Institute of Business Administration

MS Data Science Program

Hotel Management System Data Warehouse Project

From RDBMS to Star Schema with ETL Pipeline & BI Dashboard

Course Title

Data Analytics and Warehousing

Instructor

Dr. Tariq Mehmood

Submitted by

Syed Muhammad Meesum Abbas
Syed Daniyal Hussain

Date

18 May 2025

Table of Contents

Project Structure Overview

```
hotel_project/
├── csv_exports/          # Likely contains raw CSVs exported from SQLite (RDBMS)
├── dags/                 # Airflow DAG folder (contains ETL pipeline script)
├── output_data/          # Final cleaned CSVs after transformation (for DWH)
├── scripts/              # Contains your utility or helper scripts (e.g., schema creation)
├── venv/                 # Virtual environment (Python dependencies)
├── docker-compose.yml      # Docker config for running Airflow
├── export_to_csv.py       # Script to export RDBMS tables to CSV
├── hotel_erd.png          # ER Diagram of relational schema
├── hotel_management.db     # SQLite database (RDBMS)
├── hotel_star_schema.png   # Star schema diagram
├── insert_data.py         # Script to insert data into RDBMS
├── Project Report.pdf     # Final report
├── WorkFlow_Snapshots.pdf # Snapshot documentation of ETL process
├── Hotel Dashboard.pbix   # Dashboard on PowerBI
```

1. Project Introduction & Business Domain

This project focuses on building a comprehensive Hotel Management System using relational database concepts and realistic business logic. The goal is to simulate an end-to-end data journey — from designing a normalized RDBMS schema to preparing structured data for analytics through a data warehouse pipeline.

The business domain is a **mid-sized hotel operation** that offers accommodation, customer services, facilities, and staffing. Customers book rooms through different channels such as online portals, walk-ins, travel agents, or phone calls. Payments are made through various methods, and the hotel also tracks customer reviews, service usage, room maintenance activities, and staff roles.

This simulation helps in understanding the real-world challenges of:

- Managing transactional hotel operations
- Normalizing and enforcing data integrity
- Handling large-scale data (4200+ bookings)
- Transitioning from operational systems to analytical models

The solution is built using **SQLite** as the database engine and **Python** for data generation, ETL processing, and exporting to CSV files for downstream analytics. The system supports future integration with tools like Power BI, Apache Airflow, and star schema-based data warehouse models for performance insights.

2. Relational Database Schema Design

The relational schema for the Hotel Management System is designed to capture all critical operational aspects of a hotel, while ensuring **data normalization**, **referential integrity**, and **scalability**.

A total of **12 interrelated tables** were created. These include core transactional tables such as Bookings, Payments_Details, Service_Usage, and Reviews, as well as dimension-like static tables such as Customers, Rooms, Booking_Channel, and Payments.

2.1 Key Schema Highlights

- **Normalization**: The schema is normalized up to **Third Normal Form (3NF)**. Redundancy is minimized and all foreign key relationships are explicitly defined.
- **Referential Integrity**: Proper foreign key constraints ensure consistency across entities (e.g., a payment must be linked to a valid booking).
- **Realistic Structure**: Tables reflect real hotel operations — including multiple booking channels, service tracking, and staff-managed room maintenance.

2.2 List of Tables and Purpose

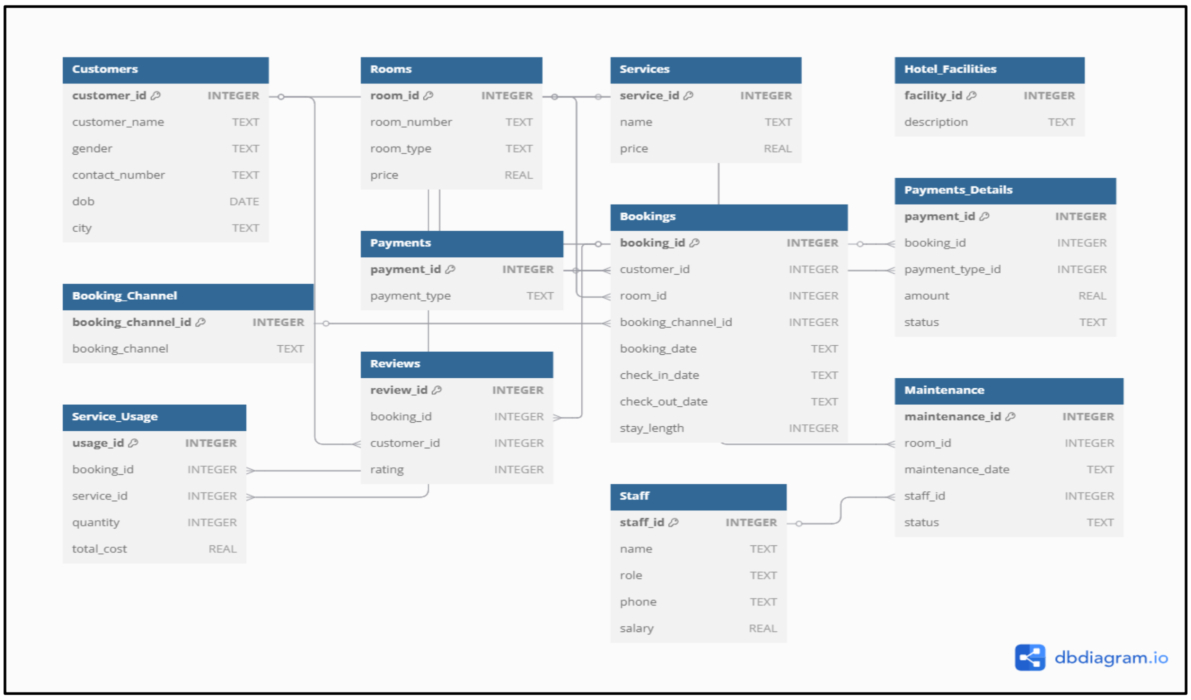
Table Name	Description
Customers	Stores personal details of customers including name, contact, DOB, and city
Rooms	Contains 30 physical rooms (Single, Double, Suite), with unique room numbers
Services	Defines additional services like laundry, spa, and room service
Hotel_Facilities	Static reference for on-site facilities like gym and conference room
Booking_Channel	Predefined methods of reservation (Online, Walk-in, etc.)
Payments	Payment types such as Cash, Card, and Online
Bookings	Core transaction table linking customer, room, and booking details
Payments_Details	Captures total amount paid per booking, including room + service charges
Service_Usage	Tracks additional services availed during a booking
Reviews	Stores customer feedback through ratings (1–5)
Staff	Maintains staff profiles and salary info
Maintenance	Records room maintenance requests and staff assignments

2.3 Entity Relationship Summary

The schema has been carefully designed with clear one-to-one, one-to-many, and many-to-one relationships between tables. Here's a breakdown of how the entities are connected:

- **Customers ↔ Bookings**
  Each customer can make **multiple bookings**. A booking must be linked to **one customer**.
- **Bookings ↔ Rooms**
  Each booking is made for **one room**, but a room can be booked **multiple times** across different customers and dates.
- **Bookings ↔ Booking_Channel**
  Every booking is associated with a **single booking channel** (e.g., Online, Walk-in), but each channel may be used in **many bookings**.
- **Bookings ↔ Payments_Details**
  There is a **one-to-one relationship**: each booking results in exactly **one payment record**. The total payment amount includes both **room charges** and **service costs**.
- **Payments_Details ↔ Payments**
  Each payment is made using **one payment type** (e.g., Cash), and each type can be associated with **many transactions**.
- **Bookings ↔ Service_Usage**
  Each booking may have **multiple service usage entries** (e.g., laundry, gym), forming a **one-to-many relationship**. Each record logs the quantity and total cost of that specific service.
- **Service_Usage ↔ Services**
  Each service usage entry is linked to **one service**. A single service (e.g., Spa) may be used in **many bookings**.
- **Bookings ↔ Reviews**
  A booking may optionally have a **review**, creating a **one-to-one or zero** relationship. One customer can submit **many reviews** if they've made multiple bookings.
- Rooms ↔ Maintenance
  Each room may have multiple maintenance requests, and each request is linked to one room.
- **Maintenance ↔ Staff**
  Every maintenance record is assigned to **one staff member**, and each staff member can be responsible for **many tasks**.
- **Hotel_Facilities**
  This is a **standalone static table** used only for display/reference and is not linked to other transactional tables.

This enhanced relational structure ensures data consistency and provides a solid foundation for **ETL workflows**, **reporting**, and **data warehouse transformation**.



3. Data Generation & Insertion Strategy

To simulate realistic hotel operations, the dataset was entirely generated using a combination of **custom name lists**, the **Faker library**, and **predefined dictionaries** for domain-specific fields. The goal was to ensure that the data reflected practical hotel scenarios while maintaining referential integrity across tables.

3.1 Data Generation Overview

The following approach was used to populate each table:

Table	Strategy
Customers	50 total (35 male + 15 female) with custom names, randomized DOB, city, contact
Rooms	30 unique physical rooms: 10 Single, 10 Double, 10 Suite, with unique room numbers
Services	4 predefined hotel services (Laundry, Room Service, Spa, Gym)
Hotel_Facilities	5 static entries (e.g., Gymnasium, Conference Room)
Booking_Channel	4 fixed types: Online, Walk-in, Travel Agent, Phone Call
Payments	3 payment modes: Online, Cash, Card
Staff	15 staff members with random names, phone numbers, roles, and salaries
Maintenance	20 tasks linking staff to rooms with random dates and status
Bookings	~4200 bookings across 2024, with 8–15 bookings per day
Service_Usage	0–3 services randomly assigned per booking, with quantity and cost
Payments_Details	Calculated as (room price × stay length) + service cost per booking
Reviews	~65% of bookings have a rating from 1 to 5

3.2 Controlled Randomization with Domain Logic

Instead of using blind random data generation, controlled logic was applied:

- **Room numbers** were made unique using structured codes like SI01, DO05, SU10.
- **DOBs** ranged from 1950 to 2005 to keep age calculations realistic.
- **Stay lengths** were constrained to 1–7 days to reflect typical hotel visits.
- **Service usage** was selectively added (0 to 3 per booking) to allow natural variation.
- **Reviews** were added probabilistically (~65%) to simulate real customer feedback behavior.

3.3 Tools and Libraries Used

- Faker: For realistic names, dates, phone numbers, and randomness
- Pandas: For data manipulation and CSV exporting
- SQLite3: For database operations using Python
- Manual dictionaries: For booking channels, payment types, and service definitions

This structured data creation process resulted in a clean, relationally consistent dataset, suitable not just for transactional operations, but also for data warehousing and business intelligence.

4. Data Export for Data Warehouse

After the successful generation and population of all tables in the hotel_management.db database, the next critical step was to make the data available for **data warehouse processing and dashboarding**. To support this, all relational tables were exported as **individual CSV files**, forming the raw input for the star schema transformation.

4.1 Why Export to CSV?

- **Decoupling**: Exporting to CSV separates operational data from analytical systems.
- **Interoperability**: CSV is widely supported by ETL tools, Power BI, Excel, Python, and SQL engines.
- **Versioning & Reuse**: CSVs can be version-controlled, reloaded, or reused for various data warehouse designs.

4.2 Export Methodology

A dedicated Python export script was created to:

- Connect to the SQLite database (hotel_management.db)
- Query each of the 12 tables using pandas.read_sql()
- Export each table to a CSV file using to_csv()
- Store all files in a clean folder named csv_exports/

4.3 Exported Tables

CSV File Name	Source Table in RDBMS
customers.csv	Customers
rooms.csv	Rooms
services.csv	Services
hotel_facilities.csv	Hotel_Facilities
booking_channel.csv	Booking_Channel
payments.csv	Payments
bookings.csv	Bookings
payments_details.csv	Payments_Details
service_usage.csv	Service_Usage
reviews.csv	Reviews
staff.csv	Staff
maintenance.csv	Maintenance

Each CSV file contains a **flat snapshot** of the table with all foreign keys intact, making it directly usable for **star schema transformation or ETL pipelines**.

4.4 Folder Structure

The export script creates a folder like this:

```
hotel_rdbms/
├── hotel_management.db
├── insert_data.py
├── export_to_csv.py
└── csv_exports/
    ├── bookings.csv
    ├── payments_details.csv
    ├── ...
```

This step prepares the data for the transition from transactional to analytical systems, and is essential for the next phase — **designing a star schema and ETL flow**.

5. Data Warehouse Design (Star Schema)

5.1 Business Scenario & Analytical Need

This project revolves around a hotel management system that tracks customer bookings, service usage, payments, and room maintenance through a transactional relational database. While this system works efficiently for day-to-day operations like check-ins, payments, and room assignment, it is not designed to support analytical needs such as:

- Understanding customer behavior patterns
- Tracking service revenue trends
- Measuring booking performance by channel or season
- Evaluating satisfaction through ratings

To meet these analytical objectives, we transitioned from a normalized relational database structure to a denormalized star schema suitable for business intelligence and reporting.

5.2 Purpose of the Star Schema

Relational databases are optimized for data integrity and transactional consistency. However, when it comes to analytics—especially when using tools like Power BI or Tableau—performance becomes an issue due to complex joins across many tables.

The star schema is a dimensional model designed to solve this problem. By denormalizing relevant data into a central fact table surrounded by multiple descriptive dimension tables, we achieve:

- Faster aggregations and summarizations
- Simpler and more readable SQL queries
- Better compatibility with OLAP engines and BI dashboards
- Improved flexibility for slicing and filtering data across business categories

5.3 Dimensional Modeling Overview

Our star schema revolves around one fact table, fact_reservation, which captures all core booking metrics. This table connects to five supporting dimension tables, each offering a specific analytical lens.

Table Name	Type	Purpose
fact_reservation	Fact	Tracks metrics like stay length, service amount, total paid, and rating
dim_customer	Dimension	Contains customer demographic data (ID, name, gender, city)
dim_room	Dimension	Holds room details (room ID and type)
dim_payment	Dimension	Describes payment methods (cash, card, online)
dim_booking_channel	Dimension	Categorizes how bookings were made (walk-in, online, phone)
dim_date	Dimension	Breaks down check-in dates by day, month, year, and calendar attributes

5.4 Mapping from RDBMS to Star Schema

Our star schema was constructed from selected tables in the normalized RDBMS, with necessary transformations and joins handled in the ETL process.

Star Table	Derived From RDBMS Tables	Notes
fact_reservation	Bookings + Payments_Details + Rooms + Reviews + Service_Usage	Aggregates stay length, room cost, service total, and rating
dim_customer	Customers	Filtered for ID, name, gender, city
dim_room	Rooms	Includes room ID and type
dim_payment	Payments	Extracted payment types only
dim_booking_channel	Booking_Channel	Four predefined values
dim_date	Generated independently	Based on check-in date from Bookings table

5.5 Date Dimension Strategy

The dim_date table was generated independently to support time-based analysis such as monthly revenue trends or weekday vs. weekend usage. It includes one row for each day in the year 2024.

Each row includes:

- date_key (YYYYMMDD format)
- Full date
- day, month, year
- day_name, week_number, quarter
- is_weekend flag

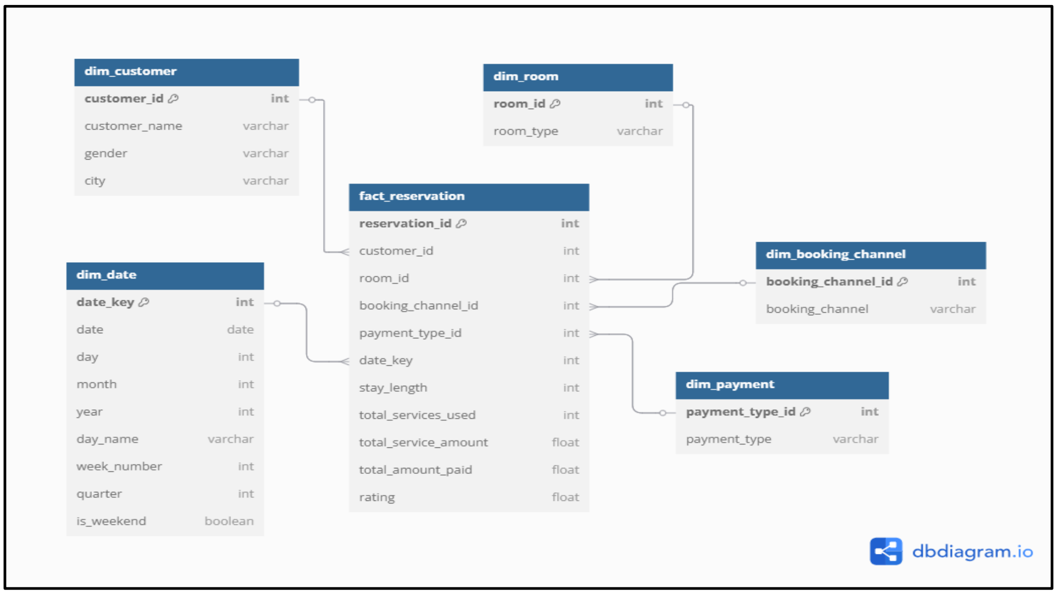
This structure enables flexible temporal analysis across daily, weekly, monthly, and quarterly intervals.

5.6 Benefits of the Star Schema Design

- **Query Performance**: Reduces the number of joins required for common queries
- **BI Compatibility**: Ideal structure for Power BI, Tableau, and OLAP-based dashboards
- **Analytics Flexibility**: Supports grouping and filtering by customer type, booking channel, room type, etc.
- **Scalability**: Can be extended easily by adding new metrics or dimensions
- **ETL Readiness**: Clean mapping to Airflow-based ETL pipelines using Docker containers

5.7 Star Schema Visualization

The diagram below visually represents how the central fact_reservation table is connected to all supporting dimensions.



This visual clarifies the one-to-many relationships and showcases how the model supports multidimensional analysis.

6. ETL Pipeline Architecture

6.1 Purpose of the ETL Process

The ETL (Extract, Transform, Load) pipeline plays a central role in reshaping transactional data from the RDBMS into a clean, denormalized format suitable for analytics. This transformation is essential because the original schema is normalized and contains many-to-many relationships, redundant columns, and complex joins all of which require complex queries that are slow to execute and difficult to visualize efficiently in business intelligence (BI) tools. The ETL process enables automation, repeatability, and prepares the data for downstream reporting and dashboarding.

6.2 Overview of the Pipeline Architecture

The architecture follows a modular, DAG-based structure implemented using **Apache Airflow**, and runs inside **Docker**. The ETL workflow reads CSV exports from the transactional database (csv_exports/), transforms them using custom Python logic, and saves the results to a folder (output_data/) in the form of clean dimension and fact tables.

Each task in the pipeline is represented as an Airflow task node, enabling control over dependencies, retries, and logging.

6.3 Transformation Strategy

The transformation logic was driven by the requirements of the star schema, which simplifies querying by using a central fact table surrounded by supporting dimensions. The following strategies were used:

- **Column Selection**: Only relevant attributes from each table were extracted (e.g., customer name, gender, city) to reduce unnecessary data volume.
- **Foreign Key Mapping**: Complex fields like check_in_date in the bookings.csv file were mapped to a date_key from a pre-generated dim_date table, enabling time-based analysis without joins on full date columns.
- **Aggregation**: For service data, we grouped rows by booking to calculate:
  - total_services_used (count of services)
  - total_service_amount (sum of costs)
- **Joining Multiple Tables**: The fact table combines data from bookings, payments_details, rooms, reviews, and service_usage, creating a single row per reservation (reservation_id) with all the relevant metrics.
- **Handling Missing Values**: Optional values like service usage or reviews were filled with 0 or left as NULL where applicable (e.g., rating).
- **Data Cleaning**: As the data was already clean, no imputation or outlier handling was needed — only validation to confirm the absence of missing or inconsistent values in star schema columns.

These transformations were applied inside modular Python functions stored in scripts/etl_functions.py, each dedicated to a specific star schema table.

6.4 DAG Design in Apache Airflow

The Airflow DAG, named etl_star_schema, defines six core tasks using PythonOperator. Five tasks process dimension tables in parallel:

- process_dim_customer
- process_dim_room
- process_dim_payment
- process_dim_booking_channel
- process_dim_date

The final task, process_fact_reservation, depends on all of the above and generates the central fact table by joining the appropriate dimension-linked values. This design ensures that the fact table is always built after all dimensions are complete and available.

6.5 Docker-Based Execution

All Airflow services were containerized using Docker for easy setup and consistency across environments. The docker-compose.yml file defines services for:

- **PostgreSQL** – for Airflow metadata
- **Airflow Webserver** – hosts the UI at localhost:8080
- **Airflow Scheduler** – triggers DAGs based on schedule or manual input
- **Airflow Init** – initializes the metadata database during first-time setup

Airflow interacts with mounted folders from the host machine, so all ETL code and input/output files remain accessible in the local development environment.

6.6 Folder Structure

The project is organized as follows:

```
hotel_etl/
├── dags/              → Airflow DAG (etl_star_schema.py)
├── scripts/           → Python ETL logic (etl_functions.py)
├── csv_exports/         → Input RDBMS tables (CSV)
├── output_data/         → Star schema output tables (CSV)
├── docker-compose.yml     → Docker service definitions
```

This structure keeps the pipeline modular, debuggable, and easily portable to new machines or cloud environments.

6.7 Benefits of the Architecture

- **Automation**: The entire ETL flow can be triggered with a single DAG run
- **Transparency**: Logs and retries are managed by Airflow
- **Scalability**: More tables or logic can be added without breaking existing flow
- **Separation of Concerns**: Data logic and orchestration are cleanly separated
- **Reusability**: Functions and pipeline structure can be reused for other business domains

7. BI Analysis & Star Schema Queries

7.1 Purpose of BI Analysis

The primary objective of transforming the transactional data into a star schema was to make analysis faster, simpler, and more accessible. While the normalized schema was ideal for day-to-day operations, it was inefficient for analytics due to its complex joins and scattered data. By converting the data into a denormalized star schema, we enabled easy exploration of business metrics, trends, and customer behavior using business intelligence tools.

7.2 Star Schema as an Analytical Model

Our final analytical model consists of a central fact table, fact_reservation, connected to descriptive dimension tables such as dim_customer, dim_room, dim_date, dim_payment, and dim_booking_channel. This structure allows for intuitive filtering, slicing, and metric aggregation, enabling business users to derive insights without deep technical knowledge.

7.3 Business Questions Answered by the Star Schema

The transformed schema supports quick and efficient answers to common business questions, including:

- Which cities generate the most booking revenue?
- What is the average stay duration by booking channel?
- How much additional income is earned through services?
- Which room types receive the highest customer ratings?
- Are weekends more active in terms of reservations?

These questions are critical for strategic decision-making in areas such as pricing, customer engagement, service design, and operational optimization.

7.4 Sample SQL Queries on the Star Schema

Q1: Which booking channel has the highest average revenue per reservation in each city?

This query helps identify which sales or booking channels (e.g., online, walk-in, travel agent) are the most profitable across different cities.

```
SELECT
   c.city,
   b.booking_channel,
   ROUND(AVG(f.total_amount_paid), 2) AS avg_revenue_per_reservation

FROM fact_reservation f
JOIN dim_customer c ON f.customer_id = c.customer_id
JOIN dim_booking_channel b ON f.booking_channel_id = b.booking_channel_id
GROUP BY c.city, b.booking_channel
ORDER BY c.city, avg_revenue_per_reservation DESC;
```

Q2: Which month had the highest average service usage per reservation, and how does it compare to service revenue?

This analysis tracks seasonal behavior in service consumption, helping hotel managers anticipate demand fluctuations.

```
SELECT
   d.month,
   ROUND(AVG(f.total_services_used), 2) AS avg_services_used,
   ROUND(AVG(f.total_service_amount), 2) AS avg_service_revenue
FROM fact_reservation f
JOIN dim_date d ON f.date_key = d.date_key
GROUP BY d.month
ORDER BY d.month;
```

Q3: What is the revenue contribution of each room type across different cities?

This question breaks down how room types perform geographically — useful for regional pricing or promotional strategy.

```
SELECT
   r.room_type,
   c.city,
   SUM(f.total_amount_paid) AS total_revenue
FROM fact_reservation f
JOIN dim_room r ON f.room_id = r.room_id
JOIN dim_customer c ON f.customer_id = c.customer_id
GROUP BY r.room_type, c.city
ORDER BY r.room_type, total_revenue DESC;
```

These types of questions demonstrate how the star schema enables layered analysis that would otherwise be difficult or time-consuming with a normalized schema. The design simplifies multi-dimensional queries and supports better data-driven decisions.

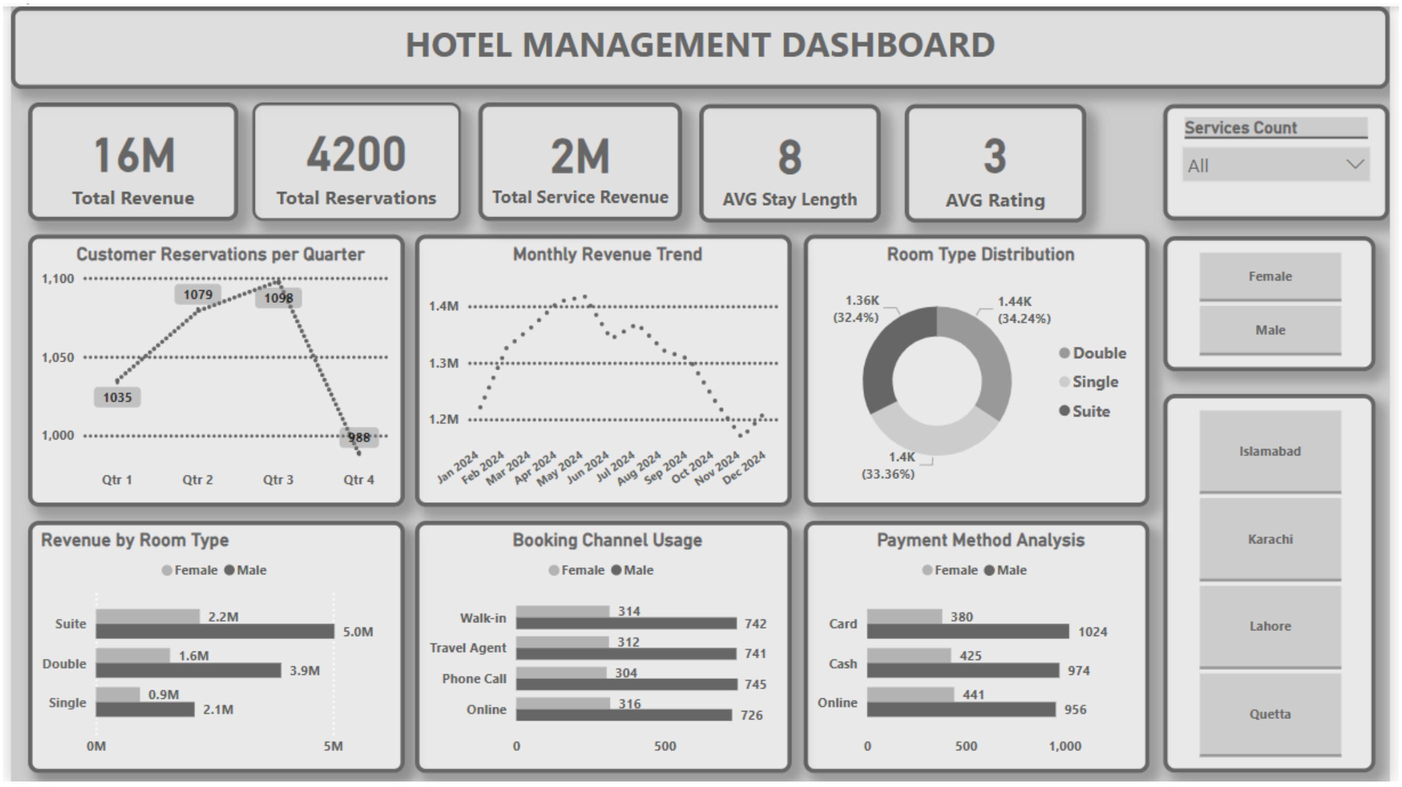
7.5 Use of BI Tools (Power BI)

The star schema was loaded into Power BI to build an interactive dashboard for business analysis. Power BI was able to automatically detect relationships between the fact and dimension tables using foreign keys. This enabled smooth development of dashboards with filters, charts, and slicers — all powered by the structured data model.

The use of Power BI provided a user-friendly interface for interacting with the data, running ad hoc analysis, and visualizing trends across time, cities, booking types, room categories, and more.

7.6 Power BI Dashboard Snapshot & Insights

Below is a snapshot of the final Power BI dashboard created using the star schema:



Beneath the dashboard, we summarize the key insights observed during analysis:

Overall Business Performance

- The hotel generated 16M in total revenue, indicating strong performance across bookings and services.
- 4,200 reservations were recorded during the analysis period (Jan–Dec 2024).
- Service revenue (spa, meals, etc.) totaled 2M, accounting for approximately 12.5% of total revenue, highlighting successful upselling.

Reservation & Revenue Trends

- The highest number of reservations occurred in Q3 (1098 bookings), closely followed by Q2 (1079).
- Q4 saw a significant decline (988 bookings), possibly due to seasonal slowdowns or off-peak demand.
- The monthly revenue trend peaked between May and July, followed by a gradual decline toward the end of the year.

Room Type Analysis

- Suites generated the highest revenue (5M), followed by Double (3.9M) and Single rooms (2.1M).

- Room bookings are evenly distributed across types, each making up roughly one-third of total reservations.
- This suggests that premium offerings like Suites are both popular and highly profitable.

Customer Demographics

- Female customers represent a significant share of the customer base.
- Filter analysis shows they are slightly more active in premium room bookings and service usage.

Booking Channel Insights

- Online and Phone Call bookings are the most popular, each with over 740 reservations.
- Walk-in and Travel Agent bookings remain steady but lower.
- This points to a clear customer preference for digital and remote booking channels.

Payment Method Insights

- Card payments (1024 transactions) are the most used, followed by Online (956) and Cash (974).
- This reflects a tech-savvy, digital-first customer base comfortable with non-cash options.

Customer Ratings & Stay Length

- The average customer rating is 3/5, indicating room for improvement in customer experience or service delivery.
- The average stay length is 8 days, which is relatively long and suggests many bookings are for vacation or extended business stays.

8. Conclusion & Reflection

This project demonstrates a complete data engineering and analytics workflow starting from schema design and data generation to ETL orchestration and dashboard visualization.

We designed a transactional hotel management schema in SQLite and generated synthetic but realistic data. After validating and exporting the data, we constructed a star schema with one fact table and five dimensions. A custom ETL pipeline was implemented using Python and Apache Airflow to automate the transformation process, and the entire pipeline was containerized using Docker for portability and consistency.

The final star schema enabled fast querying and meaningful analysis through both SQL and Power BI. The dashboard provided actionable insights into customer behavior, service usage, and revenue trends all of which would have been difficult to achieve with the original RDBMS structure.

Reflection

Through this project, we gained hands-on experience in:

- Database normalization and schema design
- Data generation and quality control
- Star schema modeling for analytical systems
- Python-based data transformation
- Airflow DAG orchestration
- Docker-based environment management
- Dashboard development and storytelling using Power BI

This end-to-end experience not only strengthened our technical skills but also deepened our understanding of how raw data can be turned into decision-supporting insights through proper data engineering and analytics practices.