|  |  |
| --- | --- |
| style step  ONLINE SHOE MENEGEMENT SYSTEM | about  Shoe Management System is to ensure that the platform meets the needs of different users and  provides a seamless online shopping experience for customers  Sir Mazhar Javaid Awan  Database management system |

**Project Report**

**STYLE STEP**

**(ONLINE SHOE MENEGEMENT SYSTEM)**

****

**Submitted By:**

**MUHAMMAD ZAHID F2021065216**

**CH. ABDULLAH MUSHTAQ F2021065345**

**Muhammad arsal F2021065038**

**University of Management and Technology**

**C-II Johar Town Lahore Pakistan**

****

**DATABASE SYSTEM PROJECT**

**Project Requirement Documentation**

# Project Name:

**Style Step**

**A shoe management system**

# Analyze your environment:

* **Requirement ID:** REQ1

Name: User Registration

Description: The system should allow users to register by providing their name, email, phone number, and address, date of birth, gender, and password.

* Requirement ID: REQ2

Name: Shoe Purchase

Description: Customers should be able to browse the inventory, select a shoe, specify the size and quantity, and make a purchase. The system should record the customer's ID, shoe ID, shoe size, price, quantity, and purchase date.

* Requirement ID: REQ3

Name: Complaint Management

Description: Customers should be able to submit complaints regarding their purchases. The system should record the customer's ID, shoe ID, complaint date, complaint description, and complaint status.

* Requirement ID: REQ4

Name: Order Processing

Description: Staff members should be able to process customer orders. The system should record the order ID, customer ID, shoe ID, shoe size, shoe color, shoe price, shoe quantity, and order date.

* Requirement ID: REQ5

Name: Payment Processing

Description: The system should support different payment methods for customers to complete their purchases. It should record the payment ID, order ID, payment method, payment amount, and payment date.

* Requirement ID: REQ6

Name: Shipping and Delivery

Description: Once an order is processed, the system should generate shipping orders with a unique shipping ID. It should record the order ID, shipping address, shipping method, shipping cost, and shipping date.

* Requirement ID: REQ7

Name: Inventory Management

Description: The system should track the inventory levels of shoes. It should record the shoe ID, shoe name, shoe brand, shoe category, shoe quantity, shoe gender, shoe season, and shoe release date.

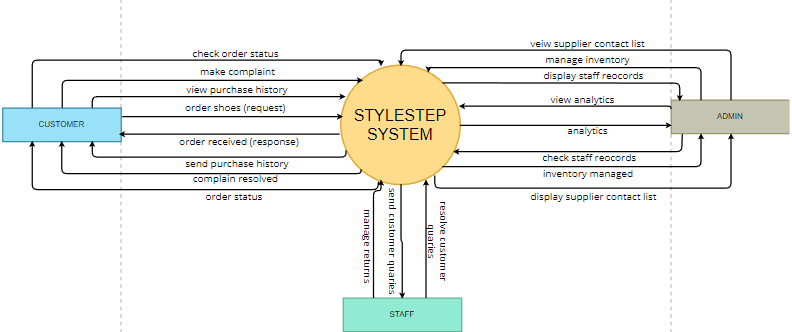
* Requirement ID: REQ8

Name: Analytics and Reporting

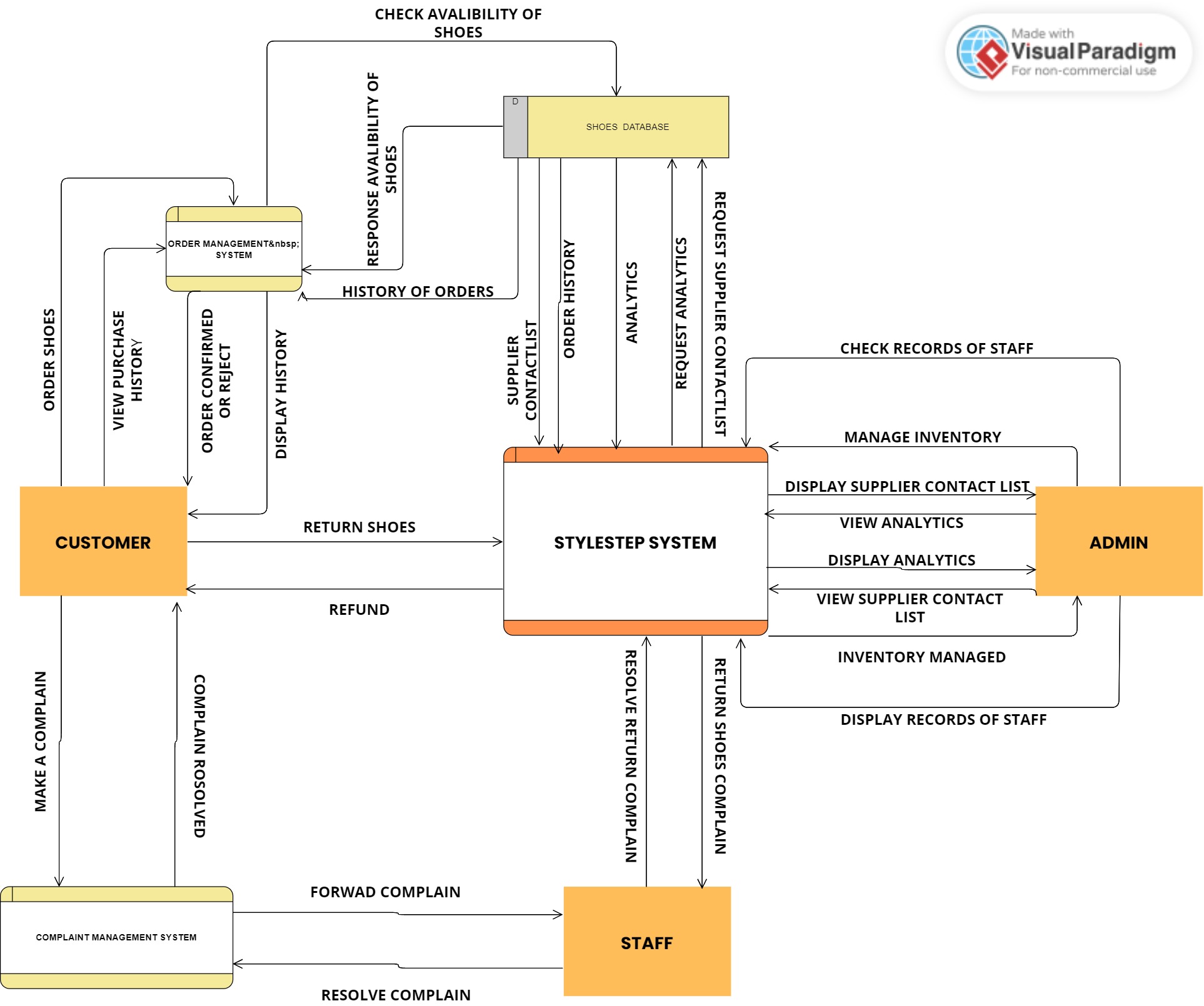
Description: The system should provide analytics and reporting capabilities, including tracking total sales, total profits, total returns, total complaints, shoe inventory levels, and monthly sales.

# Develop Logical Model:

## CONTEXT DIAGRAM:



## DATA FLOW DIAGRAM LEVEL-1:



# Tables/Entity Names:

## Customer:

* Customer ID (Primary Key)
* Customer Name
* Customer Email
* Customer Phone Number
* Customer Address
* Customer Password
* Customer Date of Birth
* Customer Gender

## Purchase History:

* Customer ID (Foreign Key)
* IdShoe (Foreign Key)
* Shoe ID (Foreign Key)
* Shoe Size
* Price
* Quantity
* Shoe Purchase Date

## Inventory:

* Shoe ID (Primary Key)
* Shoe Name
* Shoe Brand
* Shoe Category
* Shoe Quantity
* Shoe Gender
* Shoe Season
* Shoe Release Date

## Shoes:

* Shoe ID (Foreign Key)
* IdShoe (primary key)
* Shoe Size
* Shoe Color
* Shoe Price
* Shoe Description

## Complaint:

* Complaint ID (Primary Key)
* Customer ID (Foreign Key)
* Shoe ID (Foreign Key)
* Complaint Date
* Complaint Description
* Complaint Status

## Orders:

* Order ID (Primary Key)
* Customer ID (Foreign Key)
* Shoe ID (Foreign Key)
* Shoe Size
* Shoe Color
* Shoe Price
* Shoe Quantity
* Shoe Purchase Date

## Payments:

* Payment ID
* Order ID (Foreign Key)
* Payment Method
* Payment Amount
* Payment Date

## Shiping\_orders:

* Shipping ID (Primary Key)
* Order ID (Foreign Key)
* Shipping Address
* Shipping Method
* Shipping Cost
* Shipping Date

## Analytics:

* total Sales
* total Profits
* total Returns
* total Complaints
* Shoe Inventory Levels
* Month
* Id (Primary Key)

## Admin:

* Admin ID (Primary Key)
* Admin Name
* Admin Email
* Admin Phone Number
* Admin Address
* Admin Password
* Admin Role

## Staff:

* Staff (Primary Key)
* Staff Name
* Staff Email
* Staff Phone Number
* Staff Address
* Staff Password
* Staff Role

## Supplier:

* Supplier ID (Primary Key)
* Supplier Name
* Supplier Email
* Supplier Phone Number
* Supplier Address

**Weak Entities:**

* Purchase History of Customer
* Complaint
* Payments
* Shiping\_orders

**Strong Entities:**

* Customer
* Inventory
* Shoes
* Orders
* Analytics
* Admin
* Staff
* Supplier

# Single, multi, Stored, simple, composite derived Attributes:

## Customer:

* Customer ID (Single, Stored)
* Customer Name (Single, Stored)
* Customer Email (Single, Stored)
* Customer Phone Number (Single, Stored)
* Customer Address (Single, Stored)
* Customer Password (Single, Stored)
* Customer Date of Birth (Single, Stored)
* Customer Gender (Single, Stored)

## Purchase History of Customer:

* Customer ID (Multi, Stored)
* Shoe ID (Multi, Stored)
* Shoe Size (Single, Stored)
* Price (Single, Stored)
* Quantity (Single, Stored)
* Shoe Purchase Date (Single, Stored)

## Inventory:

* Shoe ID (Single, Stored)
* Shoe Name (Single, Stored)
* Shoe Brand (Single, Stored)
* Shoe Category (Single, Stored)
* Shoe Quantity (Single, Stored)
* Shoe Gender (Single, Stored)
* Shoe Season (Single, Stored)
* Shoe Release Date (Single, Stored)

## Shoes:

* Shoe ID (Single, Stored)
* Shoe Size (Single, Stored)
* Shoe Color (Single, Stored)
* Shoe Price (Single, Stored)
* Shoe Description (Single, Stored)

## Complaint:

* Complaint ID (Single, Stored)
* Customer ID (Multi, Stored)
* Shoe ID (Multi, Stored)
* Complaint Date (Single, Stored)
* Complaint Description (Single, Stored)
* Complaint Status (Single, Stored)

## Orders:

* Order ID (Single, Stored)
* Customer ID (Multi, Stored)
* Shoe ID (Multi, Stored)
* Shoe Size (Single, Stored)
* Shoe Color (Single, Stored)
* Shoe Price (Single, Stored)
* Shoe Quantity (Single, Stored)
* Shoe Purchase Date (Single, Stored)

## Payments:

* Payment ID (Single, Stored)
* Order ID (Multi, Stored)
* Payment Method (Single, Stored)
* Payment Amount (Single, Stored)
* Payment Date (Single, Stored)

## Shipping\_orders:

* Shipping ID (Single, Stored)
* Order ID (Multi, Stored)
* Shipping Address (Single, Stored)
* Shipping Method (Single, Stored)
* Shipping Cost (Single, Stored)
* Shipping Date (Single, Stored)

## Analytics:

* Total Sales (Single, Derived)
* Total Profits (Single, Derived)
* Total Returns (Single, Derived)
* Total Complaints (Single, Derived)
* Shoe Inventory Levels (Single, Derived)
* Month (Single, Derived)
* ID (Single, Stored)

## Admin:

* Admin ID (Single, Stored)
* Admin Name (Single, Stored)
* Admin Email (Single, Stored)
* Admin Phone Number (Single, Stored)
* Admin Address (Single, Stored)
* Admin Password (Single, Stored)
* Admin Role (Single, Stored)

## Staff:

* Staff ID (Single, Stored)
* Staff Name (Single, Stored)
* Staff Email (Single, Stored)
* Staff Phone Number (Single, Stored)
* Staff Address (Single, Stored)
* Staff Password (Single, Stored)
* Staff Role (Single, Stored)

## Supplier:

* Supplier ID (Single, Stored)
* Supplier Name (Single, Stored)
* Supplier Email (Single, Stored)
* Supplier Phone Number (Single, Stored)
* Supplier Address (Single, Stored)

## Primary Key, Super key, Composite Key, alternative key, foreign Key:

## Customer:

* Primary Key: Customer ID

## Purchase History of Customer:

* Foreign Key: Customer ID
* Foreign Key: IdShoe

## Inventory:

* Primary Key: Shoe ID

## Shoes:

* Primary Key: Idshoe
* Composite Key: Shoe ID + IdShoe

## Complaint:

* Primary Key: Complaint ID
* Foreign Key: Customer ID
* Foreign Key: Shoe ID

## Orders:

* Primary Key: Order ID
* Foreign Key: Customer ID
* Foreign Key: Shoe ID

## Payments:

* Primary Key: Payment ID
* Foreign Key: Order ID

## Shipping\_orders:

* Primary Key: Shipping ID
* Foreign Key: Order ID

## Analytics:

* No primary key or foreign keys mentioned

## Admin:

* Primary Key: Admin ID

## Staff:

* Primary Key: Staff ID

## Supplier:

* Primary Key: Supplier ID

# Types of relationship (One to one, one to many and many to many) of different entities with Cardinality:

## Customer (One) to Purchase History of Customer (Many):

* Cardinality: One customer can have multiple purchase history records.
* Relationship: One-to-many relationship.

## Customer (One) to Complaint (Many):

* Cardinality: One customer can have multiple complaints.
* Relationship: One-to-many relationship.

## Customer (One) to Orders (Many):

* Cardinality: One customer can place multiple orders.
* Relationship: One-to-many relationship.

## Orders (One) to Payments (Many):

* Cardinality: One order can have multiple payments.
* Relationship: One-to-many relationship.

## Orders (One) to Shipping Orders (One):

* Cardinality: One order can have one shipping order.
* Relationship: One-to-one relationship.

## Shoes (One) to Purchase History of Customer (Many):

* Cardinality: One shoe can appear in multiple purchase history records.
* Relationship: One-to-many relationship.

## Shoes (One) to Orders (Many):

* Cardinality: One shoe can appear in multiple orders.
* Relationship: One-to-many relationship.

## Admin (One) to Staff (Many):

* Cardinality: One admin can have multiple staff members.
* Relationship: One-to-many relationship.

## Shoes (One) to Inventory (One):

* Cardinality: One shoe can have one inventory record.
* Relationship: One-to-one relationship.

## Supplier (One) to Inventory (Many):

* Cardinality: One supplier can supply multiple shoes in the inventory.
* Relationship: One-to-many relationship.

# Existence and Referential dependencies:

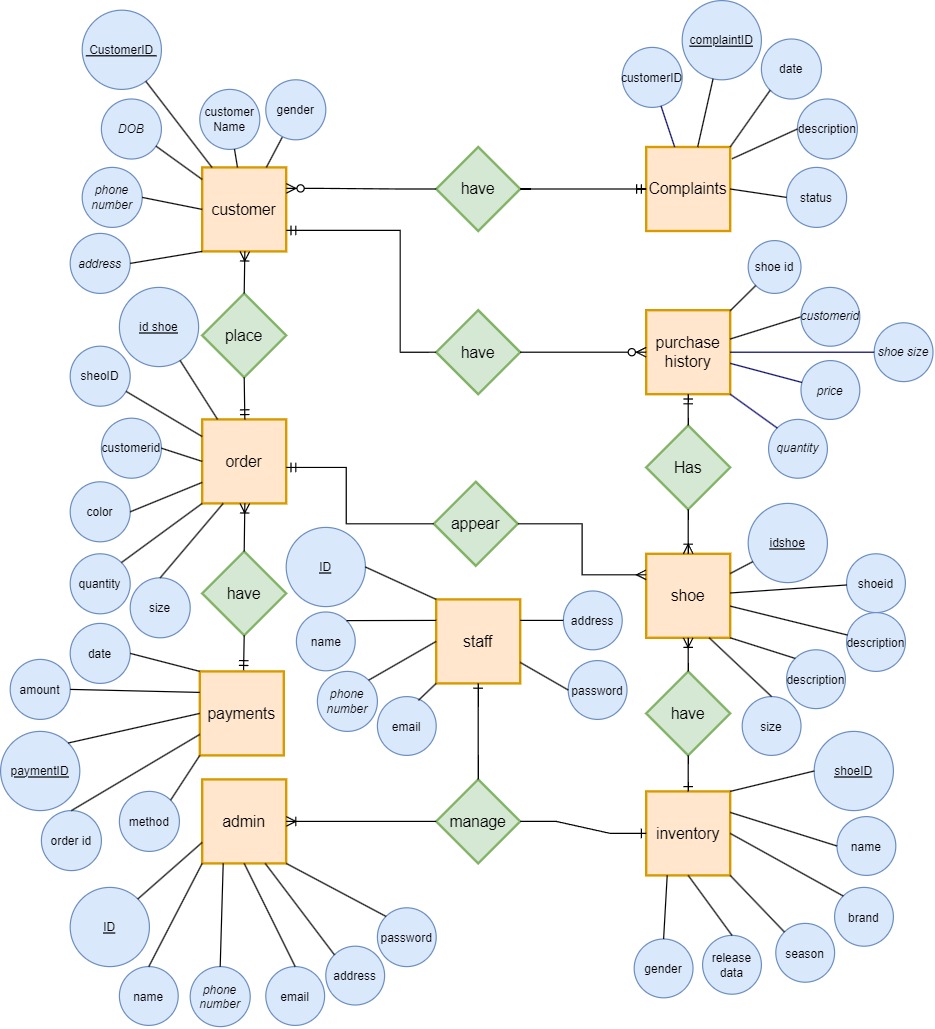
## Existence Dependencies:

* Purchase History of Customer depends on the existence of a Customer. A purchase history record cannot exist without a corresponding customer.
* Complaint depends on the existence of a Customer and a Shoe. A complaint cannot exist without a corresponding customer and shoe.
* Orders depend on the existence of a Customer and a Shoe. An order cannot exist without a corresponding customer and shoe.
* Payments depend on the existence of an Order. A payment cannot exist without a corresponding order.
* Shipping Orders depend on the existence of an Order. A shipping order cannot exist without a corresponding order.

## Referential Dependencies:

* Purchase History of Customer has a foreign key dependency on Customer. The Customer ID in Purchase History of Customer references the Customer table.
* Purchase History of Customer has a foreign key dependency on Shoes. The Shoe ID in Purchase History of Customer references the Shoes table.
* Complaint has a foreign key dependency on Customer. The Customer ID in Complaint references the Customer table.
* Complaint has a foreign key dependency on Shoes. The Shoe ID in Complaint references the Shoes table.
* Orders has a foreign key dependency on Customer. The Customer ID in Orders references the Customer table.
* Orders has a foreign key dependency on Shoes. The Shoe ID in Orders references the Shoes table.
* Payments has a foreign key dependency on Orders. The Order ID in Payments references the Orders table.
* Shipping Orders has a foreign key dependency on Orders. The Order ID in Shipping Orders references the Orders table.
* Shoes has a foreign key dependency on Inventory. The Shoe ID in Shoes references the Inventory table.
* Supplier has a foreign key dependency on Inventory. The Supplier ID in Supplier references the Inventory table.

# ERD:



# Functional Dependencies:

## Customer:

* Customer ID uniquely determines Customer Name, Customer Email, Customer Phone Number, Customer Address, Customer Password, Customer Date of Birth, and Customer Gender.

## Purchase History of Customer:

* Customer ID and Shoe ID together uniquely determine Shoe Size, Price, Quantity, and Shoe Purchase Date.

## Inventory:

* Shoe ID uniquely determines Shoe Name, Shoe Brand, Shoe Category, Shoe Quantity, Shoe Gender, Shoe Season, and Shoe Release Date.

## Shoes:

* Shoe ID and IdShoe together uniquely determine Shoe Size, Shoe Color, Shoe Price, Shoe Description, and Shoe Image.

## Complaint:

* Complaint ID uniquely determines Customer ID, Shoe ID, Complaint Date, Complaint Description, and Complaint Status.

## Orders:

* Order ID uniquely determines Customer ID, Shoe ID, Shoe Size, Shoe Color, Shoe Price, Shoe Quantity, and Shoe Purchase Date.

## Payments:

* Payment ID uniquely determines Order ID, Payment Method, Payment Amount, and Payment Date.

## Shipping Orders:

* Shipping ID uniquely determines Order ID, Shipping Address, Shipping Method, Shipping Cost, and Shipping Date.

## Admin:

* Admin ID uniquely determines Admin Name, Admin Email, Admin Phone Number, Admin Address, Admin Password, and Admin Role.

## Staff:

* Staff ID uniquely determines Staff Name, Staff Email, Staff Phone Number, Staff Address, Staff Password, and Staff Role.

## Supplier:

* Supplier ID uniquely determines Supplier Name, Supplier Email, Supplier Phone Number, and Supplier Address.

# Normalize of tables (1st NF, 2nd NF, 3rd NF):

## Customer (1NF, 2NF, 3NF):

* + The table has a primary key (Customer ID), and all attributes are single-valued.
  + No partial dependencies or transitive dependencies exist, as there are no composite or derived attributes.
  + The table is in 3NF as all non-key attributes depend only on the key attribute.

## Purchase History of Customer (1NF, 2NF, 3NF):

* + The table has a composite primary key (Customer ID, Shoe ID).
  + No partial dependencies or transitive dependencies exist, as all attributes depend on the entire primary key.
  + The table is in 3NF as all non-key attributes depend only on the key attributes.

## Inventory (1NF, 2NF, 3NF):

* + The table has a primary key (Shoe ID), and all attributes are single-valued.
  + No partial dependencies or transitive dependencies exist, as there are no composite or derived attributes.
  + The table is in 3NF as all non-key attributes depend only on the key attribute.

## Shoes (1NF, 2NF, 3NF):

* + The table has a composite primary key (Shoe ID, IdShoe).
  + No partial dependencies or transitive dependencies exist, as all attributes depend on the entire primary key.
  + The table is in 3NF as all non-key attributes depend only on the key attributes.

## Complaint (1NF, 2NF, 3NF):

* + The table has a primary key (Complaint ID), and all attributes are single-valued.
  + No partial dependencies or transitive dependencies exist, as there are no composite or derived attributes.
  + The table is in 3NF as all non-key attributes depend only on the key attribute.

## Orders (1NF, 2NF, 3NF):

* + The table has a primary key (Order ID), and all attributes are single-valued.
  + No partial dependencies or transitive dependencies exist, as there are no composite or derived attributes.
  + The table is in 3NF as all non-key attributes depend only on the key attribute.

## Payments (1NF, 2NF, 3NF):

* + The table has a primary key (Payment ID), and all attributes are single-valued.
  + No partial dependencies or transitive dependencies exist, as there are no composite or derived attributes.
  + The table is in 3NF as all non-key attributes depend only on the key attribute.

## Shipping Orders (1NF, 2NF, 3NF):

* + The table has a primary key (Shipping ID), and all attributes are single-valued.
  + No partial dependencies or transitive dependencies exist, as there are no composite or derived attributes.
  + The table is in 3NF as all non-key attributes depend only on the key attribute.

## Analytics, Admin, Staff, and Supplier tables:

* + Since the attribute details are not provided, it is not possible to determine the normalization levels for these tables without knowing their functional dependencies.

## Applying Physical Model

# Create Query: Create all the tables:

-- Create the database

CREATE DATABASE shoe\_store;

-- Use the shoe\_store database

USE shoe\_store;

-- Create the Customer table

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY,

CustomerName VARCHAR(255) NOT NULL,

CustomerEmail VARCHAR(255) NOT NULL,

CustomerPhoneNumber VARCHAR(20) NOT NULL,

CustomerAddress VARCHAR(255) NOT NULL,

CustomerPassword VARCHAR(255) NOT NULL,

CustomerDateOfBirth DATE NOT NULL,

CustomerGender VARCHAR(10) NOT NULL

);

-- Create the Inventory table

CREATE TABLE Inventory (

ShoeID INT PRIMARY KEY,

ShoeName VARCHAR(255),

ShoeBrand VARCHAR(255),

ShoeCategory VARCHAR(255),

ShoeQuantity INT,

ShoeGender VARCHAR(10),

ShoeSeason VARCHAR(50),

ShoeReleaseDate DATE

);

-- Create the Shoes table

CREATE TABLE Shoes (

ShoeID INT,

IdShoe int primary key,

ShoeSize INT,

ShoeColor VARCHAR(50),

ShoePrice DECIMAL(10, 2),

ShoeDescription TEXT,

FOREIGN KEY (ShoeID) REFERENCES Inventory(ShoeID)

);

-- Create the Complaint table

CREATE TABLE Complaint (

ComplaintID INT PRIMARY KEY,

CustomerID INT,

ShoeID INT,

ComplaintDate DATE,

ComplaintDescription TEXT,

ComplaintStatus VARCHAR(50),

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

FOREIGN KEY (ShoeID) REFERENCES Shoes(ShoeID)

);

-- Create the Orders table

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

ShoeID INT,

ShoeSize INT,

ShoeColor VARCHAR(50),

ShoePrice DECIMAL(10, 2),

ShoeQuantity INT,

ShoePurchaseDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

FOREIGN KEY (ShoeID) REFERENCES Shoes(ShoeID)

);

-- Create the Purchase History table

CREATE TABLE PurchaseHistory (

CustomerID INT,

IdShoe INT,

OrderID int,

ShoeSize INT,

Price DECIMAL(10, 2),

Quantity INT,

ShoePurchaseDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

FOREIGN KEY (IdShoe) REFERENCES Shoes(IdShoe),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

-- Create the Payments table

CREATE TABLE Payments (

PaymentID INT,

OrderID INT,

PaymentMethod VARCHAR(50),

PaymentAmount DECIMAL(10, 2),

PaymentDate DATE,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

-- Create the Shipping\_orders table

CREATE TABLE Shipping\_orders (

ShippingID INT PRIMARY KEY,

OrderID INT,

ShippingAddress VARCHAR(255),

ShippingMethod VARCHAR(50),

ShippingCost DECIMAL(10, 2),

ShippingDate DATE,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

-- Create the Analytics table

CREATE TABLE Analytics (

TotalSales DECIMAL(10, 2),

TotalProfits DECIMAL(10, 2),

TotalReturns INT,

TotalComplaints INT,

ShoeInventoryLevels INT,

Month VARCHAR(20),

ID INT PRIMARY KEY AUTO\_INCREMENT

);

-- Create the Admin table

CREATE TABLE Admin (

AdminID INT PRIMARY KEY,

AdminName VARCHAR(255),

AdminEmail VARCHAR(255),

AdminPhoneNumber VARCHAR(20),

AdminAddress VARCHAR(255),

AdminPassword VARCHAR(255),

AdminRole VARCHAR(50)

);

-- Create the Staff table

CREATE TABLE Staff (

StaffID INT PRIMARY KEY,

StaffName VARCHAR(255),

StaffEmail VARCHAR(255),

StaffPhoneNumber VARCHAR(20),

StaffAddress VARCHAR(255),

StaffPassword VARCHAR(255),

StaffRole VARCHAR(50)

);

-- Create the Supplier table

CREATE TABLE Supplier (

SupplierID INT PRIMARY KEY,

SupplierName VARCHAR(255),

SupplierEmail VARCHAR(255),

SupplierPhoneNumber VARCHAR(20),

SupplierAddress VARCHAR(255)

);

# Insert Query: Insert one value in each table:

INSERT INTO Customer (CustomerID, CustomerName, CustomerEmail, CustomerPhoneNumber, CustomerAddress, CustomerPassword, CustomerDateOfBirth, CustomerGender)

VALUES (1, 'John Doe', 'johndoe@example.com', '1234567890', '123 Main St', 'password123', '1990-01-01', 'Male'),

(2, 'Jane Smith', 'janesmith@example.com', '9876543210', '456 Elm St', 'qwerty456', '1995-05-10', 'Female'),

(3, 'Alice Johnson', 'alicejohnson@example.com', '1231231234', '789 Oak St', 'pass1234', '1992-09-15', 'Female'),

(4, 'Bob Anderson', 'bobanderson@example.com', '9879879876', '321 Pine St', 'securepwd', '1985-07-20', 'Male'),

(5, 'Emily Davis', 'emilydavis@example.com', '5555555555', '456 Maple Ave', 'mypassword', '1998-12-03', 'Female'),

(6, 'Michael Wilson', 'michaelwilson@example.com', '9998887777', '987 Cedar Ln', 'password1', '1993-04-28', 'Male'),

(7, 'Olivia Thompson', 'oliviathompson@example.com', '1112223333', '654 Birch Rd', 'letmein', '1997-11-11', 'Female'),

(8, 'David Lee', 'davidlee@example.com', '4444444444', '789 Elm St', 'password123', '1991-06-02', 'Male'),

(9, 'Sophia Martinez', 'sophiamartinez@example.com', '7777777777', '123 Oak St', '12345678', '1994-03-18', 'Female'),

(10, 'James Taylor', 'jamestaylor@example.com', '6666666666', '321 Pine Ave', 'abc123', '1988-09-09', 'Male'),

(11, 'Emma Clark', 'emmaclark@example.com', '2223334444', '456 Maple St', 'qwerty', '1996-02-14', 'Female'),

(12, 'Daniel Hernandez', 'danielhernandez@example.com', '8888888888', '987 Cedar Ave', 'password456', '1990-08-07', 'Male');

INSERT INTO Inventory (ShoeID, ShoeName, ShoeBrand, ShoeCategory, ShoeQuantity, ShoeGender, ShoeSeason, ShoeReleaseDate)

VALUES (1, 'Running Shoes', 'Nike', 'Sports', 50, 'Male', 'Summer', '2022-03-01'),

(2, 'Sneakers', 'Adidas', 'Casual', 30, 'Female', 'All Seasons', '2022-06-15'),

(3, 'Sandals', 'Reef', 'Casual', 20, 'Male', 'Summer', '2022-04-10'),

(4, 'Boots', 'Timberland', 'Outdoor', 40, 'Male', 'Winter', '2022-09-20'),

(5, 'Heels', 'Jimmy Choo', 'Formal', 15, 'Female', 'All Seasons', '2022-05-05'),

(6, 'Flip Flops', 'Havaianas', 'Casual', 25, 'Female', 'Summer', '2022-06-30'),

(7, 'Loafers', 'Gucci', 'Formal', 35, 'Male', 'All Seasons', '2022-02-15'),

(8, 'Slippers', 'UGG', 'Casual', 10, 'Female', 'Winter', '2022-11-10');

INSERT INTO Shoes (IdShoe, ShoeID, ShoeSize, ShoeColor, ShoePrice, ShoeDescription)

VALUES (1, 1, 9, 'Blue', 99.99, 'Comfortable running shoes'),

(2, 1, 10, 'Red', 99.99, 'Comfortable running shoes'),

(3, 2, 7, 'Black', 79.99, 'Stylish sneakers'),

(4, 2, 8, 'White', 79.99, 'Stylish sneakers'),

(5, 3, 11, 'Brown', 49.99, 'Casual sandals'),

(6, 3, 12, 'Black', 49.99, 'Casual sandals'),

(7, 4, 10, 'Brown', 149.99, 'Durable boots'),

(8, 5, 8, 'Silver', 199.99, 'Elegant heels'),

(9, 6, 9, 'Blue', 29.99, 'Comfortable flip flops'),

(10, 7, 9, 'Brown', 199.99, 'Stylish loafers'),

(11, 8, 7, 'Pink', 49.99, 'Cozy slippers');

INSERT INTO Orders (OrderID, CustomerID, ShoeID, ShoeSize, ShoeColor, ShoePrice, ShoeQuantity, ShoePurchaseDate)

VALUES (1, 1, 2, 9, 'Blue', 99.99, 1, '2023-05-15'),

(2, 3, 5, 8, 'Silver', 199.99, 2, '2023-05-20'),

(3, 2, 3, 11, 'Brown', 49.99, 1, '2023-05-22'),

(4, 4, 1, 10, 'Red', 99.99, 1, '2023-05-25'),

(5, 1, 4, 9, 'Blue', 29.99, 1, '2023-05-26'),

(6, 2, 1, 8, 'Black', 79.99, 1, '2023-05-27'),

(7, 3, 6, 7, 'White', 49.99, 1, '2023-05-28'),

(8, 4, 3, 10, 'Brown', 79.99, 1, '2023-05-28');

INSERT INTO PurchaseHistory (CustomerID, IdShoe, ShoeSize, Price, Quantity, ShoePurchaseDate)

VALUES (1, 1, 9, 99.99, 2, '2023-05-15'),

(1, 2, 7, 79.99, 1, '2023-05-18'),

(2, 1, 8, 99.99, 1, '2023-05-20'),

(2, 2, 7, 79.99, 3, '2023-05-22'),

(3, 1, 9, 99.99, 1, '2023-05-24'),

(4, 2, 8, 79.99, 2, '2023-05-25'),

(5, 1, 10, 99.99, 2, '2023-05-26'),

(5, 2, 7, 79.99, 1, '2023-05-26'),

(6, 1, 9, 99.99, 1, '2023-05-27'),

(6, 2, 7, 79.99, 1, '2023-05-27');

INSERT INTO Complaint (ComplaintID, CustomerID, ShoeID, ComplaintDate, ComplaintDescription, ComplaintStatus)

VALUES (1, 1, 2, '2023-05-15', 'The shoes arrived damaged.', 'Open'),

(2, 3, 5, '2023-05-20', 'Received the wrong shoe size.', 'Open'),

(3, 2, 3, '2023-05-22', 'Shoe color is different from the picture.', 'Open'),

(4, 4, 1, '2023-05-25', 'The shoe sole is coming apart.', 'Open'),

(5, 1, 4, '2023-05-26', 'Shoe size is smaller than expected.', 'Open');

INSERT INTO Payments (PaymentID, OrderID, PaymentMethod, PaymentAmount, PaymentDate)

VALUES (1, 1, 'Credit Card', 99.99, '2023-05-15'),

(2, 2, 'PayPal', 399.98, '2023-05-20'),

(3, 3, 'Credit Card', 49.99, '2023-05-22'),

(4, 4, 'Cash', 99.99, '2023-05-25'),

(5, 5, 'Credit Card', 29.99, '2023-05-26'),

(6, 6, 'PayPal', 79.99, '2023-05-27'),

(7, 7, 'Credit Card', 49.99, '2023-05-28'),

(8, 8, 'Credit Card', 79.99, '2023-05-28');

INSERT INTO Shipping\_orders (ShippingID, OrderID, ShippingAddress, ShippingMethod, ShippingCost, ShippingDate)

VALUES (1, 1, '123 Main Street, City A', 'Standard Shipping', 5.99, '2023-05-15'),

(2, 2, '456 Elm Street, City B', 'Express Shipping', 12.99, '2023-05-20'),

(3, 3, '789 Oak Street, City C', 'Standard Shipping', 5.99, '2023-05-22'),

(4, 4, '987 Pine Street, City D', 'Pickup', 0.00, '2023-05-25'),

(5, 5, '321 Cedar Street, City E', 'Standard Shipping', 5.99, '2023-05-26'),

(6, 6, '654 Maple Street, City F', 'Express Shipping', 12.99, '2023-05-27'),

(7, 7, '987 Oak Street, City G', 'Standard Shipping', 5.99, '2023-05-28'),

(8, 8, '789 Pine Street, City H', 'Standard Shipping', 5.99, '2023-05-28');

INSERT INTO Analytics (TotalSales, TotalProfits, TotalReturns, TotalComplaints, ShoeInventoryLevels, Month)

VALUES (5000.00, 2500.00, 100.00, 5, 1000, 'May 2023'),

(6000.00, 3000.00, 150.00, 8, 950, 'June 2023'),

(4500.00, 2000.00, 75.00, 3, 900, 'July 2023');

INSERT INTO Admin (AdminID, AdminName, AdminEmail, AdminPhoneNumber, AdminAddress, AdminPassword, AdminRole)

VALUES (1, 'John Doe', 'johndoe@example.com', '1234567890', '123 Main Street, City A', 'admin123', 'Administrator'),

(2, 'Jane Smith', 'janesmith@example.com', '9876543210', '456 Elm Street, City B', 'admin456', 'Administrator');

INSERT INTO Staff (StaffID, StaffName, StaffEmail, StaffPhoneNumber, StaffAddress, StaffPassword, StaffRole)

VALUES (1, 'Mike Johnson', 'mike@example.com', '1234567890', '123 Main Street, City A', 'staff123', 'Sales Representative'),

(2, 'Sarah Davis', 'sarah@example.com', '9876543210', '456 Elm Street, City B', 'staff456', 'Customer Support');

INSERT INTO Supplier (SupplierID, SupplierName, SupplierEmail, SupplierPhoneNumber, SupplierAddress)

VALUES (1, 'Nike', 'nike@example.com', '+1 (123) 456-7890', '123 Main Street, City A'),

(2, 'Adidas', 'adidas@example.com', '+1 (987) 654-3210', '456 Elm Street, City B');

# Alter Query: alter few things in your project tables:

* ALTER TABLE Customer ADD COLUMN CustomerStatus VARCHAR(20);
* ALTER TABLE Customer MODIFY COLUMN CustomerPhoneNumber VARCHAR(15);
* ALTER TABLE Inventory CHANGE COLUMN ShoeBrand Brand VARCHAR(50);
* ALTER TABLE Shoes MODIFY COLUMN ShoePrice DECIMAL(10, 2) NOT NULL;
* ALTER TABLE PurchaseHistory ADD CONSTRAINT FK\_CustomerID FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);
* ALTER TABLE Inventory MODIFY COLUMN ShoeQuantity INT;
* ALTER TABLE Customer DROP COLUMN CustomerStatus;

# Drop Query: drop few tables:

* DROP TABLE Complaint;
* DROP TABLE Orders;
* DROP TABLE Payments;

# Select with Where Clause all types retrieval:

* SELECT \* FROM Inventory WHERE ShoeBrand = 'Nike';
* SELECT \* FROM Shoes WHERE ShoeColor = 'Black' AND ShoePrice BETWEEN 50 AND 100;
* SELECT \* FROM Complaint WHERE ComplaintStatus = 'Open';
* SELECT \* FROM Shipping\_orders WHERE ShippingMethod = 'Express';
* SELECT SUM(Price) AS TotalSales, SUM(Price \* Quantity) AS TotalProfits FROM PurchaseHistory;
* SELECT COUNT(\*) AS TotalReturns FROM PurchaseHistory WHERE Quantity > 0;
* SELECT COUNT(\*) AS TotalComplaints FROM Complaint;
* SELECT \* FROM Customer WHERE CustomerGender = 'Female';
* SELECT \* FROM Inventory WHERE ShoeQuantity > 10;
* SELECT \* FROM Inventory WHERE ShoeSeason = 'Summer';
* SELECT \* FROM Complaint WHERE ComplaintDate BETWEEN '2023-01-01' AND '2023-06-30';
* SELECT \* FROM Orders WHERE ShoeColor = 'Red' AND ShoeQuantity < 5;
* SELECT \* FROM Payments WHERE PaymentAmount BETWEEN 100 AND 500;
* SELECT \* FROM Shipping\_orders WHERE ShippingCost BETWEEN 10 AND 20;
* SELECT \* FROM Customer WHERE MONTH(CustomerDateOfBirth) = 5;

# Select Query: Retrieve few values from your table (using where and Order, group by, having by Clause use in it):

* SELECT \* FROM Customer WHERE CustomerGender = 'Female' ORDER BY CustomerName;
* SELECT ShoeBrand, SUM(ShoeQuantity) AS TotalQuantity, AVG(ShoePrice) AS AveragePrice FROM Inventory,Shoes GROUP BY ShoeBrand;
* SELECT OrderID, SUM(ShoePrice \* ShoeQuantity) AS TotalPrice FROM Orders GROUP BY OrderID HAVING TotalPrice > 100;
* SELECT CustomerID, COUNT(\*) AS ComplaintCount FROM Complaint GROUP BY CustomerID HAVING ComplaintCount < 3;
* SELECT Customer.CustomerID, Customer.CustomerName, SUM(PurchaseHistory.Price) AS TotalPurchaseAmount FROM PurchaseHistory JOIN Customer ON PurchaseHistory.CustomerID = Customer.CustomerID GROUP BY Customer.CustomerID, Customer.CustomerName HAVING TotalPurchaseAmount BETWEEN 100 AND 500 ORDER BY TotalPurchaseAmount DESC;
* SELECT ShoeCategory, SUM(ShoeQuantity) AS TotalQuantity FROM Inventory GROUP BY ShoeCategory ORDER BY TotalQuantity DESC;

# Apply Referential Integrity Constraint:

* ALTER TABLE PurchaseHistory ADD CONSTRAINT FK\_PurchaseHistory\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID);
* ALTER TABLE Complaint ADD CONSTRAINT FK\_Complaint\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID);
* ALTER TABLE PurchaseHistory ADD CONSTRAINT FK\_PurchaseHistory\_Inventory FOREIGN KEY (ShoeID) REFERENCES Inventory (ShoeID);
* ALTER TABLE Orders ADD CONSTRAINT FK\_Orders\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID);
* ALTER TABLE Orders ADD CONSTRAINT FK\_Orders\_Inventory FOREIGN KEY (ShoeID) REFERENCES Inventory (ShoeID);

# Apply Joins between tables (ALL TYPES):

* SELECT Inventory.ShoeName,Shoes.ShoePrice FROM Inventory inner JOIN Shoes ON Inventory.ShoeID = Shoes.ShoeID;
* SELECT Customer.CustomerName, PurchaseHistory.IdShoe, PurchaseHistory.Price FROM Customer LEFT JOIN PurchaseHistory ON Customer.CustomerID = PurchaseHistory.CustomerID;
* SELECT Customer.CustomerName, PurchaseHistory.IdShoe, PurchaseHistory.Price FROM Customer RIGHT JOIN PurchaseHistory ON Customer.CustomerID = PurchaseHistory.CustomerID;
* SELECT Customer.CustomerName, PurchaseHistory.IdShoe, PurchaseHistory.Price FROM Customer CROSS JOIN PurchaseHistory ON Customer.CustomerID = PurchaseHistory.CustomerID;
* SELECT Customer.CustomerName, Inventory.ShoeName FROM Customer INNER JOIN PurchaseHistory ON Customer.CustomerID = PurchaseHistory.CustomerID INNER JOIN Inventory ON PurchaseHistory.IdShoe = Inventory.ShoeID;
* SELECT Customer.CustomerName, Inventory.ShoeName, Payments.PaymentMethod FROM Customer JOIN Orders ON Customer.CustomerID = Orders.CustomerID JOIN Inventory ON Orders.ShoeID = Inventory.ShoeID JOIN Payments ON Orders.OrderID = Payments.OrderID WHERE Orders.OrderID=1;
* SELECT Customer.CustomerName, SUM(PurchaseHistory.Price) AS TotalPurchaseAmount FROM Customer JOIN PurchaseHistory ON Customer.CustomerID = PurchaseHistory.CustomerID GROUP BY Customer.CustomerName;

# Sub Query:

* SELECT CustomerName FROM Customer WHERE CustomerID IN ( SELECT CustomerID FROM PurchaseHistory );
* SELECT CustomerName, (SELECT COUNT(\*) FROM PurchaseHistory WHERE CustomerID = Customer.CustomerID) AS PurchaseCount FROM Customer;
* SELECT IdShoe, AVG(Price) AS AveragePrice FROM PurchaseHistory GROUP BY IdShoe HAVING AVG(Price) > (SELECT AVG(Price) FROM PurchaseHistory);

Email: mazhar.awan@umt.edu.pk