Integrity Constraints and Data Generation - DrugVeda

Anirudh S. Kumar
Roll Number - 2021517
IIIT - Delhi
anirudh21517@iiitd.ac.in

Aakarsh Jain
Roll Number - 2021507
IIIT - Delhi
aakarsh21507@iiitd.ac.in

February 10, 2023

1 Data Generation and Population

1.1 Data Generation

- Primary Data was generated through 2 sources
  - For `products`, `tests` and `brands`, we scrapped Pharmeasy
  - For `users`, `retailers`, `medical_labs` and `suppliers`, we used Mockaroo to generate the data
- Secondary data, like the orders and inventory for retailers, was made with the help of python.
- We made sure that whatever data we were generating would be consistent with the logic of our application while also having variety.

1.2 Data Population

All the data generated in the above process was stored inside JSON files, which were then parsed via Python to insert into the database.

- We used SQLAlchemy as ORM to make the creation of the database process easier.
- We then used `mysql.connector` module in python to parse the JSON files and insert the data into the database

2 Integrity Constraints

For most of the tables, we tried to make a special ID column to ensure there is guaranteed one unique column among all the columns, which also served as the foreign key to different tables. For example :-

- Primary Key for `customers` table is set to `CustomerID`
- Primary key for `appointments` table is set to `AppointmentID`

However, certain tables do not have an explicit primary key but instead use a combination of foreign keys to identify a row uniquely. An example of this is the `inventory` table, where every row has a uniq `BatchID` and `RetailerID` pair.