**A close up of a logo

Description automatically generated**

Final Project

Pharmacy Claims

Submitted by: Yi Yang

ALY 6030-Data Warehousing and SQL

Professor Na Yu

Fall 2020

Due December 12, 2020

**Background:**

The excel file “**ALY 6030 Final Project Data Set.xlsx”** contains sample pharmacy claims for made-up members of an insurance company that pays for pharmacy via a third party known as a *Pharmacy Benefit Manager* or PBM. The file also contains a data dictionary referring to the variables in the data along with their format for your reference.

The PBM has given you these sample records as a starting point so that you as the **Developer** can set up a **test database** and pre-program some common SQL query reports that will be expected from the reporting analysts and business users working at your company, once the *full* claims detail is made available in production in a few months from the PBM. Your work on the test warehouse will help ensure a smooth and successful rollout once the data warehouse goes live in production.

**Part 1) Normalization**

Your task is to eventually load this sample data into your test database and create a **snowflake** schema (as opposed to a star schema).

However before you do, you notice that the **raw data does not meet 3NF**. You make a note to let the PBM know they need to change their formats for a relational database. You also see **some of your sample members appear more than once** because they filled a different drug which further complicates the format.

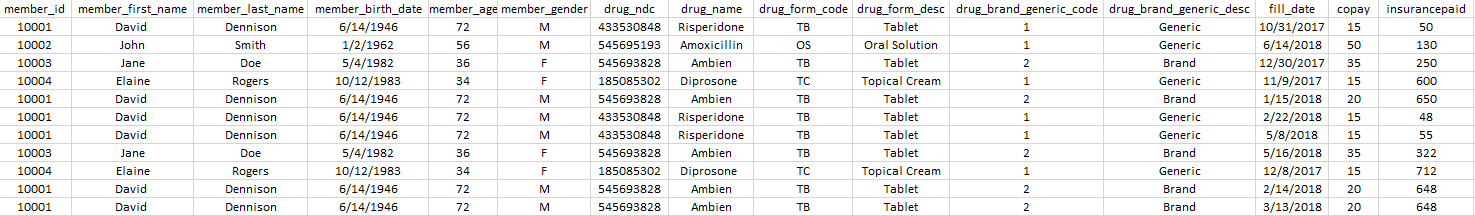
For now you decide to fix this yourself directly in excel.

**Tasks**

* Convert the raw data into a set of relational tables that meet 3NF standards. Feel free to do this directly in excel.
* Each table you create should be either a **fact** table or a **dimension** table
* You do not need to create a separate date dimension joined on date keys. Just include the dates as they appear in the data.
* Save each table as its own .csv file. Indicate in your .csv filename whether your table is a fact table or a dimension table (e.g. name it dim\_tablename.csv, fact\_tablename.csv, etc.).
* At this time you do not need to create or designate primary or foreign keys, that will be done in Part 2 using MySQL
* Answer the following questions:
  1. For each fact variable in your fact table, what type of fact is it? Additive, semi-additive, or non-additive?
  2. In your fact table, describe the *grain* in one sentence. What does each fact row represent?

**Answer:** After reviewing the original data file, we noticed that the table does not meet 1nf standard since there are some attributes repeating in the dataset including fill\_date1, fill\_date2, fill\_date3, copay1, copay2, copay3, insurancepaid1, insurancepaid2, and insurancepaid3.

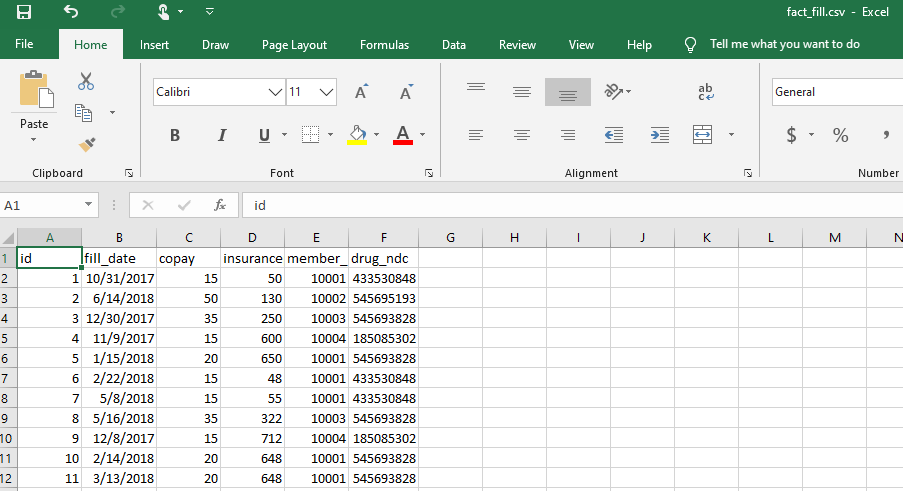
Therefore, we need to rearrange the values under these columns to eliminate duplicate attributes and null values, so that we can make this table meet the 1nf standard. The result of rearranging shows below.

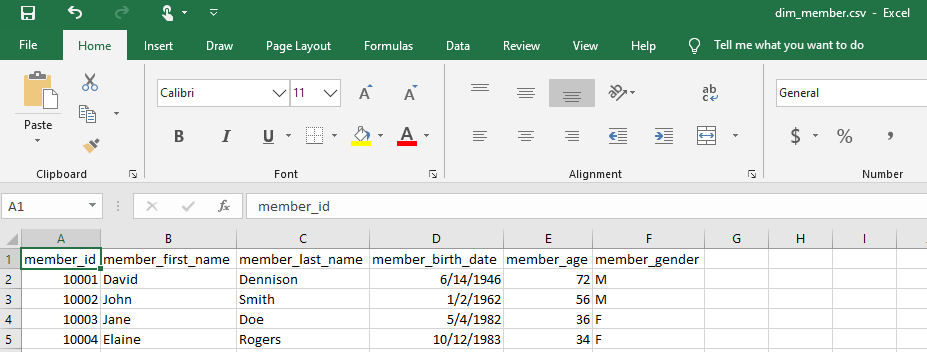


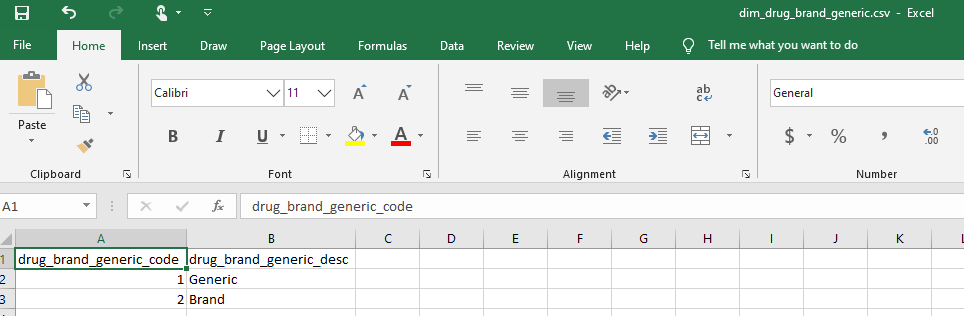
Next, after meeting the 1nf standard, we also find that the table does not meet 2nf, because there exists Partial Dependency. In order to meet the 2nf, we divided the one table into multiple tables, which are two dimension tables that are named dim\_member and dim\_drug, and 1 fact table named fact\_fill. Furthermore, to meet the 3nf, we could divide the dim\_drug table into 2 tables that are dim\_drug\_form and dim\_drug\_brand\_generic to eliminate the Transitive Dependency. There are 5 tables in total.

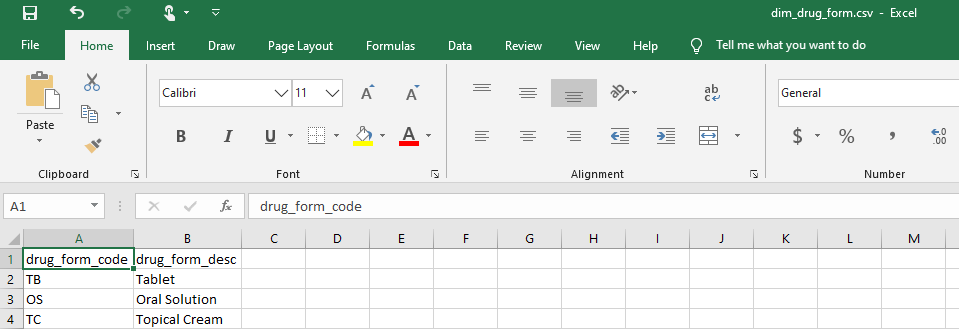
At the same time, we make sure that we deleted all the repetitive rows in each of the 5 tables. Also, specifically in fact\_fill table, we added one single column named id to serve as the primary key for this table.

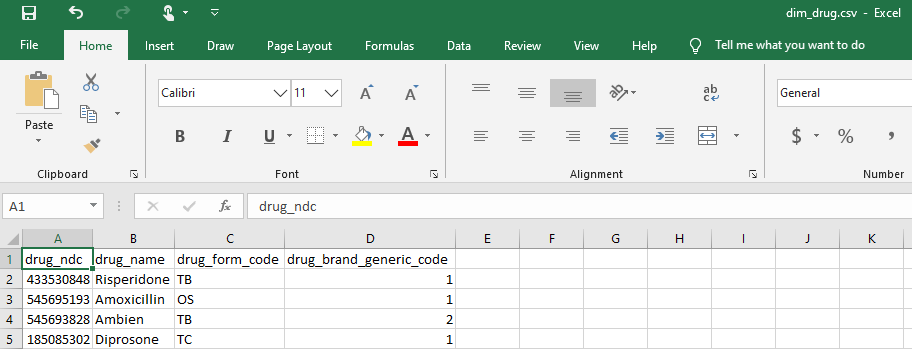
Below are the single fact table named fact\_fill and 4 dimension tables named dim\_member, dim\_drug, dim\_ drug\_brand\_generic and dim\_drug\_form that we obtained.











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

In the fact\_fill table, besides the id columns, the fact variables are filldate, copay and insurancepaid. copay and insurancepaid are additive type of fact, because they can be used with any aggregation functions like the sum() and avg(). While filldate is non-additive type of fact since it cannot be used to do calculation.

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

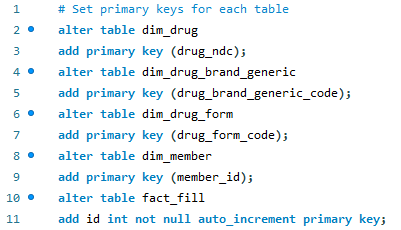
The level of detail(grain) of the fact\_fill table contains patients’ drug filling record like drugs and filldate, and their payment history like copay and how much insurance paid.

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

After you’ve converted the raw table into your 3NF fact and dimension table csv files, you are now ready to upload the data and create the snowflake schema in MySQL as well as designate the primary and foreign keys. All of your output to the tasks below should be included in your .sql code file, except for the three questions below which you should answer directly in this word document.

**Tasks**

* Import the set of 3NF csv files into MySQL either into a new database or one you’ve worked with before, that’s up to you.
* Designate the **primary keys** **(PKs)** in MySQL similar to what we did in Week 5 lab. Indicate whether you want to create a primary key as a **natural key** or a **surrogate key** using the SQL code from the lab. Either choice is valid to use so long as it meets the requirement of a PK that uniquely identifies the rows in your tables.



* Designate the **foreign keys** **(FKs)** using MySQL similar to Week 5 lab. Indicate what you want MySQL to do with the FKs in case of **DELETION** or **UPDATE**. Select either **CASCADE**, **SET** **NULL**, or **RESTRICT** for each of your FKs.



* Answer the following questions here in MS Word:

**1.What are the primary keys you designated for each of your tables? For each PK, is it a natural key or a surrogate key?**

* The PK of Table of fact\_fill is id, and it is a surrogate key.
* The PK of Table dim\_drug is drug\_ndc, and it is a natural key.
* The PK of Table dim\_drug\_brand\_generic is drug\_brand\_generic\_code, and it is a natural key.
* The PK of Table dim\_drug\_form is drug\_form\_code, and it is a natural key.
* The PK of Table dim\_member is member\_id, and it is a natural key.

**2.What are the foreign keys you designated for each of your tables? For each FK, which table did you reference where that FK is listed as the PK?**

In the fact\_fill table, the FK are member\_id and drug\_ndc. The member\_id is PK of the dim\_member table, and the drug\_ndc is the PK of dim\_drug table.

In the table of dim\_drug, the FKs are drug\_brand\_generic\_code and drug\_form\_code. The drug\_brand\_generic\_code is PK of dim\_drug\_brand\_generic table, and drug\_form\_code is PK of dim\_drug\_form.

**3.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?**

Here I’m not necessarily looking for a right or wrong answer to this question, i.e. if you selected SET NULL but you should have selected CASCADE…that’s not the goal; rather I want to make sure you understand the *differences* between these options enough to *justify* your selection. Again it matters less which one you choose vs. that you explain your choice to me in a way that demonstrates comprehension.

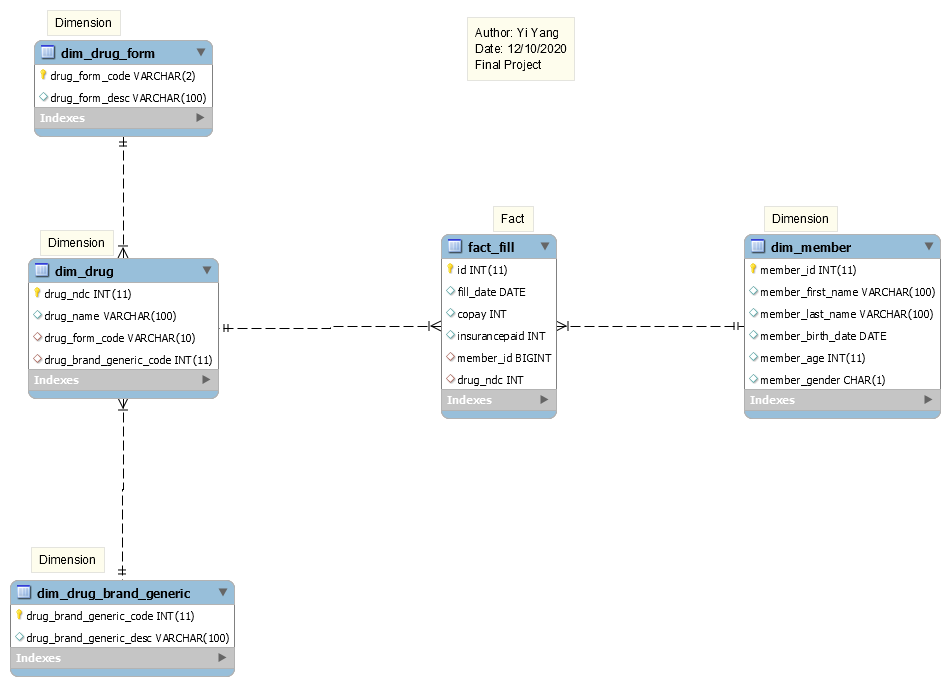
SET NULL means setting to null the value of the field in the child table when updating or deleting the parent table record while CASCADE means updating or delating records from the child table when updating or deleting a record from the parent table. RESTRICT means refusing to update or delate a record from the parent table if there are records in the child table.

In this case, we chose SET NULL for each FK because with this option when we delete the parent table row it will set the column value to NULL in child table which is easier for us to make changes whenever we want.

However, CASCADE will propagate the change when the parent table changes. For instance, if we delete a row, rows in constrained tables that reference that row will also be deleted and we don’t want this happen. Besides, RESTRICT leads us cannot delete a given parent row if a child row exists that references the value for that parent row.

**Part 3) Entity Relationship Diagram**

Now that you’ve created your database, imported your data, and designated your primary and foreign keys, you’ll need to create the official ERD to be able to communicate this table structure to all business users as well as send back to the PBM so that they have for reference.



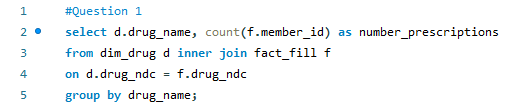
**Part 4) Analytics and Reporting**

With your database set up and ERD drawn, you want to also provide the business users with some sample queries they will likely need to analyze the production data once it is sent from the PBM.

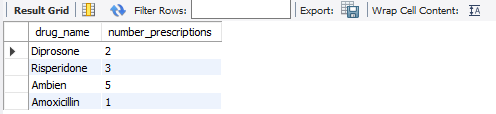
For now you develop the queries on the sample data even though you don’t really need them since it’s not very large, however in the future this will be expanded to several thousand per month so it’s good to plan ahead for the go-live date.

**Tasks**

1.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.

The SQL code is showing below:

And the result is showing below:

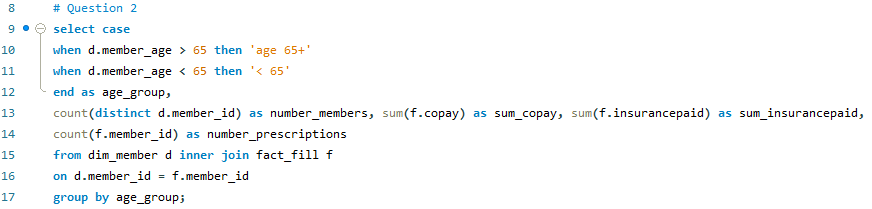


Also answer this question: How many prescriptions were filled for the drug Ambien?

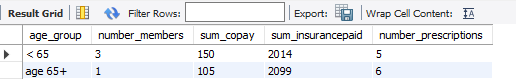
From the result table, for the drug Ambien, there are 5 prescriptions.

2.Write a SQL query that counts total prescriptions, counts unique (i.e. *distinct*) members, sums copay $$, and sums insurance paid $$, for members grouped as either ‘age 65+’ or ’ < 65’. **Use case statement logic** to develop this query similar to lecture 3. Paste your output in the space below here; your code should be included in your .sql file.

The SQL CODE is showing below:



The results are showing below:



Also answer these questions: How many unique members are over 65 years of age?

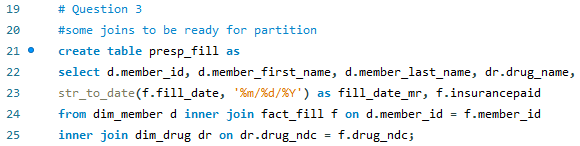
Just one member that is over 65 years of age.

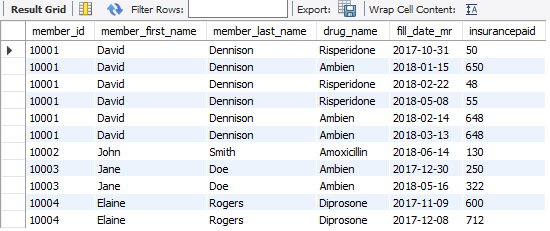
How many prescriptions did they fill?

This single member filled 6 prescriptions.

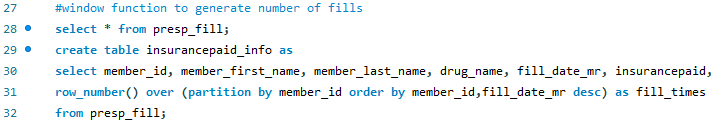
3.Write a SQL query that identifies the **amount paid by the insurance** for the **most recent prescription fill date**. Use the format that we learned with SQL Window functions. Your output should be a table with **member\_id, member\_first\_name, member\_last\_name, drug\_name, fill\_date (*most recent*), and most recent insurance paid.** Paste your output in the space below here; your code should be included in your .sql file.

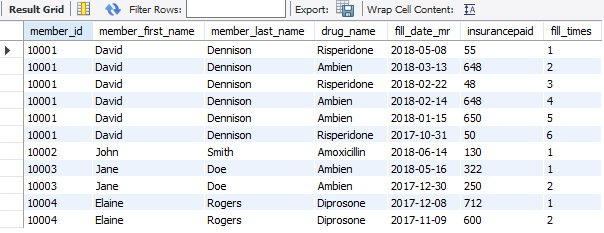
Step 1:



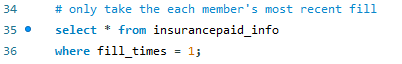


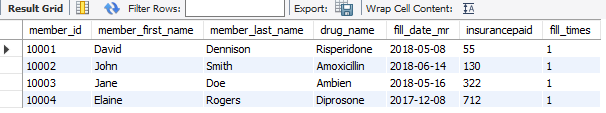
Step 2:





Step 3:





Also answer these questions: **For member ID 10003**, what was the drug name listed on their most recent fill date? How much did their insurance pay for that medication?

Based on the above table, for member ID 10003, Ambien is the drug name listed on his/her most recent fill date.

For member ID 10003, the insurance paid 322 for his/her most recent fill.

Others like member ID 10001, the insurance paid 55 for his/her most recent fill.

And for member ID 10002, the insurance paid 130 for his/her most recent fill.

And for member ID 10004, the insurance paid 712 for his/her most recent fill.