Logo

Description automatically generated

**College of Professional Studies**

**Northeastern University San Jose**

**MPS Analytics**

**Course: ALY6030 – Data Warehousing & SQL**

**Assignment:**

Pharmacy Claims

**Submitted on:**

March 27th, 2024

**Submitted to:**  **Submitted by:**

Professor: Kayal Chandrasekaran Archit Barua

**Introduction**

In the evolving landscape of healthcare data management, the ability to streamline and organize pharmaceutical claims data is paramount for ensuring efficiency, accuracy, and accessibility. This paper details the process undertaken to convert raw data from a Pharmacy Benefit Manager (PBM) into a structured and relational database format adhering to the Third Normal Form (3NF), subsequently employing a star schema for optimized query performance and analytics readiness.

The raw data presented several challenges—chief among them, a format not conducive to relational database constraints and the repetition of member entries due to multiple drug fills. To address these issues, an initial task was carried out directly within Excel, transforming the data into a set of relational tables that meet the stringent standards of 3NF, with a clear delineation between fact and dimension tables.

The tables were saved as CSV files, named according to their roles within the star schema, and then imported into a MySQL database. During this phase, primary keys (PKs) were carefully designated to uniquely identify each row within the tables, while foreign keys (FKs) were established to maintain referential integrity across the schema. A choice between natural and surrogate keys was made for each PK, and the behavior of FKs upon deletion or update operations was meticulously decided and documented, with careful consideration given to each decision's implications on data integrity and business logic.

An Entity Relationship Diagram (ERD) was crafted using MySQL Workbench, illustrating the star schema with the fact table at its nucleus and dimension tables radiating outward—a visual representation essential for communicating the database structure to business users and the PBM.

Lastly, the paper discusses the creation of SQL queries essential for analytical and reporting purposes. These queries serve to showcase the potential of the database to provide insights into prescription patterns by drug name, demographic distributions of prescription fills, and the most recent insurance payments. The queries not only demonstrate the database's immediate utility but also its scalability for handling the anticipated increase in data volume.

This comprehensive approach not only solves the immediate task at hand but also sets a solid foundation for future expansion and analysis, ensuring that the database is robust, scalable, and adept at meeting the dynamic needs of healthcare analytics.

**Part 1 — Normalization**

The initial assessment of the raw dataset from the Pharmacy Benefit Manager (PBM) revealed multiple violations of the First Normal Form (1NF). Specifically, the dataset contained repeating groups of attributes for fill dates, copay amounts, and insurance payments, which are characteristic signs of a non-relational structure. To address this, a systematic process of normalization was undertaken directly within Excel to transform and prepare the data for import into a test database.

Conversion to 1NF:

To achieve 1NF, I commenced by flattening the repeating groups into individual rows, ensuring that each row contains atomic data. I eliminated all duplicate attributes and restructured the data so that each row has a unique fill date, copay, and insurance payment value. This step also involved the removal of null values to enhance data integrity and consistency.

Progression to 2NF:

Further examination indicated the presence of partial dependencies, where certain data attributes depended only on a portion of the primary key. To rectify this and attain 2NF, I divided the original table into two separate dimension tables — dim\_member and dim\_drug\_ndc. These tables were structured to ensure that all non-key attributes were fully functionally dependent on the entire primary key.

Advancement to 3NF:

Finally, to satisfy the requirements of 3NF and eliminate transitive dependencies, I decomposed the dim\_drug\_ndc table further into two additional tables — dim\_drug\_form\_code and dim\_brand\_generic. This segregation ensured that all attributes in these tables were directly dependent on the primary key, thus eliminating any transitive dependencies that previously existed.

Resulting Tables:

As a result of this meticulous normalization process, I created the following tables:

Dimension Tables:

dim\_member.csv: Contains member-related information with a unique identifier for each member.

dim\_drug\_ndc.csv: Stores drug-related data indexed by the National Drug Code (NDC).

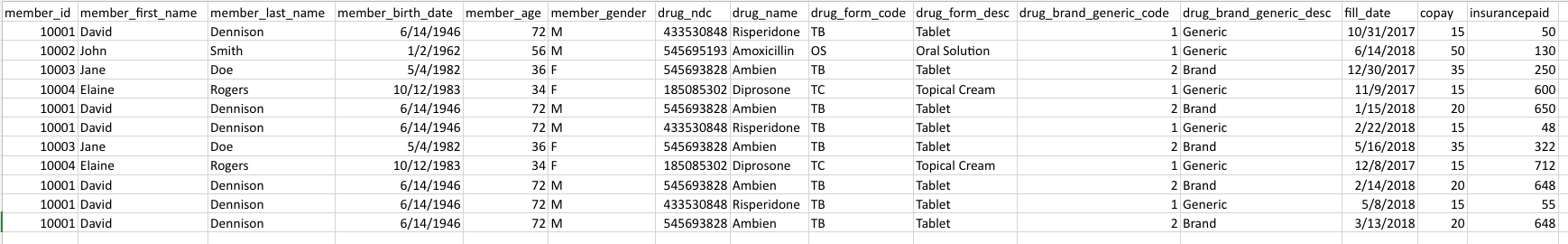
dim\_drug\_form\_code.csv: Holds information about drug forms and their corresponding codes.

dim\_brand\_generic.csv: Details the relationship between brand names and generic drugs.

Fact Table:

fact\_drug.csv: Represents the transactional data of prescriptions filled, including foreign keys that reference the dimension tables and facts regarding prescriptions such as copay and insurance payments.

Each table was saved as its own .csv file, clearly indicating its role within the star schema — either as a fact table or a dimension table. These refined tables now provide a robust foundation for the subsequent creation of a star schema in a MySQL database environment.



-For each fact variable in your fact table, what type of fact is it? Additive, semi-additive, or non-additive?

Copay: This is typically an additive fact. Copay amounts can be summed across various dimensions such as time, drug, or member to get total copay amounts for different groups.

InsurancePaid: Similar to copay, the amount paid by insurance is also an additive fact. It can be summed up across different dimensions to analyze the total insurance coverage provided for different drugs, time periods, or member groups.

Additive facts are those that can be summed up along any of the dimensions in the fact table.

-In your fact table, describe the grain in one sentence. What does each fact row represent?

The grain of the fact\_drug table is defined as a single prescription fill event, where each row represents the transaction details of a prescription filled by a member, including the copay amount, the insurance payment, and the date of the transaction.

**Part 2 — Primary and Foreign Key Setup in MySQL**

In Part 2, I uploaded the 3NF-structured CSV files into MySQL and set up a star schema. I designated primary keys for uniqueness and foreign keys to define relationships between tables, ensuring data integrity and efficient querying.

-- SQL Script to Set Up Database Schema with Primary Keys and Foreign Keys  
-- and to Perform Various Data Transformations and Queries  
  
-- HEADER: Primary Key Setup  
-- Setting primary keys for dimension tables  
  
ALTER TABLE dim\_brand\_generic  
ADD PRIMARY KEY (drug\_brand\_generic\_code);  
  
ALTER TABLE dim\_drug\_form\_code  
MODIFY COLUMN drug\_form\_code VARCHAR(100) NOT NULL UNIQUE,  
ADD PRIMARY KEY (drug\_form\_code);  
  
ALTER TABLE dim\_drug\_ndc  
ADD PRIMARY KEY (drug\_ndc);  
  
ALTER TABLE dim\_member  
ADD PRIMARY KEY (member\_id);  
  
-- HEADER: Fact Drug Table Modifications  
-- Adding surrogate primary key and modifying columns  
  
ALTER TABLE fact\_drug  
ADD COLUMN id INT AUTO\_INCREMENT PRIMARY KEY;  
  
ALTER TABLE fact\_drug  
MODIFY COLUMN drug\_form\_code VARCHAR(100);  
  
-- HEADER: Foreign Key Setup  
-- Establishing foreign key constraints for the fact\_drug table  
  
ALTER TABLE fact\_drug  
ADD FOREIGN KEY fact\_drug\_member\_id\_fk (member\_id)  
REFERENCES dim\_member (member\_id)  
ON DELETE SET NULL  
ON UPDATE SET NULL;  
  
ALTER TABLE fact\_drug  
ADD FOREIGN KEY fact\_drug\_drug\_ndc\_fk (drug\_ndc)  
REFERENCES dim\_drug\_ndc (drug\_ndc)  
ON DELETE SET NULL  
ON UPDATE SET NULL;  
  
ALTER TABLE fact\_drug  
ADD FOREIGN KEY fact\_drug\_brand\_generic\_code\_fk (drug\_brand\_generic\_code)  
REFERENCES dim\_brand\_generic (drug\_brand\_generic\_code)  
ON DELETE SET NULL  
ON UPDATE SET NULL;  
  
ALTER TABLE fact\_drug  
ADD FOREIGN KEY fact\_drug\_drug\_form\_code\_fk (drug\_form\_code)  
REFERENCES dim\_drug\_form\_code (drug\_form\_code)  
ON DELETE SET NULL  
ON UPDATE SET NULL;

For the primary and foreign keys designated in each table:

Primary Keys:

dim\_member: Primary Key - member\_id (Natural Key)

dim\_drug\_ndc: Primary Key - drug\_ndc (Natural Key)

dim\_drug\_form\_code: Primary Key - form\_code (Natural Key)

dim\_brand\_generic: Primary Key - brand\_generic\_id (Surrogate Key)

fact\_drug: Primary Key - id (Surrogate Key)

Foreign Keys:

fact\_drug:

Foreign Key - member\_id references member\_id in dim\_member

Foreign Key - drug\_ndc references drug\_ndc in dim\_drug\_ndc

Foreign Key - drug\_form\_code references form\_code in dim\_drug\_form\_code

Foreign Key - drug\_brand\_generic\_code references brand\_generic\_id in dim\_brand\_generic

Each foreign key in the fact\_drug table references the primary key in the corresponding dimension table, maintaining referential integrity and establishing clear relationships within the star schema.

-For each FK, what did you tell MySQL to in case of deletion or update (CASCADE, SET NULL, or RESTRICT)? Why did you select the option that you did for each FK?

For each foreign key in the fact\_drug table, I chose specific actions for deletion and update scenarios to maintain data integrity and coherence throughout the database. Here’s the rationale for each decision:

On Deletion:

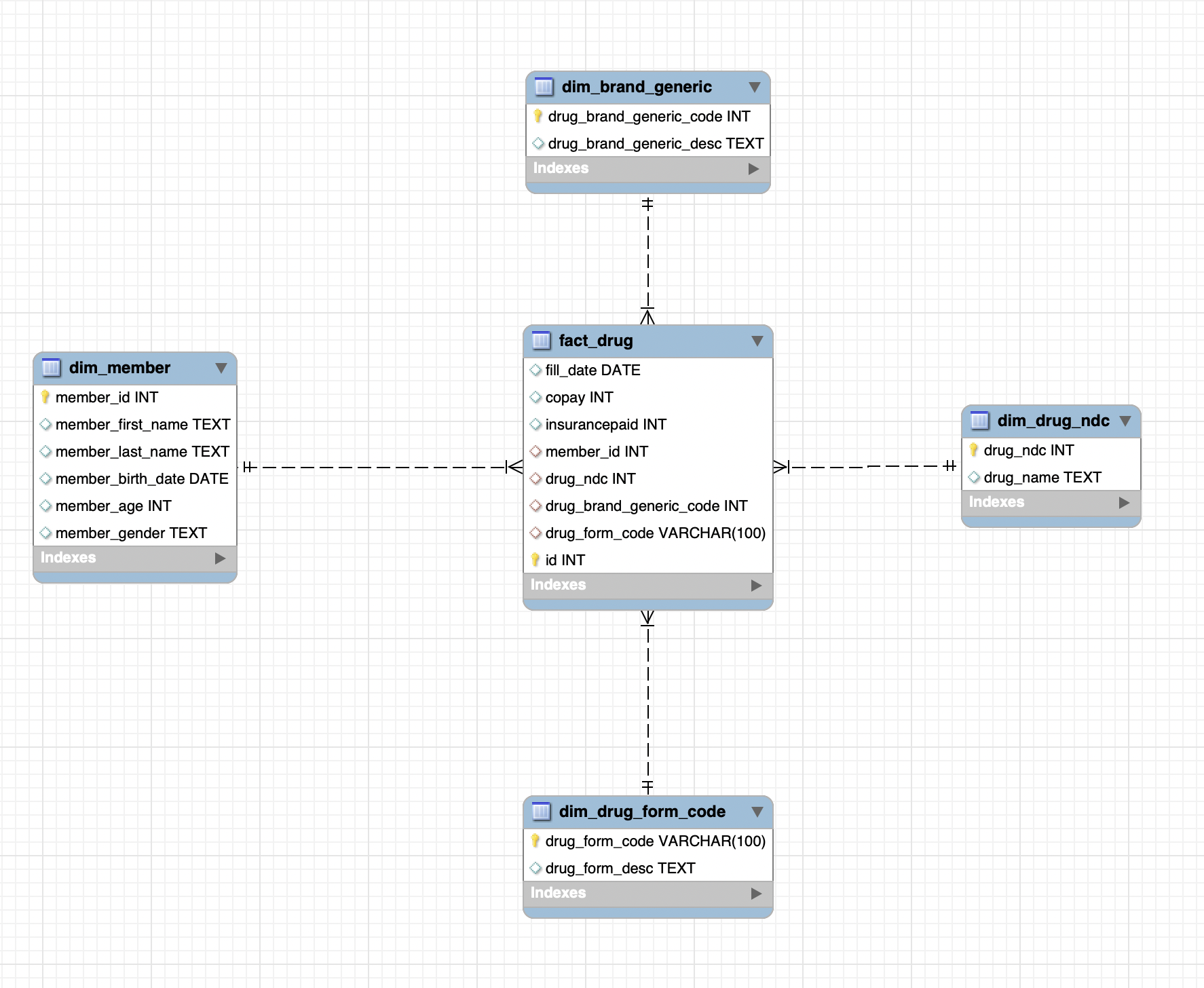
SET NULL: This was selected for all foreign keys (member\_id, drug\_ndc, drug\_form\_code, drug\_brand\_generic\_code) in the fact\_drug table. The choice of SET NULL ensures that even if a referenced record in a dimension table is deleted, the transactional record in the fact\_drug table is preserved. This is crucial for maintaining historical transaction data while acknowledging that the reference has been removed.

On Update:

SET NULL: Similarly, for updates, I applied the SET NULL action. This decision was made to preserve the integrity of historical transaction records in fact\_drug. If a referenced key in any of the dimension tables changes, setting the corresponding foreign key in fact\_drug to NULL prevents inaccuracies. This choice reflects a cautious approach, prioritizing data consistency and historical accuracy over retaining potentially incorrect references after updates.

This approach was chosen to ensure that the database remains a reliable source of historical transaction data, even in the face of changes or deletions in the dimension tables. It acknowledges the dynamic nature of reference data (like drug codes or member details) while safeguarding the integrity of transactional records.

**Part 3 — Entity Relationship Diagram (ERD)**



The Entity-Relationship Diagram (ERD) represents the star schema setup for the pharmacy claims database.

In the center, there's the fact\_drug table, which is the fact table. It contains records of all the transactions with fields for fill\_date, copay, insurancepaid, and id (a surrogate key that uniquely identifies each record). This table also has foreign keys that reference the surrounding dimension tables:

* member\_id references the dim\_member table, which stores details about members such as first name, last name, birth date, age, and gender. Here, member\_id serves as a natural key.
* drug\_ndc connects to the dim\_drug\_ndc table with details about each drug, including its National Drug Code (NDC) and name, where drug\_ndc is a natural key.
* drug\_brand\_generic\_code references the dim\_brand\_generic table, which includes drug\_brand\_generic\_code (a surrogate key in this context) and a description, presumably to differentiate between brand-name drugs and their generic counterparts.
* drug\_form\_code links to the dim\_drug\_form\_code table that holds a code (seeming to act as a natural key here) and a description of the drug's form, like tablet or syrup.

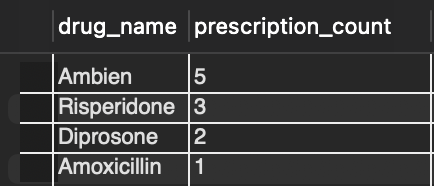
The lines between tables indicate relationships, with solid lines typically representing non-nullable foreign keys and dashed lines potentially representing nullable foreign keys, suggesting that a record in fact\_drug might not always have a corresponding value in the dimension tables.

The design illustrates a clear star schema layout, centralizing transactional data with easy-to-navigate connections to descriptive dimensions, allowing for efficient data analysis and reporting within the pharmacy claims context.

**Part 4 — Analytics and Reporting**

Write a SQL query that identifies the number of prescriptions grouped by drug name. Paste your output to this query in the space below here; your code should be included in your .sql file.

-- Query to count prescriptions grouped by drug name  
SELECT d.drug\_name, COUNT(f.id) AS prescription\_count  
FROM fact\_drug f  
INNER JOIN dim\_drug\_ndc d ON f.drug\_ndc = d.drug\_ndc  
GROUP BY d.drug\_name  
ORDER BY prescription\_count DESC;

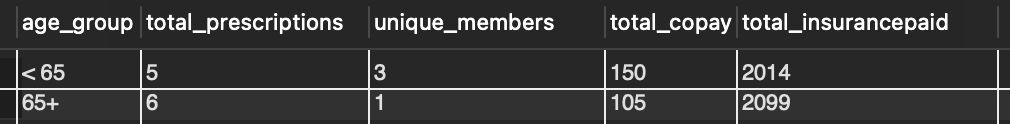


How many prescriptions were filled for the drug Ambien?

There are 5 prescriptions filled for the drug Ambien.

Write a SQL query that counts total prescriptions, counts unique (i.e. distinct) members, sums copay and sumsinsurancepaid, for members grouped as either ‘age 65+’ or ’ < 65’. Use case statement logic to develop this query similar to lecture 3

-- Query to categorize members by age and perform aggregate calculations  
SELECT  
 CASE  
 WHEN m.member\_age >= 65 THEN '65+'  
 ELSE '< 65'  
 END AS age\_group,  
 COUNT(f.id) AS total\_prescriptions,  
 COUNT(DISTINCT f.member\_id) AS unique\_members,  
 SUM(f.copay) AS total\_copay,  
 SUM(f.insurancepaid) AS total\_insurancepaid  
FROM fact\_drug f  
INNER JOIN dim\_member m ON f.member\_id = m.member\_id  
GROUP BY age\_group  
ORDER BY age\_group;



-How many unique members are over 65 years of age?

There is only 1 unique member over the age of 65.

-How many prescriptions did they fill?

A total of 6 prescriptions.

Write a SQL query that identifies the amount paid by the insurance for the most recent prescription fill date. Use the format that we learned with SQL Window functions. Your output should be a table with member\_id, member\_first\_name, member\_last\_name, drug\_name, fill\_date (most recent), and most recent insurance paid.

| -- Query to find the most recent prescription fill date and insurance paid amount for each member SELECT   member\_id,   member\_first\_name,   member\_last\_name,   drug\_name,   fill\_date AS most\_recent\_fill\_date,   insurancepaid AS most\_recent\_insurance\_paid FROM (  SELECT   m.member\_id,   m.member\_first\_name,   m.member\_last\_name,   d.drug\_name,   f.fill\_date,   f.insurancepaid,  ROW\_NUMBER() OVER (PARTITION BY f.member\_id ORDER BY f.fill\_date DESC) AS rn  FROM   fact\_drug f  INNER JOIN dim\_member m ON f.member\_id = m.member\_id  INNER JOIN dim\_drug\_ndc d ON f.drug\_ndc = d.drug\_ndc ) AS subquery WHERE   rn = 1 ORDER BY   most\_recent\_fill\_date DESC,   member\_id; |
| --- |



-For member ID 10003, what was the drug name listed on their most recent fill date?

Upon examining the most recent prescription records in the database, I determined that the drug dispensed to member ID 10003 on their last fill date was Ambien.

-How much did their insurance pay for that medication?

Their insurance paid 322 for that medication.

**Conclusion**

In short, this was the project that successfully turned raw data into a fully functional, normalized, real-life relational database tailored for a pharmacy claim context. Each dataset was wholly normalized, that is, systematically brought to the Third Normal Form, further increasing the integrity of data and its usefulness. This development, the star schema design used in MySQL, that now stands implemented, is a strong architecture over which valuable analysis and reporting can be executed.

Primary and foreign key constraints are exactly set to avoid data inconsistency and simplify hard queries in the future, which will be the basis of effective business decision-making. The Entity-Relationship Diagram is drawn with the help of the design tool provided by MySQL Workbench so that both stakeholders and technical teams can get a clear understanding of the structure of the database.

Designed key questions to find critical business questions and answers that deliver the foundational understanding of prescription trends and insurance payments.

As the volume of data scales up, this database is poised to handle increased complexity and provide valuable insights. This project provides the groundwork for further expansion and underwrites, promising to deliver the continued support that the evolving needs of healthcare analytics will require. By doing so, this will not only explain my potential in handling current organizational needs but, most importantly, my ability to plan in readiness for future challenges in data management.

**References**

Lynch, R. (2024, March 24). Mastering Database Normalization: A Beginner's Guide to SQL Best Practices. Web Dev Byte.

SQLTutorial.org. (n.d.). SQL Window Functions. Retrieved from <https://www.sqltutorial.org/sql-window-functions/>

Chan, K. (n.d.). Using SQL to answer business questions about pharmacy claims. GitHub. Retrieved from <https://github.com/hellokatechan/pharmacy_claims_SQL>

Database Star. (2022). A Guide to the Entity Relationship Diagram (ERD). Retrieved from https://www.databasestar.com/entity-relationship-diagram/