UNIVERSITY OF PORTSMOUTH

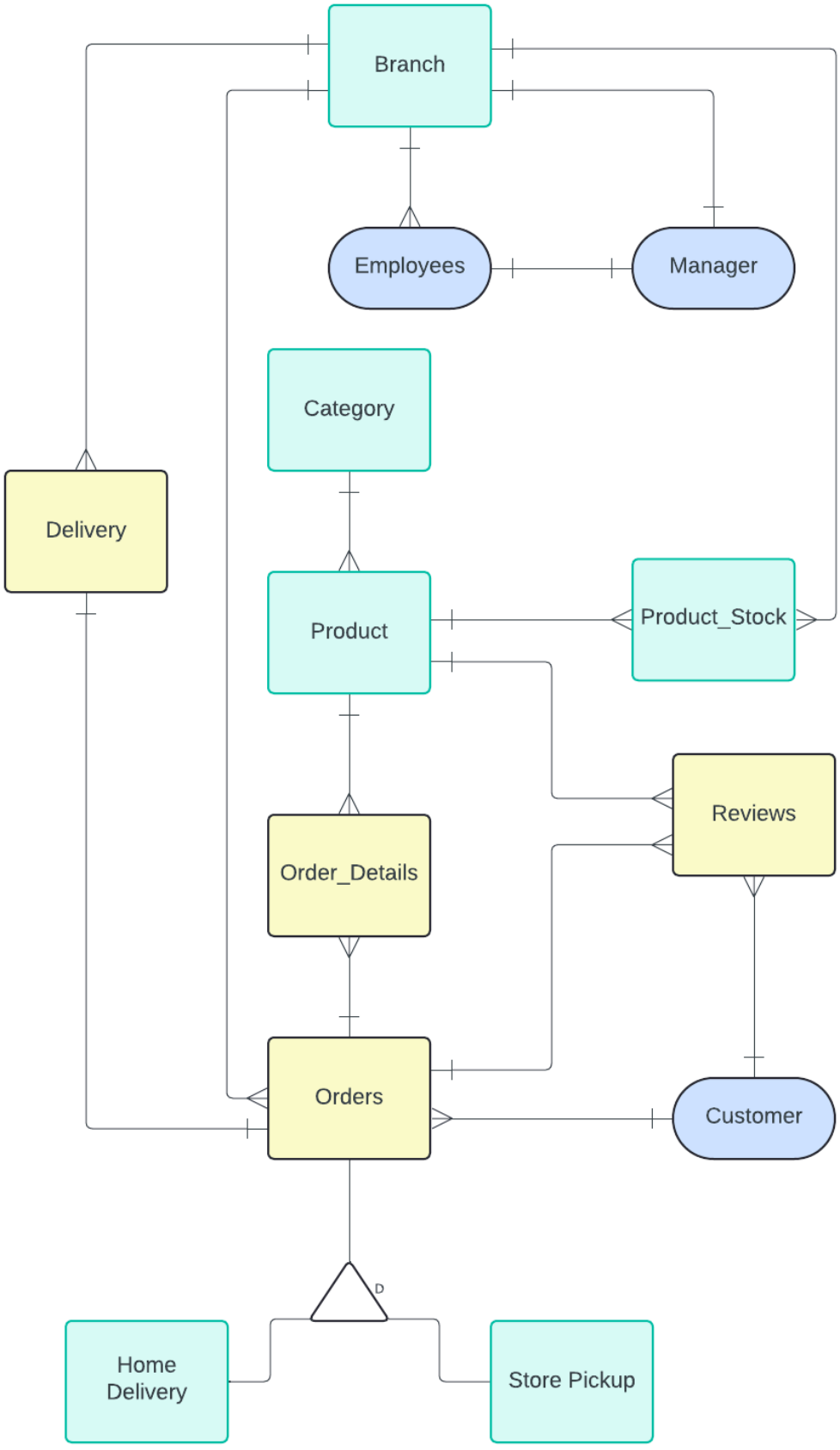
COURSEWORK 1
SUBJECT- DATA MANAGMENT

TOPIC- DESIGN AND DEVELOPMENT OF DATABASE FOR FM CLOTHING STORE USING POSGRESQL

Submitted By:

- **UP2280648**
- **UP2298397**
- **UP2299529**

Task 1: EERD (Enhanced Entity Relationship Diagram)



Explanation of the relationships shown between the entities in the above EERD (Enhanced Entity Relationship Diagram)

Branch-Employees (one-many)

- The relationship is one-many because one branch will have multiple employees but each employee will work in one specific branch.

Branch-Manager (one-one)

- The relationship is one-one because each branch will have only one manager in the branch table and each manager in the manager table will oversee only one branch.

Category-Product (one-many)

- The relationship is one-many because one category will have multiple products but each product will belong to one particular category.

Customer-Orders (one-many)

- The relationship is one-many because one particular customer can place many orders at different points of time, but each order is always linked to only one specific customer.

Branch-Orders (One-Many)

- The relationship is one-many because one branch can handle multiple orders whether store pickup or home delivery but each order is placed at only one specific branch.

Orders-Order_Details (one-many)

- The relationship is one-many because one order can have multiple products which are showcased in the Order_Details table but each entry in the Order_Details table corresponds to one specific product within a particular order.

Product-Order_Details (one-many)

- The relationship is one-many because one product can appear in multiple orders showcased by the order_details table but each record in order_details refers to only one specific product.

Orders-Product (many-many via Order_Details)

- The relationship is many-many because one order can contain multiple products and any product can be in multiple orders. The intersection table for this many-many relation is the Order_Details table.

Manager-Employees (one-one)

- The relationship is **one-to-one** because each manager is related with only one employee record in the **Employees** table which uniquely represents his role as a manager.

Orders-Delivery (one-one)

- The relationship is one-one because one order will have only one delivery and each delivery is linked to a particular order.

Product-Branch (many-many via Product_Stock table)

- The relationship is indeed a many-many relation because each branch can stock many products and the same product can be available at different branches. The product_stock table helps us track how many products of each type are available at each branch. The intersection table for this many-many relation is the Product_Stock table.

Product-Product_Stock (one-many)

- The above relationship is one-many because one product can be stocked in five different branches which means that a single product will have many entries in the Product_Stock table but each record in the Product_Stock table is compulsorily related to only one particular product.

Product_Stock-Branch (many-one)

- The relationship is many-one because one branch can stock multiple products, which means that each branch will have many entries in the Product_Stock table but for each entry in Product_Stock, there is only one branch.

Reviews-Product (Many-One)

- The relationship is many-one because a product can have many reviews because there will be many customers who have purchased the product but each review is linked to a particular product.

Reviews-Customer (many-one)

- The relationship is many-one because many reviews can be written by a single customer but each review is always linked to only a specific customer who has given the review about a certain product purchased by the customer.

Reviews-Orders (Many-one)

- The relationship is many-one because an order placed by a customer can have many reviews because the order may have more than one product but every review is always linked to one specific order.

Task 2: Rationale and Assumptions

1. The 'Branch' table in our database contains information about the five branches, namely FM Waterlooville Clothing Store, FM Fareham Clothing Store, FM Gosport Clothing Store, FM Havant Clothing Store, and FM Chichester Clothing Store, and all these branches handle both in-store pickup and online orders. We have assumed a decentralised system of branches, where there is no specific inventory or branch to deal with online orders. All five branches handle both types of orders (in-store pickup and online), and for online orders, the products are shipped from the particular branch where the order is placed. The Portsmouth head office is not included in the **'**Branch**'** entity because it only tracks and monitors sales reports and operations for all five branches and it doesn't handle orders directly.
2. Each branch has multiple employees, and we have taken into consideration that every employee works at only one specific branch, following a shift pattern—morning, evening, night, and hybrid. Each branch has a manager who oversees all the employees working at that particular branch/location, and the manager is considered an employee as well. There are a total of five managers, one for each branch. The manager entity holds information related to all the managers.
3. The products available across all the branches fall into seven main categories: Women, Men, Baby, Kids, Beauty, Sports, and Home Products. All important information related to the product, including the category in which the product is falling under, is stored in the 'Products' table. We have assumed 11 different types of products in our database falling in one of the seven mentioned categories. We have assumed that the price of a particular product remains same across all the five different branches.
4. All necessary informations related to customers are stored in the 'Customer' table. Only those customer details are recorded in the table who have placed an order either in-store pickup or online. The registration of the customer takes place just after the customer places an order whether in-person shopping or online order.
5. The 'Orders' table present in our database tracks the orders placed by customers. It includes the date on which the order is placed (both online or in-store pickup), the total amount related to that particular order, the method of delivery whether it is in-store or

online, and the branch in which the order is placed. One customer can place multiple orders and an order can have one or more than one product.

6. The Order_Details table in our database breaks down every order into the products the order contains along with the price and quantity of each product present in an order.
7. Both cash and online payment options are available, and these are tracked in the 'Orders' table. We have not included the option for payment in installment or late payment in our database to streamline the transaction process of the retailer.
8. To track the availability of products across the five main branches, a Product_Stock table is created, which records the available quantity of each product at every branch. This will help the retailer be aware of products that are in low stock.
9. For online orders, the Delivery table tracks the entire delivery process.
10. Lastly, a Review table has been created to capture customer feedback on products they've purchased, either in-store or online, to help improve the quality of products sold.

Task T3: Data Dictionary/ Scripts

1. **Data Dictionary for the 'Branch' Table:**

Branch					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Branch_ID	PK	VARCHAR (10)	PRIMARY KEY		Unique key identifying each branch
Branch_Name	AK	VARCHAR (100)	UNIQUE, NOT NULL		Name of the branch
City		VARCHAR (100)	NOT NULL		City where the branch is situated
Address		VARCHAR (300)	NOT NULL		Detail address of the branch

2. **Data Dictionary for the 'Employees' Table:**

Employees					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Employee_ID	PK	VARCHAR (10)	PRIMARY KEY		Unique key identifying each employee
First_Name		VARCHAR (100)	NOT NULL		Employee's first name
Last_Name		VARCHAR (100)	NOT NULL		Employee's last name
Designation		VARCHAR (100)	NOT NULL		Employee's job title
Salary_GBP		DECIMAL (6,2)	NOT NULL, CHECK(Salary_GBP>0)		Salary of the employee
Date_Join		DATE	NOT NULL		Date when the employee joined
Shift		VARCHAR (50)	NOT NULL, CHECK (Shift IN ('Morning', 'Evening', 'Hybrid', 'Night'))		Employee's working shift
Branch_ID	FK	VARCHAR (10)	NOT NULL	Branch (Branch_ID)	Branch where the employee works

3. **Data Dictionary for the 'Manager' Table:**

Manager					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Manager_ID	PK	VARCHAR (10)	PRIMARY KEY		Unique key identifying each manager
Employee_ID	FK	VARCHAR (10)	NOT NULL	Employees (Employee_ID)	Refers to the Employe_ID of the Employee table
Branch_ID	FK	VARCHAR (10)	NOT NULL	Branch (Branch_ID)	Branch where the manager works; refers to the Branch_ID of the Branch table

4. **Data Dictionary for the 'Category' Table:**

Category					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Category_ID	PK	VARCHAR (10)	PRIMARY KEY		Unique key identifying each category
Category_Name		VARCHAR (100)	NOT NULL		Name of the category

5. **Data Dictionary for the 'Product' Table:**

Product					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Product_ID	PK	VARCHAR (10)	PRIMARY KEY		Unique key identifying each product
Product_Description		VARCHAR (100)	NOT NULL		Detailed description of the product
Size		VARCHAR (10)	NOT NULL		Size of the product
Composition		VARCHAR (200)			Material composition of the product
Price		DECIMAL (10,2)	NOT NULL, Price>0		Price of the product in GBP
Category_ID	FK	VARCHAR (10)	NOT NULL	Category (Category_ID)	Category the product belongs to; refers to the Category_ID of the Category table

6. **Data Dictionary for the 'Customer' Table:**

Customer					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Customer_ID	PK	VARCHAR (10)	PRIMARY KEY		Unique key identifying each customer
First_Name		VARCHAR (100)	NOT NULL		First name of the customer
Last_Name		VARCHAR (100)	NOT NULL		Last name of the customer
Email		VARCHAR (100)			Email address of the customer
Phone		VARCHAR (20)	NOT NULL, VALID FOR 10 DIGITS		Phone number of the customer
Address		VARCHAR (300)	NOT NULL		Address of the customer
City		VARCHAR (100)	NOT NULL		City where the customer lives
Reg_Date		DATE	NOT NULL		Date when the customer registered

7. **Data Dictionary for the ‘Orders’ Table:**

Orders					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Order_ID	PK	VARCHAR (10)	PRIMARY KEY		Unique key identifying each order placed
Order_Date		DATE	NOT NULL		Date when the order is placed
Total_Amount		DECIMAL (10,2)	NOT NULL, Total_Amount> 0		Total amount of the order in GBP
Payment_Method		VARCHAR (50)	NOT NULL, Limited to ('Online', 'Cash')		Payment method used for the order
Delivery_Method		VARCHAR (50)	NOT NULL, Limited to ('Home Delivery', 'Store Pickup')		Delivery method for the order
Customer_ID	FK	VARCHAR (10)	NOT NULL	Customer (Customer_ID)	Customer who placed the order; refers to the Customer_ID of the Customer table
Branch_ID	FK	VARCHAR (10)	NOT NULL	Branch (Branch_ID)	Branch in which the order is placed

8. **Data Dictionary for the 'Order_Details' Table:**

Order_Details					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Order_ID	PK, FK	VARCHAR (10)	NOT NULL, Part of Composite Primary Key	Orders (Order_ID)	Key that links to the primary key of the Orders table that uniquely identify each order
Product_ID	PK, FK	VARCHAR (10)	NOT NULL, Part of Composite Primary Key	Product (Product_ID)	Key that links to the primary key of the Product table that uniquely identifies each product
Quantity		INT	NOT NULL		Quantity of the product ordered
Price_Per_Item		DECIMAL (10,2)	Price_Per_Item >0		Price of one product in GBP

9. Data Dictionary for the 'Product_Stock' Table:

Product_Stock					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Product_ID	PK, FK	VARCHAR (10)	NOT NULL, Part of Composite Primary Key	Product (Product_ID)	Key that links to the primary key of the Product table that uniquely identifies each product
Branch_ID	PK, FK	VARCHAR (10)	NOT NULL, Part of Composite Primary Key	Branch (Branch_ID)	Key that links to the primary key of the Branch table that uniquely identifies each branch
Stock_Quantity		INT	NOT NULL		Quantity of the product in stock

10.Data Dictionary for the ‘Delivery’ Table:

Delivery					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Delivery_ID	PK	VARCHAR (10)	PRIMARY KEY		Unique key identifying each delivery
Order_ID	FK	VARCHAR (10)	NOT NULL	Orders (Order_ID)	Key that links to the primary key of the Orders table that uniquely identify each order
Delivery_Date		DATE	NOT NULL		Date on which the order is delivered
Delivery_Address		VARCHAR (300)	NOT NULL		Address where the order is delivered
Delivery_Status		VARCHAR (50)	NOT NULL, LIMITED TO ('Delivered', 'Pending'))		Status of the delivery
Branch_ID	FK	VARCHAR (10)	NOT NULL	Branch (Branch_ID)	Key that links to the primary key of the Branch table that uniquely identifies each branch

11. Data Dictionary for the ‘Reviews’ Table:

Reviews					
Attribute Name	**PK or AK?**	**Data Type & Size**	**Domain and Constraints**	**FK Reference**	**Description (where non-obvious)**
Review_ID	PK	VARCHAR (10)	PRIMARY KEY		Unique key identifying each review
Product_ID	FK	VARCHAR (10)	NOT NULL	Product (Product_ID)	Product being reviewed
Customer_ID	FK	VARCHAR (10)	NOT NULL	Customer (Customer_ID)	Customer who gave the review
Order_ID	FK	VARCHAR (10)	NOT NULL	Orders (Order_ID)	Order to which the review is related
Rating		DECIMAL (2,1)	NOT NULL, CHECK (Rating BETWEEN 1 AND 5)		Rating given by the customer
Review_Note		VARCHAR (50)	NOT NULL		Comments or feedback related to the order
Review_Date		DATE	NOT NULL		Date on which the review is given

Task T4: SQL Queries

1. Queries to create the necessary tables in our database

1.1 Query to create the 'Branch' Table

```
up2280648=# create database fm_cloth_store;
CREATE DATABASE
up2280648=# \c fm_cloth_store
You are now connected to database "fm_cloth_store" as user "up2280648".
fm_cloth_store=#
fm_cloth_store=# -- Branch Table
CREATE TABLE Branch (-- a branch table is created
Branch_ID VARCHAR(10) PRIMARY KEY,-- Branch_ID column is added which is a primary key
Branch_Name VARCHAR(100) UNIQUE NOT NULL,
City VARCHAR(100) NOT NULL,
Address VARCHAR(300) NOT NULL
);
CREATE TABLE
```

1.2 Query to create the 'Employees' Table

```
fm_cloth_store=# -- Employees Table
CREATE TABLE Employees (
Employee_ID VARCHAR(10) PRIMARY KEY,
First_Name VARCHAR(100) NOT NULL,
Last_Name VARCHAR(100) NOT NULL,
Designation VARCHAR(100) NOT NULL,
Salary_GBP DECIMAL (6,2) NOT NULL CHECK (Salary_GBP > 0),
Date_Join DATE NOT NULL,
Shift VARCHAR(50) NOT NULL CHECK (Shift IN ('Morning', 'Evening', 'Hybrid', 'Night')),
Branch_ID VARCHAR(10) NOT NULL,
FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID)
);
CREATE TABLE
```

1.3 Query to create the 'Manager' Table

```
fm_cloth_store=# -- Manager Table
CREATE TABLE Manager (
Manager_ID VARCHAR(10) PRIMARY KEY,
Employee_ID VARCHAR(10) NOT NULL,
Branch_ID VARCHAR(10) NOT NULL,
FOREIGN KEY (Employee_ID) REFERENCES Employees(Employee_ID),
FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID)
);
CREATE TABLE
```

1.4 Query to create the 'Category' Table

```
fm_cloth_store=# -- Category Table
CREATE TABLE Category (
Category_ID VARCHAR(10) PRIMARY KEY,
Category_Name VARCHAR(100) NOT NULL
);
CREATE TABLE
```

1.5 Query to create the 'Product' Table

```
fm_cloth_store=# -- Product Table
CREATE TABLE Product (
Product_ID VARCHAR(10) PRIMARY KEY,
Product_Description VARCHAR(100) NOT NULL,
Size VARCHAR(10) NOT NULL,
Composition VARCHAR(200),
Price DECIMAL(10, 2) NOT NULL CHECK (Price > 0),
Category_ID VARCHAR(10) NOT NULL,
FOREIGN KEY (Category_ID) REFERENCES Category(Category_ID)
);
CREATE TABLE
```

1.6 Query to create the 'Customer' Table

```
fm_cloth_store=# -- Customer Table
CREATE TABLE Customer (
Customer_ID VARCHAR(10) PRIMARY KEY,
First_Name VARCHAR(100) NOT NULL,
Last_Name VARCHAR(100) NOT NULL,
Email VARCHAR(100),
Phone VARCHAR(20) NOT NULL CHECK (Phone ~ '^\d{10}$'),
Address VARCHAR(300) NOT NULL,
City VARCHAR(100) NOT NULL,
Reg_Date DATE NOT NULL
);
CREATE TABLE
```

1.7 Query to create the 'Orders' Table

```
fm_cloth_store=# -- Orders Table
CREATE TABLE Orders (
Order_ID VARCHAR(10) PRIMARY KEY,
Order_Date DATE NOT NULL,
Total_Amount DECIMAL(10, 2) NOT NULL CHECK (Total_Amount > 0),
Payment_Method VARCHAR(50) NOT NULL CHECK (Payment_Method IN ('Online', 'Cash')),
Delivery_Method VARCHAR(50) NOT NULL CHECK (Delivery_Method IN ('Home Delivery', 'Store Pickup')),
Customer_ID VARCHAR(10) NOT NULL,
Branch_ID VARCHAR(10) NOT NULL,
FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID),
FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID)
);
CREATE TABLE
```

1.8 Query to create the 'Order_Details' Table

```
fm_cloth_store=# -- Order_Details Table
CREATE TABLE Order_Details (
Order_ID VARCHAR(10) NOT NULL,
Product_ID VARCHAR(10) NOT NULL,
Quantity INT NOT NULL,
Price_Per_Item DECIMAL(10, 2) NOT NULL CHECK (Price_Per_Item > 0),
PRIMARY KEY (Order_ID, Product_ID),
FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);
CREATE TABLE
```

1.9 Query to create the 'Product_Stock' Table

```
fm_cloth_store=# -- Product_Stock Table
CREATE TABLE Product_Stock (
Product_ID VARCHAR(10) NOT NULL,
Branch_ID VARCHAR(10) NOT NULL,
Stock_Quantity INT NOT NULL,
PRIMARY KEY (Product_ID, Branch_ID),
FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID)
);
CREATE TABLE
```

1.10 Query to create the 'Delivery' Table

```
fm_cloth_store=# -- Delivery Table
CREATE TABLE Delivery (
Delivery_ID VARCHAR(10) PRIMARY KEY,
Order_ID VARCHAR(10) NOT NULL,
Delivery_Date DATE NOT NULL,
Delivery_Address VARCHAR(300) NOT NULL,
Delivery_Status VARCHAR(50) NOT NULL CHECK (Delivery_Status IN ('Delivered', 'Pending')),
Branch_ID VARCHAR(10) NOT NULL,
FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID)
);
CREATE TABLE
```

1.11 Query to create the 'Reviews' Table

```
fm_cloth_store=# --Reviews Table
CREATE TABLE Reviews (
Review_ID VARCHAR(10) PRIMARY KEY,
Product_ID VARCHAR(10) NOT NULL,
Customer_ID VARCHAR(10) NOT NULL,
Order_ID VARCHAR(10) NOT NULL,
Rating INT NOT NULL CHECK (Rating BETWEEN 1 AND 5),
Review_Note VARCHAR(50) NOT NULL,
Review_Date DATE NOT NULL,
FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID),
FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID)
);
CREATE TABLE
```

2. Queries to insert dummy data into the tables created above

2.1 Query to insert dummy data into the 'Branch' Table

```
fm_cloth_store=# INSERT INTO Branch (Branch_ID, Branch_Name, City, Address) VALUES
('001', 'FM Waterlooville Clothing Store', 'Waterlooville', '15 Red Cross Road, Waterlooville, WO1 2SP'),
('002', 'FM Fareham Clothing Store', 'Fareham', '12 Buckingham Street, Fareham, FO2 3ST'),
('003', 'FM Gosport Clothing Store', 'Gosport', 'Lake View Street, Gosport, GO3 2FG'),
('004', 'FM Havant Clothing Store', 'Havant', '4 Harleys Street, Havant, HP1 2FP'),
('005', 'FM Chichester Clothing Store', 'Chichester', 'Harrow Road, Chichester, CH2 4GK');
INSERT 0 5
```

2.2 Query to insert dummy data into the 'Employees' Table

```
fm_cloth_store=# INSERT INTO Employees (Employee_ID, First_Name, Last_Name, Designation, Salary_GBP, Date_Join, Shift, Branch_ID) VALUES
('1001', 'Michael', 'Holding', 'Salesman', '2000', '2023-01-01', 'Morning', '001'),
('1002', 'Lara', 'Todes', 'Salesman', '2000', '2023-01-09', 'Evening', '001'),
('1003', 'Freddie', 'Mercury', 'Manager', '5000', '2023-02-01', 'Hybrid', '002'),
('1004', 'Alfred', 'Tyson', 'Manager', '5000', '2023-02-10', 'Hybrid', '001'),
('1005', 'Brooke', 'Johnson', 'Cashier', '3000', '2023-02-25', 'Morning', '003'),
('1006', 'George', 'Bennington', 'Manager', '5000', '2023-03-01', 'Evening', '005'),
('1007', 'Rohit', 'Sharma', 'Manager', '5000', '2023-04-04', 'Hybrid', '003'),
('1008', 'Kiran', 'Shree', 'Manager', '5000', '2023-05-15', 'Hybrid', '004'),
('1009', 'John', 'Torod', 'Salesman', '2000', '2023-05-20', 'Night', '001');
INSERT 0 9
```

2.3 Query to insert dummy data into the ‘Manager’ Table

```
fm_cloth_store=# INSERT INTO Manager (Manager_ID, Employee_ID, Branch_ID) VALUES
('01', '1004', '001'),
('02', '1003', '002'),
('03', '1007', '003'),
('04', '1008', '004'),
('05', '1006', '005');
INSERT 0 5
```

2.4 Query to insert dummy data into the ‘Category’ Table

```
fm_cloth_store=# INSERT INTO Category (Category_ID, Category_Name) VALUES
('WM01', 'Women'),
('MN02', 'Men'),
('BY03', 'Baby'),
('KD04', 'Kids'),
('BT05', 'Beauty'),
('SP06', 'Sports'),
('HM07', 'Home Products');
INSERT 0 7
```

2.5 Query to insert dummy data into the ‘Product’ Table

```
fm_cloth_store=# INSERT INTO Product (Product_ID, Product_Description, Size, Composition, Price, Category_ID) VALUES
('P001', 'Yellow Hoodie', 'Small', '100% Woolen', 20.00, 'MN02'),
('P002', 'Yellow Hoodie', 'Medium', '100% Woolen', 20.00, 'MN02'),
('P003', 'Black jeans', 'Large', 'Denim', 15.00, 'MN02'),
('P004', 'Red Crop Top', 'Medium', '100% Cotton', 12.00, 'WM01'),
('P005', 'Blue Shirt Dress', 'Large', '70% cotton, 30% Polyester', 11.00, 'WM01'),
('P006', 'Fifa Ball', 'Free', 'Synthetic Leather', 100.00, 'SP06'),
('P007', 'Cricket Bat', 'Free Size', 'Wooden', 70.00, 'SP06'),
('P008', 'Baby pants', 'Small', '100% cotton', 8.00, 'BY03'),
('P009', 'Blue Kids Polo T-Shirt', 'Medium', '40% polyester, 60% cotton', 10.00, 'KD04'),
('P010', 'Beige Sofa cover', 'Free Size', '80% polyester, 20% cotton', 30.00, 'HM07'),
('P011', 'Nike Sneakers', '7', 'Air max sports shoe', 80.00, 'MN02');
INSERT 0 11
```

2.6 Query to insert dummy data into the ‘Customer’ Table

```
fm_cloth_store=# INSERT INTO Customer (Customer_ID, First_Name, Last_Name, Email, Phone, Address, City, Reg_Date) VALUES
('C001', 'Angelo', 'Thomson', 'at123@gmail.com', '0123456789', 'Harrow, LO1 2SP', 'London', '2024-09-01'),
('C002', 'Gary', 'Mehigan', 'gm321@gmail.com', '9874561230', 'South Hall, LO2 3AB', 'London', '2024-09-12'),
('C003', 'Mike', 'Paul', 'mp123@gmail.com', '0147852369', '4th Cross, MN1 3AS', 'Manchester', '2024-09-30'),
('C004', 'Sara', 'Ali', 'sa123@gmail.com', '3698520147', 'Upon Thames, LO2 3PG', 'London', '2024-09-30'),
('C005', 'Tom', 'Cruise', 'tm325@gmail.com', '8520147963', 'UOC Headquarter, CM2 5BJ', 'Cambridge', '2024-10-06'),
('C006', 'Sania', 'Todes', 'st456@gmail.com', '0147258369', 'Central Park, LO4 BG', 'London', '2024-10-12'),
('C007', 'Myke', 'Bennington', 'mb456@gmail.com', '4569871230', 'James Road, MN02 4DK', 'Manchester', '2024-10-25'),
('C008', 'Farah', 'James', 'fj897@gmail.com', '0123654789', 'Red Cross, CM1 2LM', 'Cambridge', '2024-10-30'),
('C009', 'George', 'Jackson', 'gj758@gmail.com', '9874563210', '1st Cross Lake View, MN05 7PL', 'Manchester', '2024-10-31'),
('C010', 'Shakeel', 'Ali', 'sa123@gmail.com', '4560123879', 'Guildhall Hall, PO1 2SP', 'Portsmouth', '2024-10-31');
INSERT 0 10
```

2.7 Query to insert dummy data into the ‘Orders’ Table

```
fm_cloth_store=# INSERT INTO Orders (Order_ID, Order_Date, Total_Amount,Payment_Method,Delivery_Method, Customer_ID, Branch_ID) VALUES
('OD001', '2024-09-01', 35.00,'Online','Home Delivery', 'C001', '001'),
('OD002', '2024-09-05', 20.00,'Cash','Store Pickup', 'C001', '002'),
('OD003', '2024-09-12', 35.00,'Cash','Store Pickup', 'C002', '005'),
('OD004', '2024-09-30', 40.00,'Cash','Store Pickup', 'C003', '003'),
('OD005', '2024-09-30', 18.00,'Online','Home Delivery', 'C004', '004'),
('OD006', '2024-10-06', 170.00,'Online','Home Delivery', 'C005', '005'),
('OD007', '2024-10-12', 35.00,'Online','Home Delivery', 'C006', '001'),
('OD008', '2024-10-25', 10.00,'Cash','Store Pickup', 'C007', '001'),
('OD009', '2024-10-29', 40.00,'Online','Home Delivery', 'C001', '004'),
('OD010', '2024-10-30', 30.00,'Online','Home Delivery', 'C008', '003'),
('OD011', '2024-10-31', 20.00,'Online','Home Delivery', 'C009', '001'),
('OD012', '2024-10-31', 30.00,'Cash','Store Pickup', 'C010', '002');
INSERT 0 12
```

2.8 Query to insert dummy data into the ‘Order_Details’ Table

```
fm_cloth_store=# INSERT INTO Order_Details (Order_ID, Product_ID, Quantity, Price_Per_Item) VALUES
('OD001', 'P002', 1, 20.00),
('OD001', 'P003', 1, 15.00),
('OD002', 'P001', 1, 20.00),
('OD003', 'P001', 1, 20.00),
('OD003', 'P003', 1, 15.00),
('OD004', 'P002', 2, 40.00),
('OD005', 'P008', 1, 8.00),
('OD005', 'P009', 1, 10.00),
('OD006', 'P006', 1, 100.00),
('OD006', 'P007', 1, 70.00),
('OD007', 'P001', 1, 20.00),
('OD007', 'P003', 1, 15.00),
('OD008', 'P009', 1, 10.00),
('OD009', 'P002', 2, 20.00),
('OD010', 'P003', 2, 30.00),
('OD011', 'P009', 2, 20.00),
('OD012', 'P010', 1, 30.00);
INSERT 0 17
```

2.9 Query to insert dummy data into the ‘Product_Stock’ Table

```
fm_cloth_store=# INSERT INTO Product_Stock (Product_ID, Branch_ID, Stock_Quantity) VALUES
('P001', '001', 50),
('P001', '002', 30),
('P001', '003', 10),
('P001', '004', 70),
('P001', '005', 10),
('P002', '001', 30),
('P002', '002', 40),
('P002', '003', 10),
('P002', '004', 20),
('P002', '005', 30),
('P003', '001', 15),
('P003', '002', 32),
('P003', '003', 14),
('P003', '004', 25),
('P003', '005', 32),
('P004', '001', 40),
('P004', '002', 12),
('P004', '003', 10),
('P004', '004', 32),
('P004', '005', 14),
('P005', '001', 29),
('P005', '002', 21),
('P005', '003', 14),
('P005', '004', 23),
('P005', '005', 25),
('P006', '001', 21),
('P006', '002', 10),
('P006', '003', 5),
('P006', '004', 1),
('P006', '005', 36),
('P007', '001', 45),
('P007', '002', 21),
('P007', '003', 21),
('P007', '004', 10),
('P007', '005', 23),
('P008', '001', 25),
('P008', '002', 24),
('P008', '003', 23),
('P008', '004', 25),
('P008', '005', 10),
('P009', '001', 9),
('P009', '002', 5),
('P009', '003', 42),
('P009', '004', 35),
('P009', '005', 12),
('P010', '001', 14),
('P010', '002', 25),
('P010', '003', 25),
('P010', '004', 24),
('P010', '005', 23),
('P011', '001', 21),
('P011', '002', 29),
('P011', '003', 27),
('P011', '004', 28),
('P011', '005', 24);
INSERT 0 55
```

2.10 Query to insert dummy data into the ‘Delivery’ Table

```
fm_cloth_store=# INSERT INTO Delivery (Delivery_ID, Order_ID, Delivery_Date, Delivery_Address, Delivery_Status, Branch_ID) VALUES
('DL01', 'OD001', '2024-09-03', 'Harrow, LO1 2SP', 'Delivered', '001'),
('DL02', 'OD005', '2024-10-03', 'Upon Thames, LO2 3PG', 'Delivered', '004'),
('DL03', 'OD006', '2024-10-08', 'UOC Headquarter, CM2 5BJ', 'Delivered', '005'),
('DL04', 'OD007', '2024-10-14', 'Central Park, LO4 BG', 'Delivered', '001'),
('DL05', 'OD009', '2024-10-30', 'Harrow, LO1 2SP', 'Delivered', '004'),
('DL06', 'OD010', '2024-11-01', 'Red Cross, CM1 2LM', 'Delivered', '003'),
('DL07', 'OD011', '2024-11-02', '1st Cross Lake View, MN05 7PL', 'Delivered', '001');
INSERT 0 7
```

2.11 Query to insert dummy data into the ‘Reviews’ Table

```
fm_cloth_store=# INSERT INTO Reviews (Review_ID, Product_ID, Customer_ID, Order_ID, Rating, Review_Note, Review_Date) VALUES
('RW001', 'P002', 'C001', 'OD001', 4, 'GOOD PRODUCT', '2024-09-05'),
('RW002', 'P001', 'C001', 'OD002', 5, 'EXCELLENT', '2024-09-06'),
('RW003', 'P001', 'C002', 'OD003', 3, 'Quality issue', '2024-09-13'),
('RW004', 'P003', 'C002', 'OD003', 5, 'Superb', '2024-09-13'),
('RW005', 'P003', 'C001', 'OD001', 1, 'Color faded', '2024-09-15'),
('RW006', 'P003', 'C008', 'OD010', 2, 'Color gone', '2024-11-05');

INSERT 0 6
```

3. Queries to view the respective tables where data has been inserted.

3.1 Query to view the ‘Branch’ Table

```
fm_cloth_store=# select * from branch;
 branch_id |           branch_name           |     city     |                  address
-----------+---------------------------------+--------------+-------------------------------------------
 001       | FM Waterlooville Clothing Store | Waterlooville | 15 Red Cross Road, Waterlooville, WO1 2SP
 002       | FM Fareham Clothing Store       | Fareham       | 12 Buckingham Street, Fareham, FO2 3ST
 003       | FM Gosport Clothing Store       | Gosport       | Lake View Street, Gosport, GO3 2FG
 004       | FM Havant Clothing Store        | Havant        | 4 Harleys Street, Havant, HP1 2FP
 005       | FM Chichester Clothing Store    | Chichester    | Harrow Road, Chichester, CH2 4GK
(5 rows)
```

3.2 Query to view the ‘Employees’ Table

```
fm_cloth_store=# select * from employees;
 employee_id | first_name | last_name  | designation | salary_gbp | date_join  |  shift   | branch_id
-------------+------------+------------+-------------+------------+------------+----------+-----------
 1001        | Michael    | Holding    | Salesman    |    2000.00 | 2023-01-01 | Morning  | 001
 1002        | Lara       | Todes      | Salesman    |    2000.00 | 2023-01-09 | Evening  | 001
 1003        | Freddie    | Mercury    | Manager     |    5000.00 | 2023-02-01 | Hybrid   | 002
 1004        | Alfred     | Tyson      | Manager     |    5000.00 | 2023-02-10 | Hybrid   | 001
 1005        | Brooke     | Johnson    | Cashier     |    3000.00 | 2023-02-25 | Morning  | 003
 1006        | George     | Bennington | Manager     |    5000.00 | 2023-03-01 | Evening  | 005
 1007        | Rohit      | Sharma     | Manager     |    5000.00 | 2023-04-04 | Hybrid   | 003
 1008        | Kiran      | Shree      | Manager     |    5000.00 | 2023-05-15 | Hybrid   | 004
 1009        | John       | Torod      | Salesman    |    2000.00 | 2023-05-20 | Night    | 001
(9 rows)
```

3.3 Query to view the ‘Manager’ Table

```
fm_cloth_store=# select * from manager;
 manager_id | employee_id | branch_id
------------+-------------+-----------
 01         | 1004        | 001
 02         | 1003        | 002
 03         | 1007        | 003
 04         | 1008        | 004
 05         | 1006        | 005
(5 rows)
```

3.4 Query to view the 'Category' Table

```
fm_cloth_store=# select * from category;
 category_id | category_name
-------------+---------------
 WM01        | Women
 MN02        | Men
 BY03        | Baby
 KD04        | Kids
 BT05        | Beauty
 SP06        | Sports
 HM07        | Home Products
(7 rows)
```

3.5 Query to view the 'Product' Table

```
fm_cloth_store=# select * from product;
 product_id |  product_description   |   size    |        composition        | price  | category_id
------------+------------------------+-----------+---------------------------+--------+-------------
 P001       | Yellow Hoodie          | Small     | 100% Woolen               |  20.00 | MN02
 P002       | Yellow Hoodie          | Medium    | 100% Woolen               |  20.00 | MN02
 P003       | Black jeans            | Large     | Denim                     |  15.00 | MN02
 P004       | Red Crop Top           | Medium    | 100% Cotton               |  12.00 | WM01
 P005       | Blue Shirt Dress       | Large     | 70% cotton, 30% Polyester |  11.00 | WM01
 P006       | Fifa Ball              | Free      | Synthetic Leather         | 100.00 | SP06
 P007       | Cricket Bat            | Free Size | Wooden                    |  70.00 | SP06
 P008       | Baby pants             | Small     | 100% cotton               |   8.00 | BY03
 P009       | Blue Kids Polo T-Shirt | Medium    | 40% polyester, 60% cotton |  10.00 | KD04
 P010       | Beige Sofa cover       | Free Size | 80% polyester, 20% cotton |  30.00 | HM07
 P011       | Nike Sneakers          | 7         | Air max sports shoe       |  80.00 | MN02
(11 rows)
```

3.6 Query to view the 'Customer' Table

```
fm_cloth_store=# select * from customer;
 customer_id | first_name | last_name  |      email      |   phone    |            address             |    city    |  reg_date
-------------+------------+------------+-----------------+------------+--------------------------------+------------+------------
 C001        | Angelo     | Thomson    | at123@gmail.com | 0123456789 | Harrow, LO1 2SP                | London     | 2024-09-01
 C002        | Gary       | Mehigan    | gm321@gmail.com | 9874561230 | South Hall, LO2 3AB            | London     | 2024-09-12
 C003        | Mike       | Paul       | mp123@gmail.com | 0147852369 | 4th Cross, MN1 3AS             | Manchester | 2024-09-30
 C004        | Sara       | Ali        | sa123@gmail.com | 3698520147 | Upon Thames, LO2 3PG           | London     | 2024-09-30
 C005        | Tom        | Cruise     | tm325@gmail.com | 8520147963 | UOC Headquarter, CM2 5BJ       | Cambridge  | 2024-10-06
 C006        | Sania      | Todes      | st456@gmail.com | 0147258369 | Central Park, LO4 BG           | London     | 2024-10-12
 C007        | Myke       | Bennington | mb456@gmail.com | 4569871230 | James Road, MN02 4DK           | Manchester | 2024-10-25
 C008        | Farah      | James      | fj897@gmail.com | 0123654789 | Red Cross, CM1 2LM             | Cambridge  | 2024-10-30
 C009        | George     | Jackson    | gj758@gmail.com | 9874563210 | 1st Cross Lake View, MN05 7PL  | Manchester | 2024-10-31
 C010        | Shakeel    | Ali        | sa123@gmail.com | 4560123879 | Guildhall Hall, PO1 2SP        | Portsmouth | 2024-10-31
(10 rows)
```

3.7 Query to view the ‘Orders’ Table

```
fm_cloth_store=# select * from orders;
 order_id | order_date | total_amount | payment_method | delivery_method | customer_id | branch_id
----------+------------+--------------+----------------+-----------------+-------------+-----------
 OD001    | 2024-09-01 |        35.00 | Online         | Home Delivery   | C001        | 001
 OD002    | 2024-09-05 |        20.00 | Cash           | Store Pickup    | C001        | 002
 OD003    | 2024-09-12 |        35.00 | Cash           | Store Pickup    | C002        | 005
 OD004    | 2024-09-30 |        40.00 | Cash           | Store Pickup    | C003        | 003
 OD005    | 2024-09-30 |        18.00 | Online         | Home Delivery   | C004        | 004
 OD006    | 2024-10-06 |       170.00 | Online         | Home Delivery   | C005        | 005
 OD007    | 2024-10-12 |        35.00 | Online         | Home Delivery   | C006        | 001
 OD008    | 2024-10-25 |        10.00 | Cash           | Store Pickup    | C007        | 001
 OD009    | 2024-10-29 |        40.00 | Online         | Home Delivery   | C001        | 004
 OD010    | 2024-10-30 |        30.00 | Online         | Home Delivery   | C008        | 003
 OD011    | 2024-10-31 |        20.00 | Online         | Home Delivery   | C009        | 001
 OD012    | 2024-10-31 |        30.00 | Cash           | Store Pickup    | C010        | 002
(12 rows)
```

3.8 Query to view the ‘Order_Details’ Table

```
fm_cloth_store=# select * from order_details;
 order_id | product_id | quantity | price_per_item
----------+------------+----------+----------------
 OD001    | P002       |        1 |          20.00
 OD001    | P003       |        1 |          15.00
 OD002    | P001       |        1 |          20.00
 OD003    | P001       |        1 |          20.00
 OD003    | P003       |        1 |          15.00
 OD004    | P002       |        2 |          40.00
 OD005    | P008       |        1 |           8.00
 OD005    | P009       |        1 |          10.00
 OD006    | P006       |        1 |         100.00
 OD006    | P007       |        1 |          70.00
 OD007    | P001       |        1 |          20.00
 OD007    | P003       |        1 |          15.00
 OD008    | P009       |        1 |          10.00
 OD009    | P002       |        2 |          20.00
 OD010    | P003       |        2 |          30.00
 OD011    | P009       |        2 |          20.00
 OD012    | P010       |        1 |          30.00
(17 rows)
```

3.9 Query to view the ‘Product_Stock’ Table

```
fm_cloth_store=# select * from product_stock;
 product_id | branch_id | stock_quantity
------------+-----------+----------------
 P001       | 001       |             50
 P001       | 002       |             30
 P001       | 003       |             10
 P001       | 004       |             70
 P001       | 005       |             10
 P002       | 001       |             30
 P002       | 002       |             40
 P002       | 003       |             10
 P002       | 004       |             20
 P002       | 005       |             30
 P003       | 001       |             15
 P003       | 002       |             32
 P003       | 003       |             14
 P003       | 004       |             25
 P003       | 005       |             32
 P004       | 001       |             40
 P004       | 002       |             12
 P004       | 003       |             10
 P004       | 004       |             32
 P004       | 005       |             14
 P005       | 001       |             29
 P005       | 002       |             21
 P005       | 003       |             14
 P005       | 004       |             23
 P005       | 005       |             25
 P006       | 001       |             21
 P006       | 002       |             10
 P006       | 003       |              5
 P006       | 004       |              1
 P006       | 005       |             36
 P007       | 001       |             45
 P007       | 002       |             21
 P007       | 003       |             21
 P007       | 004       |             10
 P007       | 005       |             23
 P008       | 001       |             25
 P008       | 002       |             24
 P008       | 003       |             23
 P008       | 004       |             25
 P008       | 005       |             10
 P009       | 001       |              9
 P009       | 002       |              5
 P009       | 003       |             42
 P009       | 004       |             35
 P009       | 005       |             12
 P010       | 001       |             14
 P010       | 002       |             25
 P010       | 003       |             25
 P010       | 004       |             24
 P010       | 005       |             23
 P011       | 001       |             21
 P011       | 002       |             29
 P011       | 003       |             27
 P011       | 004       |             28
 P011       | 005       |             24
(55 rows)
```

3.10 Query to view the ‘Delivery’ Table

```
fm_cloth_store=# select * from delivery;
 delivery_id | order_id | delivery_date |       delivery_address        | delivery_status | branch_id
-------------+----------+---------------+-------------------------------+-----------------+-----------
 DL01        | OD001    | 2024-09-03    | Harrow, LO1 2SP               | Delivered       | 001
 DL02        | OD005    | 2024-10-03    | Upon Thames, LO2 3PG          | Delivered       | 004
 DL03        | OD006    | 2024-10-08    | UOC Headquarter, CM2 5BJ      | Delivered       | 005
 DL04        | OD007    | 2024-10-14    | Central Park, LO4 BG          | Delivered       | 001
 DL05        | OD009    | 2024-10-30    | Harrow, LO1 2SP               | Delivered       | 004
 DL06        | OD010    | 2024-11-01    | Red Cross, CM1 2LM            | Delivered       | 003
 DL07        | OD011    | 2024-11-02    | 1st Cross Lake View, MN05 7PL | Delivered       | 001
(7 rows)
```

3.11 Query to view the 'Reviews' Table

```
fm_cloth_store=# select * from reviews;
 review_id | product_id | customer_id | order_id | rating |  review_note  | review_date
-----------+------------+-------------+----------+--------+---------------+-------------
 RW001     | P002       | C001        | OD001    |      4 | GOOD PRODUCT  | 2024-09-05
 RW002     | P001       | C001        | OD002    |      5 | EXCELLENT     | 2024-09-06
 RW003     | P001       | C002        | OD003    |      3 | Quality issue | 2024-09-13
 RW004     | P003       | C002        | OD003    |      5 | Superb        | 2024-09-13
 RW005     | P003       | C001        | OD001    |      1 | Color faded   | 2024-09-15
 RW006     | P003       | C008        | OD010    |      2 | Color gone    | 2024-11-05
(6 rows)
```

4. Queries to retrieve essential business and sales-related information for FM Clothing Store from the developed database

4.1 Query to retrieve the basic statistics of customers per branch who made purchases from all the five branches of FM Clothing Store during the months of September and October in the year 2024

```
up2280648=# \c fm_cloth_store
You are now connected to database "fm_cloth_store" as user "up2280648".
fm_cloth_store=# -- 1. Basic stats on customers per branch for a specific time period

SELECT
branch.branch_name,
customer.customer_id,
CONCAT(customer.first_name, ' ', customer.last_name) AS Name,
customer.email,
customer.phone,
customer.address
FROM
customer
JOIN
orders
ON
customer.customer_id = orders.customer_id
JOIN
branch
ON
branch.branch_id=orders.branch_id
WHERE
orders.order_date BETWEEN '2024-08-30' AND '2024-11-02'  -- Specify the time period
GROUP BY
branch.city,branch.branch_name,customer.customer_id, customer.first_name, customer.last_name, customer.email, customer.phone, customer.address
ORDER BY
branch.branch_name;
          branch_name          | customer_id |      name       |      email      |   phone    |            address
-------------------------------+-------------+-----------------+-----------------+------------+-------------------------------
 FM Chichester Clothing Store  | C002        | Gary Mehigan    | gm321@gmail.com | 9874561230 | South Hall, LO2 3AB
 FM Chichester Clothing Store  | C005        | Tom Cruise      | tm325@gmail.com | 8520147963 | UOC Headquarter, CM2 5BJ
 FM Fareham Clothing Store     | C001        | Angelo Thomson  | at123@gmail.com | 0123456789 | Harrow, LO1 2SP
 FM Fareham Clothing Store     | C010        | Shakeel Ali     | sa123@gmail.com | 4560123879 | Guildhall Hall, PO1 2SP
 FM Gosport Clothing Store     | C003        | Mike Paul       | mp123@gmail.com | 0147852369 | 4th Cross, MN1 3AS
 FM Gosport Clothing Store     | C008        | Farah James     | fj897@gmail.com | 0123654789 | Red Cross, CM1 2LM
 FM Havant Clothing Store      | C001        | Angelo Thomson  | at123@gmail.com | 0123456789 | Harrow, LO1 2SP
 FM Havant Clothing Store      | C004        | Sara Ali        | sa123@gmail.com | 3698520147 | Upon Thames, LO2 3PG
 FM Waterlooville Clothing Store | C001      | Angelo Thomson  | at123@gmail.com | 0123456789 | Harrow, LO1 2SP
 FM Waterlooville Clothing Store | C006      | Sania Todes     | st456@gmail.com | 0147258369 | Central Park, LO4 BG
 FM Waterlooville Clothing Store | C007      | Myke Bennington | mb456@gmail.com | 4569871230 | James Road, MN02 4DK
 FM Waterlooville Clothing Store | C009      | George Jackson  | gj758@gmail.com | 9874563210 | 1st Cross Lake View, MN05 7PL
(12 rows)

fm_cloth_store=#
```

Explanation of the Query:

1. **SELECT statement:**
    a. **branch_name** column is selected from the **branch** table.
    b. **customer_id**, **email, phone,** and **address** columns are selected from the **customer** table.
    c. **first_name** and **last_name** columns are selected from the **customer** table and concatenated into a single **Name** using the concat function.
2. **JOINS:**
    a. **customer** and **orders** table are joined on the basis of **customer_id**
    b. **branch** and **orders** table are joined on the basis of **branch_id**
3. **WHERE clause**:
    a. It is used to filter data based on the order_date column of the orders table to retrieve data only for September and October between '2024-08-30' AND '2024-11-02'
4. **GROUP BY:**

a. The result is grouped by the **city** and **branch_name** column of the **branch** table and **customer_id**, **first_name**, **last_name, email, phone,** and **address** column of the **customer** table.

5. **ORDER BY:**
    a. The final result is ordered by the name of the branch in ascending order.

Usefulness of the Query: The result portrays essential information about customers who made purchases across the five branches of F.M Clothing Store. This data will help the business keep records of its customers so they can be reached when new products are launched or during discount offers on the products being sold.

4.2 Query to retrieve information on all products, including description, size, composition, price, and category of the products

```
fm_cloth_store=# -- 2. All products with description and prices

SELECT
Product.Product_ID,
Product.Product_Description,
Product.Size,
Product.Composition,
Product.Price,
Category.Category_Name
FROM
Product
JOIN
Category ON Product.Category_ID = Category.Category_ID
ORDER BY
Product.Product_ID;
 product_id |  product_description   |   size    |        composition         | price  | category_name
------------+-----------------------+-----------+----------------------------+--------+---------------
 P001       | Yellow Hoodie         | Small     | 100% Woolen                |  20.00 | Men
 P002       | Yellow Hoodie         | Medium    | 100% Woolen                |  20.00 | Men
 P003       | Black jeans           | Large     | Denim                      |  15.00 | Men
 P004       | Red Crop Top          | Medium    | 100% Cotton                |  12.00 | Women
 P005       | Blue Shirt Dress      | Large     | 70% cotton, 30% Polyester |  11.00 | Women
 P006       | Fifa Ball             | Free      | Synthetic Leather          | 100.00 | Sports
 P007       | Cricket Bat           | Free Size | Wooden                     |  70.00 | Sports
 P008       | Baby pants            | Small     | 100% cotton                |   8.00 | Baby
 P009       | Blue Kids Polo T-Shirt | Medium    | 40% polyester, 60% cotton |  10.00 | Kids
 P010       | Beige Sofa cover      | Free Size | 80% polyester, 20% cotton |  30.00 | Home Products
 P011       | Nike Sneakers         | 7         | Air max sports shoe        |  80.00 | Men
(11 rows)
```

Explanation of the Query:

1. **SELECT statement:**
    a. **Product_ID, Product_Description, Size, Composition, and Price** columns are selected from the **Product** table.
    b. **Category_Name** column is selected from the **Category** table
2. **JOINS:**
    a. **Product** and **Category** table are joined on the basis of **Category_ID**

3. **ORDER BY:**
    a. The final result is arranged by **Product_ID** in ascending order

Usefulness of the Query: The result portrays all necessary information related to the products sold by F.M Clothing Store. This will help retrieve quick information about products, such as price, description, composition, and the category under which each product falls.

4.3 Query to retrieve online orders and detailed delivery information associated with it

```
fm_cloth_store=# -- 3. Order record and delivery details

SELECT
Orders.Order_ID,
STRING_AGG(Product.Product_Description, ', ') AS Product_Descriptions, -- Concatenate product descriptions
Orders.Order_Date,
Orders.Total_Amount,
Orders.Delivery_Method,
Customer.Customer_ID,
CONCAT(customer.first_name, ' ', customer.last_name) AS Customer_Name,
Customer.Phone,
Delivery.Delivery_ID,
Delivery.Delivery_Date,
Delivery.Delivery_Address,
Branch.Branch_Name
FROM
Orders
JOIN
Customer ON Orders.Customer_ID = Customer.Customer_ID
JOIN
Delivery ON Orders.Order_ID = Delivery.Order_ID
JOIN
Branch ON Orders.Branch_ID = Branch.Branch_ID
JOIN
Order_Details ON Orders.Order_ID = Order_Details.Order_ID
JOIN
Product ON Order_Details.Product_ID = Product.Product_ID
WHERE
Orders.Delivery_Method = 'Home Delivery'
GROUP BY
Orders.Order_ID, Orders.Order_Date, Orders.Total_Amount, Orders.Delivery_Method,
Customer.Customer_ID, Customer.First_Name, Customer.Last_Name, Customer.Email, Customer.Phone,
Delivery.Delivery_ID, Delivery.Delivery_Date, Delivery.Delivery_Address, Delivery.Delivery_Status,
Branch.Branch_Name
ORDER BY
Orders.Order_ID ASC;
 order_id |        product_descriptions        | order_date | total_amount | delivery_method | customer_id | customer_name  |   phone    | delivery_id | delivery_date |       delivery_address        |          branch_name
----------+------------------------------------+------------+--------------+-----------------+-------------+----------------+------------+-------------+---------------+-------------------------------+-------------------------------
 OD001    | Yellow Hoodie, Black jeans         | 2024-09-01 |        35.00 | Home Delivery   | C001        | Angelo Thomson | 0123456789 | DL01        | 2024-09-03    | Harrow, LO1 2SP               | FM Waterlooville Clothing Store
 OD005    | Baby pants, Blue Kids Polo T-Shirt | 2024-09-30 |        18.00 | Home Delivery   | C004        | Sara Ali       | 3698520147 | DL02        | 2024-10-03    | Upon Thames, LO2 3PG          | FM Havant Clothing Store
 OD006    | Fifa Ball, Cricket Bat             | 2024-10-06 |       170.00 | Home Delivery   | C005        | Tom Cruise     | 8520147963 | DL03        | 2024-10-08    | UOC Headquarter, CM2 5BJ      | FM Chichester Clothing Store
 OD007    | Yellow Hoodie, Black jeans         | 2024-10-12 |        35.00 | Home Delivery   | C006        | Sania Todes    | 0147258369 | DL04        | 2024-10-14    | Central Park, LO4 BG          | FM Waterlooville Clothing Store
 OD009    | Yellow Hoodie                      | 2024-10-29 |        40.00 | Home Delivery   | C001        | Angelo Thomson | 0123456789 | DL05        | 2024-10-30    | Harrow, LO1 2SP               | FM Havant Clothing Store
 OD010    | Black jeans                        | 2024-10-30 |        30.00 | Home Delivery   | C008        | Farah James    | 0123654789 | DL06        | 2024-11-01    | Red Cross, CM1 2LM            | FM Gosport Clothing Store
 OD011    | Blue Kids Polo T-Shirt             | 2024-10-31 |        20.00 | Home Delivery   | C009        | George Jackson | 9874563210 | DL07        | 2024-11-02    | 1st Cross Lake View, MN05 7PL | FM Waterlooville Clothing Store
(7 rows)
```

Explanation of the Query:

1. **SELECT statement:**
    a. **Order_ID, Order_Date, Total_Amount,** and **Delivery_Method** columns are selected from the Orders table
    b. **Product_Description** column is selected and aggregated as string from the **Product** table which is stored as an alias name of **Product_Descriptions** which combines all the products available in an order.
    c. **Customer_ID, First_Name, Last_Name, Email,** and **Phone** columns are selected from the **Customer** table. **First_Name** and **Last_Name** are concatenated into a single name and stored as **Customer_Name**
    d. **Delivery_ID, Delivery_Date** and **Delivery_Address** columns are selected from the **Delivery** table.
    e. **Branch_Name** is selected from the **Branch** table
2. **JOINS:**
    a. **Customer** and **Orders** table are joined on the basis of **Customer_ID**
    b. **Delivery** and **Orders** table are joined on the basis of **Order_ID**
    c. **Branch** and **Orders** table are joined on the basis of **Branch_ID**
    d. **Orders** and **Order_Details** table are joined on the basis of **Order_ID**

e. **Product** and **Order_Details** table are joined on the basis of **Product_ID**

3. **WHERE clause:**
    a. It is used to filter data to retrieve only those fields whose delivery method is **'Home Delivery'**
4. **GROUP BY:**
    a. The result is grouped by the **Order_ID, Order_Date, Total_Amount** and **Delivery_Method** from the **Orders** table, **Customer_ID, First_Name, Last_Name, Email** and **Phone** from the **Customer** table, **Delivery_ID, Delivery_Date, Delivery_Address** and **Delivery_Status** from the **Delivery** table and finally the **Branch_Name** from the **Branch** table
5. **ORDER BY:**
    a. The final result is arranged as per **Order_ID** in ascending order

Usefulness of the Query: This query helps track the detailed delivery process of orders placed online from any of the branches of F.M Clothing Store thereby increasing delivery efficiency.

4.4 Query to retrieve detailed information on the availability of all the products and their quantities present across the five different branches.

```
fm_cloth_store=# -- 4. Report of product availability and their location
SELECT
Product.Product_ID,
Product.Product_Description,
Product.Size,
Product.Composition,
Product.Price,
Category.Category_Name,
Branch.Branch_ID,
Branch.Branch_Name,
Product_Stock.Stock_Quantity
FROM
Product_Stock
JOIN
Product ON Product_Stock.Product_ID = Product.Product_ID
JOIN
Branch ON Product_Stock.Branch_ID = Branch.Branch_ID
JOIN
Category ON Product.Category_ID = Category.Category_ID
ORDER BY
Product.Product_ID, Branch.Branch_ID;
```

product_id	product_description	size	composition	price	category_name	branch_id	branch_name	stock_quantity
P001	Yellow Hoodie	Small	100% Woolen	20.00	Men	001	FM Waterlooville Clothing Store	50
P001	Yellow Hoodie	Small	100% Woolen	20.00	Men	002	FM Fareham Clothing Store	30
P001	Yellow Hoodie	Small	100% Woolen	20.00	Men	003	FM Gosport Clothing Store	10
P001	Yellow Hoodie	Small	100% Woolen	20.00	Men	004	FM Havant Clothing Store	70
P001	Yellow Hoodie	Small	100% Woolen	20.00	Men	005	FM Chichester Clothing Store	10
P002	Yellow Hoodie	Medium	100% Woolen	20.00	Men	001	FM Waterlooville Clothing Store	30
P002	Yellow Hoodie	Medium	100% Woolen	20.00	Men	002	FM Fareham Clothing Store	40
P002	Yellow Hoodie	Medium	100% Woolen	20.00	Men	003	FM Gosport Clothing Store	10
P002	Yellow Hoodie	Medium	100% Woolen	20.00	Men	004	FM Havant Clothing Store	20
P002	Yellow Hoodie	Medium	100% Woolen	20.00	Men	005	FM Chichester Clothing Store	30
P003	Black jeans	Large	Denim	15.00	Men	001	FM Waterlooville Clothing Store	15
P003	Black jeans	Large	Denim	15.00	Men	002	FM Fareham Clothing Store	32
P003	Black jeans	Large	Denim	15.00	Men	003	FM Gosport Clothing Store	14
P003	Black jeans	Large	Denim	15.00	Men	004	FM Havant Clothing Store	25
P003	Black jeans	Large	Denim	15.00	Men	005	FM Chichester Clothing Store	32
P004	Red Crop Top	Medium	100% Cotton	12.00	Women	001	FM Waterlooville Clothing Store	40
P004	Red Crop Top	Medium	100% Cotton	12.00	Women	002	FM Fareham Clothing Store	12
P004	Red Crop Top	Medium	100% Cotton	12.00	Women	003	FM Gosport Clothing Store	10
P004	Red Crop Top	Medium	100% Cotton	12.00	Women	004	FM Havant Clothing Store	32
P004	Red Crop Top	Medium	100% Cotton	12.00	Women	005	FM Chichester Clothing Store	14
P005	Blue Shirt Dress	Large	70% cotton, 30% Polyester	11.00	Women	001	FM Waterlooville Clothing Store	29
P005	Blue Shirt Dress	Large	70% cotton, 30% Polyester	11.00	Women	002	FM Fareham Clothing Store	21
P005	Blue Shirt Dress	Large	70% cotton, 30% Polyester	11.00	Women	003	FM Gosport Clothing Store	14
P005	Blue Shirt Dress	Large	70% cotton, 30% Polyester	11.00	Women	004	FM Havant Clothing Store	23
P005	Blue Shirt Dress	Large	70% cotton, 30% Polyester	11.00	Women	005	FM Chichester Clothing Store	25
P006	Fifa Ball	Free	Synthetic Leather	100.00	Sports	001	FM Waterlooville Clothing Store	21
P006	Fifa Ball	Free	Synthetic Leather	100.00	Sports	002	FM Fareham Clothing Store	10
P006	Fifa Ball	Free	Synthetic Leather	100.00	Sports	003	FM Gosport Clothing Store	5
P006	Fifa Ball	Free	Synthetic Leather	100.00	Sports	004	FM Havant Clothing Store	1
P006	Fifa Ball	Free	Synthetic Leather	100.00	Sports	005	FM Chichester Clothing Store	36
P007	Cricket Bat	Free Size	Wooden	70.00	Sports	001	FM Waterlooville Clothing Store	45
P007	Cricket Bat	Free Size	Wooden	70.00	Sports	002	FM Fareham Clothing Store	21
P007	Cricket Bat	Free Size	Wooden	70.00	Sports	003	FM Gosport Clothing Store	21
P007	Cricket Bat	Free Size	Wooden	70.00	Sports	004	FM Havant Clothing Store	10
P007	Cricket Bat	Free Size	Wooden	70.00	Sports	005	FM Chichester Clothing Store	23
P008	Baby pants	Small	100% cotton	8.00	Baby	001	FM Waterlooville Clothing Store	25
P008	Baby pants	Small	100% cotton	8.00	Baby	002	FM Fareham Clothing Store	24
P008	Baby pants	Small	100% cotton	8.00	Baby	003	FM Gosport Clothing Store	23
P008	Baby pants	Small	100% cotton	8.00	Baby	004	FM Havant Clothing Store	25
P008	Baby pants	Small	100% cotton	8.00	Baby	005	FM Chichester Clothing Store	10
P009	Blue Kids Polo T-Shirt	Medium	40% polyester, 60% cotton	10.00	Kids	001	FM Waterlooville Clothing Store	9
P009	Blue Kids Polo T-Shirt	Medium	40% polyester, 60% cotton	10.00	Kids	002	FM Fareham Clothing Store	5
P009	Blue Kids Polo T-Shirt	Medium	40% polyester, 60% cotton	10.00	Kids	003	FM Gosport Clothing Store	42
P009	Blue Kids Polo T-Shirt	Medium	40% polyester, 60% cotton	10.00	Kids	004	FM Havant Clothing Store	35
P009	Blue Kids Polo T-Shirt	Medium	40% polyester, 60% cotton	10.00	Kids	005	FM Chichester Clothing Store	12
P010	Beige Sofa cover	Free Size	80% polyester, 20% cotton	30.00	Home Products	001	FM Waterlooville Clothing Store	14
P010	Beige Sofa cover	Free Size	80% polyester, 20% cotton	30.00	Home Products	002	FM Fareham Clothing Store	25
P010	Beige Sofa cover	Free Size	80% polyester, 20% cotton	30.00	Home Products	003	FM Gosport Clothing Store	25
P010	Beige Sofa cover	Free Size	80% polyester, 20% cotton	30.00	Home Products	004	FM Havant Clothing Store	24
P010	Beige Sofa cover	Free Size	80% polyester, 20% cotton	30.00	Home Products	005	FM Chichester Clothing Store	23
P011	Nike Sneakers	7	Air max sports shoe	80.00	Men	001	FM Waterlooville Clothing Store	21
P011	Nike Sneakers	7	Air max sports shoe	80.00	Men	002	FM Fareham Clothing Store	29
P011	Nike Sneakers	7	Air max sports shoe	80.00	Men	003	FM Gosport Clothing Store	27
P011	Nike Sneakers	7	Air max sports shoe	80.00	Men	004	FM Havant Clothing Store	28
P011	Nike Sneakers	7	Air max sports shoe	80.00	Men	005	FM Chichester Clothing Store	24

(55 rows)

Explanation of the Query:

1. **SELECT statement:**
    a. **Product_ID, Product_Description, Size, Composition,** and **Price** columns are selected from the **Product** table
    b. **Category_Name** column is selected from the **Category** table
    c. **Branch_ID and Branch_Name** columns are selected from the **Branch** table
    d. **Stock_Quantity** column is selected from the **Product_Stock** table.
2. **JOINS:**
    a. **Product** and **Product_Stock** tables are joined on the basis of **Product_ID**
    b. **Branch** and **Product_Stock** tables are joined on the basis of **Branch_ID**
    c. **Product** and **Category** tables are joined on the basis of **Category_ID**

3. **ORDER BY:**
    a. The final result is arranged in ascending order first on the basis of **Product_ID** and then on the basis of **Branch_ID**

Usefulness of the Query: This query helps keep a record of the availability of each product across all the branches

4.5 Query to calculate the monthly income generated by each of the five branches for the month of September and October 2024

```
fm_cloth_store=# -- 5. Monthly income generated per branch
SELECT
Branch.City,
TO_CHAR(Orders.Order_Date, 'YYYY-MM') AS Month,
SUM(Orders.Total_Amount) AS Monthly_Income
FROM
Orders
JOIN
Branch ON Orders.Branch_ID = Branch.Branch_ID
GROUP BY
Branch.City,
Month
ORDER BY
Month, Branch.City;
     city      |  month  | monthly_income
---------------+---------+----------------
 Chichester    | 2024-09 |          35.00
 Fareham       | 2024-09 |          20.00
 Gosport       | 2024-09 |          40.00
 Havant        | 2024-09 |          18.00
 Waterlooville | 2024-09 |          35.00
 Chichester    | 2024-10 |         170.00
 Fareham       | 2024-10 |          30.00
 Gosport       | 2024-10 |          30.00
 Havant        | 2024-10 |          40.00
 Waterlooville | 2024-10 |          65.00
(10 rows)
```

Explanation of the Query:

1. **SELECT statement:**

a. **City** column is selected from the **Branch** table
b. **Order_Date** column is selected from the **Orders** table and is converted into string and stored in year-month format and stored as **Month**
c. **Total_Amount** is selected from the Orders table and added using the **sum** function and stored the result as **Montly_Income**

2. **JOINS:**
   a. **Branch** and **Orders** table are joined on the basis of **Branch_ID**
3. **GROUP BY:**
   a. The result is grouped by the **City** column of the **Branch** table and by **Month**
4. **ORDER BY:**
   a. The final result is arranged in ascending order first on the basis of **Month** and then on the basis of **City**

Usefulness of the Query: The result conveys essential information on monthly sales per branch and helps compare sales between the branches

4.6 Query to retrieve the top 3 selling products

```
fm_cloth_store=# -- 6. Top 3 selling products
SELECT
Product.Product_ID,
Product.Product_Description,
Product.Size,
Product.Composition,
Category.Category_Name,
SUM(Order_Details.Quantity) AS Total_Quantity_Sold
FROM
Order_Details
JOIN
Product ON Order_Details.Product_ID = Product.Product_ID
JOIN
Category ON Product.Category_ID = Category.Category_ID
GROUP BY
Product.Product_ID,
Product.Product_Description,
Product.Size,
Product.Composition,
Category.Category_Name
ORDER BY
Total_Quantity_Sold DESC
LIMIT 3;
 product_id |  product_description   |  size  |        composition        | category_name | total_quantity_sold
------------+-----------------------+--------+---------------------------+---------------+---------------------
 P003       | Black jeans           | Large  | Denim                     | Men           |                   5
 P002       | Yellow Hoodie         | Medium | 100% Woolen               | Men           |                   5
 P009       | Blue Kids Polo T-Shirt | Medium | 40% polyester, 60% cotton | Kids          |                   4
(3 rows)
```

Explanation of the Query:

1. **SELECT statement:**

a. **Product_ID, Product_Description, Size** and **Composition** columns are selected from the **Product** table
b. **Qauntity** column is selected from the **Order_Details** table and aggregated using the **sum** function and stored the result as **Total_Quantity_Sold**

2. **JOINS:**
    a. **Product** and **Order_Details** table are joined on the basis of **Product_ID**
3. **GROUP BY:**
    a. The result is grouped by the **Product_ID, Product_Description, Size,** and **Composition** columns of the **Product** table and **category_name** column of the **category** table
4. **ORDER BY:**
    a. The final result is arranged in descending order on the basis of **Total_Quantity_Sold**.
5. **LIMIT:**
    a. The results are limited to only 3 top selling products

Usefulness of the Query: This query identifies the top 3 selling products across all branches. This is important for introducing more products of similar types and maintaining the same quality in other products as well.

4.7 Query to retrieve products when stock levels drop below 10 across all five branches

```
fm_cloth_store=# -- 7. Low stock alert when product quantity is less than 10
SELECT
Product.Product_ID,
Product.Product_Description,
Branch.Branch_Name,
Product_Stock.Stock_Quantity
FROM
Product_Stock
JOIN
Product ON Product_Stock.Product_ID = Product.Product_ID
JOIN
Branch ON Product_Stock.Branch_ID = Branch.Branch_ID
WHERE
Product_Stock.Stock_Quantity < 10
ORDER BY
Product.Product_ID,
Branch.Branch_ID;
 product_id |  product_description   |           branch_name           | stock_quantity
------------+-------------------------+---------------------------------+----------------
 P006       | Fifa Ball               | FM Gosport Clothing Store       |              5
 P006       | Fifa Ball               | FM Havant Clothing Store        |              1
 P009       | Blue Kids Polo T-Shirt | FM Waterlooville Clothing Store |              9
 P009       | Blue Kids Polo T-Shirt | FM Fareham Clothing Store       |              5
(4 rows)
```

Explanation of the Query:

1. **SELECT statement:**
    a. **Product_ID** and **Product_Description** tables are selected from the **Product** table
    b. **Branch_Name** is selected from the **branch** table
    c. **Stock_Quantity** is selected from the **Product_Stock** table
2. **JOINS:**
    a. **Product** and **Product_Stock** tables are joined on the basis of **Product_ID**
    b. **Branch** and **Product_Stock** tables are joined on the basis of **Branch_ID**
3. **WHERE clause:**
    a. It is used to filter the results for Stock_Quantity less than 10
4. **ORDER BY:**
    a. The final result is arranged in ascending order first on the basis of **Product_ID** from the **Product** table and then on the basis of **Branch_ID** from the branch table

Usefulness of the Query: This query helps retailers restock products when the availability of items falls below a certain predefined threshold (e.g., 10) across the branches

4.8 Query to list the products that receive a customer rating of less than 4

```
fm_cloth_store=# -- 8. Products with ratings less than 4

SELECT Product.Product_ID,
Product.Product_Description,
Product.Category_ID,
Reviews.Rating AS Product_Rating
FROM Product
JOIN Reviews ON Product.Product_ID = Reviews.Product_ID
where Rating<4
GROUP BY Product.Product_ID, Product.Product_Description, Product.Category_ID, Rating
ORDER BY Rating ASC;
 product_id | product_description | category_id | product_rating
------------+---------------------+-------------+----------------
 P003       | Black jeans         | MN02        |              1
 P003       | Black jeans         | MN02        |              2
 P001       | Yellow Hoodie       | MN02        |              3
(3 rows)
```

Explanation of the Query:

1. **SELECT statement:**
    a. **Product_ID, Product_Description, and Category_ID** columns are selected from the **Product** table
    b. **Rating** column is selected from the **Reviews** table and stored as **Product_Rating**

2. **JOINS:**

    a. **Product** and **Reviews** table are joined on the basis of **Product_ID**

3. **WHERE clause:**

    a. This retrieves only those products which has got ratings less than 4

4. **GROUP BY:**

    a. The result is grouped by **Product_ID, Product_Description** and **Category_ID** columns of the **Product** table and by **Rating**

5. **ORDER BY:**

    a. The final result is arranged in ascending order by **rating**

Usefulness of the Query: This creates an opportunity to improve products by considering customer feedback and suggestions, and enhancing the quality of the products sold.

4.9 Query to calculate the total number of orders placed (including both online and in-store pickup) per branch

```
fm_cloth_store=# -- 9. Monthly order count per branch
SELECT
Branch.City,
TO_CHAR(Orders.Order_Date, 'YYYY-MM') AS Month,
COUNT(Orders.Order_ID) AS Monthly_Order_Count
FROM
Orders
JOIN
Branch ON Orders.Branch_ID = Branch.Branch_ID
GROUP BY
Branch.City,
TO_CHAR(Orders.Order_Date, 'YYYY-MM')
ORDER BY
Month, Branch.City;
     city      |  month  | monthly_order_count
---------------+---------+---------------------
 Chichester    | 2024-09 |                   1
 Fareham       | 2024-09 |                   1
 Gosport       | 2024-09 |                   1
 Havant        | 2024-09 |                   1
 Waterlooville | 2024-09 |                   1
 Chichester    | 2024-10 |                   1
 Fareham       | 2024-10 |                   1
 Gosport       | 2024-10 |                   1
 Havant        | 2024-10 |                   1
 Waterlooville | 2024-10 |                   3
(10 rows)
```

Explanation of the Query:

1. **SELECT statement:**
    a. **City** column is selected from the **Branch** table
    b. **Order_Date** column is selected from the **Orders** table and is converted into string and stored in a year-month format which is stored as **Month**
    c. **Order_ID** column is selected from the **Orders** table and is counted using the **count** function which is stored as Monthly_Order_Count

2. **JOIN:**

    a. **Branch** and **Orders** table are joined on the basis of **Branch_ID**

3. **GROUP BY:**

   a. The result is grouped by **City** and **Month**

4. **ORDER BY:**

   a. The final result is arranged in ascending order first by **Month** and then by **City**

Usefulness of the Query: This query helps retrieve information on the number of orders placed per branch in a specific month.

4.10 Query to retrieve the number of 'Home Delivery' and 'Store Pickup' orders in the month of September and October

```
fm_cloth_store=# -- 10. No. of Home Delivery and Store Pickup orders in both the months
SELECT
TO_CHAR(Order_Date, 'YYYY-MM') AS Order_Month,
COUNT(CASE WHEN Delivery_Method = 'Home Delivery' THEN 1 END) AS Home_Delivery_Count,
COUNT(CASE WHEN Delivery_Method = 'Store Pickup' THEN 1 END) AS Store_Pickup_Count
FROM
Orders
GROUP BY
TO_CHAR(Order_Date, 'YYYY-MM')
ORDER BY
Order_Month ASC;
 order_month | home_delivery_count | store_pickup_count
-------------+---------------------+--------------------
 2024-09     |                   2 |                  3
 2024-10     |                   5 |                  2
(2 rows)
```

Explanation of the Query:

1. **SELECT statement:**
   a. **TO_CHAR(Order_Date, 'YYYY-MM') AS Order_Month:** This particular line converts the **Order_Date** column of the **Orders** table in the **year-month** format and stored as **Order_Month**.
   b. **COUNT(CASE WHEN Delivery_Method = 'Home Delivery' THEN 1 END) AS Home_Delivery_Count:** This line counts the number of rows where Delivery_Method is **"Home Delivery"** and stores the result in **Home_Delivery_Count**.
   c. **COUNT(CASE WHEN Delivery_Method = 'Store Pickup' THEN 1 END) AS Store_Pickup_Count:** This line counts the number of rows where **Delivery_Method** is **"Store Pickup"** and stores the result in **Store_Pickup_Count**
2. **FROM clause:**

    a. The data is retrieved from the **Orders** table
3. **GROUP BY:**
    a. The result is grouped by the order of month
4. **ORDER BY:**
    a. The final result is arranged in ascending order by **Order_Month.**

Usefulness of the Query: This query helps identify the number of home delivery and store pickup orders per month, which helps the business understand customer preferences for online versus offline purchases

Task 5 (Reflective Report)

Development of the Database

The first crucial step we undertook before developing the database was to understand the requirements of the FM Clothing Store. After carefully understanding the requirements of the FM Clothing Store we figured out the different types of tables to include in our database by keeping in mind the usefulness of the tables for day-to-day operations of the retailer. The tables present in our database are systematically linked to multiple tables within the database, this is done to establish important relationships throughout the database with the correct use of foreign keys. Every table has a primary key, which will help to uniquely identify each entry in the tables. We also made sure to create the names of columns in each table simple and concise so that any person can easily insert, update, and delete records in the database. The tables are also designed in the third normal form (3NF) to ensure that the database doesn't have any redundant or repeating data. Moreover, the tables are created in such a way that even a person with no technical knowledge of database can easily understand the purpose and functionality of every table created. In order to fit the database correctly with the business's actual data, the database is extensively tested with dummy data to make sure that the database works without a glitch in the actual scenario.

SQL Queries

The main queries for this database are created in such a way that it cover important aspects of F.M. Clothing Store's operations which will help to generate actionable insights and facilitate growth of the business. The queries are chosen by taking into consideration the following factors but not limited to these-

- Sales Insights: To generate monthly sales data for each branch.
- Stock Management: To track the availability of products across all branches and identify items to restock when the stock of certain products falls below a certain limit.
- Customer Details: To store details of customers who have made purchases.

- Delivery Process: To track the delivery process for orders placed online.
- Customer Feedback and Suggestions: To analyze customer feedback in order to improve the quality of products sold

Collaboration with team mates

The database could not be developed without the effective cooperation and collaboration of all the team members. The task was evenly and systematically divided among the teammates to distribute the workload equally. We held Google Meet sessions every alternate day to discuss the progress of our work. All three members in our group were open to new ideas and were always eager to accept different feedback and suggestions throughout the coursework. Lastly, and most importantly, working as a team kept us motivated and energized to successfully develop the database.

References

1. Beaulieu, A., & Beaulieu, A. (2009). *Learning sql : Master sql fundamentals*. O'Reilly Media, Incorporated. https://ebookcentral.proquest.com/lib/portsmouth-ebooks/detail.action?docID=443242&query=13:%20978-0-596-52083-0

2. Obe, R. O., & Hsu, L. S. (2017). *Postgresql : Up and running : a practical guide to the advanced open source database*. O'Reilly Media, Incorporated. https://ebookcentral.proquest.com/lib/portsmouth-ebooks/detail.action?docID=5103333&query=9781491963418
3. Russo, J. (2017). *Sql by example*. Momentum Press. https://ebookcentral.proquest.com/lib/portsmouth-ebooks/detail.action?docID=5602384&query=sql%20by%20example
4. Wilton, P., & Colby, J. (2005). *Beginning sql*. John Wiley & Sons, Incorporated. https://ebookcentral.proquest.com/lib/portsmouth-ebooks/detail.action?docID=226434&query=sql
5. Ghlala, R. (2019). *Analytic sql in sql server 2014/2016*. John Wiley & Sons, Incorporated. https://ebookcentral.proquest.com/lib/portsmouth-ebooks/detail.action?docID=5849355&query=sql

6. SQL Tutorial, (2024,10). W3schools. https://www.w3schools.com/sql/default.asp