**Introduction:**

Our project is a retail management system named Cash & Carry, which consists of a python-based GUI and a normalized database implemented in MySQL. The database consists of various entities, triggers, views, and stored procedures. The retail management system consists of all the general basic entities including customers, managers, stores, products and store staff etc.

**NORMALIZATION OF DATA:**

Firstly, we started as considering all the main attributes and tables we might require to implement our database, and increased the number of table and attributes as we realized as we progressed into the project. Our database consists of two schemas, Sales and Production which are interlinked to put forward complete relations and stabilized normalization.

The table in both the schemas are as follows:

**Sales Schema:**

1. Customers
2. Login\_info
3. Stores
4. Staffs
5. Order\_items
6. Orders
7. Admin\_login
8. Location

**Production Schema:**

1. Products
2. Brands
3. Categories
4. Stocks

First Normal Form:

The data is in first normal form since there is no multivalued attribute in the database. In other words, no attribute domain as relation as elements.

Second Normal Form:

Secondly, after implementing first normal form, we looked for any partial decencies and removed those.

**Third Normal Form:**

After ensuring the second normal form, we looked for any transitive dependencies to be removed and eliminated those by further splitting of tables as needed.

**Boyce-Cott Normal Form:**

When we assured that the table was in third normal form, where to eradicate trivial dependency, we implemented 3.5 normal form by shifting a few attributes and splitting a few tables.

**Fourth Normal Form:**

When Boyce-Cott Normal Form was ensured we moved towards fourth normal form where we removed multi-valued dependency by further split of tables, which led to final form of our database.

**DATABASE STRUCTURE:**

The entities and all the structure, in addition all the relationships between the tables are most adequately represented by means of a graphical diagram known as Entity Relation Diagram(ERD). The ERD for our database is as follows:

Diagram

Description automatically generated

To further add completeness to our database, we implemented the following views, triggers and procedures.

**VIEWS:**

Following are the available views for our database,

**For Sales Schema:**

1. Num\_of\_customers ~ displays total number of customers.
2. Affiliated\_stores ~ displays all the affiliated stores to our retail management system.
3. Working\_states ~ displays the list of states where our services operate.
4. Staff\_count ~ display the number of staff workers working under our system.
5. Order\_tally ~ keeps the count of all the available orders.

**For Production schema:**

1. Num\_of\_categories ~ displays the number of available categories.
2. Producrs\_to\_choose\_from ~ display the products from the customer can make his choice for purchase.
3. Stock\_info ~ gives the information regarding available stocks.

**TRIGGERS:**

For the triggers we were more interested in securing our system from modifications in critical data, therefore we created a third schema for our retail database which keeps the audit information regarding certain changes in certain tables and keep their audit. For both the schemas are:

1. Before\_customer\_update
2. Before\_order\_update
3. Before\_order\_items\_update
4. Before\_order\_upadate

**Stored Procedures:**

The stored procedures for our project for utility are as follows:

**For Sales:**

1. SelectCustomerFromACity()
2. zipcodeToLocation()
3. customerLoginVerification(cust\_id,cust\_pass)
4. adminLoginVerification(mang\_id,mang\_pass)

**For Production:**

1. stocksView()