QUESTION 3 Part B: Query Optimization

The original query is performing multiple joins and an aggregation (SUM(price)) on large tables (airline, flight, booking). Without indexes, the database would need to scan entire tables to find matching rows for joins and to compute the total revenue, which can be very slow, especially with a large dataset; the query is taking too long.

To speed up the query, I created indexes on the columns involved in joins and aggregation:

1. **idx_airline_id** on the airline table for the join with flight.
2. **idx_airline_id_flight** on the flight table to help with the join between airline and flight.
3. **idx_flight_id_flight** on the flight table for the join with booking.
4. **idx_flight_id_booking** on the booking table to speed up the join with flight.
5. **idx_price_booking** on the price column in the booking table to optimize the SUM(price) aggregation.

After adding the indexes, I ran the query again. The query execution time improved significantly, and the database no longer had to scan the entire tables for each join.

Results: By adding these indexes, the query became much faster because the database could quickly look up the rows needed for the joins and the aggregation. The overall performance improved, reducing the time taken to get the top 20 airlines by total revenue.