Overview & Objectives

The purpose of this project is to create and set up a Healthcare Patient Outcome Data Warehouse that will bring together and examine the medical data of hospitals covering patients, diagnoses, doctors, and departments. This will be a significant step for healthcare administrators and analysts as they will be able to see and monitor patient admissions, treatment costs, and outcomes, identify the most common diagnoses and also make decisions regarding the allocation of resources and disease management strategies.

The system takes the raw data from various hospital systems concerning patients, diagnoses and treatments and merges it into a single star schema thereby making the querying and visualization of the key performance indicators easier. The project also involves the complete ETL (Extract, Transform, Load) process using SQL and Python tools.

Schema Design Explanation

The data warehouse utilizes a Star Schema, which gives prominence to the main fact table (fact_admissions) in the middle capturing the metrics of trading events of the business like hospital admissions together with their cost of the attached treatment. Numerous dimension tables storing the descriptive attributes such as patient demographics, diagnosis information, doctor specialization, department, and time-based data surround the central table. This arrangement provides an opportunity to conduct a flexible, multi-dimensional analysis of the hospital data and to perform OLAP-style analytics such as averaging the treatment cost per department, counting admissions per diagnosis category, analyzing revenue trends by month or quarter, and examining patient demographics by disease type. The Star Schema design helps in breaking down complex queries by reducing joins, thus increasing the query performance, and at the same time, giving a clear and intuitive framework for analytical reporting and decision-making.

Table 1: Dimension Fact Tables

Table	Column	Data Type	Description
dim_patient	patient_id	INT (PK)	Surrogate key (same as OLTP).

	first_name	VARCHAR(255)	Patient first name.
	last_name	VARCHAR(255)	Patient last name.
	gender	ENUM('Male','Female','Other')	Gender.
	age	INT	Age.
	age_group	VARCHAR(20)	Derived attribute categorizing patients: Child, Adult, or Senior.
dim_diagnosis	diagnosis_id	INT (PK)	Surrogate key (same as OLTP).
	diagnosis_code	VARCHAR(50)	Diagnosis code.
	diagnosis_desc	VARCHAR(255)	Text description.
	category	VARCHAR(100)	Derived classification (e.g., Cardiology, Respiratory, General).
dim_doctor	doctor_id	INT (PK, AUTO_INCREMENT)	Unique doctor key.
	doctor_name	VARCHAR(255)	Full name of doctor.

	specialization	VARCHAR(100)	Medical department or field.
dim_department	department_id	INT (PK, AUTO_INCREMENT)	Unique department key.
	department_name	VARCHAR(100)	Name of hospital department.
	location	VARCHAR(100)	Location or floor (optional field, can be null).
dim_date	date_id	VARCHAR(8) (PK)	Surrogate date key formatted as YYYYMMDD.
	full_date	DATE	Calendar date.
	day	INT	Day of month.
	month	INT	Month number (1–12).
	month_name	VARCHAR(20)	Full month name (e.g., January).
	quarter	VARCHAR(2)	Quarter of the year (1–4).

	year	INT	Calendar year.
	weekday	VARCHAR(10)	Name of weekday (e.g., Monday).
fact_admissions	admission_id	INT (PK)	Unique admission record (from OLTP).
	patient_id	INT (FK → dim_patient.patient_id)	Links to patient dimension.
	doctor_id	INT (FK → dim_doctor.doctor_id)	Links to doctor dimension.
	diagnosis_id	INT (FK → dim_diagnosis.diagnosis_id)	Links to diagnosis dimension.
	department_id	INT (FK → dim_department.department_id)	Links to department dimension.
	date_id	VARCHAR(8) (FK → dim_date.date_id)	Links to date dimension.

	treatment_cost	DECIMAL(10,2)	Measure field for analysis (e.g., total revenue, cost per admission).

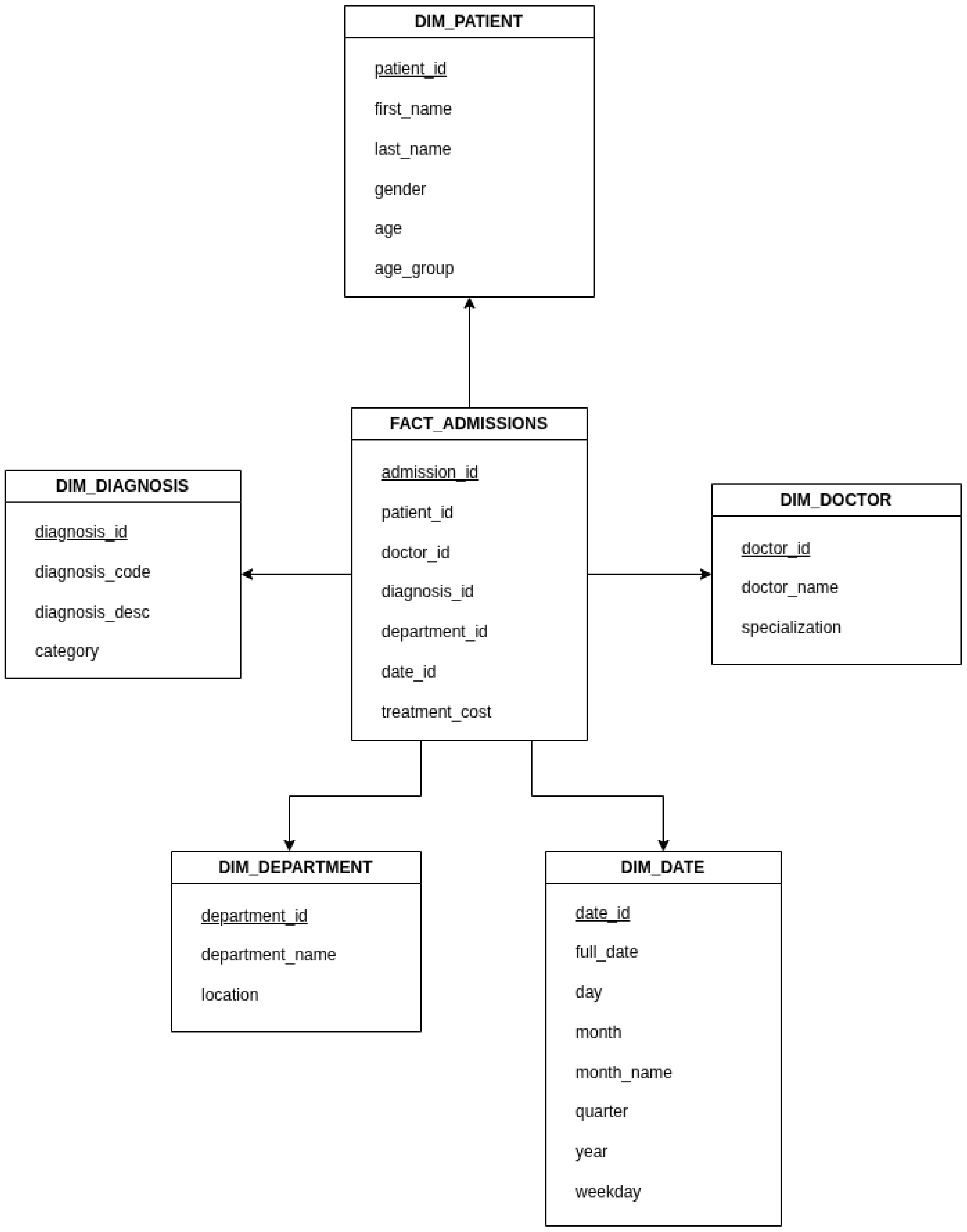

Figure 1: Entity Relationship Diagram

ETL Process Steps

Healthcare data warehouse's ETL (Extract, Transform, Load) process is a structured and filtered workflow that guarantees the final system's availability of clean, consistent, and analyzable data. It starts with an extraction phase wherein three main CSV source files, diagnoses.csv, patients.csv, and treatments.csv, are loaded into Python through the pandas library. These files are representations of the hospital's operational (OLTP) systems, which contain diagnostic codes and descriptions, patient demographic information, and treatment details such as doctors, departments, and costs. Each dataset is being guided into a pandas DataFrame, which is a flexible and efficient in-memory structure provided for subsequent cleansing and transformation of operations.

The transformation phase sees the data being subjected to extensive cleansing, standardization, and enrichment so that it meets the data warehouse quality and reliability criteria before being loaded there. The processing of the first dataset, diagnoses.csv, comprises deleting duplicate records, and in the diagnosis_code and description columns the missing values are filled with the most frequent (mode) value. Invalid placeholder entries like "XXX" are being removed, while descriptions are being unified through having spaces and mere letters properlyCapitalized, and "Unk" being changed to "Unknown." A vital part of the transformation process is the imposition of a one-to-one relationship between diagnosis descriptions and codes, which is done by recognizing and correcting the issues in the duplicates of code assignments, creating new synthetic codes if necessary to keep data integrity. Subsequently, data types are standardized, and the dataset is exported as diagnoses_cleaned.csv.

The patients.csv dataset undergoes a series of transformations that are similar yet specific to the domain. The first names that are missing are replaced with "Unknown," and the last names are first cleaned from any undesired symbols such as the "#" at the end and then they are all changed to Title Case so that they are uniform. The age column is checked for its correctness by taking away the values that are not realistic (for instance, less than 0 or more than 120) and replacing the missing ones with the mean age of the dataset. The gender column is made uniform by changing all the entries to uppercase and using full names for abbreviations like "M" and "F" (i.e. "Male" and "Female"). After removing duplicate patient records using their unique patient_id, the data cleaned and made uniform is saved in patients_cleaned.csv.

The treatments.csv dataset is intricate since it is linking patients, diagnoses, and doctors through hospital admissions. The transformation starts with deduplication and then goes on to deal with the values that are missing, filling up the absent doctor_name in the data with the most common one and replacing the treatment_cost that is not accounted for with the median so as not to affect the outliers too much. The entries that are not sure like "???" and "UnknownDept" are all converted into "Unknown Doctor" and "Unknown Department" that are easier to understand, respectively. Doctor's name and department which are text columns are rid of the extra spaces and converted to Title Case. Dates are converted into valid datetime objects, thus, time-based analysis will be done consistently. The validation of treatment costs is done in a very strict manner, as the negative values will be nullified and re-imputed, and the very high outlier above ₱200,000 will be capped to maintain the realistic range. Once the data is ready, it is saved in treatments_cleaned.csv.

During the loading phase, the MySQL-based data warehouse receives the cleaned datasets through an automated Python script (pipeline.ipynb). The script is a crucial part of the database operations and guarantees that the loading process is monitored and versatile at the same time. First, it checks whether the SQL schema file (script.sql) is present, then it connects to the MySQL database (healthcare_dw) with the provision of local file imports. The SQL file, which outlines the warehouse schema along with data loading commands, is read and divided into separate statements. Sequentially, one at a time, the statement for table creation, data loading, or foreign key constraint definition, are executed in a single transaction. There is logging done continuously to show the progress, give the executed statements, and mention any errors during the process. If things go wrong, the transaction is reversed to keep the database consistent; meanwhile, the successful ones are committed and recorded.

The SQL script actually does the work of physical data loading and modeling. It starts by turning off foreign key checks and getting rid of existing tables to make sure there's a clean start. After that, staging OLTP tables, diagnoses, patients, and treatments, are created and the LOAD DATA LOCAL INFILE command is used to directly import the cleaned CSV files into MySQL. The script then, from the mere existence of data and its descriptive attributes creates the dimension tables dim_patient, dim_diagnosis, dim_doctor, dim_department, and dim_date, and here it is that the analysts get their descriptive attributes for data analysis. A central fact table, fact_admissions, is prepared to record the treatment costs together with other measurable business events and to relate all dimension tables through foreign keys.

The entire ETL pipeline where Python is used for extraction and transformation and MySQL for loading data into structured format, guarantees data to be accurate, consistent and ready for analytics. The design is deliberate: Python takes care of the flexible and heavy data cleaning operations, while MySQL maintains the integrity of the schema and allows for multi-dimensional analysis. The process is meticulously planned for production and be easily maintained because there are strong logging, error handling, and data validation in place. At the end of this workflow, the raw, unstructured hospital data gets converted into a clean, integrated and query-optimized data warehouse that not only supports data-driven decision-making, reporting, and future analytical applications but also the entire process of turning data into information.

Key Insights from Reports

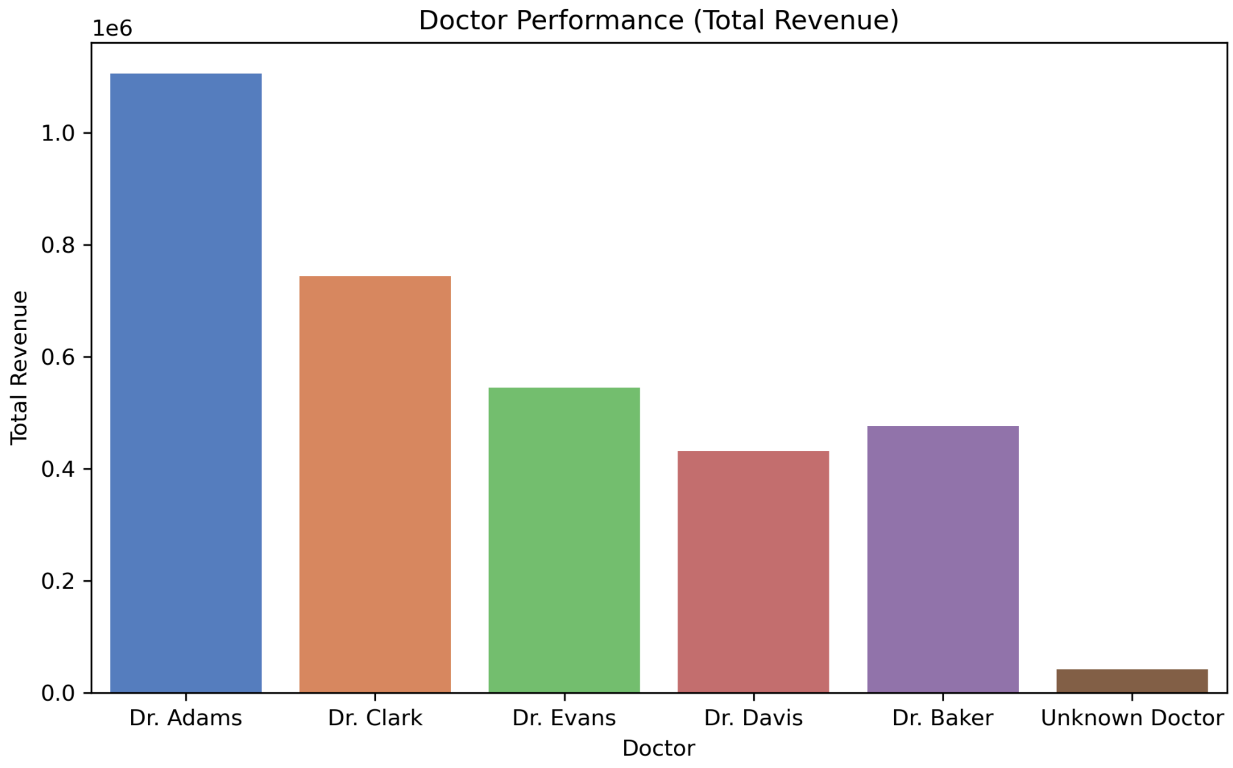

Figure 2: Doctor Performance

The bar chart shows very clearly a hierarchy in medical staff total revenue, in which Dr. Adams is the best performer with a total of about 1.1 million. This result is more than double the next closest teammate, Dr. Clark, who was responsible for approximately 750,000. Next to Dr. Clark, there is a central group of doctors that includes Dr. Evans (about 550,000), Dr. Baker (about 480,000), and Dr. Davis (about 430,000), who all have quite similar performances. The graph points out a very important issue: the "Unknown Doctor" category. It is a small amount of money (about 40,000), but its existence signals a problem with data quality or attribution in the tracking system. Hence, it can be inferred that in-depth studies will have to be conducted not only to understand what lies behind the top revenue of Dr. Adams but also to rectify the system flaws causing the revenue to be unattributed.

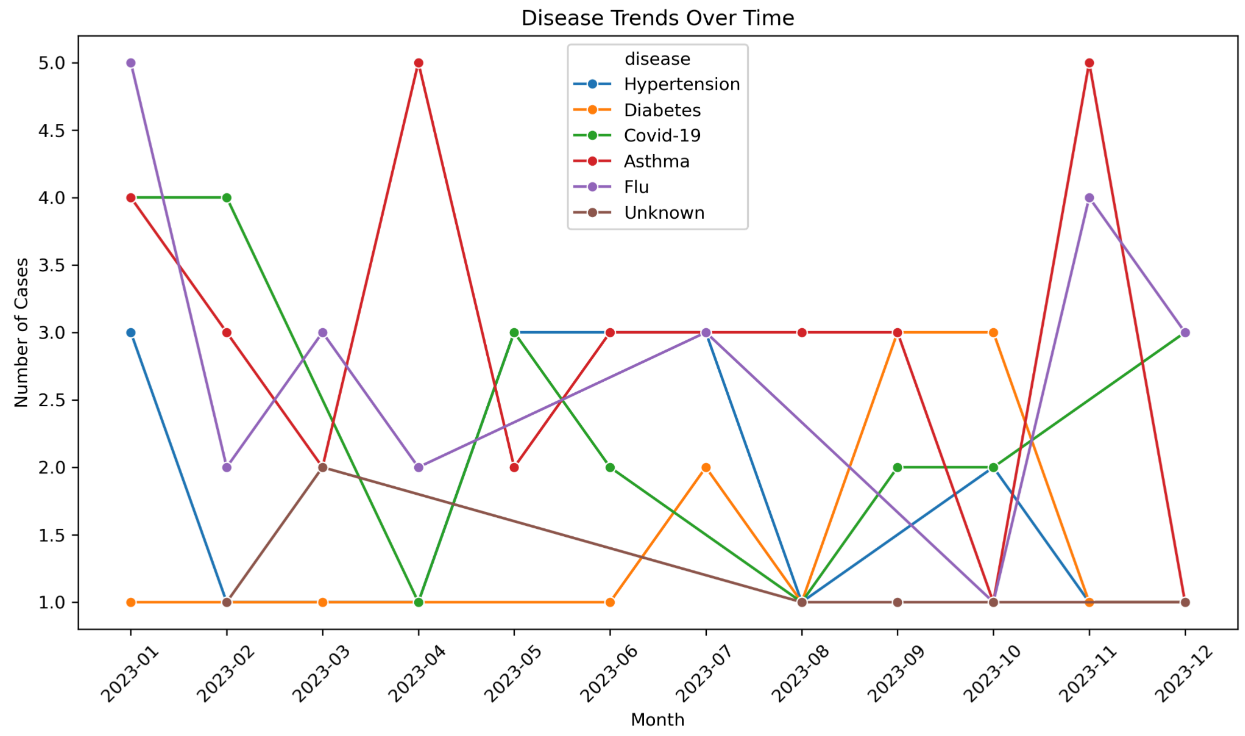

Figure 3: Disease Trend

This line graph illustrates the monthly case numbers of six different diseases during the year 2023. The data reveals a lot of ups and downs, especially in acute, seasonal diseases, and the most striking ones are when the red line of Asthma reached 5 cases in April and November, and purple line of Flu had its own peaks of 5 cases in January and 4 in November. The strong seasonality in spring and fall/winter has been established as a major finding. On the other hand, the chronic conditions like Diabetes (orange line) are stable and about 1 case in most of the year. In August 2023, a very unusual pattern happened, where almost all diseases tracked, including Hypertension, Covid-19, Asthma, and Flu, at the same time, went down to their lowest levels, which were mostly 1 case. This overall dip deserves further investigation as it might indicate a data reporting error, a clinic holiday, or a real lull. And lastly, there is a good sign concerning the data quality: “Unknown” category (brown line) began with 4 cases in January and gradually decreased to 1 at the end of June, meaning that the diagnostic or case attribution processes got a lot better as the year went by.

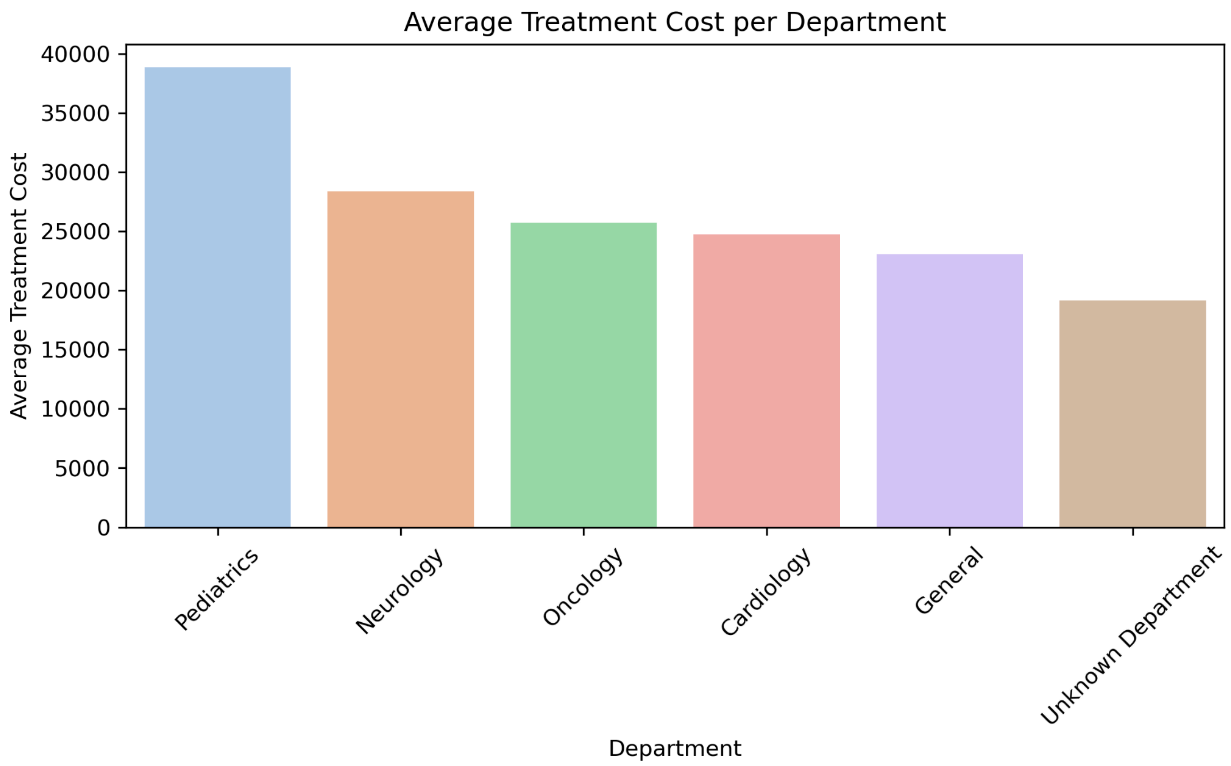

Figure 4: Average Treatment Cost per Department

The graph distinctly reveals Pediatrics to be the most notable high-cost outlier, with an average treatment cost of about 39,000. This is remarkably higher, more than 10,000 additional, compared to the second most expensive department, Neurology (approx. 28,500). After Neurology, there is a close grouping of costs for several other departments such as Oncology (approx. 25,500), Cardiology (approx. 24,500), and General (approx. 23,000). One of the most important conclusions from this data is the existence of an "Unknown Department" with an enormous average cost of around 19,000. This "Unknown" group is a major data integrity issue, since these unrecognized costs not only become unavailable for analysis but also distort the actual averages of all the other departments.