PROJECT DESIGN

GROUP 4

ClearCare Data Initiative

Sponsor: Vijay Koneru

MIS6349.001

**INDEX**

1. Introduction ............................................................................................................ 3

2. Requirements.......................................................................................................... 3

3. Data Acquisition...................................................................................................... 4

4. Data Standardization............................................................................................... 5

5. Database Architecture............................................................................................. 6

6. Data Optimization................................................................................................... 8

7. Resilience Strategy.................................................................................................. 9

8. Validation and Performance.................................................................................... 10

9. Technical Documentation....................................................................................... 10

10. Project Outcome................................................................................................... 13

10. Growth Planning.................................................................................................... 18

**INTRODUCTION**

This project provides consumers with a clearer understanding of healthcare costs, enabling them to make more informed decisions about their care. Additionally, it offers insights into pricing discrepancies across providers and payers, promoting greater accountability and transparency within the healthcare industry. This document outlines the project’s design, including data sources, architecture, processing methods, and user interface considerations, to ensure a comprehensive and accessible price transparency tool.

**REQUIREMENTS**

**USER REQUIREMENTS - Patients/Consumers**

* **View Pricing Information**: Users should be able to view standard charges for specific healthcare services, including the cash price, minimum and maximum negotiated rates, and rates associated with various insurance providers.
* **Compare Prices**: The system should allow users to compare prices for the same service across different insurance providers.
* **Search Functionality**: Users should have the ability to search for services using common terms, keywords, or codes (e.g., CPT codes) for easy navigation.
* **Data Interpretation**: Information should be presented in a clear, user-friendly tabular format to help users understand pricing variations and implications.

### **DATA REQUIREMENTS**

1. **Service Information**
   1. **Code**: A unique identifier for each healthcare service or procedure, possibly using standard codes such as CPT.
   2. **Description**: A brief description of the healthcare service or procedure, providing more context about what each code represents.
2. **Pricing Information**
   1. **Standard Charge (Gross)**: The hospital’s list or base price for the procedure or service before any discounts or negotiated rates are applied.
   2. **Standard Charge (Negotiated Dollar)**: The rate negotiated between the hospital and each insurance provider for a specific procedure or service. This is the actual amount agreed upon that an insurer will pay.
   3. **Standard Charge Min**: The lowest negotiated rate with any insurance provider for the specified procedure or service, representing the minimum charge across all payers.
   4. **Standard Charge Max**: The highest negotiated rate with any insurance provider for the specified procedure or service, representing the maximum charge across all payers.
3. **Insurance Information**
   1. **Payer Name**: The name of the insurance company that has a negotiated rate with the hospital for the service.
   2. **Plan Name**: Specific plans under each insurance company, which may have different negotiated rates.

**SYSTEM REQUIREMENTS (Backend and Database)**

* **Data Storage and Aggregation**: The system should have the capacity to store and aggregate large volumes of data from multiple providers.
* **Data Normalization**: Mechanisms to standardize data from various sources (e.g., normalize CPT codes, standardize price fields).
* **Data API**: An API for external systems or applications to retrieve pricing data for integration with other platforms.

**DATA ACQUISITION**

We collected data from various hospitals across four U.S. states—Illinois, Texas, Wisconsin, and New Jersey—focusing on information provided under price transparency laws. The data was available in multiple formats from different hospitals in line with transparency requirements. After consulting with the UI team and the client sponsor, we identified several common attributes in these datasets. We filtered these to focus on relevant fields: HIPAA codes, procedure descriptions, standard minimum and maximum charges, gross charges, negotiated dollar amounts, standard discounted charges, and plan-level standard charges across multiple insurance providers. All datasets were individually handled in the following steps and will be merged and loaded in the database at the end of the cleaning and standardizing process.

**DATA STANDARDISATION**

For this project, we are focusing on CPT codes specifically related to surgery and surgical equipment. During our initial research phase, we sourced a master list containing over 17,000 CPT codes, which has been approved by the client as the foundation for filtering and normalization. Our hospital datasets initially included various types of codes, including CPT and non-CPT codes. However, after discussions with the client and considering project timelines, we agreed to work exclusively with CPT codes. Consequently, we filtered out all non-CPT codes from the hospital price transparency datasets, ensuring that only relevant CPT codes were retained. This operation was completed using Python and the NumPy library.

After filtering, we moved on to populating short descriptions for each CPT code. We implemented two approaches for this task, ultimately combining the results to create a comprehensive description column across all datasets:

1. **LLM Model Deployment:** We deployed a Large Language Model to source descriptions from the web, allowing us to capture relevant details for each CPT code.
2. **Web Scraping:** Using Python, Beautiful Soup, and the Hugging Face API, we scraped descriptions from a reliable website to populate our master dataset.

By merging the results from both methods, we successfully populated the description column consistently across all hospital datasets, ensuring uniformity and completeness.

In addition, substantial data cleaning was performed on the datasets sourced from hospital websites to improve data quality. This process included:

* **Column Filtering:** We identified and removed columns irrelevant to the project scope, streamlining the data for analysis.
* **Handling Missing and Null Values:** Using Python, we systematically identified missing and null values. After consulting with the client, we replaced the “Standard charge - gross” field with the “Maximum charge” for relevant entries, and "negotiated charge" was standardized to "Standard charge" across datasets.

For the Minimum Viable Product (MVP), we agreed with the client to include only a select number of insurance providers and a subset of their plans. This approach enables us to create a scalable system that can accommodate more providers and plans in future iterations.

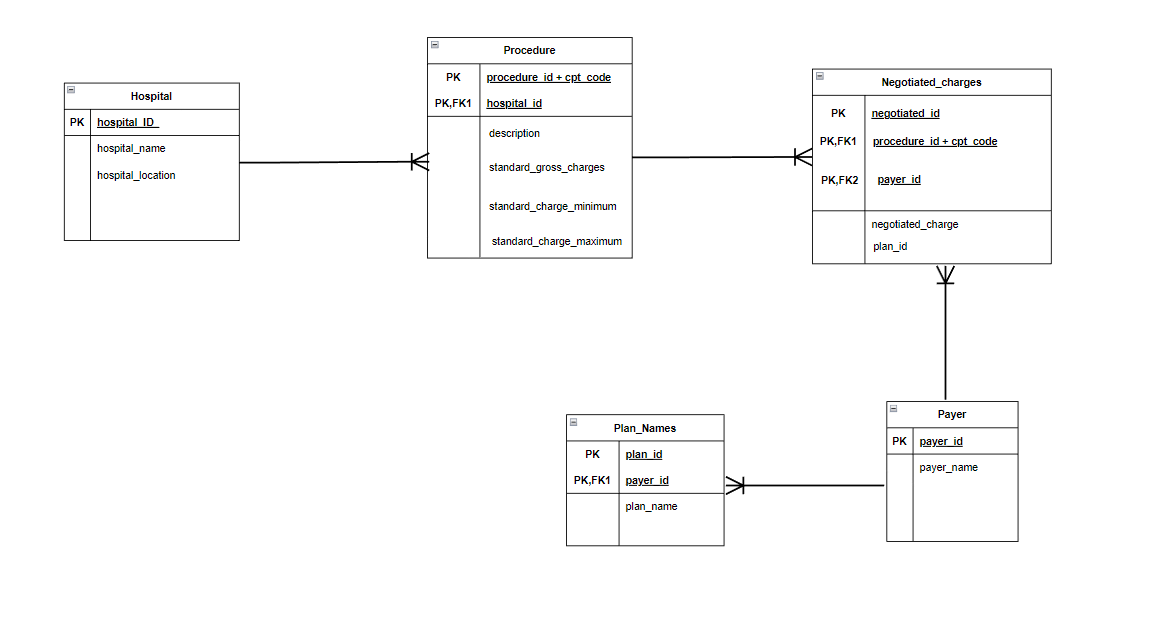
**VALIDATION AND PERFORMANCE:**

After providing the requirements for the UI Development team, we will allocate time for validation. We will do sample testing for a few HIPAA codes where we ensure proper data retrieval. We plan to verify the data for a few codes manually and if there is a mismatch, we will perform root-cause analysis to understand the reason behind the discrepancies.

We plan to coordinate with Team 2 to understand the performance of the system. For the Postgres database, we have validated the database performance by querying on few attributes and we are satisfied with the result. The database has been optimized and the technical details are mentioned in the “Data Optimization for High Performance in PostgreSQL” section of the document.

**Logical Design**

**DATABASE ARCHITECTURE**



The database architecture is designed to support an insurance transparency application that enables users to compare healthcare costs across hospitals and insurance providers. The system is structured to capture and present detailed information on hospital procedures, charges, negotiated rates with various insurance providers, and specific plans offered by these providers.

Entity-Relationship Diagram (ERD) Overview

**Hospital Table:**

Attributes: Contains details about each hospital, such as hospital\_ID (Primary Key), hospital\_name, and hospital\_location.

Purpose: Serves as the foundational entity to link procedure costs to specific hospitals.

**Procedure Table:**

Attributes: Includes procedure\_id combined with cpt\_code as a composite Primary Key, hospital\_id (Foreign Key referencing Hospital), description, standard\_gross\_charges, standard\_charge\_minimum, and standard\_charge\_maximum.

Purpose: Stores information on each medical procedure, along with standard pricing details. This entity is linked to specific hospitals and is the basis for calculating costs.

**Negotiated Charges Table:**

Attributes: Uses negotiated\_id as the Primary Key and includes composite Foreign Keys procedure\_id + cpt\_code (from Procedure) and payer\_id (from Payer). It also contains negotiated\_charge and plan\_id.

Purpose: Represents the specific charges negotiated between hospitals and different payers (insurance providers) for various procedures. This entity provides transparency into the variable costs for procedures across providers and plans.

**Payer Table:**

Attributes: Contains payer\_id (Primary Key) and payer\_name.

Purpose: Lists insurance providers offering coverage, allowing users to see which payers are associated with each hospital and procedure.

**Plan Names Table:**

Attributes: Consists of plan\_id (Primary Key), payer\_id (Foreign Key referencing Payer), and plan\_name.

Purpose: Defines specific plans offered by each payer, enabling users to view plan-level negotiated rates for procedures.

**Entity Relationships**

*Hospital and Procedure:*

One-to-Many: A hospital can perform multiple procedures, but each procedure belongs to one hospital.

Foreign Key: hospital\_id in Procedure references hospital\_ID in Hospital.

*Procedure and Negotiated Charges:*

One-to-Many: A procedure can have multiple negotiated charges with various payers.

Composite Foreign Key: procedure\_id + cpt\_code in Negotiated Charges references the same fields in Procedure.

*Payer and Plan Names:*

One-to-Many: Each payer can offer multiple plans.

Foreign Key: payer\_id in Plan Names references payer\_id in Payer.

*Payer and Negotiated Charges:*

One-to-Many: Each payer can have different negotiated charges for various procedures.

Foreign Key: payer\_id in Negotiated Charges references payer\_id in Payer.

*Plan Names and Negotiated Charges:*

One-to-Many: Each plan can have multiple negotiated charges for different procedures.

Foreign Key: plan\_id in Negotiated Charges references plan\_id in Plan Names.

**DATA OPTIMIZATION FOR HIGH PERFORMANCE IN POSTGRESQL**

To ensure high performance and minimize redundancy in the database, the following optimization strategies have been implemented:

1. Data Normalization

The database schema is normalized to 3NF to eliminate redundancy and improve data integrity:

* 1NF: All tables store atomic values with no repeating groups. For example, the Hospital table contains unique rows for each hospital, and the Procedure table links procedures to hospitals via hospital\_id.
* 2NF: All non-key attributes depend on the full primary key. For instance, in the Procedure table, description and pricing depend on the composite key (procedure\_id + cpt\_code).
* 3NF: Non-key attributes are independent of each other. The Payer table only stores payer\_id and payer\_name, avoiding redundancy in stored data.

This structure ensures efficient storage and flexibility as the system grows.

2. Indexing

Indexes are applied to improve query performance:

* Primary Key and Clustered Indexes: Each table has a primary key index, with clustered indexes on frequently queried columns:
  + Hospital: Clustered by hospital\_ID.
  + Procedure: Clustered by procedure\_id + cpt\_code.
  + Negotiated Charges: Clustered by procedure\_id + cpt\_code and payer\_id.

These indexes ensure fast data retrieval, even with large datasets.

3. Efficient Joins

For retrieving detailed cost data based on user selections (state and hospital), efficient joins are used:

* Hospital ↔ Procedure: Join on hospital\_id (indexed).
* Procedure ↔ Negotiated Charges: Join on procedure\_id + cpt\_code (clustered index).
* Negotiated Charges ↔ Payer/Plan: Join on payer\_id and plan\_id (indexed).

These optimized joins, supported by indexing, ensure fast and accurate data retrieval for comparing healthcare costs.

Through normalization, strategic indexing, and optimized joins, the database is designed for high performance and scalability, ensuring fast queries and minimal redundancy as the system grows

**RESILIENCE STRATEGY**

* Failover Solution : Implement scripts(SQL and Python) to monitor data availability and trigger failover to backup systems in real-time, if feasible.
* Role-Based Access: We will only provide limited access to certain people within team to avoid accidental deletions or unintended updates in the database
* Data Restoration: We will keep the Python script ready to load the Postgres database which was utilized to load the data in the first place. Also, we will plan to recover any unprecedented failures on the spot and with available edit access to the teammates, we will handle those adhocly.
* Training: We will have a meet-up with team 2 to onboard them with database details and attributes and how we can utilize these to pass it to UI.

**TECHNICAL DOCUMENTATION**

**Data Modeling and Schema Design**

**1. Data Dictionary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **Column Name** | **Data Type** | **Description** | **Constraints** | **Default Value** |
| **Hospital** | hospital\_id | Serial/INT | Unique identifier for each hospital | Primary Key | Auto-increment |
|  | hospital\_name | VARCHAR (255) | Name of the hospital | NOT NULL |  |
|  | hospital\_location | VARCHAR (255) | |  | | --- | |  |      |  | | --- | | Location of the hospital | |  |  |
| **Plan Names** | plan\_id | SERIAL / INT | Unique identifier for each plan | |  | | --- | |  |      |  | | --- | | Primary Key | | |  | | --- | |  |      |  | | --- | | Auto-increment | |
|  | payer\_id | INT | Unique identifier for the payer | Foreign Key |  |
|  | plan\_name | |  | | --- | |  |      |  | | --- | | VARCHAR (255) | | Name of the insurance plan | NOT NULL |  |
| **Payer** | payer\_id | SERIAL / INT | Unique identifier for the payer | Primary Key | Auto-increment |
|  | payer\_name | VARCHAR (255) | Name of the payer | NOT NULL |  |
| **Negotiated Charges** | negotiated\_id | SERIAL / INT | Unique identifier for each negotiated charge | Primary Key | Auto-increment |
|  | cpt\_code | VARCHAR (10) | CPT code for the procedure | Foreign Key |  |
|  | procedure\_id | INT | References the procedure being negotiated | Foreign Key |  |
|  | payer\_id | INT | |  | | --- | |  |      |  | | --- | | Unique identifier for the payer | | Foreign Key |  |
|  | plan\_id | INT | |  | | --- | |  | |  |  |
|  | negotiated\_charges | DECIMAL(10, 2) | Negotiated charges for the procedure |  |  |
| **Procedure** | procedure\_id | SERIAL/INT | Unique identifier for each procedure | Primary key |  |
|  | hospital\_id | INT | Unique identifier for the hospital | Foreign Key |  |
|  | cpt\_code | VARCHAR | Code identifying the procedure |  |  |
|  | description | VARChar | Description of the procedure |  |  |
|  | standard\_gross\_charge | DECIMAL (10, 2) | Standard gross charges for the procedure |  |  |
|  | standard\_charge\_minimumm | DECIMAL (10, 2) | Minimum standard charge for the procedure |  |  |
|  | Standard\_charge\_maximum | DECIMAL (10, 2) | Maximum standard charge for the procedure |  |  |

**Table Descriptions:**

**1. Hospital:**

* hospital\_id (INT, PK): Unique identifier for each hospital
* hospital\_name (VARCHAR): Name of the hospital
* hospital\_location (VARCHAR): Location of the hospital

**2. Procedure:**

* procedure\_id (INT, PK): Unique identifier for each procedure
* cpt\_code (VARCHAR): CPT code associated with the procedure
* description (TEXT): Description of the procedure
* standard\_charge (DECIMAL): Standard charge for the procedure (optional)
* hospital\_id (INT, FK): Foreign key referencing the Hospital table

**3. Negotiated\_Charges:**

* negotiated\_id (INT, PK): Unique identifier for each negotiated charge
* procedure\_id (INT, FK): Foreign key referencing the Procedure table
* payer\_id (INT, FK): Foreign key referencing the Payer table
* negotiated\_charge (DECIMAL): Negotiated charge for the procedure and payer combination
* plan\_id (INT, FK): Foreign key referencing the Plan\_Names table

**4. Payer:**

* payer\_id (INT, PK): Unique identifier for each payer
* payer\_name (VARCHAR): Name of the payer (e.g., insurance company)

**5. Plan\_Names:**

* plan\_id (INT, PK): Unique identifier for each plan
* payer\_id (INT, FK): Foreign key referencing the Payer table
* plan\_name (VARCHAR): Name of the insurance plan

**Relationships:**

* **Hospital-Procedure:** One-to-many relationship. A hospital can offer multiple procedures.
* **Procedure-Negotiated\_Charges:** One-to-many relationship. A procedure can have multiple negotiated charges with different payers and plans.
* **Payer-Plan\_Names:** One-to-many relationship. A payer can offer multiple plans.
* **Negotiated\_Charges-Payer:** Many-to-one relationship. Multiple negotiated charges can be associated with a single payer.
* **Negotiated\_Charges-Plan\_Names:** Many-to-one relationship. Multiple negotiated charges can be associated with a single plan.

### **2. Data Flow**

**ETL Process**:

* **Extract**: Collect data from hospitals, payers, and insurance plans.
* **Transform**: Clean, normalize, and format the data according to the schema.
* **Load**: Store the transformed data in the PostgreSQL database.

**Considerations:**

* **Data Quality and Validation:** Implemented data validation rules to ensure data consistency and accuracy.
* **Scalability:** Considered indexing frequently queried columns (e.g., hospital\_id, procedure\_id, payer\_id, plan\_id) to improve query performance.
* **Security:** Implemented appropriate security measures to protect sensitive patients and financial data.
* **Data Privacy:** Adhered to relevant data privacy regulations and ensure data is handled responsibly

**PROJECT OUTCOME**

The primary outcome of this project is a **user-friendly price transparency tool** that empowers consumers to make informed healthcare decisions by comparing costs across hospitals and insurance providers. This tool will provide:

1. **Detailed Cost Comparisons**: Users will be able to view and compare healthcare costs for specific procedures across multiple hospitals and insurance providers. The system will show essential pricing elements like standard charges, negotiated rates, minimum and maximum costs, and cash prices for each procedure, aiding users in making cost-effective choices.
2. **Intuitive Search and Filter Functionality**: The tool will allow users to search in database as well as UI by common terms, keywords, and CPT codes, with search results easily sortable by cost, hospital, and provider. This feature ensures that users can locate relevant information quickly and accurately.
3. **Clear Data Visualization**: Data will be presented in a tabular, easy-to-read format, which simplifies the comparison process for consumers and highlights significant price variations across insurers and plans.
4. **Backend Data Integrity and High Performance**: The database architecture, designed to handle extensive data with optimized indexing and normalization, ensures fast and accurate data retrieval, even with large datasets. This will provide a seamless experience for users and a reliable backend for future scalability.
5. **Data Consistency and Accuracy**: By implementing data standardization practices, such as unifying CPT code descriptions and handling missing values, the tool will ensure consistent and reliable pricing information across all procedures, locations, and insurers.

### **Expected Impact**

This tool is expected to facilitate informed healthcare choices for consumers and foster a more competitive healthcare marketplace by making price variations clear and accessible. Additionally, it sets a standard for healthcare transparency that can encourage further adoption and refinement of such practices within the industry.

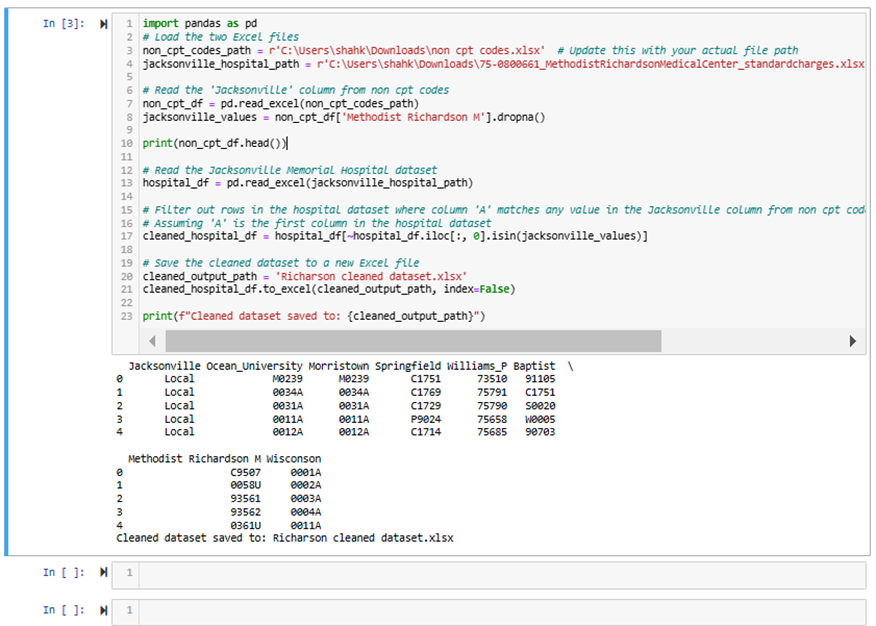
# **Technologies Used**

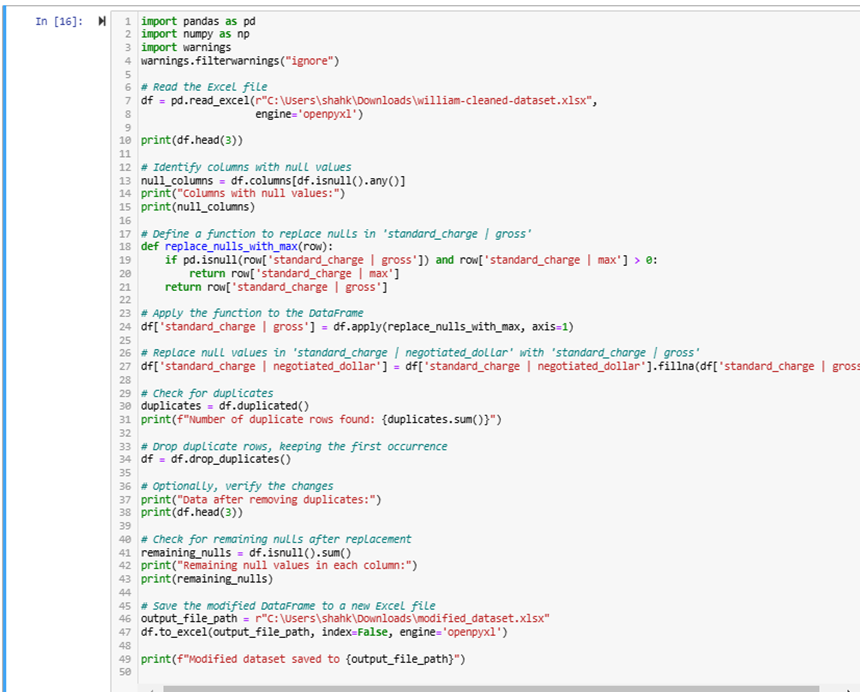
1**.PostgreSQL for Database Efficiency**: We’re using PostgreSQL to store and organize our data. Its advanced indexing capabilities help us handle complex queries and large datasets efficiently. With PostgreSQL, we can ensure quick searches, efficient data retrieval, and scalability as the data grows.

2. **Python for Data Processing and Cleanup**: Python is our primary tool for data cleaning, validation, and transformation. Python scripts are used to normalize and standardize data (e.g., mapping CPT codes consistently across datasets) and to fill in missing values based on predefined logic. By automating these processes with Python, we ensure consistency, save time, and reduce errors in data handling.

**Relevant Screenshots of the outcomes we achieved so far**

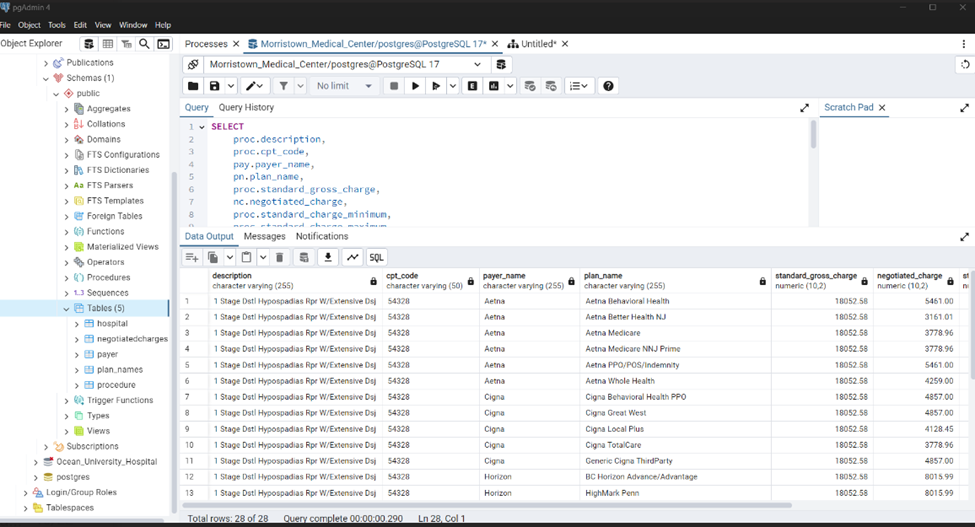
**Data Cleaning Using Python: -**



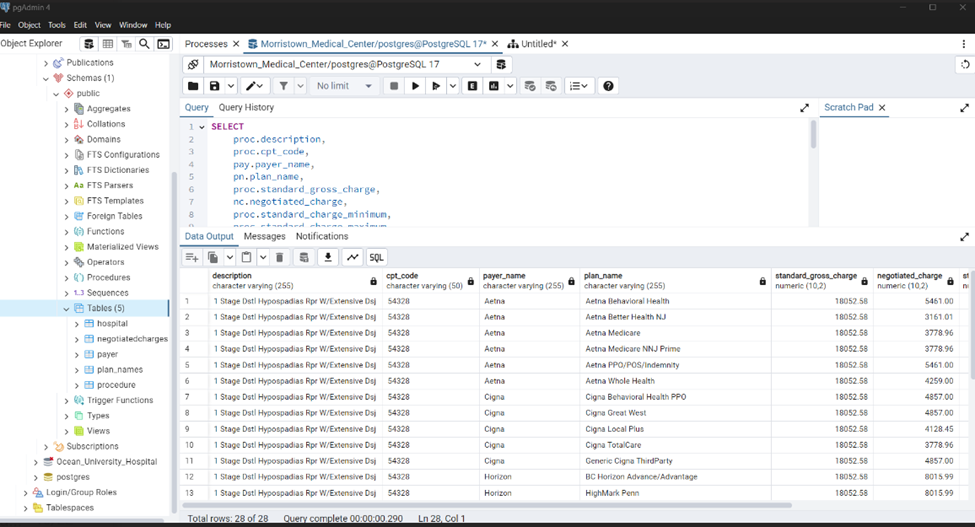


**Database Screenshots:**

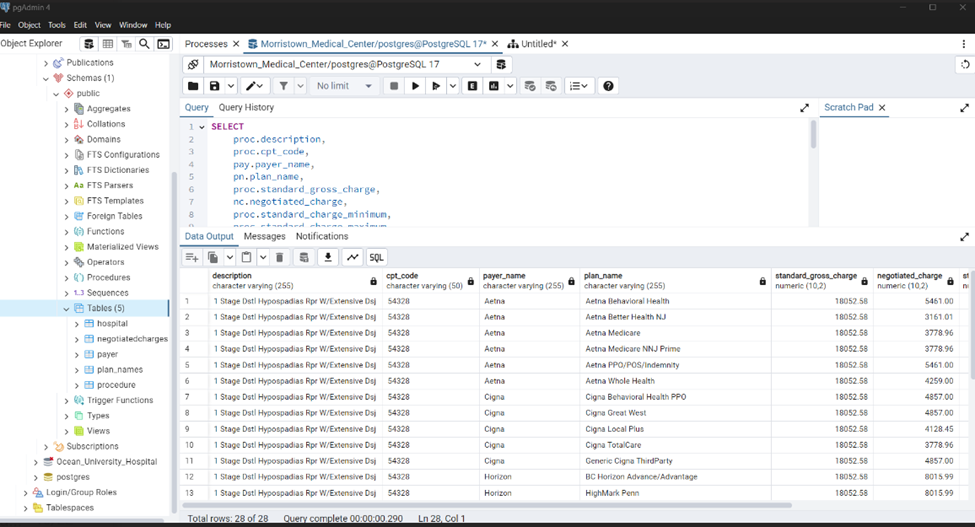
Procedure Table



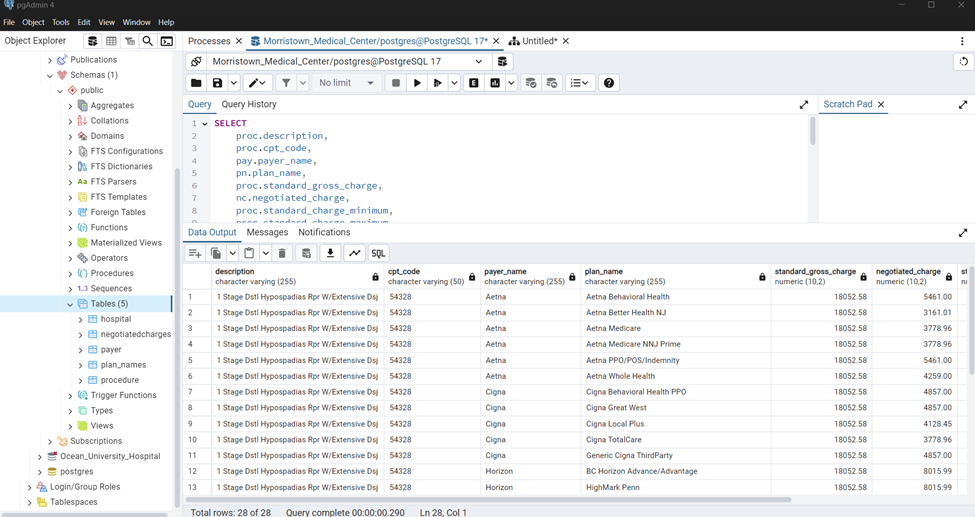
Plan Names:

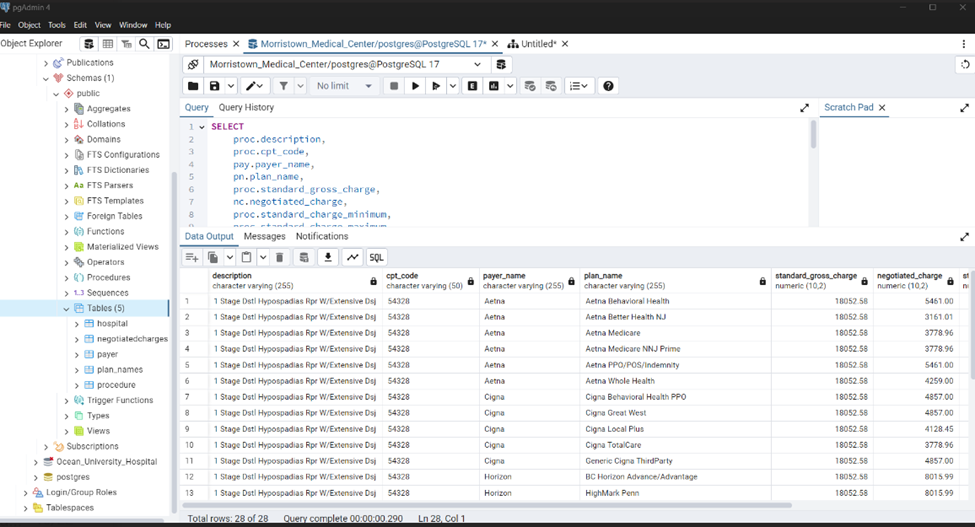


Negotiated Charges



Outputs:





**Physical Infrastructure**

### **Hardware:**

* **CPU**: Intel i7
* **RAM**: 16 GB or more
* **Storage**: 512 GB SSD or larger
* **Display**: 15-inch Full HD or higher

### **2. Software:**

* **OS**: Windows 10/11, macOS

**Development Tools**:

* **IDE**: VS Code, Jupyter
* **Libraries**: Python, Pandas, NumPy, Matplotlib
* **Database**: PostgreSQL, MY SQL, MY SQL Work Bench, Draw.io

### **3. Network & Security:**

* **VPN**: For secure data access
* **Firewall**: Ensure protection

**4. Collaboration**:

* Microsoft Teams for team communication.
* Outlook for project management and communication.

**GROWTH PLANNING**

The growth planning strategy for ClearCare Data Initiative focuses on creating a flexible, scalable, and maintainable system. By adopting microservices architecture, implementing robust data management practices, and maintaining a proactive maintenance approach, the project can efficiently handle future expansion and technological changes.

Microservices Architecture

* Break down the ClearCare Data Initiative into independent, modular services
* Each service will handle specific functions like:
  + Data Collection
  + Data Processing
  + Data Visualization
  + User Authentication

## **Technology Stack for Scalability**

Recommended Technology Stack

* Data Processing: Python (with scalable libraries)
* Database: PostgreSQL with horizontal scaling capabilities
* Cloud Infrastructure: AWS or Azure for:
  + Elastic compute resources
  + Managed database services
  + Auto-scaling capabilities

## **Data Management and Scalability**

Data Collection Framework

* Create standardized data collection templates
* Develop robust data normalization processes
* Implement data validation and cleaning mechanisms

Scalable Data Handling

* Use sharding techniques for large datasets
* Implement data replication for redundancy
* Design flexible database schemas that can accommodate future data types.

Schema Design

* Create normalized tables for hospitals, insurance providers, and treatments
* Implement versioning for data changes
* Design flexible schemas to accommodate future data fields

**Future Enhancements**

* Create a robust algorithms to query data based on user requirements
* Side-by-side plan comparison
* Treatment cost calculators
* Insurance coverage analyzer
* Trend analysis for healthcare pricing

**Client Sponsor’s Approval**

