AirlineDB

The Airline Database is a database used as the basis for tutorials to teach SQL. The database contains the flights data related to transactions, tickets, flights, airports and flight schedules.

AirlineDB SQL Questions and Queries

Task 1:

Question: How many tickets are there without boarding passes?

Expected Output: A single count of tickets that don't have any associated boarding passes.

Query Written:

```
SELECT COUNT(DISTINCT t.ticket_no) AS tickets_without_boarding_passes
FROM tickets t
LEFT JOIN boarding_passes bp ON t.ticket_no = bp.ticket_no
WHERE bp.ticket_no IS NULL;
```

Task 2:

Question: Represent the book_date column in 'yyyy-mm-dd' format using the Bookings Table.

Expected Output: All booking records with book_date formatted as 'yyyy-mm-dd'.

Query Written:

```
SELECT
  book_ref,
  TO_CHAR(book_date, 'YYYY-MM-DD') AS formatted_book_date,
  total_amount
FROM bookings;
```

Task 3:

Question: Rank Airports based on the number of flights departing from them.

Expected Output: List of airports with their departure flight counts and rank.

Query Written:

```
SELECT
  departure_airport,
  COUNT(flight_id) AS total_flights,
  RANK() OVER( ORDER BY COUNT(flight_id) DESC) AS airport_rank
FROM flights
GROUP BY departure_airport;
```

Task 4:

Question: Find the total revenue generated from each aircraft model, along with the average ticket price per flight for that model. Show only aircraft models that have generated more than 1,000,000 in total revenue.

Expected Output: Aircraft model, total revenue, and average ticket price per flight, filtered by revenue threshold.

Query Written:

```
SELECT
   ac.model AS aircraft_model,
   SUM(tf.amount) AS total_revenue,
   AVG(tf.amount) AS avg_ticket_price
FROM aircrafts ac
INNER JOIN flights f ON ac.aircraft_code = f.aircraft_code
INNER JOIN ticket_flights tf ON f.flight_id = tf.flight_id
GROUP BY ac.model
HAVING SUM(tf.amount) > 1000000
ORDER BY total_revenue DESC;
```

Task 5:

Question: For each passenger, display their ticket number, total amount paid, and classify them as 'Premium' if they paid more than the average ticket price, 'Standard' if equal to average, or 'Budget' if below average.

Expected Output: Ticket number, passenger name, total amount, and passenger category based on spending.

Query Written:

```
SELECT
  t.ticket_no,
  t.passenger_name,
  SUM(tf.amount) AS total_paid,
  CASE
    WHEN SUM(tf.amount) > (SELECT AVG(amount) FROM ticket_flights) THEN 'Premium'
    WHEN SUM(tf.amount) = (SELECT AVG(amount) FROM ticket_flights) THEN 'Standard'
    ELSE 'Budget'
  END AS passenger_category
FROM tickets t
INNER JOIN ticket_flights tf ON t.ticket_no = tf.ticket_no
GROUP BY t.ticket_no, t.passenger_name
ORDER BY total_paid DESC;
```

Task 6:

Question: Calculate the running total of bookings and the difference from the previous day's booking count for each date. Also show the 7-day moving average of total bookings.

Expected Output: Date, daily booking count, running total, difference from previous day, and 7-day moving average.

Query Written:

```
SELECT
   DATE(book_date) AS booking_date,
   COUNT(*) AS daily_bookings,
   SUM(COUNT(*)) OVER (ORDER BY DATE(book_date)) AS running_total,
   COUNT(*) - LAG(COUNT(*)) OVER (ORDER BY DATE(book_date)) AS
diff_from_previous_day,
   AVG(COUNT(*)) OVER (ORDER BY DATE(book_date) ROWS BETWEEN 6 PRECEDING
AND CURRENT ROW) AS moving_avg_7days
FROM bookings
GROUP BY DATE(book_date)
ORDER BY booking_date;
```

Task 7:

Question: Find all flights where the actual departure was delayed by more than 2 hours compared to scheduled departure. For each delayed flight, show the departure airport name, arrival airport name, flight number, delay duration in hours, and the status (classify as 'Minor Delay' if 2-4 hours, 'Major Delay' if 4-8 hours, 'Severe Delay' if more than 8 hours).

Expected Output: Flight details with delay classification for flights delayed more than 2 hours.

Query Written:

```
SELECT
   dep.airport_name AS departure_airport,
   arr.airport_name AS arrival_airport,
```

```
  f.flight_no,
  EXTRACT(EPOCH FROM (f.actual_departure - f.scheduled_departure)) / 3600 AS
delay_hours,
  CASE
    WHEN EXTRACT(EPOCH FROM (f.actual_departure - f.scheduled_departure)) / 3600
BETWEEN 2 AND 4 THEN 'Minor Delay'
    WHEN EXTRACT(EPOCH FROM (f.actual_departure - f.scheduled_departure)) / 3600
BETWEEN 4 AND 8 THEN 'Major Delay'
    WHEN EXTRACT(EPOCH FROM (f.actual_departure - f.scheduled_departure)) / 3600
> 8 THEN 'Severe Delay'
  END AS delay_status
FROM flights f
INNER JOIN airports dep ON f.departure_airport = dep.airport_code
INNER JOIN airports arr ON f.arrival_airport = arr.airport_code
WHERE EXTRACT(EPOCH FROM (f.actual_departure - f.scheduled_departure)) / 3600 >
2
  AND f.actual_departure IS NOT NULL
ORDER BY delay_hours DESC;
```

Task 8:

Question: Display the top 3 most popular routes (departure-arrival airport pairs) by number of flights, along with the percentage of total flights each route represents. Also show the most common aircraft model used on each route.

Expected Output: Route information with flight count, percentage, and most used aircraft model.

Query Written:

```
WITH route_stats AS (
  SELECT
    f.departure_airport,
    f.arrival_airport,
    COUNT(*) AS flight_count,
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM flights), 2) AS
percentage_of_total
  FROM flights f
  GROUP BY f.departure_airport, f.arrival_airport
),
route_aircraft AS (
  SELECT
    f.departure_airport,
    f.arrival_airport,
    f.aircraft_code,
    ROW_NUMBER() OVER (PARTITION BY f.departure_airport, f.arrival_airport
                ORDER BY COUNT(*) DESC) AS aircraft_rank
  FROM flights f
  GROUP BY f.departure_airport, f.arrival_airport, f.aircraft_code
)
SELECT
  dep.airport_name AS departure_airport,
```

```
    arr.airport_name AS arrival_airport,
    rs.flight_count,
    rs.percentage_of_total,
    ac.model AS most_common_aircraft
FROM route_stats rs
INNER JOIN airports dep ON rs.departure_airport = dep.airport_code
INNER JOIN airports arr ON rs.arrival_airport = arr.airport_code
INNER JOIN route_aircraft ra ON rs.departure_airport = ra.departure_airport
    AND rs.arrival_airport = ra.arrival_airport
INNER JOIN aircrafts ac ON ra.aircraft_code = ac.aircraft_code
WHERE ra.aircraft_rank = 1
ORDER BY rs.flight_count DESC
LIMIT 3;
```

Task 9:

Question: For each booking, calculate the total number of passengers, total amount paid, number of flight segments, and the seat class distribution (Economy/Comfort/Business). Show only bookings with more than 2 passengers.

Expected Output: Booking details with passenger count, amount, flight segments, and count of passengers in each fare class.

Query Written:

```
SELECT
    b.book_ref,
    COUNT(DISTINCT t.ticket_no) AS total_passengers,
    b.total_amount,
    COUNT(DISTINCT tf.flight_id) AS flight_segments,
    SUM(CASE WHEN tf.fare_conditions = 'Economy' THEN 1 ELSE 0 END) AS
economy_count,
```

```
  SUM(CASE WHEN tf.fare_conditions = 'Comfort' THEN 1 ELSE 0 END) AS
comfort_count,
  SUM(CASE WHEN tf.fare_conditions = 'Business' THEN 1 ELSE 0 END) AS
business_count
FROM bookings b
INNER JOIN tickets t ON b.book_ref = t.book_ref
INNER JOIN ticket_flights tf ON t.ticket_no = tf.ticket_no
GROUP BY b.book_ref, b.total_amount
HAVING COUNT(DISTINCT t.ticket_no) > 2
ORDER BY total_passengers DESC;
```

Task 10:

Question: Find airports that have both domestic and international flights. For each such airport, show the count of domestic flights, international flights, and calculate what percentage of their total flights are international. (Assume domestic flights are within the same country based on airport timezone).

Expected Output: Airport name, domestic flight count, international flight count, and percentage of international flights.

Query Written:

```
WITH flight_classification AS (
  SELECT
    f.departure_airport,
    dep.airport_name,
    CASE
      WHEN SUBSTRING(dep.timezone FROM 1 FOR 3) = SUBSTRING(arr.timezone
FROM 1 FOR 3)
      THEN 'Domestic'
      ELSE 'International'
    END AS flight_type
  FROM flights f
```

```
    INNER JOIN airports dep ON f.departure_airport = dep.airport_code
    INNER JOIN airports arr ON f.arrival_airport = arr.airport_code
)
SELECT
    airport_name,
    SUM(CASE WHEN flight_type = 'Domestic' THEN 1 ELSE 0 END) AS domestic_flights,
    SUM(CASE WHEN flight_type = 'International' THEN 1 ELSE 0 END) AS
international_flights,
    ROUND(
        SUM(CASE WHEN flight_type = 'International' THEN 1 ELSE 0 END) * 100.0 /
        COUNT(*), 2
    ) AS international_percentage
FROM flight_classification
GROUP BY departure_airport, airport_name
HAVING SUM(CASE WHEN flight_type = 'Domestic' THEN 1 ELSE 0 END) > 0
    AND SUM(CASE WHEN flight_type = 'International' THEN 1 ELSE 0 END) > 0
ORDER BY international_percentage DESC;
```
