Database for SQL Database Analysis

```
-- Create Database
CREATE DATABASE AirlineDB;
USE AirlineDB;

-- Airports
CREATE TABLE Airports (
    airport_code CHAR(3) PRIMARY KEY,
    airport_name VARCHAR(100) NOT NULL,
    city VARCHAR(50),
    country VARCHAR(50)
);

-- Aircrafts
CREATE TABLE Aircrafts (
    aircraft_code CHAR(3) PRIMARY KEY,
    model VARCHAR(100) NOT NULL,
    range_km INT NOT NULL
);

-- Seats
CREATE TABLE Seats (
    aircraft_code CHAR(3),
    seat_no VARCHAR(5),
    fare_conditions ENUM('Economy','Comfort','Business'),
    PRIMARY KEY (aircraft_code, seat_no),
    FOREIGN KEY (aircraft_code) REFERENCES Aircrafts(aircraft_code)
);
```

```
-- Flights
CREATE TABLE Flights (
    flight_id INT AUTO_INCREMENT PRIMARY KEY,
    flight_no VARCHAR(10) NOT NULL,
    scheduled_departure DATETIME NOT NULL,
    scheduled_arrival DATETIME NOT NULL,
    departure_airport CHAR(3),
    arrival_airport CHAR(3),
    status ENUM('Scheduled','On Time','Delayed','Departed','Arrived','Cancelled') DEFAULT
'Scheduled',
    aircraft_code CHAR(3),
    actual_departure DATETIME,
    actual_arrival DATETIME,
    FOREIGN KEY (departure_airport) REFERENCES Airports(airport_code),
    FOREIGN KEY (arrival_airport) REFERENCES Airports(airport_code),
    FOREIGN KEY (aircraft_code) REFERENCES Aircrafts(aircraft_code)
);

-- Bookings
CREATE TABLE Bookings (
    book_ref CHAR(6) PRIMARY KEY,
    book_date DATETIME NOT NULL,
    total_amount DECIMAL(10,2) NOT NULL
);

-- Tickets (Passenger details are stored here)
CREATE TABLE Tickets (
    ticket_no CHAR(13) PRIMARY KEY,
    book_ref CHAR(6),
    passenger_name VARCHAR(100) NOT NULL,
    passenger_id VARCHAR(20), -- ID doc number
```

```
    contact_data JSON,       -- can hold email/phone
    FOREIGN KEY (book_ref) REFERENCES Bookings(book_ref)
);

-- Ticket_Flights (link tickets to multiple flights)
CREATE TABLE Ticket_Flights (
    ticket_no CHAR(13),
    flight_id INT,
    fare_conditions ENUM('Economy','Comfort','Business'),
    amount DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (ticket_no, flight_id),
    FOREIGN KEY (ticket_no) REFERENCES Tickets(ticket_no),
    FOREIGN KEY (flight_id) REFERENCES Flights(flight_id)
);

-- Boarding Passes
CREATE TABLE Boarding_Passes (
    ticket_no CHAR(13),
    flight_id INT,
    boarding_no INT,
    seat_no VARCHAR(5),
    PRIMARY KEY (ticket_no, flight_id),
    FOREIGN KEY (ticket_no, flight_id) REFERENCES Ticket_Flights(ticket_no, flight_id),
    FOREIGN KEY (flight_id) REFERENCES Flights(flight_id)
);
```

SQL Queries to Extract Key Insights

1)Top 5 busiest routes (by passenger bookings)

```
SELECT a1.city AS source_city,
    a2.city AS destination_city,
    COUNT(tf.ticket_no) AS total_passengers
FROM Ticket_Flights tf
JOIN Flights f ON tf.flight_id = f.flight_id
JOIN Airports a1 ON f.departure_airport = a1.airport_code
JOIN Airports a2 ON f.arrival_airport = a2.airport_code
GROUP BY a1.city, a2.city
ORDER BY total_passengers DESC
LIMIT 5;
```

source_city	destination_city	total_passengers
Delhi	Dubai	12,340
New York	London	11,275
Mumbai	Singapore	9,845
Dubai	Sydney	9,102
Paris	Rome	8,755

2) Percentage of flights delayed beyond 30 minutes

```
SELECT
ROUND(100.0 * SUM(CASE
WHEN TIMESTAMPDIFF(MINUTE, f.scheduled_departure, f.actual_departure) > 30
THEN 1 ELSE 0 END) / COUNT(*), 2)
AS delayed_percentage
FROM Flights f
WHERE f.actual_departure IS NOT NULL;
```

delayed_percentage
18.35

3) High-value frequent flyers (spent > 100,000 total)

```
SELECT t.passenger_name,
    SUM(tf.amount) AS total_spent,
    COUNT(DISTINCT tf.flight_id) AS flights_taken
FROM Tickets t
JOIN Ticket_Flights tf ON t.ticket_no = tf.ticket_no
GROUP BY t.passenger_name
HAVING SUM(tf.amount) > 100000
ORDER BY total_spent DESC;
```

passenger_name	total_spent	flights_taken
Rahul Sharma	1,25,800.00	36
Aisha Khan	1,18,500.00	29
John Doe	1,01,200.00	22

4) Average ticket price by class

```
SELECT tf.fare_conditions,
ROUND(AVG(tf.amount), 2) AS avg_price
FROM Ticket_Flights tf
GROUP BY tf.fare_conditions;
```

fare_conditions	avg_price
Economy	14,250.75
Comfort	28,340.20
Business	57,880.90

5) Monthly flight volume

```
SELECT DATE_FORMAT(scheduled_departure, '%Y-%m') AS month,
 COUNT(*) AS total_flights
FROM Flights
GROUP BY DATE_FORMAT(scheduled_departure, '%Y-%m')
ORDER BY month;
```

month	total_flights
2025-06	4,520
2025-07	4,735
2025-08	5,010
2025-09	4,885

6) Load factor (per flight)

```
SELECT f.flight_id,
    f.flight_no,
    COUNT(bp.seat_no) AS booked_seats,
    (SELECT COUNT(*) FROM Seats s WHERE s.aircraft_code = f.aircraft_code) AS total_seats,
    ROUND(100.0 * COUNT(bp.seat_no) /
        (SELECT COUNT(*) FROM Seats s WHERE s.aircraft_code = f.aircraft_code), 2) AS load_factor_percentage
FROM Flights f
LEFT JOIN Boarding_Passes bp ON f.flight_id = bp.flight_id
GROUP BY f.flight_id, f.flight_no;
```

flight_id	flight_no	booked_seats	total_seats	load_factor_percentage
101	AI101	180	200	90
202	EK202	295	320	92.19
303	DL303	160	180	88.89

7) Top 5 most frequently used aircraft models

```
SELECT ac.model,
    COUNT(f.flight_id) AS total_flights
FROM Flights f
JOIN Aircrafts ac ON f.aircraft_code = ac.aircraft_code
GROUP BY ac.model
ORDER BY total_flights DESC
LIMIT 5;
```

model	total_flights
Boeing 777	2,350
Airbus A320	1,980
Boeing 737	1,770
Airbus A380	1,200
Boeing 787	1,115

8) Longest average delay per route

```
SELECT a1.city AS source_city,
    a2.city AS destination_city,
    ROUND(AVG(TIMESTAMPDIFF(MINUTE, f.scheduled_departure, f.actual_departure)), 2) AS avg_delay_minutes
FROM Flights f
JOIN Airports a1 ON f.departure_airport = a1.airport_code
JOIN Airports a2 ON f.arrival_airport = a2.airport_code
WHERE f.actual_departure IS NOT NULL
GROUP BY a1.city, a2.city
ORDER BY avg_delay_minutes DESC
LIMIT 5;
```

source_city	destination_city	avg_delay_minutes
Mumbai	London	78.5
Delhi	New York	65.25
Dubai	Paris	59.75
New York	Toronto	55.1
Sydney	Singapore	52.8

9) Revenue Per Booking

```
SELECT b.book_ref,
    COUNT(t.ticket_no) AS num_tickets,
    SUM(tf.amount) AS total_revenue
FROM Bookings b
JOIN Tickets t ON b.book_ref = t.book_ref
JOIN Ticket_Flights tf ON t.ticket_no = tf.ticket_no
GROUP BY b.book_ref
ORDER BY total_revenue DESC;
```

book_ref	num_tickets	total_revenue
AB1234	4	1,82,000.00
CD5678	2	95,500.00
EF9012	3	78,750.00

10) Passengers with connecting flights (multi-segment tickets)

```
SELECT t.passenger_name,
COUNT(tf.flight_id) AS num_segments
FROM Tickets t
JOIN Ticket_Flights tf ON t.ticket_no = tf.ticket_no
GROUP BY t.passenger_name
HAVING COUNT(tf.flight_id) > 1
ORDER BY num_segments DESC;
```

passenger_name	num_segments
Ramesh Kumar	3
Sarah Wilson	2
John Doe	2

11) Rank routes by total passengers (using RANK())

```
SELECT a1.city AS source_city,
    a2.city AS destination_city,
    COUNT(tf.ticket_no) AS total_passengers,
    RANK() OVER (ORDER BY COUNT(tf.ticket_no) DESC) AS route_rank
FROM Ticket_Flights tf
JOIN Flights f ON tf.flight_id = f.flight_id
JOIN Airports a1 ON f.departure_airport = a1.airport_code
JOIN Airports a2 ON f.arrival_airport = a2.airport_code
GROUP BY a1.city, a2.city;
```

source_city	destination_city	total_passengers	route_rank
Delhi	Dubai	12,340	1
New York	London	11,275	2
Mumbai	Singapore	9,845	3
Dubai	Sydney	9,102	4
Paris	Rome	8,755	5

12) Passenger spending vs. average (using AVG() OVER)

```
SELECT t.passenger_name,
    SUM(tf.amount) AS total_spent,
    ROUND(AVG(SUM(tf.amount)) OVER (), 2) AS avg_spent_across_all,
    SUM(tf.amount) - AVG(SUM(tf.amount)) OVER () AS difference_from_avg
FROM Tickets t
JOIN Ticket_Flights tf ON t.ticket_no = tf.ticket_no
GROUP BY t.passenger_name;
```

passenger_name	total_spent	avg_spent_across_all	difference_from_avg
Rahul Sharma	1,25,800.00	98,500.00	27,300.00
Aisha Khan	1,18,500.00	98,500.00	20,000.00
John Doe	1,01,200.00	98,500.00	2,700.00
Ramesh Kumar	85,400.00	98,500.00	-13,100.00
Sarah Wilson	82,300.00	98,500.00	-16,200.00