

A white paper with black text

Description automatically generated

**Introduction:**The project focuses on organizing and analyzing pharmacy claims data using a relational database approach. The raw data was normalized from its original format into 3NF (Third Normal Form) to eliminate redundancies and ensure that the data is well-structured for efficient querying. The project involved creating a star schema with fact and dimension tables to facilitate easy data retrieval for business analysis. Key operations included designing primary and foreign keys for data integrity and implementing SQL queries to provide insights into prescription trends, member demographics, and insurance payments.

**Part -1:**

A close-up of a list

Description automatically generated**Table converted to INF:** Initially violated the First Normal Form (1NF) due to repeating groups of attributes fill dates, copay amounts, and insurance payments. To address this, I flattened the repeating groups into individual rows, every contained atomic data and unique values for fill dates, copays, and insurance payments.

A screenshot of a computer

Description automatically generated**Fact table 1:fact\_drug.csv**

Moving to Second Normal Form (2NF) To resolve this, I split the original table into two ensuring that all non-key attributes were fully dependent on the entire PK.

Finally, to achieve 3NF and eliminate transitive dependencies, I further decomposed **into** 2 tables. So by all means we have 4 dim and 1 fact

**Dim table 1:dim\_member.csv:**

**A white grid with black and blue text

Description automatically generated**

**Dim table 2: dim\_drug\_ndc.csv**

A screenshot of a computer

Description automatically generated

**Dimension table 3: dim\_drug\_form\_code.csv**

**A screenshot of a computer

Description automatically generated**

**Dim table 4: dim\_brand\_generic.csv**

**A screenshot of a computer

Description automatically generated**

****

**Copay: This is an additive fact, as it can be summed across various dimensions**

**InsurancePaid: This is also an additive fact. can be summed across dimensions to calculate total insurance coverage.**

****

**Grain of the fact table: The grain of the fact\_drug table refers to an individual prescription fill event, where each row captures the specific details of a prescription filled by a member, including the copay amount, insurance payment, and the date of the transaction.**

**Part -2:**

Database Setup and Updates

* Database Creation:  
  Imported the provided 3NF-structured CSV files into a new database, final\_project, to establish a star schema for analysis.
* All 5 files converted from 5 excel to csv and imported csv files to my sql

**A white table with black text

Description automatically generated**

**A table with text on it

Description automatically generated**



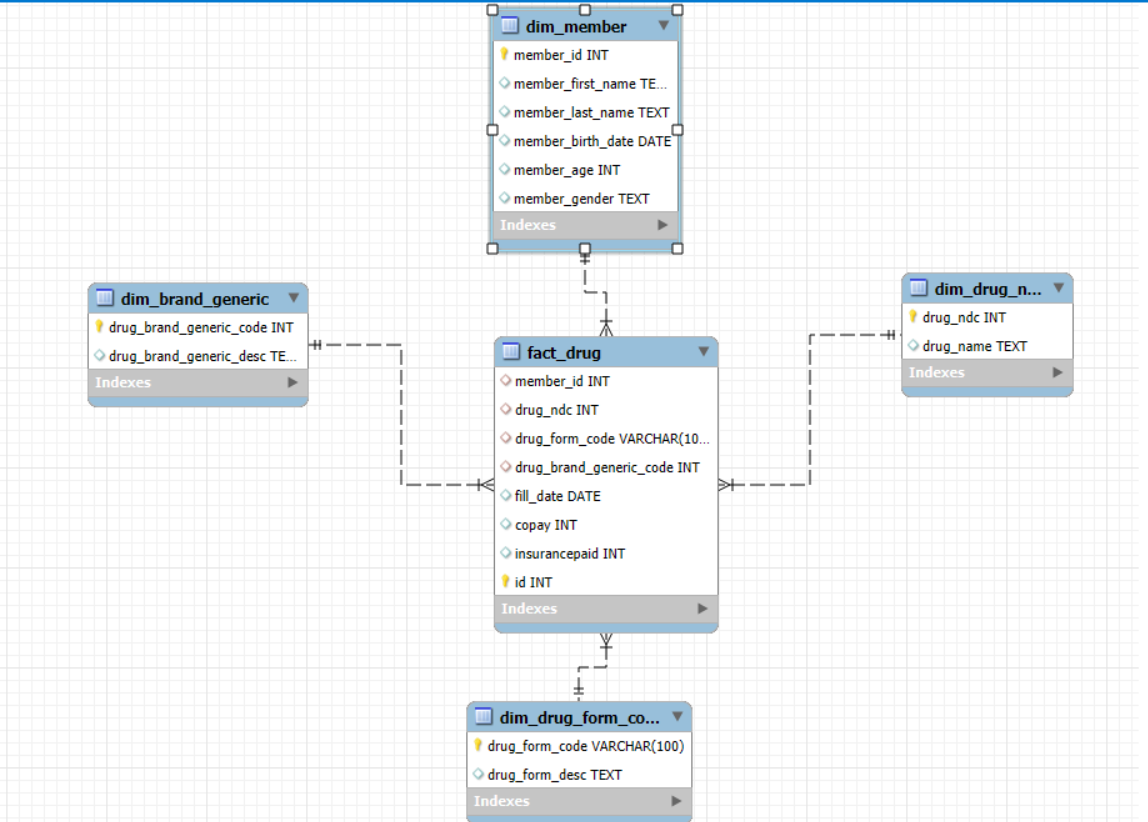
|  |  |  |  |
| --- | --- | --- | --- |
| **Foreign Key (FK)** | **Action on DELETE** | **Action on UPDATE** | **Justification** |
| fact\_drug.member\_id | SET NULL | CASCADE | **DELETE**: Keeps historical data intact by setting member\_id to NULL if a member is deleted. **UPDATE**: Ensures updates cascade to maintain consistency. |
| fact\_drug.drug\_ndc | SET NULL | CASCADE | **DELETE**: Nullifies references to deleted drugs but retains transactions. **UPDATE**: Cascades updates for accurate referencing. |
| fact\_drug.drug\_form\_code | SET NULL | CASCADE | **DELETE**: Preserves historical data by setting drug\_form\_code to NULL if a drug form is deleted. **UPDATE**: Cascades updates for consistency. |
| fact\_drug.drug\_brand\_generic\_code | SET NULL | CASCADE | **DELETE**: Nullifies references to deleted brand-generic codes to retain transaction integrity. **UPDATE**: Cascades updates to avoid data issues. |

Conversion and Formatting

* Date Fields: Converted fill\_date and member\_birth\_date to MySQL DATE type for efficient queries and proper handling of date operations.
* Column Adjustments: Modified data types (e.g., VARCHAR(100)) to meet MySQL requirements for primary keys and foreign key constraints.

This approach aligns with best practices for ensuring data integrity and optimizing database design for analytical queries.

Part -3:



A close up of black text

Description automatically generatedPart 4 :

Query -1: Prescriptions Grouped by Drug Name

A screenshot of a computer

Description automatically generatedA total of 5 prescriptions were filled for the drug Ambien.

A close up of a text

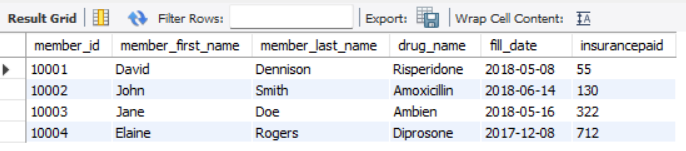
Description automatically generatedQuery 2:

A screenshot of a computer

Description automatically generated

* There is 1 unique member over the age of 65.
* They filled a total of 6 prescriptions.

Query -3:

A close-up of a question

Description automatically generated

The

* Upon reviewing the latest prescription records, I found that the drug dispensed to member ID 10003 on their most recent fill date was **Ambien.**
* Insurance payment for this medication amounted to **322**.

**Conclusion:**  
The database setup and normalization process resulted in a well-organized schema that allows for efficient querying and reporting. By applying normalization techniques, we ensured that the data is stored without redundancy and with integrity. The queries provided useful insights into prescription trends, member demographics, and insurance payments, which will aid in future decision-making and reporting for pharmacy claims management. The overall setup aligns with best practices for data warehousing and SQL, making the data both accessible and actionable for analysis.

**References:**

* **Ben-Gan, I. (2020). *SQL Server 2019: The complete guide to database normalization* [Blog post].** [**https://www.sqlservercentral.com/articles/sql-server-2019-the-complete-guide-to-database-normalization**](https://www.sqlservercentral.com/articles/sql-server-2019-the-complete-guide-to-database-normalization)

**A white background with black text

Description automatically generated**