End-to-End Hospital Appointment & Billing Analysis

Python → SQL → Power BI

Name: Harsh Patwa
Tools: Python, PostgreSQL (SQL), Power BI
Dataset: Hospital Appointments, Billing, Patients, Doctors, Treatments

1. Project Overview

This project demonstrates a complete **end-to-end data analytics workflow**, starting from raw hospital operational data to interactive dashboard creation.

The objective of this project is to:

- Clean and prepare raw hospital data using Python
- Store and analyze structured data using SQL (PostgreSQL)
- Answer real-world healthcare business questions
- Build an interactive Power BI dashboard for decision-making

This project closely reflects **real industry-level healthcare data analysis** performed by data analysts.

2. Dataset Description

The project uses multiple related datasets representing hospital operations:

- **Appointments:** appointment date, time, doctor, reason for visit, status
- **Billing:** bill amount, payment status, payment method, billing date
- **Patients:** patient demographics and identifiers
- **Doctors:** doctor information and specialties

- **Treatments:** treatment type and appointment mapping

Total records: ~200 appointments and billing transactions.

The project uses multiple related datasets representing hospital operations:

A	B	C	D	E	F	G
appointment_id	patient_id	doctor_id	appointment_date	appointment_time	reason_for_visit	status
A001	P034	D009	09-08-2023	15:15:00	Therapy	Scheduled
A002	P032	D004	09-06-2023	14:30:00	Therapy	No-show
A003	P048	D004	28-06-2023	08:00:00	Consultation	Cancelled
A004	P025	D006	01-09-2023	09:15:00	Consultation	Cancelled
A005	P040	D003	06-07-2023	12:45:00	Emergency	No-show
A006	P045	D006	19-06-2023	16:15:00	Checkup	Scheduled
A007	P001	D007	09-04-2023	10:30:00	Consultation	Scheduled
A008	P016	D010	24-05-2023	08:45:00	Consultation	Cancelled
A009	P039	D010	05-03-2023	13:45:00	Follow-up	Scheduled
A010	P005	D003	13-01-2023	15:30:00	Therapy	Completed
A011	P022	D007	12-11-2023	16:00:00	Checkup	No-show
A012	P029	D003	07-05-2023	10:00:00	Follow-up	Completed
A013	P003	D002	16-08-2023	12:00:00	Emergency	Scheduled
A014	P012	D010	25-05-2023	10:30:00	Emergency	Cancelled
A015	P026	D004	15-01-2023	17:15:00	Consultation	No-show
A016	P016	D008	30-06-2023	11:00:00	Consultation	Scheduled
A017	P037	D009	11-07-2023	17:00:00	Emergency	Scheduled
A018	P022	D007	14-11-2023	09:45:00	Consultation	Cancelled
A019	P029	D001	06-02-2023	15:30:00	Checkup	Cancelled
A020	P014	D003	05-12-2023	15:15:00	Consultation	Completed
A021	P028	D009	24-04-2023	10:00:00	Therapy	No-show
A022	P005	D001	14-11-2023	13:00:00	Consultation	No-show
A023	P047	D009	09-05-2023	14:30:00	Follow-up	Cancelled

Figure 1: Raw hospital datasets used for analysis

3. Data Cleaning & Preparation (Python)

Purpose

Python was used to inspect, clean, and validate raw hospital datasets before loading them into the SQL database.

Steps Performed

- Loaded all datasets using Pandas
- Checked for missing values and duplicates
- Standardized categorical columns (payment status, treatment type)
- Converted date and time columns to proper formats
- Created derived fields such as:
    - Appointment month
    - Billing month
- Exported cleaned datasets for SQL analysis

Sample Code

```
import pandas as pd

appointments = pd.read_csv("appointments.csv")
appointments['appointment_date'] =
pd.to_datetime(appointments['appointment_date'])

appointments['appointment_month'] =
appointments['appointment_date'].dt.to_period('M').astype(str)

appointments.to_csv("appointments_cleaned.csv", index=False)
```

[1]:
```
import pandas as pd
```

[88]:
```
appointment=pd.read_csv(
    r"C:\Users\harsh\OneDrive\Desktop\Hospital appointment analysis\appointments.csv")
billing=pd.read_csv(r"C:\Users\harsh\OneDrive\Desktop\Hospital appointment analysis\billing.csv")
treatment=pd.read_csv(r"C:\Users\harsh\OneDrive\Desktop\Hospital appointment analysis\treatments.csv")
doctor=pd.read_csv(r"C:\Users\harsh\OneDrive\Desktop\Hospital appointment analysis\doctors.csv")
patient=pd.read_csv(r"C:\Users\harsh\OneDrive\Desktop\Hospital appointment analysis\patients.csv")
```

[89]:
```
appointment.head()
```

[89]:

	appointment_id	patient_id	doctor_id	appointment_date	appointment_time	reason_for_visit	status
0	A001	P034	D009	2023-08-09	15:15:00	Therapy	Scheduled
1	A002	P032	D004	2023-06-09	14:30:00	Therapy	No-show
2	A003	P048	D004	2023-06-28	8:00:00	Consultation	Cancelled
3	A004	P025	D006	2023-09-01	9:15:00	Consultation	Cancelled
4	A005	P040	D003	2023-07-06	12:45:00	Emergency	No-show

[90]:
```
billing.head()
```

[90]:

	bill_id	patient_id	treatment_id	bill_date	amount	payment_method	payment_status
0	B001	P034	T001	2023-08-09	3941.97	Insurance	Pending
1	B002	P032	T002	2023-06-09	4158.44	Insurance	Paid
2	B003	P048	T003	2023-06-28	3731.55	Insurance	Paid
3	B004	P025	T004	2023-09-01	4799.86	Insurance	Failed

```
appointment['appointment_date'] = pd.to_datetime(appointment['appointment_date'])
treatment['treatment_date'] = pd.to_datetime(treatment['treatment_date'])


billing['bill_date'] = pd.to_datetime(billing['bill_date'])


patient['date_of_birth'] = pd.to_datetime(patient['date_of_birth'])
patient['registration_date'] = pd.to_datetime(patient['registration_date'])
```

```
appointment.dtypes
```

```
appointment_id              object
patient_id                  object
doctor_id                   object
appointment_date    datetime64[ns]
appointment_time            object
reason_for_visit            object
status                      object
dtype: object
```

```
appointment['appointment_month'] = appointment['appointment_date'].dt.to_period('M')
```

```
appointment[['appointment_date', 'appointment_month']].head()
```

	appointment_date	appointment_month
0	2023-08-09	2023-08
1	2023-06-09	2023-06

Figure 2: Data cleaning and feature engineering using Python (Pandas)

4. Data Storage & Modeling (SQL)

The cleaned datasets were loaded into a **PostgreSQL database** with properly defined relationships:

Key Relationships

- Patients → Appointments (1-to-Many)
- Doctors → Appointments (1-to-Many)
- Appointments → Treatments (1-to-Many)
- Treatments → Billing (1-to-Many)

- Patients → Billing (1-to-Many)

SQL was used to:

- Perform aggregations
- Analyze appointment trends
- Track revenue and payment behavior
- Support Power BI reporting

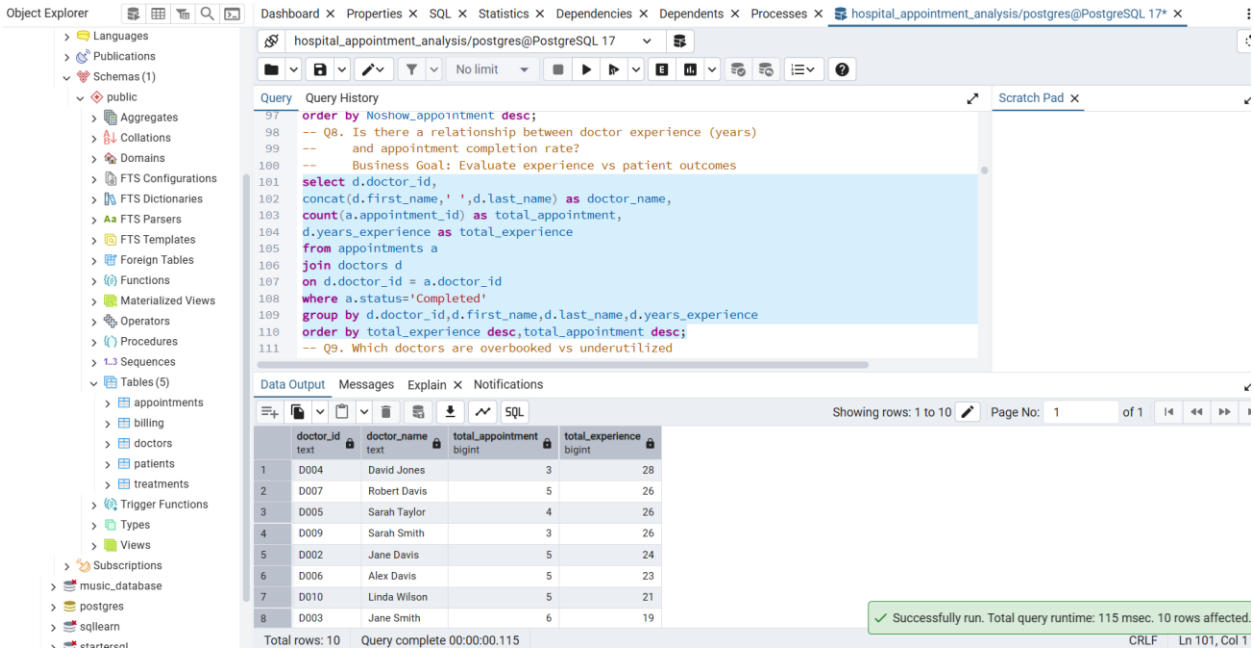


Figure 4: PostgreSQL database schema and table relationships

5. Business Questions & SQL Analysis

This section answers key business questions using SQL to evaluate **operational efficiency, patient behavior, scheduling effectiveness, and doctor performance** in the hospital.

Q1. What percentage of total appointments result in No-Shows, Cancellations, Completions, and Scheduled status?

Business Goal:

Measure operational efficiency and patient reliability.

SQL Query:

```
SELECT
    status,
    COUNT(*) AS total_appointment,
    ROUND(
        COUNT(*) * 100 / (SELECT COUNT(*) FROM appointments),
        2
    ) AS percentage
FROM appointments
GROUP BY status
ORDER BY percentage DESC;
```

Insight:

- Completed and Scheduled appointments make up the majority, indicating generally good patient engagement.
- However, a significant percentage of **No-Shows and Cancellations** highlights operational inefficiencies and potential revenue leakage.
- These findings suggest the need for better confirmation mechanisms and reminder systems.

Q2. Which appointment time slots (hour-wise) have the highest no-show rates?

Business Goal:
Redesign scheduling strategies to reduce no-shows.

SQL Query:

```
SELECT
    appointment_hour,
    COUNT(appointment_id) AS total_appointments,
    COUNT(appointment_id) FILTER (WHERE status = 'No-show') AS
no_show_count,
    ROUND(
        COUNT(appointment_id) FILTER (WHERE status = 'No-show') *
```

```
100.0 / COUNT(appointment_id),
        2
    ) AS no_show_percentage
FROM appointments
GROUP BY appointment_hour
ORDER BY no_show_percentage DESC;
```

Insight:

- Certain time slots show **disproportionately high no-show percentages**.
- Early morning or late-day hours tend to have higher patient drop-offs.
- The hospital can optimize scheduling by:
    - Overbooking high-risk time slots
    - Introducing stricter confirmation rules for these hours

Q3. Which days of the week experience the highest appointment cancellations?

Business Goal:
Improve staffing plans and slot allocation.

SQL Query:

```
SELECT
    TO_CHAR(appointment_date, 'day') AS day_of_week,
    COUNT(*) AS total_appointment,
    COUNT(*) FILTER (WHERE status = 'Cancelled') AS
cancelled_appointment,
    ROUND(
        COUNT(*) FILTER (WHERE status = 'Cancelled') * 100 / COUNT(*),
        2
    ) AS cancelled_percentage
FROM appointments
GROUP BY day_of_week
ORDER BY cancelled_percentage DESC;
```

Insight:

- Specific weekdays experience **higher cancellation rates**.
- This indicates demand variability across the week.
- Staffing levels and appointment capacity should be adjusted based on day-wise cancellation behavior.

Q4. Which appointment reasons are most likely to result in no-shows?

Business Goal:
Refine booking and confirmation policies.

SQL Query:

```
SELECT
    reason_for_visit,
    COUNT(*) AS total_appointment,
    COUNT(*) FILTER (WHERE LOWER(TRIM(status)) = 'no-show') AS
not_shown
FROM appointments
GROUP BY reason_for_visit
ORDER BY not_shown DESC;
```

Insight:

- Certain visit types (such as routine consultations or follow-ups) are more prone to no-shows.
- High-priority visits like emergencies show better attendance.
- This insight supports differentiated reminder strategies based on visit type.

Q5. What is the average number of appointments per day, and how does it fluctuate month-over-month?

Business Goal:
Demand forecasting and capacity planning.

SQL Queries:

Average appointments per day:

```
SELECT AVG(daily_appointment) AS avg_appointment_per_day
FROM (
    SELECT
        appointment_date,
        COUNT(*) AS daily_appointment
    FROM appointments
    GROUP BY appointment_date
) AS daily_count;
```

Monthly appointment trend:

```
SELECT
    DATE_TRUNC('month', appointment_date) AS months,
    COUNT(*) AS total_appointment
FROM appointments
GROUP BY months
ORDER BY months;
```

Insight:

- The hospital handles a stable average number of daily appointments.
- Month-over-month analysis reveals fluctuations, indicating seasonal or behavioral trends.
- These insights help in proactive capacity planning and resource allocation.

Doctor Performance & Capacity Utilization

Q6. Which doctors have the highest number of completed appointments?

Business Goal:
Identify top-performing doctors and workload distribution.

SQL Query:

```
SELECT
    d.doctor_id,
    CONCAT(d.first_name, ' ', d.last_name) AS doctor_name,
    COUNT(a.appointment_id) AS completed_appointment
FROM appointments a
JOIN doctors d
    ON a.doctor_id = d.doctor_id
WHERE a.status = 'Completed'
GROUP BY d.doctor_id, d.first_name, d.last_name
ORDER BY completed_appointment DESC;
```

Insight:

- A small group of doctors handle a disproportionately high number of completed appointments.
- These doctors demonstrate high efficiency and patient trust.
- The hospital can use this insight for:
    - Performance recognition
    - Balanced workload distribution
    - Best-practice replication among other doctors

6. Dashboard Development (Power BI)

An **interactive Power BI dashboard** was created to visualize hospital performance metrics.

Key Performance Indicators (KPIs)

- **Total Revenue:** ₹173.42K
- **Total Appointments:** 200
- **Paid Revenue:** ₹173.42K
- **Average Bill Amount:** ₹2.76K
- **Total Patients:** 50

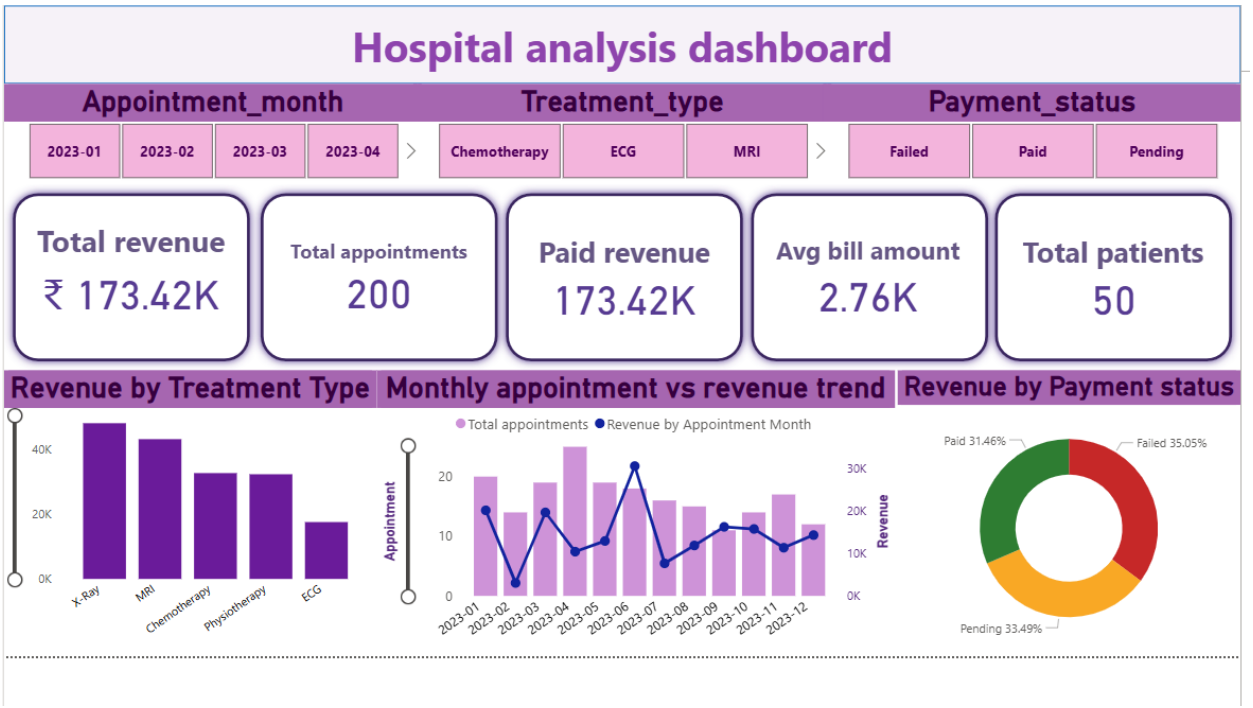
Visualizations

- **Revenue by Treatment Type:** Identifies high-earning services
- **Monthly Appointment vs Revenue Trend:** Shows operational vs financial trends
- **Revenue by Payment Status:** Highlights payment distribution and risks

Interactive Filters (Slicers)

- **Appointment Month:** Time-based analysis
- **Treatment Type:** Service-level analysis
- **Payment Status:** Financial performance filtering

The dashboard allows **dynamic filtering and intuitive storytelling** for stakeholders.



7. Key Insights Summary

- Diagnostic treatments such as **X-Ray and MRI generate the highest revenue**

- Appointment volume varies month-to-month, but **revenue depends heavily on treatment type**
- **Paid transactions dominate revenue**, while pending and failed payments indicate areas for improvement in collections
- Average billing value suggests a mix of both routine and high-value treatments
- Monthly trends reveal **operational load vs financial return differences**

8. Conclusion

This project successfully demonstrates a **complete data analytics lifecycle**:

- Data preparation using **Python**
- Business analysis using **SQL**
- Insight visualization using **Power BI**

It reflects **real-world healthcare analytics**, focusing on operational efficiency and revenue performance.

9. Future Enhancements

- Add **doctor-level performance analysis**
- Include **patient demographics and age-based insights**
- Build **appointment no-show prediction models**
- Automate data refresh using Power BI Service
- Add drill-through pages for detailed treatment and billing analysis