Clothing Store Database Guidebook

A Comprehensive Guide to Managing Retail Data Efficiently

By Manveer Kaur

Data Analyst, St. Clair College, Toronto

Table of Contents

1. Introduction: The **Clothing Store Database Guidebook** provides a structured overview of the database designed to manage various aspects of a clothing retail business. This database is optimized for handling **customers, products, suppliers, orders, inventory, and categories** efficiently.

The guidebook includes:

- A **detailed explanation** of the database structure
- **Descriptions of all tables** and their relationships
- **SQL scripts** for creating tables, inserting data, and retrieving information
- **Best practices** for database management

This document is useful for **database administrators, developers, and business managers** who need to understand and interact with the system effectively. Whether you're managing inventory, tracking sales, or analyzing customer behavior, this guide will help you navigate and utilize the database efficiently.

2. Database Structure The database consists of seven primary tables:

- **Customers**: Stores customer details.
- **Categories**: Defines product categories.
- **Suppliers**: Manages supplier information.
- **Products**: Contains details of clothing items.
- **Orders**: Stores customer order data.
- **Order_Items**: Links orders with products.
- **Inventory**: Tracks stock levels.

3. Table Descriptions

- **Customers Table**:
    - customer_id: Unique ID for each customer.
    - first_name, last_name: Customer name.
    - email: Unique email.
    - phone, address: Contact details.
    - created_at: Timestamp of account creation.
- **Categories Table**:
    - category_id: Unique category ID.
    - category_name: Name of category.
    - description: Category details.
- **Suppliers Table**:
    - supplier_id: Unique supplier ID.
    - supplier_name: Name of supplier.
    - contact_name, email, phone, address: Contact details.
- **Products Table**:
    - product_id: Unique product ID.
    - product_name: Name of product.
    - category_id: Foreign key linking to Categories.

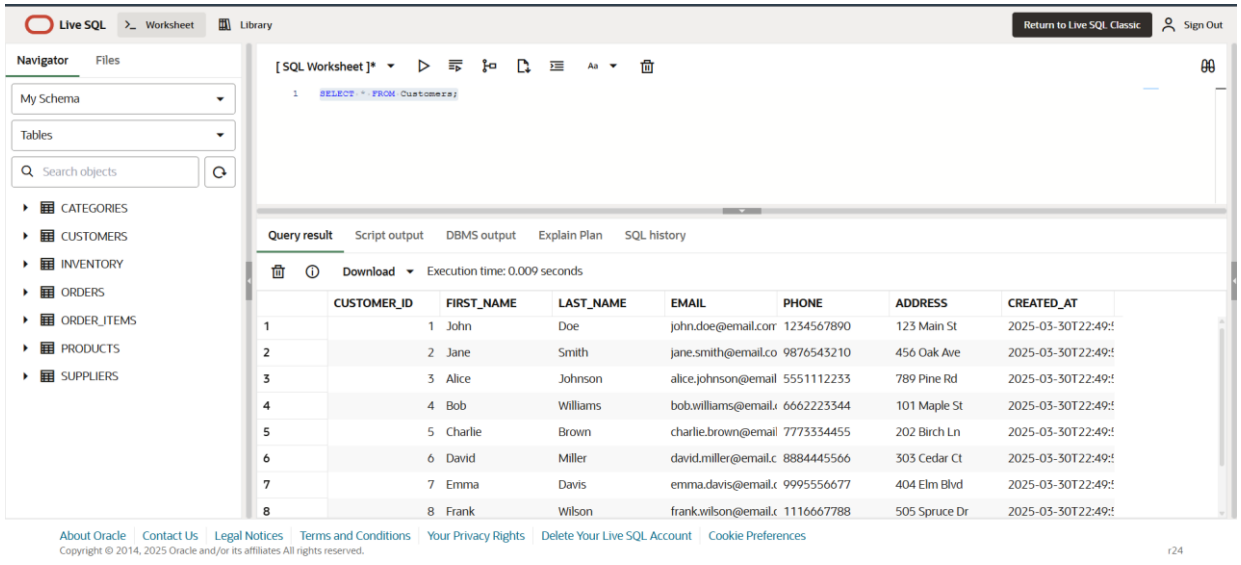
    - price: Product price.
    - product_size, color: Attributes of the product.
    - stock_quantity: Available stock.
    - supplier_id: Foreign key linking to Suppliers.
- **Orders Table**:
    - order_id: Unique order ID.
    - customer_id: Foreign key linking to Customers.
    - order_date: Date of order.
    - total_amount: Total order cost.
    - status: Order status (Pending, Shipped, etc.).
- **Order_Items Table**:
    - order_item_id: Unique ID.
    - order_id: Foreign key linking to Orders.
    - product_id: Foreign key linking to Products.
    - quantity: Number of items ordered.
    - subtotal: Price for the quantity ordered.
- **Inventory Table**:
    - inventory_id: Unique ID.
    - product_id: Foreign key linking to Products.

- quantity: Stock level.
- last_updated: Timestamp of last update.

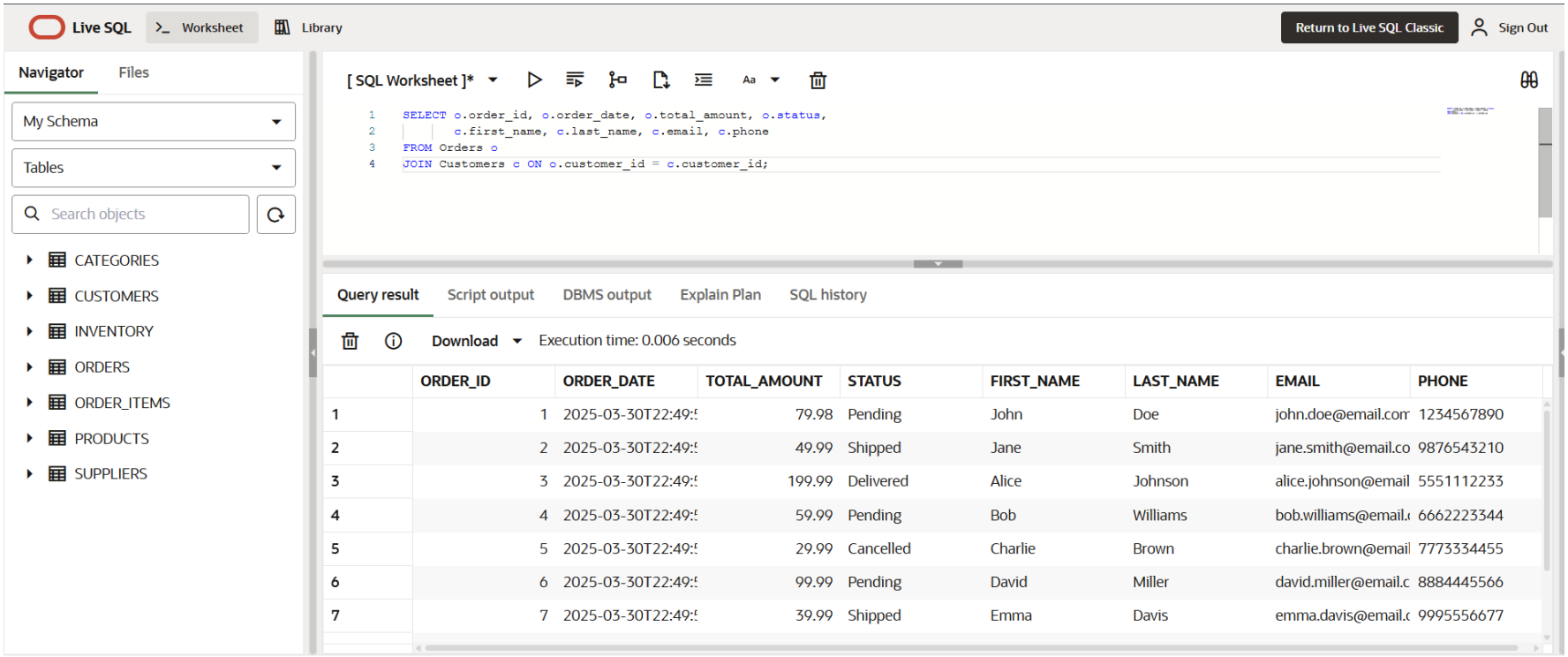
4. Key SQL Queries

View all Customers:

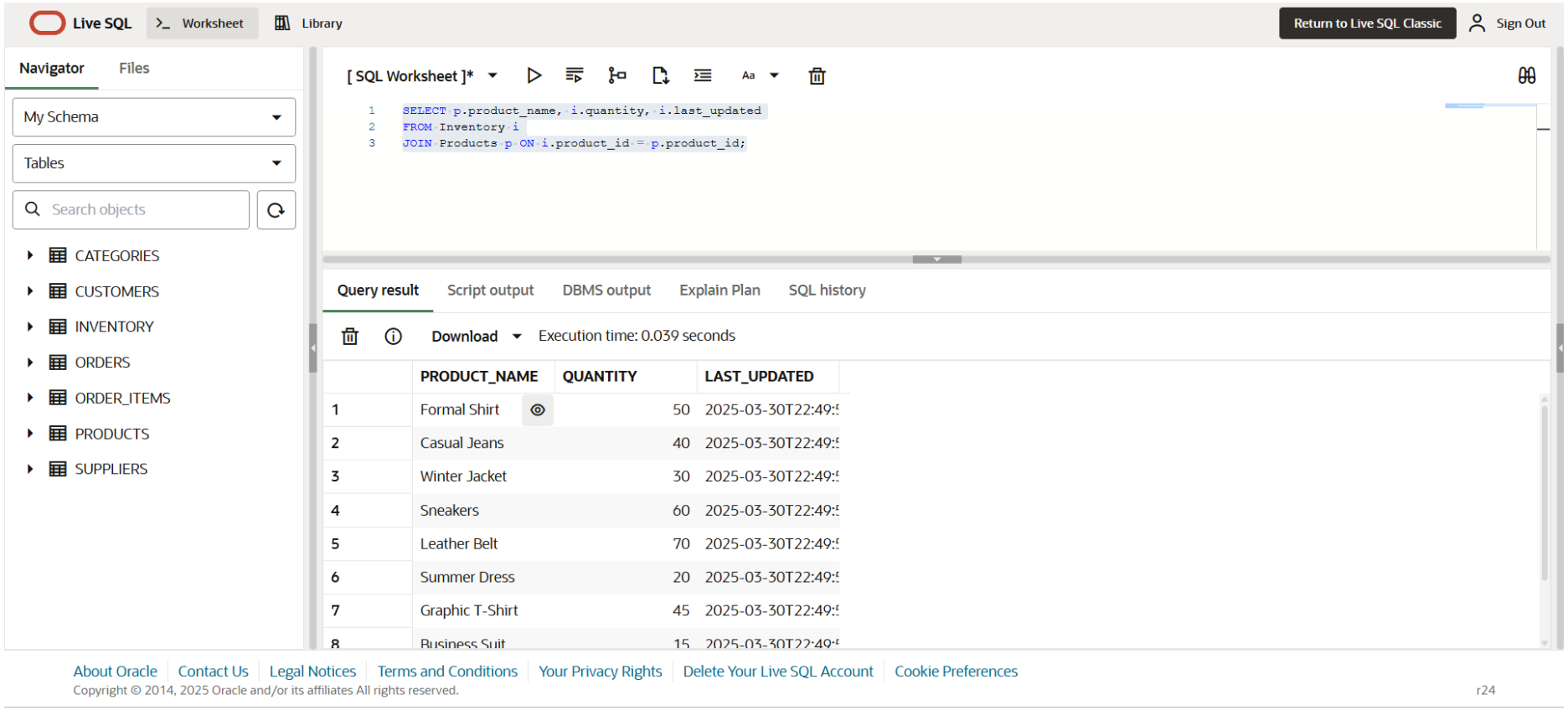
SELECT * FROM Customers;



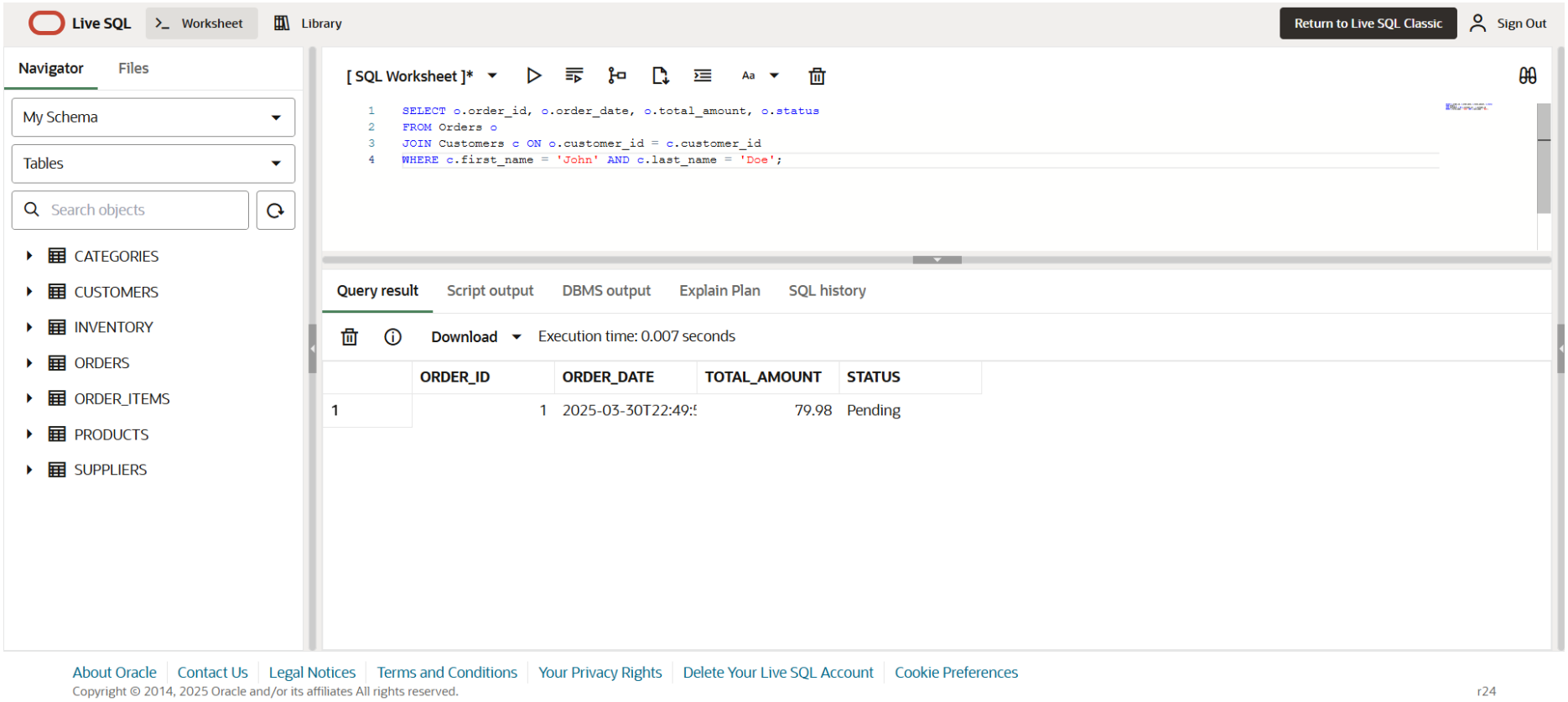
Retrieve all Orders with Customer Details:



View Inventory Levels:



Find Orders for a Specific Customer (Example: John Doe)



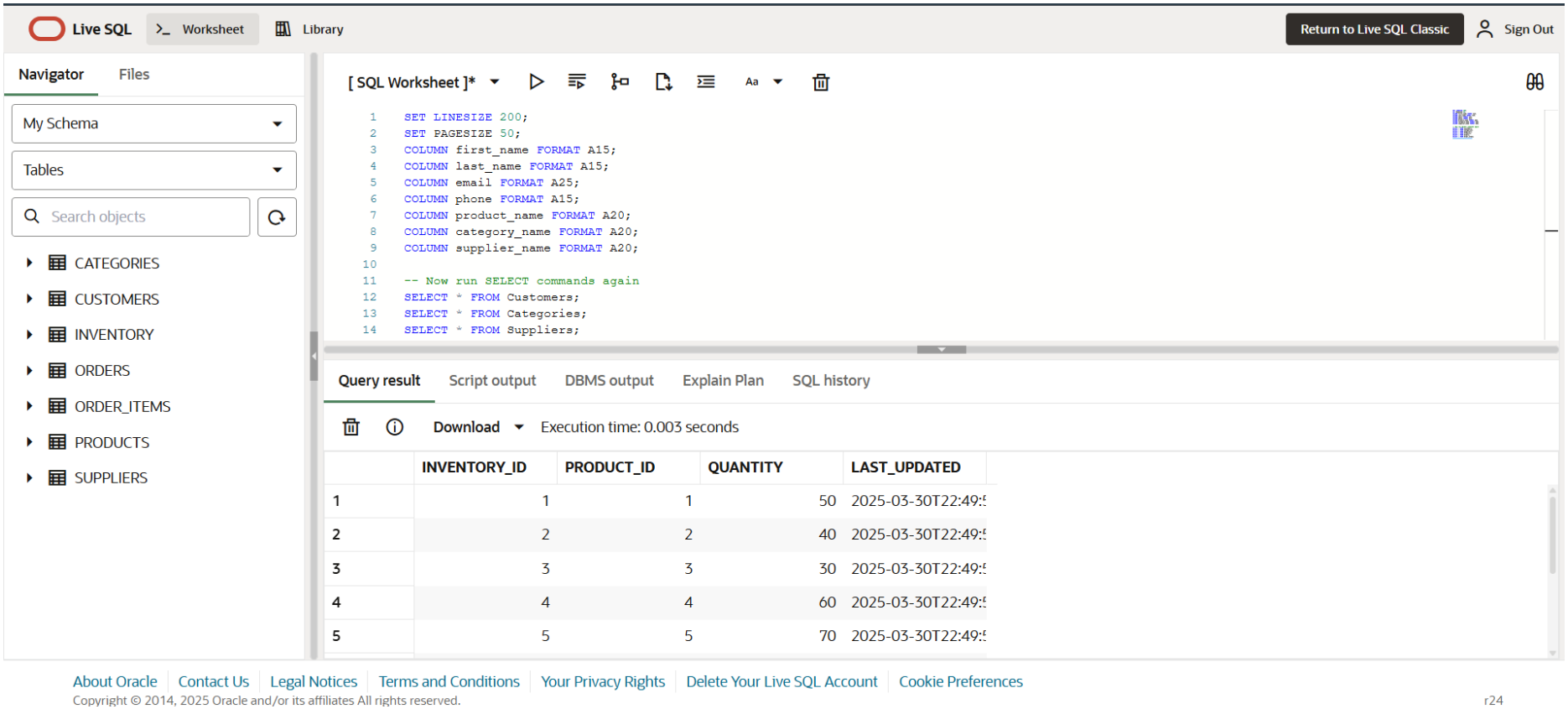
Alternative: View Data in a More Readable Format

If you want a more formatted display, use column alignment in SQL*Plus or SQLcl:

```
SET LINESIZE 200;
```

```
SET PAGESIZE 50;
COLUMN first_name FORMAT A15;
COLUMN last_name FORMAT A15;
COLUMN email FORMAT A25;
COLUMN phone FORMAT A15;
COLUMN product_name FORMAT A20;
COLUMN category_name FORMAT A20;
COLUMN supplier_name FORMAT A20;

-- Now run SELECT commands again
SELECT * FROM Customers;
SELECT * FROM Categories;
SELECT * FROM Suppliers;
SELECT * FROM Products;
SELECT * FROM Orders;
SELECT * FROM Order_Items;
SELECT * FROM Inventory;
```



5. Database Management Best Practices

- Regularly update stock levels in the Inventory table.
- Ensure email addresses are unique in the Customers table.
- Use foreign keys to maintain referential integrity between tables.
- Backup the database periodically to prevent data loss.

```
-- Drop tables if they already exist
BEGIN
   EXECUTE IMMEDIATE 'DROP TABLE Order_Items CASCADE CONSTRAINTS';
   EXECUTE IMMEDIATE 'DROP TABLE Orders CASCADE CONSTRAINTS';
   EXECUTE IMMEDIATE 'DROP TABLE Inventory CASCADE CONSTRAINTS';
   EXECUTE IMMEDIATE 'DROP TABLE Products CASCADE CONSTRAINTS';
   EXECUTE IMMEDIATE 'DROP TABLE Suppliers CASCADE CONSTRAINTS';
   EXECUTE IMMEDIATE 'DROP TABLE Categories CASCADE CONSTRAINTS';
   EXECUTE IMMEDIATE 'DROP TABLE Customers CASCADE CONSTRAINTS';
EXCEPTION
   WHEN OTHERS THEN NULL;
END;
/


-- Customers Table
CREATE TABLE Customers (
   customer_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
   first_name VARCHAR2(50) NOT NULL,
```

```
    last_name VARCHAR2(50) NOT NULL,
    email VARCHAR2(100) UNIQUE NOT NULL,
    phone VARCHAR2(20),
    address CLOB,
    created_at TIMESTAMP DEFAULT SYSTIMESTAMP
);

-- Categories Table
CREATE TABLE Categories (
    category_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    category_name VARCHAR2(50) UNIQUE NOT NULL,
    description CLOB
);

-- Suppliers Table
CREATE TABLE Suppliers (
    supplier_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    supplier_name VARCHAR2(100) NOT NULL,
    contact_name VARCHAR2(50),
```

```
    email VARCHAR2(100) UNIQUE,
    phone VARCHAR2(20),
    address CLOB
);

-- Products Table
CREATE TABLE Products (
    product_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    product_name VARCHAR2(100) NOT NULL,
    category_id NUMBER,
    price NUMBER(10,2) NOT NULL,
    product_size VARCHAR2(10), -- Renamed from SIZE to product_size
    color VARCHAR2(50),
    stock_quantity NUMBER DEFAULT 0,
    supplier_id NUMBER,
    FOREIGN KEY (category_id) REFERENCES Categories(category_id) ON DELETE CASCADE,
    FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id) ON DELETE SET NULL
);
```

```
-- Orders Table
CREATE TABLE Orders (
    order_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    customer_id NUMBER,
    order_date TIMESTAMP DEFAULT SYSTIMESTAMP,
    total_amount NUMBER(10,2),
    status VARCHAR2(20) DEFAULT 'Pending' CHECK (status IN ('Pending', 'Shipped', 'Delivered', 'Cancelled')),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id) ON DELETE CASCADE
);

-- Order_Items Table
CREATE TABLE Order_Items (
    order_item_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    order_id NUMBER,
    product_id NUMBER,
    quantity NUMBER NOT NULL,
    subtotal NUMBER(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES Products(product_id) ON DELETE CASCADE
```

```
);


-- Inventory Table
CREATE TABLE Inventory (
    inventory_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    product_id NUMBER,
    quantity NUMBER NOT NULL,
    last_updated TIMESTAMP DEFAULT SYSTIMESTAMP,
    FOREIGN KEY (product_id) REFERENCES Products(product_id) ON DELETE CASCADE
);
```

```
-- Insert into Customers
INSERT INTO Customers (first_name, last_name, email, phone, address) VALUES
('John', 'Doe', 'john.doe@email.com', '1234567890', '123 Main St'),
('Jane', 'Smith', 'jane.smith@email.com', '9876543210', '456 Oak Ave'),
('Alice', 'Johnson', 'alice.johnson@email.com', '5551112233', '789 Pine Rd'),
('Bob', 'Williams', 'bob.williams@email.com', '6662223344', '101 Maple St'),
('Charlie', 'Brown', 'charlie.brown@email.com', '7773334455', '202 Birch Ln'),
('David', 'Miller', 'david.miller@email.com', '8884445566', '303 Cedar Ct'),
```

```
('Emma', 'Davis', 'emma.davis@email.com', '9995556677', '404 Elm Blvd'),
('Frank', 'Wilson', 'frank.wilson@email.com', '1116667788', '505 Spruce Dr'),
('Grace', 'Moore', 'grace.moore@email.com', '2227778899', '606 Walnut Pl'),
('Henry', 'Taylor', 'henry.taylor@email.com', '3338889900', '707 Chestnut Ln');

-- Insert into Categories
INSERT INTO Categories (category_name, description) VALUES
('Shirts', 'Casual and formal shirts'),
('Pants', 'Denim, chinos, and joggers'),
('Jackets', 'Winter and casual jackets'),
('Shoes', 'Casual and formal footwear'),
('Accessories', 'Belts, wallets, and more'),
('Dresses', 'Casual and party wear dresses'),
('T-Shirts', 'Graphic and plain t-shirts'),
('Suits', 'Formal and business suits'),
('Hoodies', 'Winter and casual hoodies'),
('Shorts', 'Casual summer shorts');

-- Insert into Suppliers
```

```
INSERT INTO Suppliers (supplier_name, contact_name, email, phone, address) VALUES
('ABC Textiles', 'Mike Johnson', 'mike@abc.com', '4445556666', 'New York, USA'),
('XYZ Clothing', 'Sara Lee', 'sara@xyz.com', '7778889999', 'Los Angeles, USA'),
('Fashion Hub', 'Tom Clark', 'tom@fashionhub.com', '1231231234', 'Paris, France'),
('Trend Wear', 'Lily Adams', 'lily@trendwear.com', '4564564567', 'London, UK'),
('Urban Styles', 'Robert Brown', 'robert@urban.com', '7897897890', 'Milan, Italy'),
('Classic Attire', 'Emma White', 'emma@classic.com', '3213213210', 'Berlin, Germany'),
('Elite Fashion', 'William Black', 'william@elite.com', '6546546543', 'Tokyo, Japan'),
('Luxury Designs', 'Sophia Green', 'sophia@luxury.com', '9879879876', 'Seoul, South Korea'),
('Budget Clothes', 'David Carter', 'david@budget.com', '1471471470', 'Bangkok, Thailand'),
('Street Wear Co.', 'Olivia Scott', 'olivia@streetwear.com', '2582582581', 'Sydney, Australia');


-- Insert into Products
INSERT INTO Products (product_name, category_id, price, product_size, color, stock_quantity, supplier_id) VALUES
('Formal Shirt', 1, 29.99, 'M', 'Blue', 50, 1),
('Casual Jeans', 2, 49.99, 'L', 'Black', 40, 2),
('Winter Jacket', 3, 99.99, 'XL', 'Red', 30, 3),
('Sneakers', 4, 59.99, '10', 'White', 60, 4),
('Leather Belt', 5, 19.99, 'L', 'Brown', 70, 5),
```

```
('Summer Dress', 6, 39.99, 'S', 'Pink', 20, 6),
('Graphic T-Shirt', 7, 25.99, 'M', 'Green', 45, 7),
('Business Suit', 8, 199.99, 'L', 'Navy', 15, 8),
('Hooded Sweatshirt', 9, 45.99, 'XL', 'Gray', 35, 9),
('Denim Shorts', 10, 35.99, 'M', 'Blue', 25, 10);

-- Insert into Orders
INSERT INTO Orders (customer_id, total_amount, status) VALUES
(1, 79.98, 'Pending'),
(2, 49.99, 'Shipped'),
(3, 199.99, 'Delivered'),
(4, 59.99, 'Pending'),
(5, 29.99, 'Cancelled'),
(6, 99.99, 'Pending'),
(7, 39.99, 'Shipped'),
(8, 19.99, 'Delivered'),
(9, 45.99, 'Pending'),
(10, 35.99, 'Shipped');
```

```
-- Insert into Order_Items
INSERT INTO Order_Items (order_id, product_id, quantity, subtotal) VALUES
(1, 1, 2, 59.98),
(2, 2, 1, 49.99),
(3, 8, 1, 199.99),
(4, 4, 1, 59.99),
(5, 1, 1, 29.99),
(6, 3, 1, 99.99),
(7, 6, 1, 39.99),
(8, 5, 1, 19.99),
(9, 9, 1, 45.99),
(10, 10, 1, 35.99);

-- Insert into Inventory
INSERT INTO Inventory (product_id, quantity) VALUES
(1, 50),
(2, 40),
(3, 30),
(4, 60),
```

```
(5, 70),
(6, 20),
(7, 45),
(8, 15),
(9, 35),
(10, 25);

COMMIT;
```

Conclusion & Benefits

Conclusion

The **Clothing Store Database** provides a structured and efficient way to manage essential retail operations, including **customer management, product inventory, order processing, and supplier coordination**. By implementing this system, businesses can streamline workflows, reduce manual errors, and gain valuable insights into sales and inventory trends. This guidebook serves as a blueprint for setting up and optimizing the database for maximum efficiency.

Benefits of This Database System

- **Efficient Data Management** – Organizes customer, product, and order details systematically.
- **Improved Inventory Tracking** – Helps maintain accurate stock levels, reducing shortages or overstocking.
- **Enhanced Customer Insights** – Tracks customer purchases, enabling personalized marketing and better service.

- **Seamless Order Processing** – Automates order management, reducing delays and errors.
- **Supplier Coordination** – Maintains supplier records for easier procurement and restocking.
- **Scalability & Performance** – Designed to handle business growth and increased data volume efficiently.
- **Data Security & Integrity** – Ensures accurate and secure storage of business-critical information.

Future Enhancements & Use Cases

- **Integration with E-commerce Platforms** – Connect the database with online stores for real-time inventory updates.
- **Advanced Analytics & Reporting** – Implement business intelligence tools to generate sales and customer insights.
- **Automation & AI-driven Insights** – Use AI to predict sales trends and optimize stock levels.
- **Mobile Accessibility** – Develop a mobile-friendly dashboard for easy data access on the go.
- **Loyalty Programs & Customer Engagement** – Enhance customer retention with loyalty points and targeted promotions.
- **Multi-Store Support** – Expand the database to manage multiple retail locations efficiently.

Platform Used

This database was designed and implemented using **Oracle Live SQL**, a cloud-based SQL development environment. Oracle Live SQL allows users to write, test, and execute SQL queries in a web-based interface without requiring local database installation.

🔗 **Platform Link:** Oracle Live SQL

[1]

[1] This guidebook serves as a foundational resource for database management in a clothing retail environment. It is intended for developers, business owners, and IT professionals seeking to implement and optimize a structured database system. Future adaptations may be required to align with emerging retail trends and technological advancements.