Data Management Coursework

FM Clothing Store Database

Group Members

UP2293300

UP2302347

UP2089114

FM Clothing Store Database Solution

The purpose of the FM Clothing Store database is to provide a structured, centralized relational database solution that can help maximize effective management of store operations. The database revolves around optimization of inventory, customer transactions and product availability at multiple branch locations.

Not only does the FM Clothing Store database save essential store information—it allows one to analyze product availability, monthly revenue per location and customer orders by city. Such a design drives optimized operations, data integrity and helpful perspectives of business performance, which aligns with FM Clothing's purpose of, improved customer service at all branches as well as informed decisions.

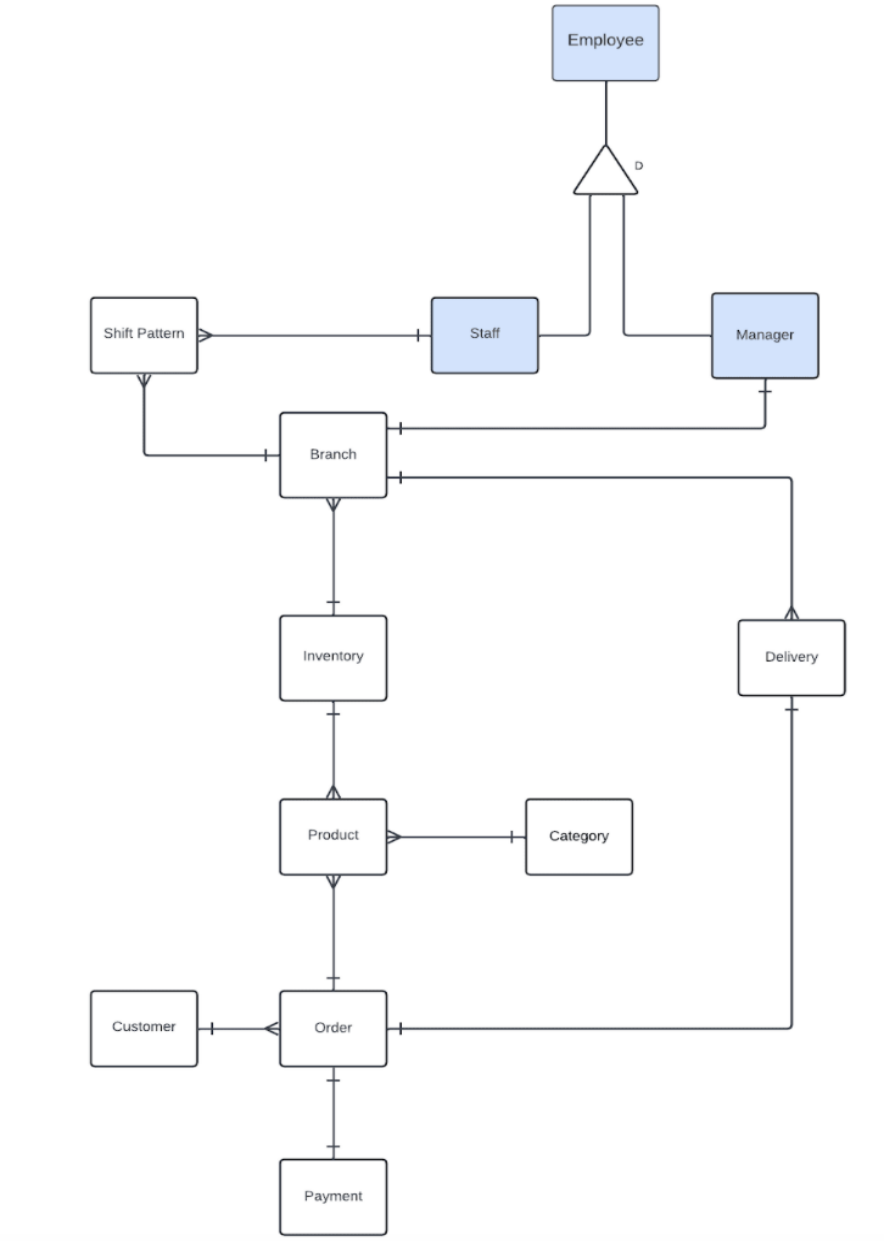
Table of Contents

Task T1: EERD

This EERD diagram shows the entity relationship between tables of database for FM Clothing Store of Hampshire.

The EERD diagram generated using Lucid Chart. The diagram below shows the FM Clothing Store EERD and it has 12 tables created for the FM Clothing Store Database system.

Some assumptions were made during the development of this EERD diagram, which are explained in Task 2.



Task T2: Rationale and Assumptions

- We assumed that Inventory is centralized, it is a one single factory, which has all inventory, and from there the inventory is distributed to all branches rather than each branch having their own inventory. Which makes this design a simple stock management and restocking becomes easier process across each branch.

- The employee table represents all employees of FM Clothing Store including company management team as well. Therefore, the employee table specializes and has sub tables staff (which regular staff of each branch) and manager as each branch has its own manager reporting to head office. This is to streamline roles assignment.

- To know whether a customer has an account to retrieve their order history and details, we did not create a separate table for customer account, but instead included a Boolean value within the customer table to confirm or deny whether a customer has an account registered with FM Clothing Store or not.

Task T3: Data Dictionary/ Scripts

Branch					
Attribute Name	**PK/ AK**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Branch_ID	PK	INT	PRIMARY KEY, UNIQUE NOT NULL		All FK references are uniquely identified by each branch.
Manager_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Manager	
Shift_Pattern_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Shift_Pattern	
Branch_Name		VARCHAR(20)	NOT NULL		
Branch_Address		VARCHAR(255)	NOT NULL		
Branch_Phone_Number	AK	INT	UNIQUE NOT NULL		Contacting branches is vital for FM clothing's survival, so contact information cannot be null. Unique identifier of each branch

Branch_Email	AK	VARCHAR(100)	UNIQUE NOT NULL		Unique identifier of each branch

Shift Pattern					
Attribute Name	**PK/ AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Shift_Pattern_ID	PK	INT	PRIMARY KEY, UNIQUE, NOT NULL		Each member of staff's shift work is uniquely identified by a shift pattern_ID, each shift_pattern ID belongs to a particular branch
Staff_ID		INT	FOREIGN KEY,, UNIQUE, NOT NULL	Staff	
Shift_Start		TIME	NOT NULL		
Shift_End		TIME	NOT NULL,		
Shift_Date		DATE	NOT NULL		

Employee					
Attribute Name	**PK/ AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Employee_id	PK	INT	PRIMARY KEY, UNIQUE, NOT NULL		
First_Name		VARCHAR (20)	NOT NULL		
Last_Name		VARCHAR (20)	NOT NULL		
Job_Title		VARCHAR (50)	NOT NULL		
Email	AK	VARCHAR (100)	UNIQUE, NOT NULL		
Phone_Number		INT	UNIQUE,		

Staff					
Attribute Name	**PK/ AK**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Staff_ID	PK	INT	PRIMARY KEY, UNIQUE,		Staff details already specified in the employee table, no further attributes required.

			NOT NULL		
Shift_Pattern_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Shift_Pattern	
Employee_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Employee	

Manager					
Attribute Name	**PK/ AK**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Manager_ID	PK	INT	PRIMARY KEY, UNIQUE, NOT NULL		Each manager_id is unique and belongs to a particular branch
Branch_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Branch	
Employee_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Employee	

Product

Attribute Name	PK/ AK	Data Type & Size	Domain and Constraints	FK Reference	Description (where non-obvious)
Product_ID	PK	INT	PRIMARY KEY, UNIQUE NOT NULL		
Category_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Product_Category	
Product_Description		TEXT	NOT NULL		
Size		TEXT	NOT NULL		
Composition		TEXT	NOT NULL		
Price		DECIMAL (10,2)	CHECK (Product_Price > 0)		

Product Category					
Attribute Name	**PK/ AK**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Category_ID	PK	INT	PRIMARY KEY, UNIQUE, NOT NULL		

Category_Name		VARCHAR(20)	NOT NULL		

Inventory					
Attribute Name	**PK/ AK**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Inventory_ID	PK	INT	PRIMARY KEY, UNIQUE, NOT NULL		
Product_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Product	
Branch_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Branch	
Stock_Quantity		INT	NOT NULL, CHECK		Can be equal to zero as stock can run out.
Restock_Date		DATE	NOT NULL		

Customer					
Attribute Name	**PK/ AK**	**Data Type & Size**	**Domain and**	**FK Reference**	**Description (where non-obvious)**

			Constraints		
Customer_ID	PK	INT	PRIMARY KEY, UNIQUE, NOT NULL		
First_Name		VARCHAR (20)	NOT NULL		
Last_Name		VARCHAR (20)	NOT NULL		
Email	AK	VARCHAR (100)	UNIQUE, NOT NULL		Cannot be null, for order confirmation purposes and proof of purchase
Phone_Number	AK	VARCHAR (15)	UNIQUE		Can be null, customer does not have to give a phone number to because of privacy.
City		VARCHAR (50)	NOT NULL		
Account_Registered		BOOLEAN	NOT NULL		True or False response, confirming a customer account has been created to store order history, so customer can retrieve order/delivery details whenever they want post-purchase.

Payment

Attribute Name	PK/ AK	Data Type & Size	Domain and Constraints	FK Reference	Description (where non-obvious)
Payment_ID	PK	INT	PRIMARY KEY, UNIQUE, NOT NULL		Each payment is unique to an order.
Order_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	"Order"	
Amount_Paid		Decimal (10,2)	NOT NULL CHECK (Amount_ Paid > 0)		No discounts are offered so amount paid should be greater than zero
Payment_Mod e		CHAR(2 0)	NOT NULL		
Payment_Date		DATE	NOT NULL		

Order					
Attribute Name	**PK/ AK**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Order_ID	PK	INT	PRIMARY KEY, UNIQUE, NOT NULL		Each order_ID is unique to every customer that placed an order, each order is specific to a branch when purchased in-store, and each product purchased from in store or inventory has a unique identifier.

Customer_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Customer	
Product_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Product	
Branch_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	Branch	
Order_Date		DATE	NOT NULL		
Order_Total		DECIMAL (10,2)	NOT NULL CHECK (Order_total >0)		Every order should be greater than zero, since no discounts are provided.
Quantity		INT	NOT NULL CHECK (Quantity >0)		

Delivery

Attribute Name	PK/ AK	Data Type & Size	Domain and Constraints	FK Reference	Description (where non-obvious)
Delivery_ID	PK	INT	PRIMARY KEY, UNIQUE, NOT NULL		Each delivery is specific to an order.
Order_ID		INT	FOREIGN KEY, UNIQUE, NOT NULL	“Order”	
Delivery_Mode		VARCHAR(100)	NOT NULL		
Delivery_Address		VARCHAR(255)	NOT NULL		
Delivery_Date		DATE	NOT NULL		

This data dictionary allowed us to generate the following queries to create tables.

Queries to create tables

Employee Table

```
CREATE TABLE Employee (
  Employee_ID INT PRIMARY KEY UNIQUE NOT NULL,
  First_Name VARCHAR(20) NOT NULL,
  Last_Name VARCHAR(20) NOT NULL,
  Job_title VARCHAR(50) NOT NULL,
  Email VARCHAR(100) UNIQUE,
  Phone_Number INT UNIQUE);
```

Manager

```
CREATE TABLE Manager (
  Manager_ID INT PRIMARY KEY UNIQUE NOT NULL,
  Branch_ID INT,
  Employee_ID INT,
  FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID),
  FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID));
```

Shift Pattern

```
CREATE TABLE Shift_Pattern (
  Shift_Pattern_ID INT PRIMARY KEY UNIQUE NOT NULL,
  Shift_Start TIME NOT NULL,
  Shift_End TIME NOT NULL,
  Shift_Date DATE NOT NULL,
  Branch_ID INT,
  Staff_ID INT,
  FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID),
  FOREIGN KEY (Staff_ID) REFERENCES Staff(Staff_ID));
```

Branch

```
CREATE TABLE Branch (
  Branch_ID INT PRIMARY KEY UNIQUE NOT NULL,
  Branch_name VARCHAR(20) NOT NULL,
  Manager_ID INT,
  Branch_Address VARCHAR(255) NOT NULL,
  Branch_Phone_Number INT UNIQUE NOT NULL,
  Branch_Email VARCHAR(100) UNIQUE NOT NULL,
  Shiftpattern_ID INT,
  FOREIGN KEY (Manager_ID) REFERENCES Manager (Manager_ID),
  FOREIGN KEY (Shiftpattern_ID) REFERENCES Shift_Pattern (Shift_Pattern_ID));
```

Staff

```
CREATE TABLE Staff (
  Staff_ID INT PRIMARY KEY UNIQUE NOT NULL,
  Shiftpattern_ID INT,
  Employee_ID INT,
  FOREIGN KEY (Shiftpattern_ID) REFERENCES Shift_Pattern(Shift_Pattern_ID),
  FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID));
```

Product Category

```
CREATE TABLE ProductCategory (
  Category_ID INT PRIMARY KEY UNIQUE NOT NULL,
  Category_Name VARCHAR(20) NOT NULL);
```

Product

```
CREATE TABLE Product (
  Product_ID INT PRIMARY KEY UNIQUE NOT NULL,
  Product_Description TEXT NOT NULL,
  Size TEXT NOT NULL,
  Composition TEXT NOT NULL,
  Price DECIMAL(10, 2) NOT NULL CHECK (Price > 0),
  Category_ID INT,
  FOREIGN KEY (Category_ID) REFERENCES ProductCategory(Category_ID));
```

Inventory

```
CREATE TABLE Inventory (
  Inventory_ID INTEGER PRIMARY KEY UNIQUE NOT NULL,
  Product_ID INTEGER NOT NULL,
  Branch_ID INTEGER NOT NULL,
  Stock_Quantity INTEGER NOT NULL,
  Restock_Date DATE NOT NULL,
  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
  FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID));
```

Customer

```
CREATE TABLE Customer (
  Customer_ID INT PRIMARY KEY UNIQUE NOT NULL,
  First_Name VARCHAR(20) NOT NULL,
  Last_Name VARCHAR(20) NOT NULL,
  Email VARCHAR(100) UNIQUE NOT NULL,
  Phone_Number VARCHAR(15) UNIQUE,
  City VARCHAR(50),
  Account_Registered BOOLEAN NOT NULL);
```

Order

```
CREATE TABLE "Order" (
  Order_ID INT PRIMARY KEY UNIQUE NOT NULL, O
  Order_Date DATE NOT NULL,
  Order_Total DECIMAL(10, 2) NOT NULL CHECK(Order_Total > 0),
  Customer_ID INT NOT NULL,
  Branch_ID INT NOT NULL,
  Product_ID INT NOT NULL,
  Quantity INT NOT NULL CHECK (Quantity > 0),
  FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID),
  FOREIGN KEY (Branch_ID) REFERENCES Branch (Branch_ID),
  FOREIGN KEY (Product_ID) REFERENCES Product (Product_ID));
```

Payment

```
CREATE TABLE Payment (
  Payment_ID INT PRIMARY KEY UNIQUE NOT NULL,
  Order_id INT,
  Amount_Paid DECIMAL(10, 2) NOT NULL,
  Payment_Mode CHAR(20) NOT NULL,
  Payment_Date DATE NOT NULL,
  FOREIGN KEY (Order_id) REFERENCES "Order" (Order_ID));
```

Delivery

```
CREATE TABLE Delivery (
  Delivery_ID INT PRIMARY KEY UNIQUE NOT NULL,
  Delivery_Address VARCHAR(255) NOT NULL,
  Delivery_Mode VARCHAR(100) NOT NULL,
  Delivery_Date DATE NOT NULL,
  Order_ID INT NOT NULL,
  FOREIGN KEY (Order_id) REFERENCES "Order" (Order_ID));
```

Task T4: SQL Queries

Query One:

```
SELECT
        Branch.Branch_Name
        Customer.Customer_ID,
        CONCAT(Customer.First_Name, ' ', Customer.Last_Name) AS Full_Name,
        COUNT("Order".Order_ID) AS Total_Order,
        "Order".Order_Date,
        SUM(Payment.Amount_Paid) AS Total_Spent
FROM
        Customer
JOIN
        "Order" ON Customer.Customer_ID = "Order".Customer_ID
JOIN
        Branch ON Branch.Branch_ID = "Order".Branch_ID
JOIN
        Payment ON "Order".Order_ID = Payment.Order_ID
WHERE
         "Order".Order_Date BETWEEN '2024-01-01' AND '2024-12-31'
GROUP BY
        Branch.Branch_Name, Customer.Customer_ID, Customer.First_Name,
        Customer.Last_Name, "Order".Order_Date
        ORDER BY
           Total_Spent DESC;
```

Description: This query generates basic statistics on customers per city for a specific time period by recording customer's order activity at each branch. Such statistics include customer's full name, total orders they have placed and total paid for each order within the year of 2024. Data is grouped by customer, branch, a particular order date, with the output listed in an descending order of total income; doing so allows the company to identify highest-spending customers across locations.

Query Two:

```
SELECT
        Product.Product_ID,
        Product.Product_Description,
        Product.Size,
        Product.Composition,
        Product.Price,
        ProductCategory.Category_Name
        FROM
```

```
        Product
JOIN
        ProductCategory ON Product.Category_ID =   ProductCategory.Category_ID
ORDER BY
        ProductCategory.Category_Name ASC,
        Product.Price ASC,
        Product.Size ASC;
```

Description: This query provides information regarding all products produced by FM Clothing Store with description and prices included. Information about each product includes its ID number, description, size, composition, category name and price. By joining 'Product' to the 'Product Category' table to include category name for each product as well as sorting data in ascending order by category name, price per category and product size, the company can evaluate products by category, price, size and composition more efficiently.

Query Three:

```
SELECT
   "Order".Order_ID,
   "Order".Order_Date,
   "Order".Order_Total,
   Delivery.Delivery_Address,
   Delivery.Delivery_Mode,
   Delivery.Delivery_Date
FROM
   "Order"
JOIN
   Delivery ON "Order".Order_ID = Delivery.Order_ID;
```

Description: This query generates data on order record and its associated delivery details. It gathers information regarding order ID, date of order, amount paid for order, delivery address, delivery mode and delivery date. The query combines the 'Order' and 'Delivery' tables to generate order/delivery details in a single output, a more efficient and easier method to track each order purchased alongside its subsequent delivery information.

Query Four:

```
SELECT
   Product.Product_ID,
   Product.Product_Description,
```

```
    Branch.Branch_Name,
    Inventory.Stock_Quantity,
    Inventory.Restock_Date
FROM
    Inventory
JOIN
    Product ON Inventory.Product_ID = Product.Product_ID
JOIN
    Branch ON Inventory.Branch_ID = Branch.Branch_ID
WHERE
    Inventory.Stock_Quantity >= 0;
```

Description: This query displays product availability and their location, providing product inventory details for every branch, consisting of the product ID, branch name, stock quantity and restock date. By joining 'Inventory' table to 'Product' and 'Branch' tables, it clearly presents which products are in stock at which branch. Attention to stock quantity is further enforced by the 'WHERE' clause to filter for products with stock amount equal to or greater than zero, ensuring items out of stock are not listed.

Query Five:

```
SELECT
    Branch.Branch_Name,
    TO_CHAR("Order".Order_Date, 'YYYY-MM') AS Month,
    SUM(Payment.Amount_Paid) AS Total_Income
FROM
    Branch
JOIN
    "Order" ON Branch.Branch_ID = "Order".Branch_ID
JOIN
    Payment ON "Order".Order_ID = Payment.Order_ID
GROUP BY
    Branch.Branch_Name,
    Month
ORDER BY
    Month,
    Branch.Branch_Name;
```

Description: This query retrieves the monthly income generated by each branch. Joining the 'Branch,' 'Order,' and 'Payment' tables combine branch names, monthly income and order details in one view.

Demonstration

Query 1 Screenshot

```
fmstore=# SELECT
fmstore-#     Branch.Branch_Name,
fmstore-#     Customer.Customer_ID,
fmstore-#     CONCAT(Customer.First_Name, ' ', Customer.Last_Name) AS Full_Name,
fmstore-#     COUNT("Order".Order_ID) AS Total_Order,
fmstore-#     "Order".Order_Date,
fmstore-#     SUM(Payment.Amount_Paid) AS Total_Spent
fmstore-# FROM
fmstore-#     Customer
fmstore-# JOIN
fmstore-#     "Order" ON Customer.Customer_ID = "Order".Customer_ID
fmstore-# JOIN
fmstore-#     Branch ON Branch.Branch_ID = "Order".Branch_ID
fmstore-# JOIN
fmstore-#     Payment ON "Order".Order_ID = Payment.Order_ID
fmstore-# WHERE
fmstore-#     "Order".Order_Date BETWEEN '2024-01-01' AND '2024-12-31'
fmstore-# GROUP BY
fmstore-# Branch.Branch_Name, Customer.Customer_ID, Customer.First_Name,  Customer.Last_Name, "Order".Order_Date
fmstore-# ORDER BY
fmstore-#     Total_Spent DESC;
  branch_name  | customer_id |     full_name     | total_order | order_date | total_spent
---------------+-------------+-------------------+-------------+------------+-------------
 Portsmouth    |       10006 | Terrye Caddens    |           1 | 2024-12-12 |     1200.29
 Havant        |       10007 | Clint Breache     |           1 | 2024-10-14 |      949.89
 Portsmouth    |       10001 | Steffane Stachini |           1 | 2024-02-20 |      921.29
 Chichester    |       10004 | Inga Yesichev     |           1 | 2024-06-12 |      762.56
 Havant        |       10004 | Inga Yesichev     |           2 | 2024-07-10 |      754.67
 Fareham       |       10003 | Georgeanna Jagson |           1 | 2024-04-25 |      678.86
 Gosport       |       10001 | Steffane Stachini |           1 | 2024-11-01 |      514.43
 Waterlooville |       10003 | Georgeanna Jagson |           1 | 2024-03-14 |      467.35
 Gosport       |       10010 | Grantley Mannie   |           1 | 2024-09-03 |      456.79
 Havant        |       10006 | Terrye Caddens    |           1 | 2024-08-10 |      394.88
 Gosport       |       10009 | Ophelie Charlot   |           1 | 2024-10-17 |      360.29
 Chichester    |       10005 | Lennard Snow      |           1 | 2024-07-04 |      264.44
 Fareham       |       10002 | Chester Dicey     |           1 | 2024-05-25 |      180.01
 Waterlooville |       10002 | Chester Dicey     |           1 | 2024-03-20 |      160.29
 Gosport       |       10008 | Jayme Blunkett    |           1 | 2024-01-19 |      137.27
(15 rows)
```

Query 2 Screenshot

```
fmstore=# SELECT
fmstore-# Product.Product_ID,
fmstore-# Product.Product_Description,
fmstore-# Product.Size,
fmstore-# Product.Composition,
fmstore-# Product.Price,
fmstore-# ProductCategory.Category_Name
fmstore-# FROM
fmstore-# Product
fmstore-# JOIN
fmstore-# ProductCategory ON Product.Category_ID =   ProductCategory.Category_ID
fmstore-# ORDER BY
fmstore-# ProductCategory.Category_Name ASC,
fmstore-# Product.Price ASC,
fmstore-# Product.Size ASC;
 product_id |    product_description    | size |                              composition                               | price | category_name
------------+---------------------------+------+------------------------------------------------------------------------+-------+---------------
     777567 | Beauty Hydrating Serum    | L    | Water 65% - Glycerin 15% - Hyaluronic Acid 5% - Vitamin C 2% - Other 13% | 19.80 | Beauty
     666901 | Velvet Cushion Cover      | S    | EVA 10%                                                                |  9.99 | Home
     888345 | Rattan Laundry Basket     | M    | Rattan 100%                                                            | 18.90 | Home
     505234 | Knitted Baby Blanket      | XL   | Polyester 85% - Nylon 15%                                              | 14.75 | Kids
     789789 | Kids' Denim Overalls      | M    | Acrylic 50% - Polyester 38% - Wool 12%                                 | 20.00 | Kids
     123123 | Padded Ski Jacket         | M    | Acrylic 45%- Nylon 35%- Wool 20%                                       | 10.50 | Men
     101012 | Men's Casual Chinos       | M    | Cotton 98% - Spandex 2%                                                | 12.25 | Men
     999678 | Sporty Running Shoes      | M    | Polyester 60% - Rubber 30%                                             | 22.50 | Sport
     234890 | Faux Leather Biker Jacket | S    | Cotton 80% - Acrylic 20%                                               | 11.50 | Women
     456456 | Luxe Silk Scarf           | XS   | Silk 100%                                                              | 15.75 | Women
(10 rows)
```

Query 3 Screenshot

```
fmstore=# SELECT
fmstore-#     "Order".Order_ID,
fmstore-#     "Order".Order_Date,
fmstore-#     "Order".Order_Total,
fmstore-#     Delivery.Delivery_Address,
fmstore-#     Delivery.Delivery_Mode,
fmstore-#     Delivery.Delivery_Date
fmstore-# FROM
fmstore-#     "Order"
fmstore-# JOIN
fmstore-#     Delivery ON "Order".Order_ID = Delivery.Order_ID;
 order_id | order_date | order_total |     delivery_address     |  delivery_mode  | delivery_date
----------+------------+-------------+--------------------------+-----------------+---------------
     4567 | 2024-11-01 |      514.43 | Manchester High street   | delivery        | 2024-11-25
     1230 | 2024-03-14 |      180.01 | Commercial Road          | delivery        | 2024-03-18
     7891 | 2024-05-25 |      264.44 | High Street -street 2    | in store pickup | 2024-06-02
     6789 | 2024-07-10 |      599.04 | Derby road street 3      | in store pickup | 2024-10-19
     6789 | 2024-07-10 |      599.04 | Sheffield Street         | delivery        | 2024-07-27
     7892 | 2024-07-04 |      394.88 | Northbrook road street 4 | in store pickup | 2024-07-26
     8901 | 2024-08-10 |      360.29 | Nottingham Road          | delivery        | 2024-08-29
     1234 | 2024-10-14 |      949.89 | Bristol Street 4         | delivery        | 2024-10-29
     2346 | 2024-01-19 |      137.27 | Saint Nicholas street 5  | in store pickup | 2024-10-30
     7893 | 2024-10-17 |      155.63 | Leeds Street 3           | delivery        | 2024-09-03
    54321 | 2024-02-20 |      921.29 | Manchester High Street   | delivery        | 2024-02-22
    91234 | 2024-03-20 |      160.29 | Commercial Road          | delivery        | 2024-03-22
    45075 | 2024-06-12 |      762.56 | Sheffield Street         | delivery        | 2024-06-14
    13486 | 2024-09-03 |      456.79 | Northbrook Road Street 4 | in store pickup | 2024-09-05
   234354 | 2024-12-12 |     1200.29 | Nottingham Road          | delivery        | 2024-12-14
    89034 | 2024-04-25 |      678.86 | High Street - Street 2   | in store pickup | 2024-04-27
(16 rows)
```

Query 4 Screenshot

```
fmstore=# SELECT
fmstore-#     Product.Product_ID,
fmstore-#     Product.Product_Description,
fmstore-#     Branch.Branch_Name,
fmstore-#     Inventory.Stock_Quantity,
fmstore-#     Inventory.Restock_Date
fmstore-# FROM
fmstore-#     Inventory
fmstore-# JOIN
fmstore-#     Product ON Inventory.Product_ID = Product.Product_ID
fmstore-# JOIN
fmstore-#     Branch ON Inventory.Branch_ID = Branch.Branch_ID
fmstore-# WHERE
fmstore-#     Inventory.Stock_Quantity >= 0;
 product_id |    product_description    |  branch_name  | stock_quantity | restock_date
------------+--------------------------+---------------+----------------+--------------
     999678 | Sporty Running Shoes      | Waterlooville |            150 | 2024-11-01
     456456 | Luxe Silk Scarf           | Fareham       |            120 | 2024-11-01
     234890 | Faux Leather Biker Jacket | Gosport       |            120 | 2024-11-01
     101012 | Men’s Casual Chinos       | Havant        |              0 | 2024-11-01
     666901 | Velvet Cushion Cover      | Chichester    |             80 | 2024-11-01
     888345 | Rattan Laundry Basket     | Portsmouth    |            130 | 2024-11-01
     123123 | Padded Ski Jacket         | Waterlooville |            200 | 2024-11-01
     789789 | Kids' Denim Overalls      | Fareham       |             80 | 2024-11-01
     777567 | Beauty Hydrating Serum    | Gosport       |            110 | 2024-11-01
     505234 | Knitted Baby Blanket      | Havant        |            150 | 2024-11-01
(10 rows)
```

Query 5 Screenshot

```
fmstore=# SELECT
fmstore-#     Branch.Branch_Name,
fmstore-#     TO_CHAR("Order".Order_Date, 'YYYY-MM') AS Month,
fmstore-#     SUM(Payment.Amount_Paid) AS Total_Income
fmstore-# FROM
fmstore-#     Branch
fmstore-# JOIN
fmstore-#     "Order" ON Branch.Branch_ID = "Order".Branch_ID
fmstore-# JOIN
fmstore-#     Payment ON "Order".Order_ID = Payment.Order_ID
fmstore-# GROUP BY
fmstore-#     Branch.Branch_Name,
fmstore-#     Month
fmstore-# ORDER BY
fmstore-#     Month,
fmstore-#     Branch.Branch_Name;
  branch_name   |  month  | total_income
----------------+---------+--------------
 Gosport        | 2024-01 |       137.27
 Portsmouth     | 2024-02 |       921.29
 Waterlooville  | 2024-03 |       627.64
 Fareham        | 2024-04 |       678.86
 Fareham        | 2024-05 |       180.01
 Chichester     | 2024-06 |       762.56
 Chichester     | 2024-07 |       264.44
 Havant         | 2024-07 |       754.67
 Havant         | 2024-08 |       394.88
 Gosport        | 2024-09 |       456.79
 Gosport        | 2024-10 |       360.29
 Havant         | 2024-10 |       949.89
 Gosport        | 2024-11 |       514.43
 Portsmouth     | 2024-12 |      1200.29
(14 rows)
```

Reflection

Developing the FM Clothing Store database provided me the opportunity to apply newly learnt PostgreSQL theoretical knowledge in a practical setting. The primary task was to design a relational database to centralise this store's operations. We achieved this by creating an EERD, data dictionary - aiding in creating table queries - followed by queries to generate reports on various measures. My background in PostgreSQL is very little to none, although the design of this database allowed me to understand PSQL quite in depth. Nevertheless, after attending various university lectures/workshops and online courses, it was very exciting and exhilarating to be able to showcase this new knowledge in creating a functional database.

To complete this project, I collaborated with two group members. Given that we all have little experience with PSQL, we deemed it most appropriate to delegate the first three tasks evenly then come together to edit and finalise each task. We chose to perform task 4 together in generating the queries since it is a large proportion of marks in the report. My role was highlighted mainly in the data dictionary. I valued my role in this task as it gave me the opportunity to thoroughly learn about data types and sizes and the various constraints and domains existing in PSQL. Regular check-ins among the group allowed us to address errors in each task, for instance, identifying where additional foreign keys were needed to strengthen the integrity and efficiency of the database, for instance, allowing me to amend the data dictionary accordingly.

To select queries, we focused on meeting the operational needs of the company. The first query aimed to produce customer statistics which helps the company in their targeted marketing strategies towards locations with fewer orders. The second focused on summarising stock quantity, product types and categories – crucial to inventory management. To help the company monitor the efficiency of its delivery, it was only appropriate to generate a query for order and delivery details. Furthermore, to give an overview of inventory levels across branches, we produced a query regarding product availability, and finally, the last query intended on assisting the store in monitoring profitability across branches by calculating monthly income by location.

Overall, this project enhanced my skill and practical experience in database management via PSQL.