SQL Queries for eCommerce Database

# Database Schema Documentation

## Overview

This document describes the database schema for managing a food eCommerce application. The schema includes tables for Users, Orders, Products, Order\_Items, and Subscriptions, along with their relationships.

**Structure:**

1. **Users Table** (Customers, Admins)
2. **Products Table** (Information about products)
3. **Subscriptions Table** (Customer subscriptions)
4. **Orders Table** (Products purchased by users)
5. **Order\_Items Table** (Linking orders and products for multiple items in an order)

**Relationships:**

* **One-to-many relationship** between Users and Orders.
* **Many-to-many relationship** between Orders and Products through the Order\_Items table.
* **One-to-one or one-to-many relationship** between Users and Subscriptions.

## Tables

## 1. Users

Stores user information.

* **user\_id** (VARCHAR(255), Primary Key) — Unique identifier for each user.
* **username** (VARCHAR(255), Not Null) — Username of the user.
* **email** (VARCHAR(255), Not Null) — Email address of the user.
* **password** (VARCHAR(255), Not Null) — Password for user authentication.
* **user\_type** (ENUM('CUSTOMER', 'ADMIN'), Default 'CUSTOMER') — Type of user (CUSTOMER or ADMIN).
* **created\_at** (TIMESTAMP, Default CURRENT\_TIMESTAMP) — Timestamp of when the user was created.

### 2. ****Orders****

Stores order information.

* **order\_id** (INT, Primary Key, Auto Increment) — Unique identifier for each order.
* **user\_id** (VARCHAR(255), Foreign Key) — References Users.user\_id.
* **order\_date** (TIMESTAMP) — Date and time when the order was placed.
* **total\_amount** (DECIMAL(10, 2)) — Total amount for the order.

### 3. ****Order\_Items****

Junction table to handle the many-to-many relationship between Orders and Products.

* **order\_id** (INT, Foreign Key) — References Orders.order\_id.
* **product\_id** (INT, Foreign Key) — References Products.product\_id.
* **quantity** (INT) — Quantity of the product ordered.
* **price** (DECIMAL(10, 2)) — Price of the product at the time of order.

### 4. ****Products****

Stores product information.

* **product\_id** (INT, Primary Key, Auto Increment) — Unique identifier for each product.
* **product\_name** (VARCHAR(255), Not Null) — Name of the product.
* **price** (DECIMAL(10, 2)) — Price of the product.
* **stock\_quantity** (INT) — Quantity of the product in stock.

### 5. ****Subscriptions****

Stores subscription information.

* **subscription\_id** (INT, Primary Key, Auto Increment) — Unique identifier for each subscription.
* **user\_id** (VARCHAR(255), Foreign Key) — References Users.user\_id.
* **product\_id** (INT, Foreign Key) — References Products.product\_id.
* **start\_date** (DATE) — Date when the subscription starts.
* **end\_date** (DATE) — Date when the subscription ends.
* **is\_active** (BOOLEAN) — Indicates whether the subscription is active.

## Relationships

1. **Users** — **Orders**: One-to-Many
   * Each user can have multiple orders.
   * Each order is associated with only one user.
2. **Orders** — **Products**: Many-to-Many (through Order\_Items)
   * Each order can include multiple products.
   * Each product can be included in multiple orders.
3. **Users** — **Subscriptions**: One-to-Many
   * Each user can have one or multiple subscriptions.
   * Each subscription is associated with only one user.

# SQL Queries:

# 1. Create Database and Use It

CREATE DATABASE ecommerce\_db;  
USE ecommerce\_db;

# 2. Create Tables

CREATE TABLE users (  
 user\_id VARCHAR(255) PRIMARY KEY,  
 username VARCHAR(50) NOT NULL,  
 email VARCHAR(100) NOT NULL UNIQUE,  
 password VARCHAR(255) NOT NULL,  
 user\_type ENUM('CUSTOMER', 'ADMIN') DEFAULT 'CUSTOMER',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
CREATE TABLE Product (  
 productId VARCHAR(50) PRIMARY KEY,  
 productName VARCHAR(100) NOT NULL,  
 productDescription TEXT,  
 price DOUBLE NOT NULL,  
 isActive BOOLEAN DEFAULT TRUE  
);  
  
CREATE TABLE IF NOT EXISTS subscriptions (  
 subscriptionId VARCHAR(255) PRIMARY KEY,  
 productId VARCHAR(255),  
 customerId VARCHAR(255),  
 type VARCHAR(50) NOT NULL,  
 startDate DATE,  
 endDate DATE,  
 isActive BOOLEAN DEFAULT TRUE,  
 FOREIGN KEY (productId) REFERENCES Product(productId),  
 FOREIGN KEY (customerId) REFERENCES users(user\_id)  
);  
  
CREATE TABLE orders (  
 order\_id VARCHAR(255) PRIMARY KEY,  
 user\_id VARCHAR(255) NOT NULL,  
 total\_amount DECIMAL(10, 2) NOT NULL,  
 order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 status ENUM('PENDING', 'SHIPPED', 'DELIVERED') DEFAULT 'PENDING',  
 FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE  
);  
  
CREATE TABLE order\_items (  
 order\_item\_id VARCHAR(255) PRIMARY KEY,  
 order\_id VARCHAR(255) NOT NULL,  
 product\_id VARCHAR(255) NOT NULL,  
 quantity INT NOT NULL,  
 price DECIMAL(10, 2) NOT NULL,  
 FOREIGN KEY (order\_id) REFERENCES orders(order\_id) ON DELETE CASCADE,  
 FOREIGN KEY (product\_id) REFERENCES Product(productId) ON DELETE CASCADE  
);

# 3. Insert Dummy Data

INSERT INTO Product (productId, productName, productDescription, price, isActive) VALUES  
('P001', 'Pizza Margherita', 'Classic pizza with tomato, mozzarella, and basil', 218.08, TRUE),  
('P002', 'Veggie Burger', 'A healthy burger with fresh vegetables and vegan cheese', 89.00, TRUE),  
('P003', 'Caesar Salad', 'Fresh romaine lettuce with Caesar dressing and croutons', 78.67, TRUE),  
('P004', 'Chocolate Cake', 'Rich and moist chocolate cake', 44.00, TRUE),  
('P005', 'Spaghetti Bolognese', 'Traditional Italian pasta with a rich meat sauce', 99.99, TRUE);  
  
INSERT INTO users (user\_id, username, email, password, user\_type)  
VALUES  
('U001', 'Aarav Sharma', 'aarav.sharma@gmail.com', 'A8d$kP!w7Z', 'ADMIN'),  
('U002', 'Isha Patel', 'isha.patel@yahoo.in', 'J@4rL9\*eQ', 'CUSTOMER'),  
('U003', 'Rohan Gupta', 'rohan.gupta@gmail.com', 'X2b&V6#sP', 'CUSTOMER'),  
('U004', 'Priya Singh', 'priya.singh@gmail.com', 'H5v%T3^wR', 'CUSTOMER'),  
('U005', 'Vikram Reddy', 'vikram.reddy@yahoo.in', 'K9u@L1\*oN', 'CUSTOMER');  
  
INSERT INTO subscriptions (subscriptionId, productId, customerId, type, startDate, endDate, isActive)  
VALUES  
('S001', 'P001', 'U002', 'MONTHLY', '2024-08-01', '2024-08-31', TRUE),  
('S002', 'P002', 'U003', 'YEARLY', '2024-07-15', '2025-07-14', TRUE),  
('S003', 'P003', 'U004', 'MONTHLY', '2024-08-10', '2024-09-09', TRUE),  
('S004', 'P004', 'U005', 'WEEKLY', '2024-08-17', '2024-08-24', TRUE),  
('S005', 'P005', 'U003', 'YEARLY', '2024-06-20', '2025-06-19', TRUE);

INSERT INTO orders (order\_id, user\_id, total\_amount, status) VALUES

('O001', 'U002', 307.08, 'PENDING'),

('O002', 'U003', 89.00, 'SHIPPED'),

('O003', 'U004', 78.67, 'DELIVERED'),

('O004', 'U005', 143.99, 'PENDING'),

('O005', 'U003', 123.67, 'DELIVERED');

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity, price) VALUES ('OI001', 'O001', 'P001', 1, 218.08),

('OI002', 'O001', 'P003', 1, 78.67),

('OI003', 'O002', 'P002', 1, 89.00),

('OI004', 'O003', 'P003', 2, 78.67),

('OI005', 'O004', 'P005', 1, 99.99),

('OI006', 'O004', 'P004', 1, 44.00),

('OI007', 'O005', 'P001', 1, 218.08),

('OI008', 'O005', 'P002', 1, 89.00);

**Relationships Recap:**

* **Users to Orders**: One-to-Many (one user can have multiple orders).
* **Orders to Products**: Many-to-Many (handled by order\_items table).
* **Users to Subscriptions**: One-to-Many (one user can have multiple subscriptions).