2024-12-05 09:00:00', '
    2024-12-02 18:00:00'),
('OI4' ,18, 'Maintenance', '2024-12-05 09:00:00', '
    2024-12-02 18:00:00');


INSERT INTO Flight (flight_id, origin, destination, date,
    flight_status, operating_airline, available_seats) VALUES
```

```sql
('FL001', 'LAX', 'JFK', '2024-12-01', 'Scheduled', 'Airline A', 180),
('FL002', 'JFK', 'LAX', '2024-12-01', 'Scheduled', 'Airline A', 160),
('FL003', 'LAX', 'ORD', '2024-12-02', 'Scheduled', 'Airline B', 366),
('FL004', 'ORD', 'LAX', '2024-12-02', 'Delayed', 'Airline B', 100),
('FL005', 'LAX', 'SFO', '2024-12-03', 'Scheduled', 'Airline C', 250),
('FL006', 'SFO', 'LAX', '2024-12-03', 'Cancelled', 'Airline C', 296),
('FL007', 'LAX', 'SEA', '2024-12-04', 'Scheduled', 'Airline D', 78),
('FL008', 'SEA', 'LAX', '2024-12-04', 'Scheduled', 'Airline D', 500),
('FL009', 'LAX', 'DEN', '2024-12-05', 'Scheduled', 'Airline E', 239),
('FL010', 'DEN', 'LAX', '2024-12-05', 'Scheduled', 'Airline E', 375);


INSERT INTO Booking (customer_id, flight_id, seat_class, seat_number, price, payment_status, booking_status, booking_date) VALUES
(1, 'FL001', 'Economy', '12A', 200.00, 'Paid', 'Active', '2024-11-01'),
(2, 'FL002', 'Business', '3C', 500.00, 'Paid', 'Active', '2024-11-02'),
(3, 'FL003', 'First', '1A', 800.00, 'Pending', 'Active', '2024-11-03'),
(4, 'FL004', 'Economy', '22B', 180.00, 'Paid', 'Active', '2024-11-04'),
(5, 'FL005', 'Economy', '15D', 150.00, 'Paid', 'Cancelled', '2024-11-05'),
(6, 'FL006', 'Business', '2B', 600.00, 'Cancelled', 'Cancelled', '2024-11-06'),
(7, 'FL007', 'Economy', '16E', 220.00, 'Paid', 'Active', '2024-11-07'),
(8, 'FL008', 'First', '1B', 900.00, 'Pending', 'Active', '2024-11-08'),
(9, 'FL009', 'Economy', '18F', 210.00, 'Paid', 'Active', '2024-11-09'),
(10, 'FL010', 'Economy', '20C', 175.00, 'Pending', 'Active', '2024-11-10');

INSERT INTO Maintenance_event (event_id, airport_code, subsystem_code, event_type) VALUES
('A01', 'JFK', 'SYS000', 'AOS'),
('A02', 'LAX', 'SYS001', 'AOS'),
('AO3', 'ORD', 'SYS002', 'AOS'),
('AO4', 'BCN', 'SYS003', 'AOS'),
('OI1', 'JFK', 'SYS004', 'OI'),
('OI2', 'JFK', 'SYS005', 'OI'),
('OI3', 'BCN', 'SYS006', 'OI'),
('OI4', 'BER', 'SYS007', 'OI');

INSERT INTO operational_interruption (event_id, flight_id,
    interruption_type) VALUES
('OI1', 'FL001', 'DELAY'),
('OI2', 'FL001', 'DELAY'),
('OI3', 'FL002', 'SAFETY'),
('OI4', 'FL004', 'DELAY');

INSERT INTO aircraft_out_of_service (event_id, service_type,
     duration, duration_unit) VALUES
('AO1', 'MAINTENANCE', 4, 'HOURS'),
('AO2', 'MAINTENANCE', 2, 'HOURS'),
('AO3', 'REVISION', 24, 'DAYS'),
('AO4', 'MAINTENANCE', 18, 'HOURS');

INSERT INTO Work_order (work_order_id, event_id,
    execution_place, execution_date, work_kind,
    duration_hours) VALUES
(1, 'AO1', 'Hangar 1a', '2024-10-25', 'Scheduled', 1),
(2, 'AO1', 'Hangar 1b', '2024-10-25', 'Scheduled', 1.5),
(3, 'AO3', 'Hangar 1', '2024-10-25', 'Scheduled', 30),
(4, 'AO3', 'Hangar 2', '2024-10-26', 'Scheduled', 64),
(5, 'AO3', 'Hangar 3', '2024-10-27', 'Unscheduled', 10),
(6, 'AO3', 'Hangar 2', '2024-10-28', 'Unscheduled', 4),
(7, 'AO4', 'Hangar 1c', '2024-10-28', 'Unscheduled', 4),
(8, 'OI1', 'Parking 32a', '2024-10-28', 'Unscheduled', 1),
(9, 'OI1', 'Parking 12b', '2024-10-28', 'Scheduled', 0.5);

INSERT INTO work_order_scheduled( work_order_id,
    forecasted_date, forecasted_manhours) VALUES
(1, '2024-10-25', 5),
(2, '2024-10-25', 3),
(3, '2024-10-26',120),
(4, '2024-10-29', 220),
(9, '2024-10-28', 5);

INSERT INTO work_order_unscheduled( work_order_id,
    reporter_class, reporter_id, due_date) VALUES
(5, 'Pilot', 'PL007', '2024-11-27'),
(6, 'Pilot', 'PL008', '2025-01-27'),
(7, 'Maintenance Personal', 'ME007', '2024-11-29'),
(8, 'Maintenance Personal', 'ME008', '2024-11-29');


```

```
INSERT INTO Flight_schedule (flight_id,
    actual_departure_time, actual_arrival_time, delay_code,
    num_passengers, num_cabin_crew, num_flight_crew,
    is_cancelled) VALUES
('FL001', '2024-12-01 08:15:00', '2024-12-01 16:10:00', NULL
    , 170, 5, 2, FALSE),
('FL002', '2024-12-01 17:30:00', '2024-12-02 01:20:00', '
    D001', 150, 4, 2, FALSE),
('FL003', '2024-12-02 09:45:00', '2024-12-02 15:45:00', NULL
    , 360, 6, 3, FALSE),
('FL004', '2024-12-02 10:20:00', '2024-12-02 16:20:00', '
    D002', 95, 3, 2, TRUE),
('FL005', '2024-12-03 13:05:00', '2024-12-03 14:40:00', NULL
    , 240, 5, 2, FALSE),
('FL006', '2024-12-03 15:15:00', '2024-12-03 16:45:00', NULL
    , 0, 4, 2, TRUE),
('FL007', '2024-12-04 10:10:00', '2024-12-04 13:05:00', NULL
    , 75, 2, 2, FALSE),
('FL008', '2024-12-04 14:10:00', '2024-12-04 17:15:00', NULL
    , 490, 8, 3, FALSE),
('FL009', '2024-12-05 08:40:00', '2024-12-05 12:10:00', NULL
    , 230, 5, 2, FALSE),
('FL010', '2024-12-05 13:10:00', '2024-12-05 16:40:00', '
    D003', 370, 6, 2, FALSE);


INSERT INTO Aircraft_Inventory (aircraft_id, available)
    VALUES
(1, TRUE),
(2, TRUE),
(3, FALSE),
(4, TRUE),
(5, TRUE),
(6, FALSE),
(7, TRUE),
(8, TRUE),
(9, FALSE),
(10, TRUE);
```

Listing 2: Random Data Generation