**Introduction**

This report will contain the aims, objectives, and the design of the database system for Betty’s Sweet Shop. Betty’s Sweet Shop is an ecommerce business that sells sweets in the UK entirely through their own website.

There are many requirements that have been set and many requirements which have been met for Betty’s Sweet Shop. The aim for the company is to imprint themselves into the ecommerce world and lead their market staying ahead of their competitors. The database will contain all the information needed for the business to run, this varies from the sweets available, customer information, warehouse, and distribution centres information, placing orders for the customer and for the warehouse.

This database will contain a vast amount of information that will be needed in order for Betty’s Sweet Shop to run efficiently. The database had been grouped up with differently names and information for them to be efficient and for the records to be recorded accurately. This ranges from Warehouse Details, Distribution Centre details to information that regards the orders the store makes from their warehouse. Furthermore, customers details will be stored in this database that will range from their contact numbers to their name. Each and every table in this database has a unique identity field that allows the admins to identify and verify any information which they wish to recall. Furthermore, the accurate order for customers is recorded correctly in which the admins can see exactly how much each customer has spent on an order.

Lastly, in order for this database to meet the requirements that were set for Betty’s Sweet Shop, this report will highlight and show a step-by-step process of the creation and implementation of this database that was created.

This will start with the development and design of the class diagram which is a blueprint for the relational model. This allowed me to gain an idea and have a grasp of how my relationships in the database will look. Next is the relational model which will show the relationships between each table in the database grouped correctly and efficiently and each field name and type in each type. This is the most efficient way to managing data in a simple and precise structure. As the database has been designed, next is the create table statements which will be used to implement the structure that was shown in the relational model. The create table statements will highlight each field, their name, and data type. Once all the tables have been created, data will be implemented in into the tables to ensure their accuracy and efficiency.

Moreover, the next phase will be testing the database, this will be running SQL queries that will range from inserting data into the database to retrieving data from the database. This wil highlight if the database meets the requirements set by the client and if any changes need to be made. Lastly, the conclusion of the report will include an evaluation of the database and the product, the flaws that have been found in the database and any changes that could’ve been made.

**Diagram

Description automatically generatedClass Diagram**

**Assumptions –**

* A customer can either have 1 or 0 shopping baskets.
* A Customer can either have 1 or 0 shopping lists.
* For every order there will be 1 payment information.
* A sweet can either have 1 offer or none.
* Many sweets can be in 1 category, or 1 category can have many sweets.
* There is 1 distribution centre for every store order
* There can only be 1 warehouse for every store order
* Many orders can contain many sweets, or many sweets can have many orders.

**Design**

To meet the requirements set by the client I had to model the database correctly and identify what the relationships are between the several tables.

Another requirement that had to be met was that customers will be shown different offers of sweets. For me to meet this requirement I identified that I had to have an offers table. This offers table will store all the current offers that are available for sweets and when the offer ends in date format. In the offers table it will have a field named sweet\_ID. which is a foreign key that references from the sweet table. This will allow the client and user to identify which specific sweets of what offer on them currently. I then decided that the offers and sweets table will have a 0..1 relationship, this will allow me to highlight that 1 sweet can either have no offers or 1 offer.

Another requirement that was set was the need for customers to bookmark and view sweets which they are interested in. This will show in the customers shopping list. What I first planned on doing was only having 3 fields for the Shopping List table, this included List\_ID to uniquely identify each row, Sweet\_ID for the users to identify what sweets they have placed in their list and Customer to identify what customer this list belongs too. However, after consideration, I decided to add 2 more rows that would show when the user or client added the sweets to the list and when the sweets were removed from the list.

Furthermore, the feedback I got from my first-class diagram draft I took into consideration and used it to improve my tables and database. One of the key points that was stated was that I have a relationship between customer and orders but no relationship between order and sweets. After taking this into consideration, I then decided to make a relationship between the 2 tables Orders and Sweets. This relationship would be in form of a foreign key Sweet\_ID referencing the primary key in the sweets table being in the order tables. This was done to make sure that my database met the requirements set by the client. This will allow the user/client to see exactly what orders the customers made and what sweets the customers ordered.

**The Relational ModelDiagram

Description automatically generated**

**Text

Description automatically generatedImplementation**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generatedText

Description automatically generated**

**Text

Description automatically generated**

**Conclusion**

Throughout the process of designing, implementing, and testing my database I have learnt a lot that I know will benefit me in the long run.

In the first stage of this exercise, we had to create a class diagram for Betty’s Sweet shop. The creation of the class diagram taught me a lot as I learnt how to properly identify what fields are in what tables, the different relationships between each table and how to make sure my class diagram meets the client’s needs. Moreover, when I first submitted my first draft, I believe the feedback I received assisted me a lot into improving my class diagram and taught me how to perfect it in the future. This ranged from me correcting my relationships, making sure the tables fit the client’s needs, making sure each field is correct. This also made me go back through the learning materials that were on Canvas and using google less as it was less reliable.

Next was starting the database and creating the tables. So far, I have identified the relationships between my tables in my class diagram and what tables are linked with each other. This stage of the process I had to implement my class diagram and ensure that the relationships were valid, and my database was efficient. However, this stage of the exercise was the easiest for me as I have used ORACLE APEX before, and I believe to be proficient in SQL. This allowed me to utilise the skills and knowledge I gained beforehand to ensure my database came out correct and efficient. I had to create several tables with each having a different unique primary key, I had to ensure that my tables had a relationship with each other thus I added a foreign key for the ones that needed it. However, while creating the tables I had to go back to the assignment brief and ensure that my database met the requirements of the clients as I believe I was overcomplicating or drifting off a bit from the aim. I also watched a couple youtube videos on how to properly show a relationship between 2 tables, alter a table to add a composite key or foreign key and alter table to edit any columns I wanted. All this allowed me follow exactly what the client wanted, the relationships they wanted for example, customers being able to see sweets, customers being able to see their shopping basket and list and being able to edit them.

Next when it came to adding data to my database I had to go slowly and make sure that the data added was correct, each field accepted the correct data type and the necessary fields were null. I first tested my database by using single line insert statements to see if my tables accepted data correctly. After I decided they did, I resulted in using mock data generated online from Mockeroo. Mockeroo allowed me to furthermore fit the client’s needs, I inserted a formula while creating the mock data in mockeroo for the customer orders table which allowed my database to add up the costs correctly and show the actual total order cost.