Table of Contents

1.Business Context

SkyLite Air, based in London, operates a global network, focusing on efficient and reliable air travel. The Operations Department oversees flight scheduling, crew coordination, aircraft maintenance, and route profitability to ensure smooth operations and cost-effectiveness.

Recently, SkyLite has experienced stagnant revenue growth, signaling a need to identify opportunities for expansion and optimize existing operations. The Operations Department lacks actionable insights to identify underperforming routes, optimize fleet allocation, and control expenses, necessitating a data-driven approach to improve route profitability and operational efficiency.

This project aims to develop a data product that equips the Operations Department with insights to optimize flight schedules, enhance route profitability, reduce costs, and improve fleet utilization, supporting data-driven decision-making.

Database Scope and Entities

The data product captures key operational data across interconnected entities:

- Flights: Details including flight ID, route, aircraft, schedule and seats booked.
- Transaction: Payment amounts and seat categories per booking.
- Aircraft: Fleet details, including model, capacity, and maintenance limits.
- Routes & Airports: Flight durations and airport details.
- Cost: Operational expenses such as fuel, crew salary, and turnaround costs.

Expected Reports and Insights

The purpose of this data product is to optimize flight scheduling, manage fleet and route profitability, track costs, and improve resource allocation. By analyzing data across these entities, the database will generate key reports such:

Operational Efficiency

- Peak Travel Day Analysis: Identifies busy days based on bookings to adjust schedules and aircraft allocation.
- Flight Demand: Compares seats booked vs. seat capacity to adjust fleet size and flight frequency.
- Flight Utilization: Tracks aircraft efficiency to avoid under-utilization.

Cost and Revenue Analysis

- Route Revenue: Compares revenue per route with operational costs to identify profitable routes.
- Cost Optimization: Compares revenue vs. costs (fuel, crew, turnaround) to reduce expenses.

These insights will enable SkyLite Air, to make data-driven decisions, improving operational efficiency, customer satisfaction, and profitability.

2. Database Design

The ERD for SkyLite Air has been designed using Crow's Foot Notation to clearly represent the entities, attributes, and relationships.

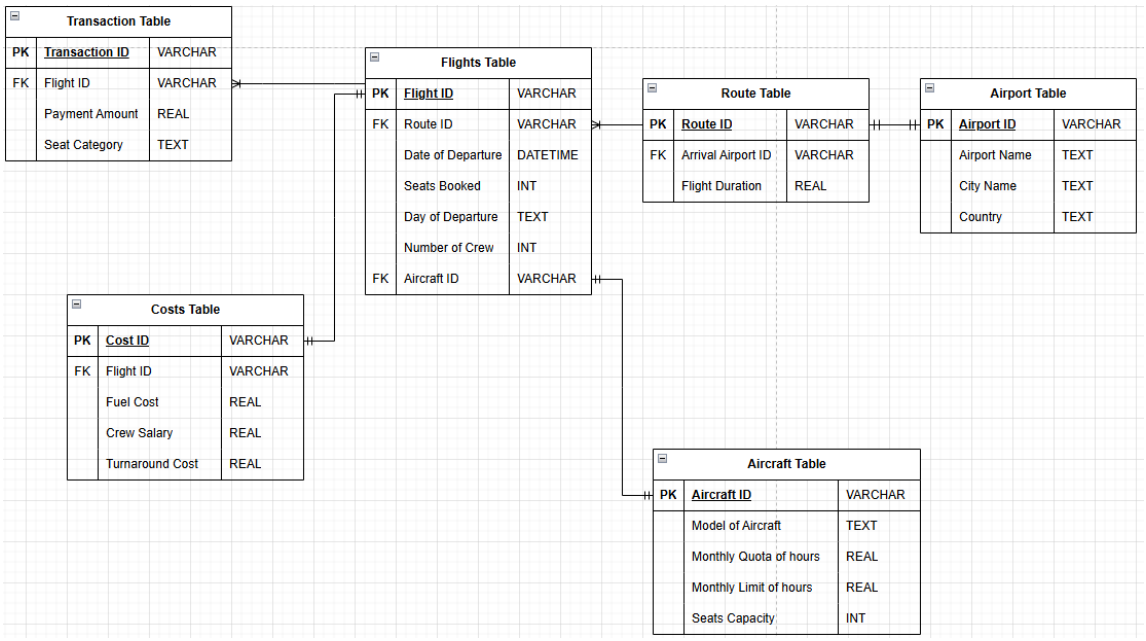

Figure 1: E-R Diagram

Entities, Attributes, and Keys

The database schema includes six core entities: Flight, Airport, Route, Aircraft, Transaction, and Cost.

Refer to Appendix A for the detailed schema documentation.

Relationships and Cardinality

Based on the design, the following relationships have been defined between the entities. Below is an explanation of the relationships and their cardinality:

Entity Relationship	Cardinality	Description
Flights → Routes	Many-to-One	Multiple flights can belong to the same route.
Flights → Aircraft	Many-to-One	One aircraft can be used for multiple flights, but each flight operates with a single aircraft.
Flights → Transactions	One-to-Many	A flight can have multiple transactions (ticket sales), but each transaction is associated with one specific flight.
Flights → Costs	One-to-One	Each flight has a single entry for its operational costs, ensuring that cost information is directly tied to a specific flight.
Routes → Airports	One-to-One	Each route leads to a unique arrival airport, considering all flights depart from London, so there will be no multiple routes for the same arrival airport.

Table 1: Cardinality of relationships

Assumptions

- Each booking corresponds to a single transaction, updating total seats booked per flight without exceeding the aircraft capacity.
- Fuel efficiency and crew salaries are fixed per flight, covering the entire crew.
- All flights depart from London Heathrow (LHR), with unique arrival airports per route.
- Flight durations are in hours, with a base price of £50/hour for cost calculations.
- Each city has one airport, so airport name isn't unique within cities.

This design ensures efficient data management, minimizes redundancy, and supports data integrity, optimizing flight scheduling, cost analysis, and operations.

3. SQL Schema Implementation

The schema is designed to ensure normalization, referential integrity, and efficient data management.

Data Types

- VARCHAR(n): For unique identifiers (*aircraft_id, airport_id, route_id, flight_id*).
- TEXT: For variable-length strings (*aircraft_model, airport_name, airport_city, city_code, seat_category*).
- DATETIME: For date variable (*departure_day*)
- INTEGER: For whole numbers (*aircraft_quota, aircraft_limit, number_of_crew, seats_booked*).
- REAL: For monetary and floating-point values (*payment_amount, fuel_cost, crew_salary, turnaround_cost*).

Schema Implementation

Refer to Appendix B for the Schema implementation code.

Constraints

1. Primary Key (PK): Uniquely identifies records in each table.
2. Foreign Key (FK): Ensures referential integrity between tables.
3. Check Constraints:
    - Transaction: Payment Amount > 0.

- Route: Flight Duration > 0.
- Cost: Flight ID must be unique to prevent duplicate cost entries.

4. Referential Integrity: Maintains consistency by setting related records to NULL when a referenced flight is deleted.
5. NOT NULL: Applied to ID fields to ensure mandatory values.

Normalization Considerations

The schema design follows the principles of normalization to reduce redundancy and ensure efficient data management:

1. **First Normal Form (1NF):** Ensured atomic values by storing indivisible data points in separate fields. For example, *seats_booked* in the flights table holds a single value, and airport details like *airport_name, airport_city,* and *airport_country* are stored in the airports table, preventing duplication.
2. **Second Normal Form (2NF):** Eliminated partial dependencies by ensuring non-key attributes depend entirely on the primary key. For instance, *departure_date* and *seats_booked* depend on *flight_id* in the flights table, while *flight_duration* is stored in the routes table to avoid redundancy.
3. **Third Normal Form (3NF):** Removed transitive dependencies by organizing data into distinct tables. Airport details reside in the airports table, referenced by *arrival_airport* in routes, while payment data is tracked in transactions, ensuring accurate revenue calculations without redundancy.

Alternative Designs and ERD Modifications

Two key modifications are proposed to enhance SkyLite Air's database: a Booking table to manage multiple transactions per booking, introducing many-to-one relationships with flights and transactions, and a Crew table to streamline crew assignment tracking, establishing a many-to-

one relationship between crew and flights. These changes reduce redundancy, improve data integrity, and better align the structure with real-world operations.

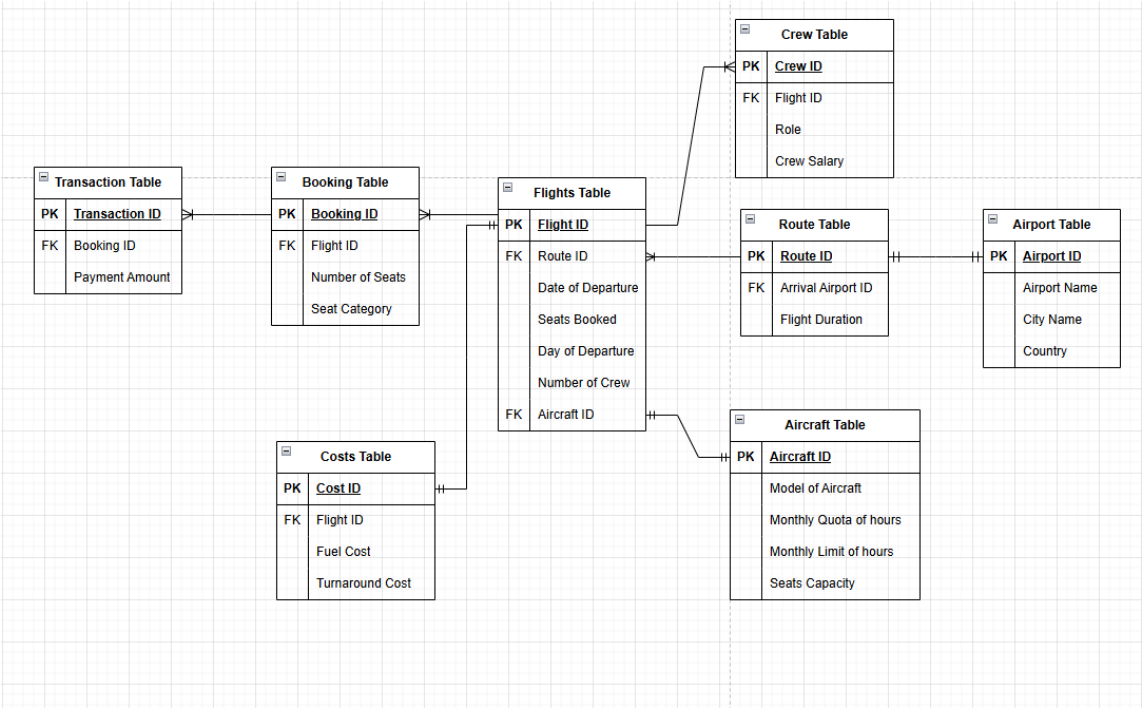

Figure 2: Alternative ERD

4. Synthetic Data Generation

Data Generation Process

To populate the SkyLite Air database, we used Mockaroo to generate synthetic data, aligning with the defined schema and ensuring realistic patterns.

1. **Schema Definition:**
   - Created tables in Mockaroo to mirror the database structure (Aircraft, Airport, Route, Flight, Transaction, Cost).
   - Specified appropriate data types and formats for each field (e.g., names, dates, numbers).
2. **Customized Data Generation:**
   - Simulated major international airports with unique codes and ensured unique arrival airports per route.

- Defined aircraft seat capacities, monthly quotas, and operational limits to reflect fleet diversity.
- Simulated diverse booking patterns by maintaining a 3:1 ratio of economy to business class seats, with business fares set at 2x the economy price for realistic pricing.
- Increased *seats_booked* for high-demand routes, adjusting ranges based on aircraft models (e.g., 100–150 for smaller aircraft, 200–500 for larger models), and assigned randomized departure dates.
- Applied peak travel day pricing: ticket prices were calculated as base rate per hour × seat category (economy = base, business = 2x), with peak days at 1.5x, Tuesdays at 1.2x, and other days at the base rate.
- Transactions were aligned with *seats_booked*, simulating realistic behavior, and costs were calculated based on flight duration and aircraft type.
- Data was generated in *Mockaroo*, imported into *SQLite*, and managed with *Python* scripts to enforce foreign key integrity and handle errors.

Assumptions

- Pricing Structure: Ticket prices depend on seat category, flight duration, and day of departure.
- Demand Patterns: Higher bookings on popular routes, with peak travel days on Wednesdays and Thursdays.
- Seat Category Ratio: Maintained a 3:1 ratio of economy to business class seats.

Refer to Appendix C for Import Script and Data Validation.

Sample Data

Here are a few example rows for each table to illustrate the data generated:

airport_id	airport_name	airport_city	city_code	airport_country
1	Athens International Airport	Athens	ATH	Greece
2	King Abdulaziz International Airport	Jeddah	JED	Saudi Arabia
3	George Bush Intercontinental Airport	Houston	IAH	United States
4	Helsinki-Vantaa Airport	Helsinki	HEL	Finland
5	Netaji Subhas Chandra Bose International Airport	Kolkata	CCU	India

Table 2: Flights Table

aircraft_id	aircraft_model	aircraft_quota	aircraft_limit	seats_capacity
1	A330	9	52	299
2	A380	10	58	538
3	A320	3	51	193
4	B777	1	60	316
5	B777	2	55	352

Table 3: Aircrafts Table

airport_id	airport_name	airport_city	city_code	airport_country
1	Athens International Airport	Athens	ATH	Greece
2	King Abdulaziz International Airport	Jeddah	JED	Saudi Arabia
3	George Bush Intercontinental Airport	Houston	IAH	United States
4	Helsinki-Vantaa Airport	Helsinki	HEL	Finland
5	Netaji Subhas Chandra Bose International Airport	Kolkata	CCU	India

Table 4: Airports Table

route_id	arrival_airport	flight_duration
01JP5JPNR6Y5KD32S8YZAAQT8R	48	2.5
01JP5JPNR61PTYNAQQTCKMFKD4	50	12
01JP5JPNR7BJBZDRBPAWEE51C1	37	2.5
01JP5JPNR72TVSKD8RCABR998H	10	11
01JP5JPNR8SKG268NAGBBE99S7	3	9

Table 5: Routes Table

route_id	arrival_airport	flight_duration
01JP5JPNR6Y5KD32S8YZAAQT8R	48	2.5
01JP5JPNR61PTYNAQQTCKMFKD4	50	12
01JP5JPNR7BJBZDRBPAWEE51C1	37	2.5
01JP5JPNR72TVSKD8RCABR998H	10	11
01JP5JPNR8SKG268NAGBBE99S7	3	9

Table 6: Transactions Table

cost_id	flight_id	fuel_cost	crew_salary	turnaround_cost
1	01JP7Q3EMY876TT2CCTRKJMHT8	14000	1475	2250
2	01JP7Q3ENCNX1JW8M6WF4M14P7	3000	795	1750
3	01JP7Q3ENEP1NT89JRBBDKNRRM	44000	6820	6500
4	01JP7Q3ENGKC7RM67RTH09R1EY	48000	7800	7000
5	01JP7Q3ENJGNMSQDGBNZKKMNV7	14000	3290	4500

Table 7: Costs Table

Challenges and Solutions

- Pricing Structure: Initially, flight duration wasn't factored into ticket prices, leading to inconsistencies between short and long-haul flights. Introducing a base rate per hour aligned pricing with flight length, ensuring fairness across routes.
- Demand Variation: Seat demand fluctuated across different days, requiring dynamic pricing adjustments. We applied multipliers to the base rate — 1.5x for peak days, 1.2x for Tuesdays, and the base rate for other days — to better capture booking trends.

- Seat Category Imbalance: The initial model allocated too many business-class seats, distorting revenue calculations. Adjusting the economy-to-business ratio to 3:1 ensured a balanced distribution, reflecting realistic airline practices.
- Foreign Key Dependencies: Ensured correct import order (Airports → Aircraft → Routes → Flights → Costs → Transactions) to maintain referential integrity.
- Realistic Seat Bookings: Applied constraints to prevent *seats_booked* from exceeding *seat_capacity,* verified through SQL checks.
- Date Consistency: Aligned departure dates with correct weekdays using *Mockaroo's* date functions and SQL validation.
- Data Import Issues: Resolved CSV import errors like missing values and type mismatches through manual cleaning and validation scripts.

Reflections on Data Generation Process

The data generation process required careful coordination to ensure realistic seat bookings, accurate flight schedules, and proper aircraft assignments across tables. Maintaining referential integrity was crucial to avoid mismatched foreign key relationships, and iterative adjustments helped handle edge cases like missing data and formatting inconsistencies. Ultimately, designing and implementing the SkyLite Air database resulted in a robust data model that provides a solid foundation for analyzing flight performance, route profitability, and operational costs, directly supporting data-driven decision-making.

6. Conclusion

SkyLite Air's analysis highlights key areas for optimization, including flight scheduling, route profitability, aircraft utilization, and cost efficiency. Identifying peak travel days enables strategic scheduling to maximize revenue. High-demand routes like Tokyo, Barcelona, and Paris offer expansion opportunities, while fill rate insights ensure optimal seat occupancy. Revenue analysis confirms the importance of premium routes, guiding pricing and service enhancements. Underutilized aircraft were reassigned based on demand and cost analysis, improving fleet efficiency. Cost estimations for new deployments support data-driven decisions, aligning expenses with profitability goals.

By implementing these insights, SkyLite Air can enhance efficiency, profitability, and customer experience. Continuous refinement of scheduling, pricing, and aircraft allocation will drive long-term growth and competitiveness.