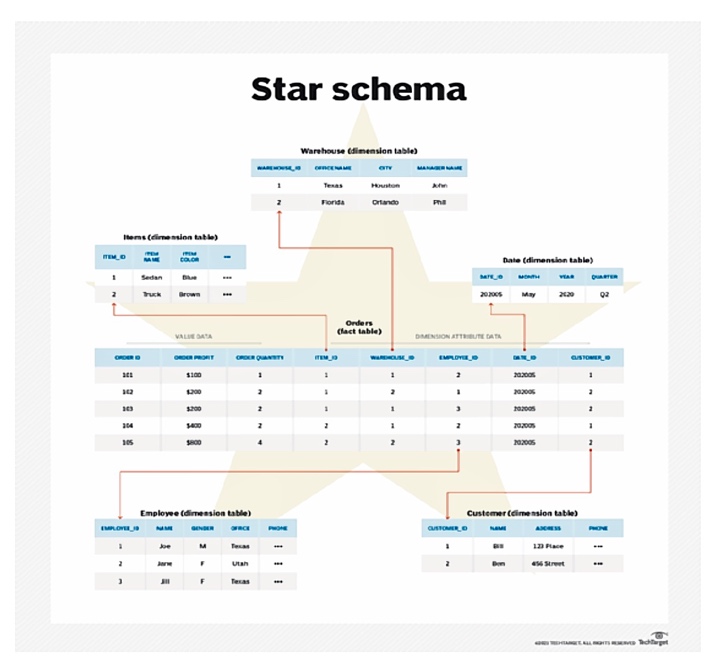
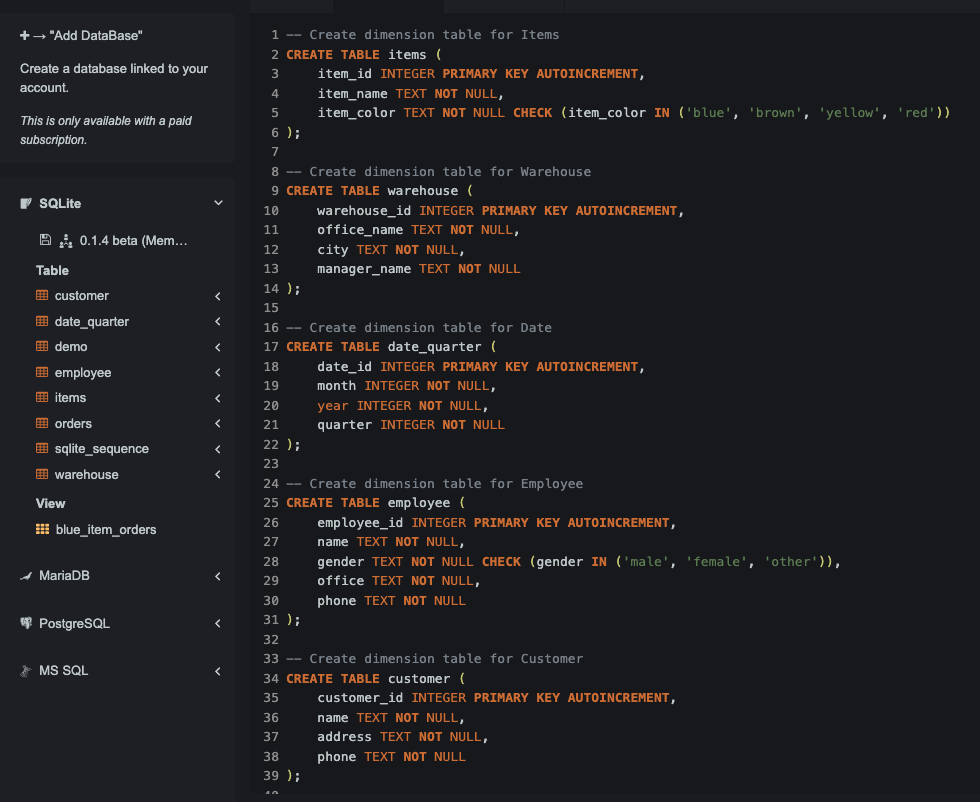
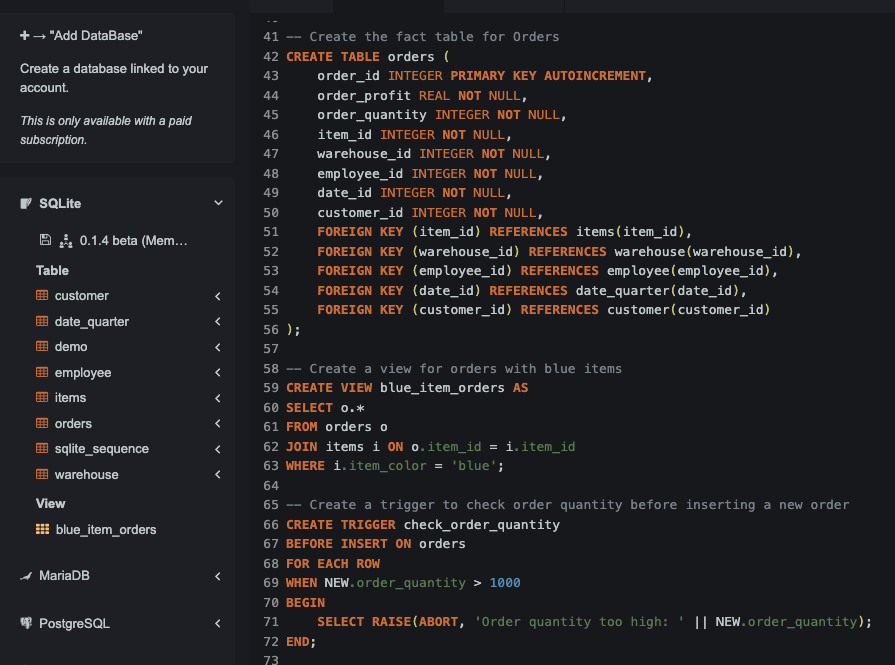
Databases and Data lakes Applied exercise Module 2

I was tasked to create a database based on the following schema.



I started by using SQL lite to create all the Tables needed for the database using the SQL Code supplied.





Each Field in the Tables were Given a Data Type and Constraints.

For example, in the “Items” Table the field “Item\_color” is a Text Data Type, Cannot be blank and must contain the colours 'blue', 'brown', 'yellow', 'red'.

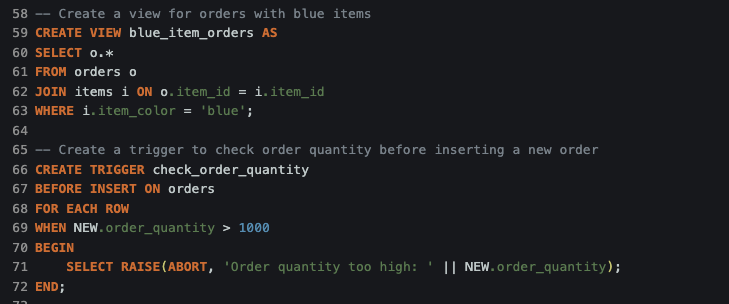


This helps to maintain Data consistency and quality. Preventing errors in future.

Each Table has also been assigned one field that is the Primary key for that table. This is the Field that we searched for most often in that table. By assigning Primary keys can Improve performance in searches as well as making it easier to maintain the table.

Following this we can see that the Order Table has Both a primary key and Several Foreign Keys Created. This is due to the Order Table is at the Centre of The Schema and uses the remaining Tables (Dimension Tables) are Linked though it via these foreign Keys.

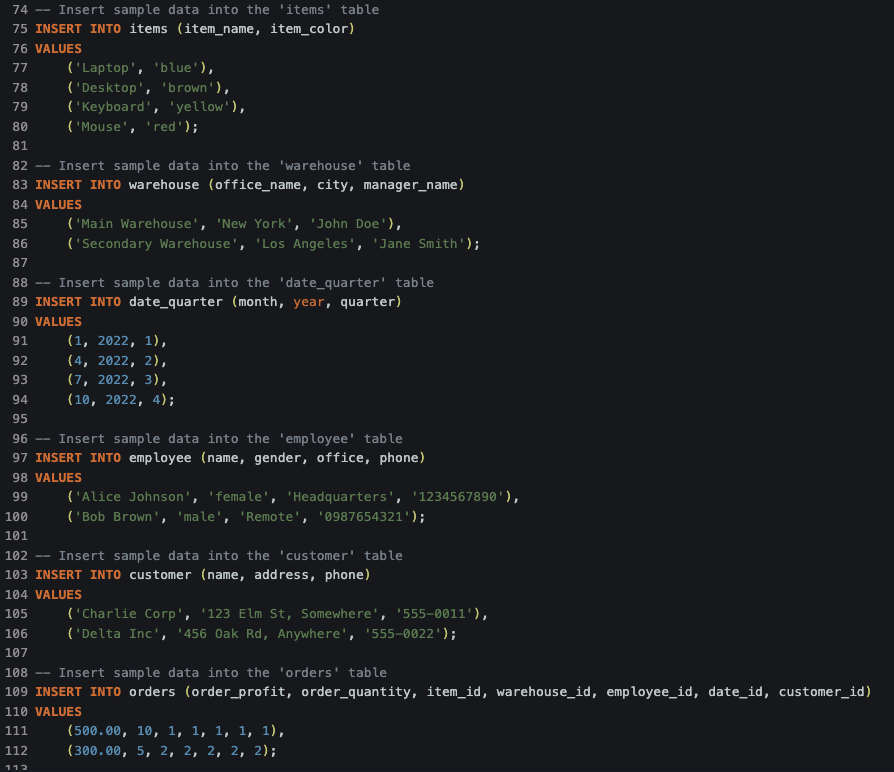
I Then went on to Adding a View, A Stored Procedure and a Trigger



The View “Blue\_item\_orders” Filters orders where Items are a “blue” Colour By Joining The Orders table with the Items Table and returns only the Records That are Blue in colour.

The stored procedure & trigger “Check\_order\_quantity” is a simple data validation check that ensures the order quantity is not unreasonably High (set at 1000 units). This trigger is called before the insertion of a new record into the orders table. This provides a quick and automated way to avoid errors in data entry.

Once the Database and Tables were created, I then used the “Insert” statement to populate the Tables.



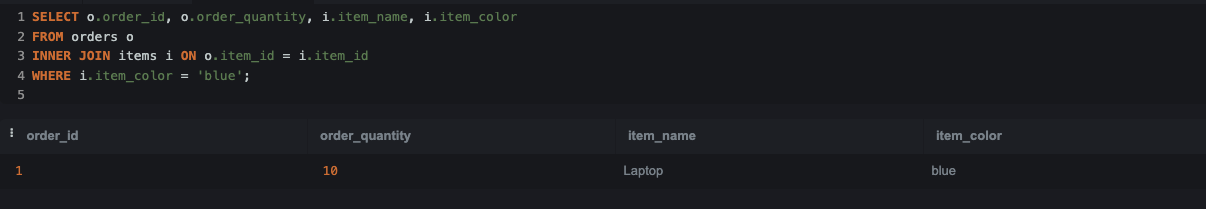
Now That The tables had been populated I could then begin to run queries of the database.

Firstly, I began by using the “Blue\_item\_orders” view that I had previously created. This returned the everything from any record that had a blue item colour.



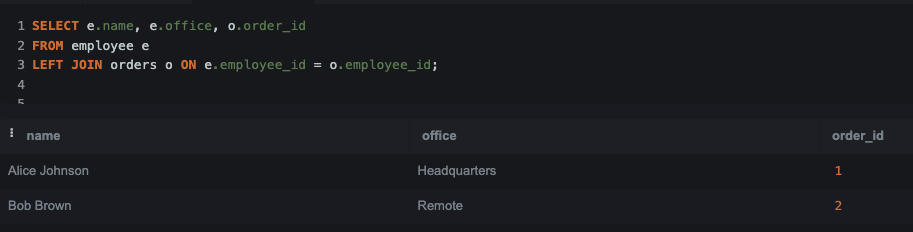
I then moved on to The other goals listed below;

Goal 1: Retrieve all orders along with the item details but only where the item colour is 'blue'. SQL Query



This query demonstrated an INNER JOIN, which returns rows when there is at least one match in both tables. It's used here to link orders with their respective items, filtering for only those items that are blue.

Goal 2: List all employees and any orders they may have processed, including employees who have not processed any orders.



The LEFT JOIN ensured that all employees are listed, regardless of whether they have processed any orders. This is crucial for full staff audits or analyses, showing unassigned employees as well.

Goal 3**:** Display all customers and any orders they have placed, including all placed orders even if the customer details are not recorded.



A RIGHT JOIN is useful as I wanted to ensure all entries in the joined table (orders in this case) are shown—even if there’s no corresponding entry in the base table (customers).

Observations and Normalisation improvements.

During the course of this exercise I have noted various Normalisation improvements that could be completed im improve performance and storage needs.

1. **Orders Table Redundancy**
   * The orders table contains multiple foreign keys linking it to different dimension tables. However, some data may still be repeated, particularly in cases where detailed customer or item attributes are frequently referenced.
2. **Customer Data Duplication**
   * Customer information (e.g., name, address, and phone) is stored directly in the customer table. However, if customers frequently place orders, referencing the same customer multiple times could lead to redundant data in queries and potential inconsistencies.
3. **Item Data Duplication**
   * The items table stores item-specific information, such as item\_name and item\_color. If an order frequently involves the same item, there could be unnecessary repetition of data.
4. **Warehouse and Employee Data**
   * If employees are frequently linked to multiple orders, storing their details directly in the orders table can create duplication.
5. **Date Table Renaming Consideration**
   * The date\_quarter table was renamed from date to prevent conflicts with SQL reserved keywords. However, the structure of the table itself can be optimized further by storing dates in a more standardized format.