AN SQL PROJECT
FOR
“Target Corporation
Dayton Hudson
Corporation”

TABLE OF CONTENTS

1. Introduction

Name: Target Corporation (formerly Dayton Hudson Corporation)

Business Sector: Retail

Description:

Initially established as Dayton Hudson Corporation in 1902, Target Corporation has developed into a dominant power in the American retail industry. Operating under the name Target, the corporation has a huge network of stores that sell a wide variety of commodities, including groceries, home goods, electronics, and clothing. Target has developed a unique brand identity and is well-known for its dedication to offering high-quality products at reasonable costs.

Establishing an effective data management system specifically designed to assist Target's retail operations is the primary aim of this project. The primary focus of this project is to establish a comprehensive data management system tailored to support Target's retail operations. Encompassing various dimensions of the organization, including product inventories, store locations, inventory levels, suppliers, customer interactions, transactions, and discount information, the database will serve as a robust platform. This will help to optimize inventory management processes, enhance operational performance, elevate the standard for customer service, enable data-driven decision-making, and advance a better understanding of sales patterns in the retail sector.

Why did we pick this organization?

The selection of Target Corporation for this project stems from its notable position in the retail sector and the range of operational difficulties it faces. Target's expansive product range, spanning from clothing to electronics and groceries, presents a complex web of interactions that warrants in-depth database analysis. Because of its complexity, it provides a rare chance to explore the obstacles of managing large inventory, a wide range of client profiles, and a large number of transactions.

Online marketplaces remain the foundation for user-friendly purchasing experiences as the worldwide increase in e-commerce continues to transform the retail environment. Target Corporation, a giant retail corporation, exhibits e-commerce's disruptive influence. Notably, Target recorded a significant 40% increase in online sales after implementing its e-commerce platform, highlighting the importance of digital retail initiatives. Target's intentional adoption of e-commerce corresponds with modern customer tastes and serves as an appealing setting for in-depth database research. By using Target's e-commerce platform as the focal point for our database model, we hope to investigate the complexities of various parts of a large-scale organization's retail operations.

2. Data sources

Data for the database entities was sourced through a combination of methods:

- Web scraping of the Target website for Products, Product_Category, Product_Inventory and Discount entity information.
- Google searches to compile comprehensive details about various Target store locations, including geographic locations and store-specific details.
- Simulated data was generated for entities such as Customers, Payment_Details, Orders, and Order_Items. This approach ensures the protection of sensitive information while mimicking the structure and relationships found in Target's actual database.

3. Entities and Attributes

Product • Product_ID (PK), • Product_Name, • Category, • Price, • Description	**Supplier** • Supplier_ID (PK), • Supplier_Name, • Contact_Name, • Contact_Email, • Contact_Phone	**Store** • Store_ID (PK), • Store_Name, • Address, • City, • State, • Zip_Code, • Inventory_ID (FK)
Customer • Customer_ID (PK), • Customer_Type(Member?Non-Member?) **Member** • First_Name, • Last_Name, • Email, Phone, **Non-Member** • Phone	**Discount** • Discount_ID (PK), • Discount_Name, • Discount_Type, • Discount_Amount, • Applicable_Products, • Start_Date, • End_Date, • Product_ID (FK)	**Sales** • Sale_ID (PK), • Quantity, • Total_Sales, • Product_Category, • Product_ID (FK), • Discount_ID (FK) • Sale_Type(Online_Sale?, Instore_Sale?) **Online_Sale** • Website **Instore_Sale** • Store_Location
Orders • Order_ID (PK), • Order_Date, • Ship_Date, • Customer_ID (FK), • Product_ID (FK), • Store_ID (FK)	**Inventory** • Inventory_ID (PK), • Quantity_In_Stock, • Reorder_Level, • Last_Restocked_Date	

4. Description of the entities and attributes:

Records detailed information about Target's products:

➔ Product:
- Product_ID (Primary Key): An identifier unique to each product.
- Product_Name: The distinctive name of the product.
- Category: The specific category to which the product belongs.
- Price: The retail price of the product.
- Description: A detailed description providing additional information about the product.
- Supplier_ID (Foreign Key): An identifier referencing the Supplier entity.
- Discount_ID (Foreign Key): An identifier referencing the Discount entity.

Overseas supplier information to ensure efficient supply chain management:

➔ Supplier:
- Supplier_ID (Primary Key): An identification code unique to each supplier.
- Supplier_Name: The official name of the supplier.
- Contact_Name: The name of the contact person at the supplier's end.
- Contact_Email: The email address of the contact person.
- Contact_Phone: The phone number of the contact person.

Target's various stores, including unique identifiers, names, addresses, and location details:

➔ Store:
- Store_ID (Primary Key): An identifier uniquely assigned to each Target store.
- Store_Name: The official name of the Target store.
- Address: The physical address of the Target store.
- City: The city in which the Target store is located.
- State: The state in which the Target store operates.
- Zip_Code: The postal code of the Target store's location.

Information about customers shopping in Target:

➔ Customer:
- Customer_ID (Primary Key): A unique identifier for each Target customer.
- First_Name: The first name of the Target customer.
- Last_Name: The last name of the Target customer.
- Email: The email address of the Target customer.
- Phone: The phone number of the Target customer.

- Customer_Type: The categorization of the Target customer (e.g., regular, premium).
- Discount_ID (Foreign Key): An identifier referencing the Discount entity.

To oversee the management of data associated with Target's discounts, such as special codes, names, quantities, and the length of the discount:

➔ Discount:
- Discount_ID (Primary Key): An identifier uniquely assigned to each discount offered by Target.
- Discount_Name: The name associated with the Target discount.
- Discount_Type: The type of discount offered by Target (e.g., percentage, fixed amount).
- Discount_Amount: The value or percentage of the Target discount.
- Applicable_Products: A list of products to which the Target discount is applicable.
- Start_Date: The starting date of the Target discount.
- End_Date: The ending date of the Target discount.

To track sales transactions, monitoring quantities sold and total sales for insights into product demand and revenue.

➔ Sales:
- Sale_ID (Primary Key): An identifier for each Target sales transaction.
- Quantity: The quantity of products sold in each Target transaction.
- Total_Sales: The total sales amount generated at Target.
- Product_Category: Each product belongs to a category at Target.
- Type: The classification or type of each product category at Target.
- Order_ID (Foreign Key): An identifier referencing the Orders entity.

To capture information about customer orders, including unique identifiers and associated product details:

➔ Orders:
- Order_ID (Primary Key): A unique identifier for each Target order placed.
- Order_Date: The date when the Target order was placed.
- Ship_Date: The date when the Target order was shipped.
- Customer_ID (Foreign Key): An identifier referencing the Target Customer entity.
- Product_ID (Foreign Key): An identifier referencing the Target Product entity.

To manage stock levels, tracking quantities, reorder levels, and last restock date, ensuring effective inventory management for product availability:

➔ Inventory:

- Inventory_ID (Primary Key): An identifier uniquely assigned to each inventory record at Target.
- Quantity_In_Stock: The quantity of a product currently available in stock at Target.
- Reorder_Level: The threshold level at which a product needs to be reordered at Target.
- Last_Restocked_Date: The date on which the inventory was last restocked at Target.
- Product_ID (Foreign Key): An identifier referencing the Target Product entity.
- Store_ID (Foreign Key): An identifier referencing the Target Store entity.

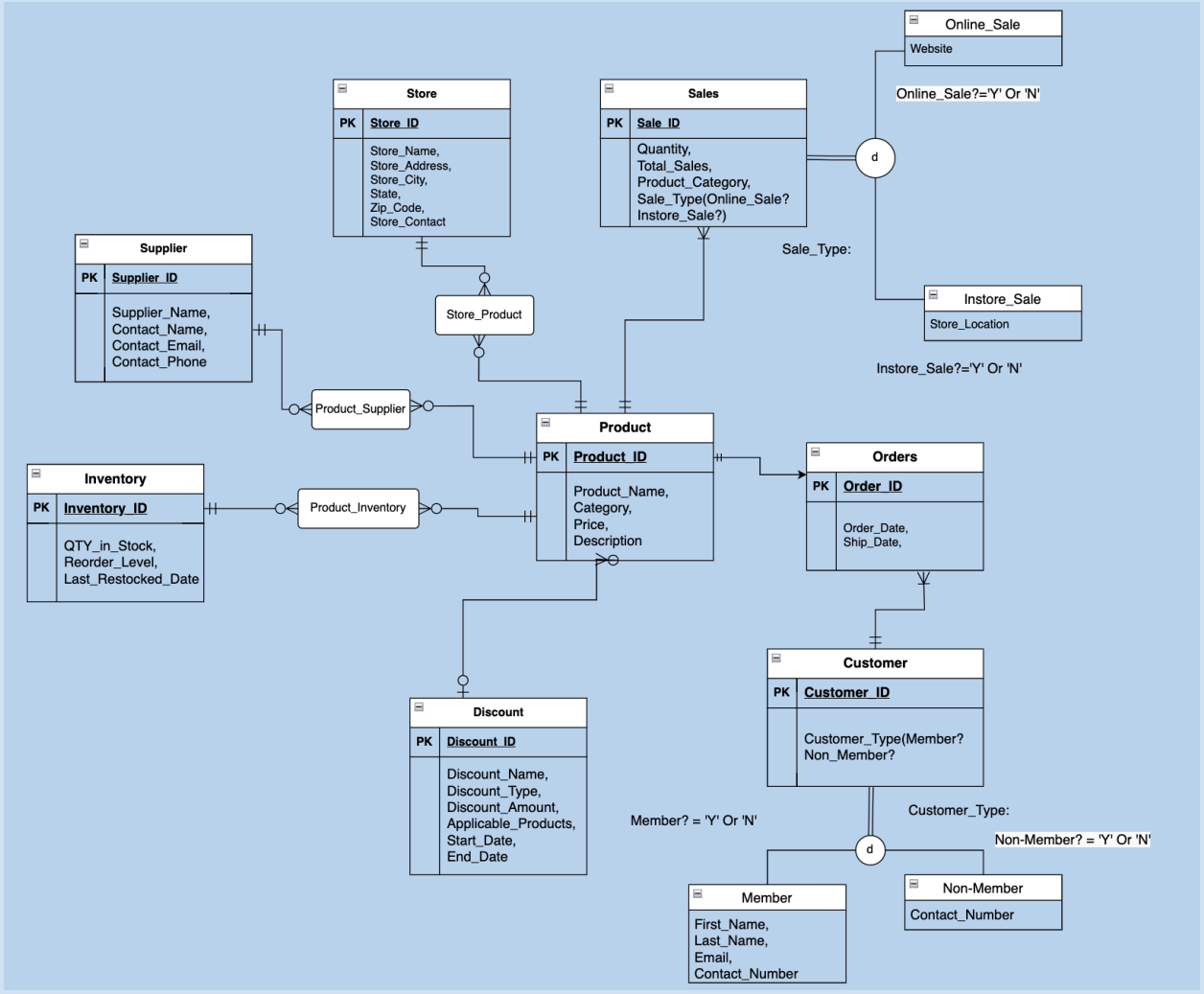
5. Business Rules

1. Customer to Orders (One-to-Many):
    a. A customer of Target Corporation can place multiple orders.
    b. Each order is associated with one customer of Target, identified by the Customer_ID foreign key.
2. Product to Sales (One-to-Many):
    a. A product available at Target can be part of multiple sales transactions.
    b. Each sale is associated with one product, identified by the Product_ID foreign key.
3. Discount to Sales (One-to-Many):
    a. A discount offered by Target can be applied to multiple sales transactions.
    b. Each sale is associated with one discount, identified by the Discount_ID foreign key.
4. Store to Orders (One-to-Many):
    a. A Target store can receive multiple orders.
    b. Each order is associated with one Target store, identified by the Store_ID foreign key.
5. Product to Product_Inventory (One-to-Many):
    a. A product available at Target can be stored in multiple inventory items.
    b. Each inventory item is associated with one product, identified by the Product_ID foreign key.
6. Product to Product_Supplier (One-to-Many):
    a. A product available at Target can be supplied by multiple suppliers.
    b. Each supplier is associated with one product, identified by the Product_ID foreign key.
7. Store to Store_Product (One-to-Many):
    a. A Target store can have multiple products for sale.
    b. Each product is associated with one Target store, identified by the Store_ID foreign key.
8. Supplier to Product_Supplier (One-to-Many):
    a. A supplier providing products to Target can supply multiple products.
    b. Each product is associated with one supplier, identified by the Supplier_ID foreign key.
9. Customer to Product (Many-to-Many through Orders):

a. A customer of Target can purchase multiple products, and a product can be purchased by multiple Target customers through orders.

10. Product to Store (Many-to-Many through Store_Product):
    a. A product available at Target can be present in multiple Target stores, and a Target store can have multiple products for sale.
11. Product to Supplier (Many-to-Many through Product_Supplier):
    a. A product available at Target can be supplied by multiple suppliers, and a supplier can supply multiple products to Target.
12. Product to Inventory (Many-to-Many through Product_Inventory):
    a. A product available at Target can be associated with multiple inventory items, and an inventory item can be associated with multiple Target products.

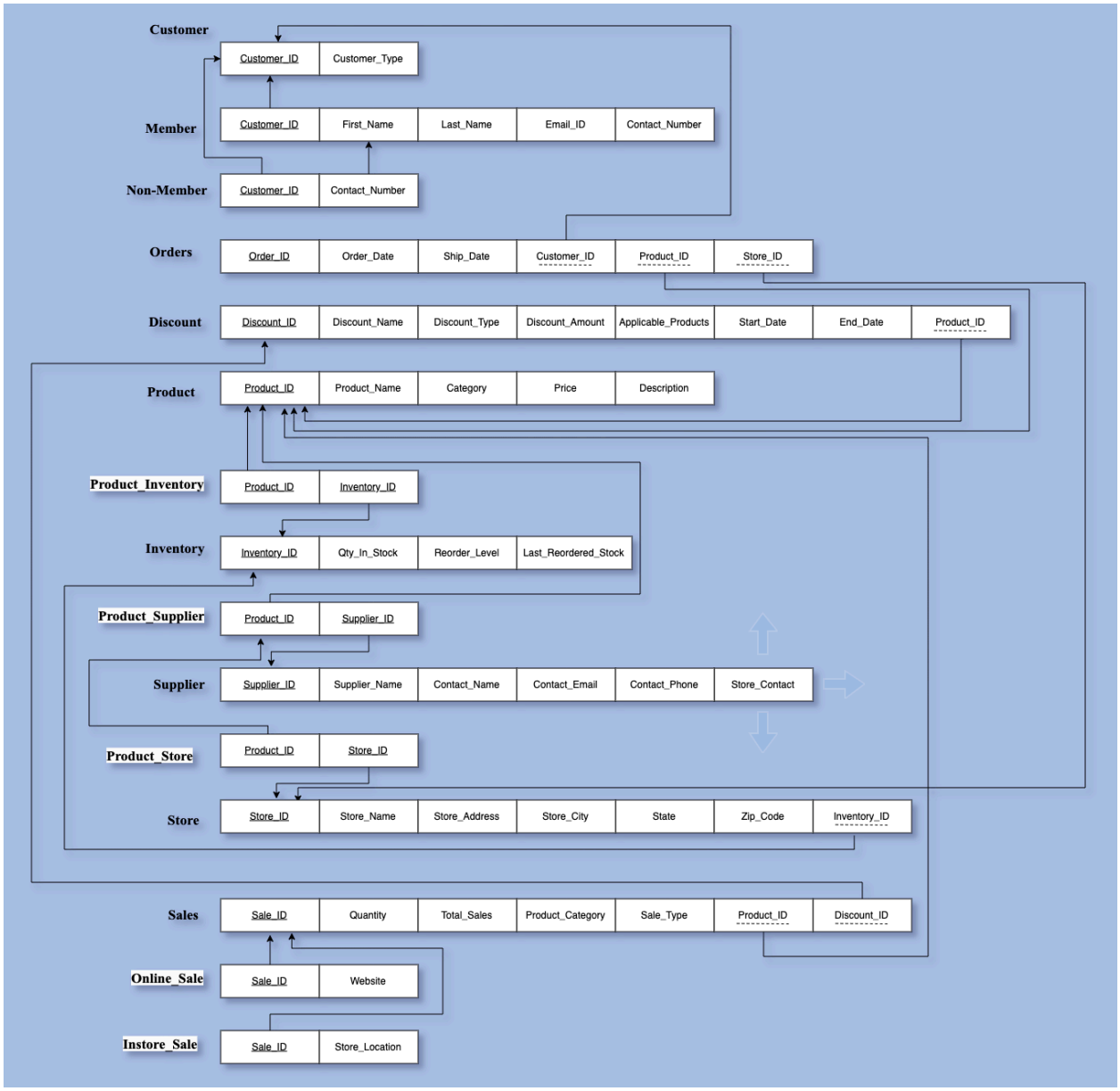
6. Enhanced Entity Relationship Diagram



7. Data Normalization

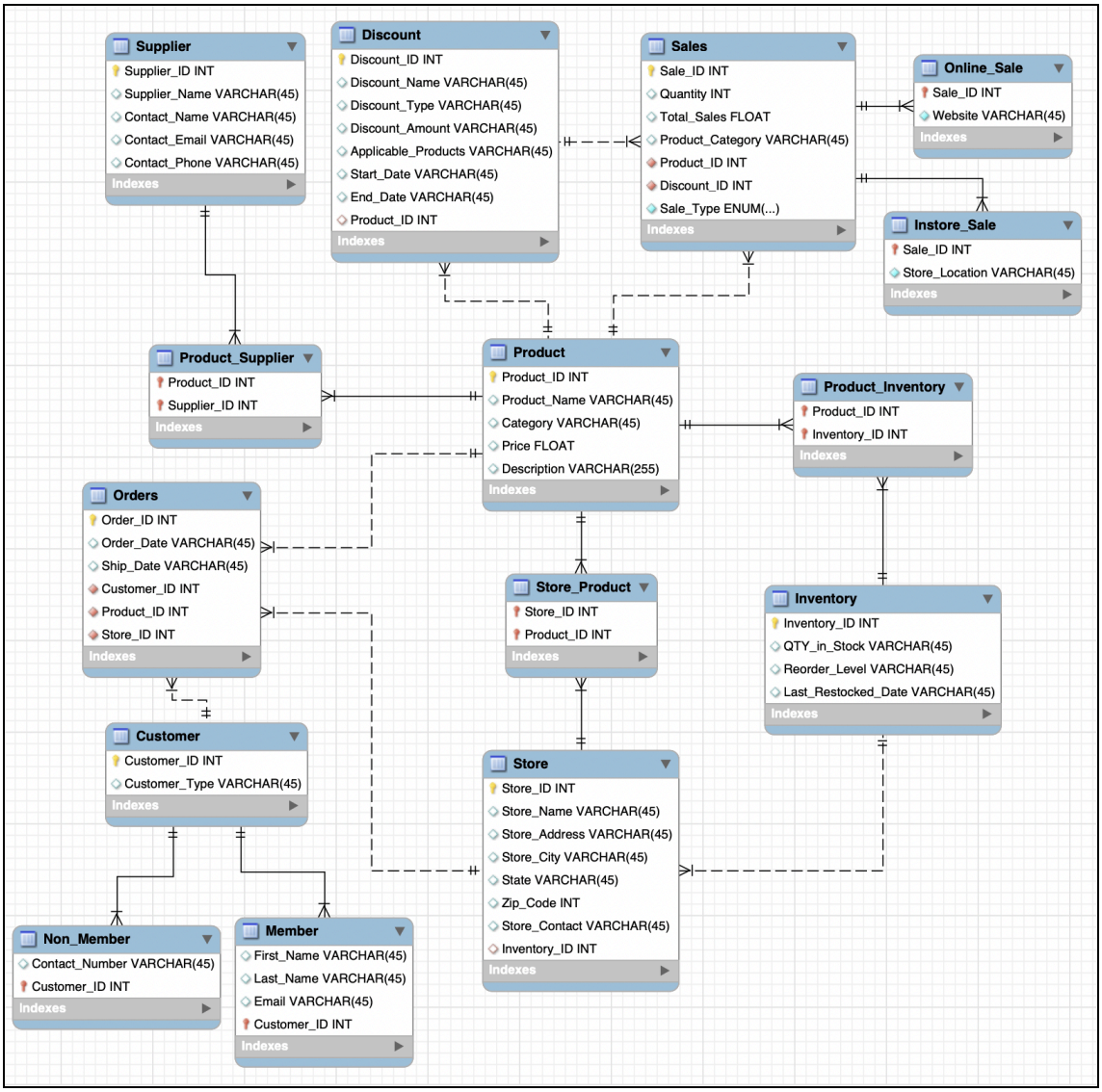
Data normalization aims to eliminate redundancy and dependency issues in database tables. This process involves breaking down complex tables into simpler, well-structured ones, adhering to specific normal forms.

The ultimate objective is to enhance data integrity, minimize anomalies, and promote efficient storage and retrieval.



- Customer Table:

    - Customer_ID (PK): Identifies each customer uniquely.
    - Customer_Type: Represents whether the customer is a Member or Non-Member.
    - No transitive dependencies exist in the Customer table.
- Discount Table:
    - Discount_ID (PK): Uniquely identifies each discount.
    - Discount_Name, Discount_Type, Discount_Amount, Applicable_Products, Start_Date, End_Date: Attributes directly depend on the Discount_ID, and there are no transitive dependencies.
- Sales Table:
    - Sale_ID (PK): Uniquely identifies each sale transaction.
    - Quantity, Total_Sales, Sale_Type, Product_Category: Directly depend on Sale_ID, and there are no transitive dependencies.
    - Product_ID (FK): References Product table, ensuring referential integrity.
- Orders Table:
    - Order_ID (PK): Uniquely identifies each order.
    - Order_Date, Ship_Date: Directly depend on Order_ID with no transitive dependencies.
    - Customer_ID (FK), Product_ID (FK), Store_ID (FK): References Customer, Product, and Store tables, respectively.
- Product Table:
    - Product_ID (PK): Uniquely identifies each product.
    - Product_Name, Category, Price, Description: Directly depend on Product_ID with no transitive dependencies.
- Inventory Table:
    - Inventory_ID (PK): Uniquely identifies each inventory item.
    - Quantity_In_Stock, Reorder_Level, Last_Restocked_Date: Directly depend on Inventory_ID with no transitive dependencies.
- Supplier Table:
    - Supplier_ID (PK): Uniquely identifies each supplier.
    - Supplier_Name, Contact_Name, Contact_Email, Contact_Phone: Directly depend on Supplier_ID with no transitive dependencies.
- Store Table:
    - Store_ID (PK): Uniquely identifies each store.
    - Store_Name, Address, City, State, Zip_Code, Store_Contact: Directly depend on Store_ID with no transitive dependencies.
    - Inventory_ID (FK): References Inventory table.
- Product_Supplier, Product_Inventory, Store_Product Tables:
    - These associative tables primarily consist of composite primary keys and foreign keys, ensuring many-to-many relationships between related entities without introducing transitive dependencies.



8. Table Creation

SQL Code

```
-- Schema TargetDB
-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `TargetDB` DEFAULT CHARACTER SET utf8 ;
USE `TargetDB` ;

-- -----------------------------------------------------
-- Table `TargetDB`.`Discount`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TargetDB`.`Discount` (
  `Discount_ID` INT NOT NULL,
  `Discount_Name` VARCHAR(45) NULL,
  `Discount_Type` VARCHAR(45) NULL,
  `Discount_Amount` VARCHAR(45) NULL,
```

```
  `Applicable_Products` VARCHAR(45) NULL,
  `Start_Date` VARCHAR(45) NULL,
  `End_Date` VARCHAR(45) NULL,
  `Product_ID` INT NULL,
  PRIMARY KEY (`Discount_ID`),
  INDEX `fk_Discount_Product_idx` (`Product_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Discount_Product`
    FOREIGN KEY (`Product_ID`)
    REFERENCES `TargetDB`.`Product` (`Product_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Product`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TargetDB`.`Product` (
  `Product_ID` INT NOT NULL,
  `Product_Name` VARCHAR(45) NULL,
  `Category` VARCHAR(45) NULL,
  `Price` FLOAT NULL,
  `Description` VARCHAR(255) NULL,
  PRIMARY KEY (`Product_ID`)
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TargetDB`.`Customer` (
  `Customer_ID` INT NOT NULL,
  `Customer_Type` VARCHAR(45) NULL,
  PRIMARY KEY (`Customer_ID`)
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Member`
-- -----------------------------------------------------
-- Corrected Member table creation
CREATE TABLE IF NOT EXISTS `TargetDB`.`Member` (
  `First_Name` VARCHAR(45) NULL,
  `Last_Name` VARCHAR(45) NULL,
  `Email` VARCHAR(45) NULL,  -- Corrected data type to VARCHAR(45)
  `Customer_ID` INT NOT NULL,
  PRIMARY KEY (`Customer_ID`),
  CONSTRAINT `fk_Member_Customer`
    FOREIGN KEY (`Customer_ID`)
    REFERENCES `TargetDB`.`Customer` (`Customer_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Non_Member`
-- -----------------------------------------------------
-- Corrected Non_Member table creation
CREATE TABLE IF NOT EXISTS `TargetDB`.`Non_Member` (
  `Contact_Number` VARCHAR(45) NULL,  -- Changed data type to VARCHAR(45)
  `Customer_ID` INT NOT NULL,
  PRIMARY KEY (`Customer_ID`),
  CONSTRAINT `fk_Non_Member_Customer`
    FOREIGN KEY (`Customer_ID`)
    REFERENCES `TargetDB`.`Customer` (`Customer_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Supplier`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TargetDB`.`Supplier` (
```

```
  `Supplier_ID` INT NOT NULL,
  `Supplier_Name` VARCHAR(45) NULL,
  `Contact_Name` VARCHAR(45) NULL,
  `Contact_Email` VARCHAR(45) NULL,
  `Contact_Phone` VARCHAR(45) NULL,
  PRIMARY KEY (`Supplier_ID`)
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Inventory`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TargetDB`.`Inventory` (
  `Inventory_ID` INT NOT NULL,
  `QTY_in_Stock` VARCHAR(45) NULL,
  `Reorder_Level` VARCHAR(45) NULL,
  `Last_Restocked_Date` VARCHAR(45) NULL,
  PRIMARY KEY (`Inventory_ID`)
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Store`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TargetDB`.`Store` (
  `Store_ID` INT NOT NULL,
  `Store_Name` VARCHAR(45) NULL,
  `Store_Address` VARCHAR(45) NULL,
  `Store_City` VARCHAR(45) NULL,
  `State` VARCHAR(45) NULL,
  `Zip_Code` INT NULL,
  `Store_Contact` VARCHAR(45) NULL,
  `Inventory_ID` INT NULL,
  PRIMARY KEY (`Store_ID`),
  INDEX `fk_Store_Inventory_idx` (`Inventory_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Store_Inventory`
    FOREIGN KEY (`Inventory_ID`)
    REFERENCES `TargetDB`.`Inventory` (`Inventory_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
) ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `TargetDB`.`Online_Sale`
-- -----------------------------------------------------
-- Create Online_Sale subtype
CREATE TABLE IF NOT EXISTS `TargetDB`.`Online_Sale` (
  `Sale_ID` INT NOT NULL,
  `Website` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Sale_ID`),
  CONSTRAINT `fk_Online_Sale_Sale`
    FOREIGN KEY (`Sale_ID`)
    REFERENCES `TargetDB`.`Sales` (`Sale_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Instore_Sale`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `TargetDB`.`Instore_Sale` (
  `Sale_ID` INT NOT NULL,
  `Store_Location` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Sale_ID`),
  CONSTRAINT `fk_Instore_Sale_Sale`
    FOREIGN KEY (`Sale_ID`)
    REFERENCES `TargetDB`.`Sales` (`Sale_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
) ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `TargetDB`.`Sales`
-- -----------------------------------------------------
-- Create Sales table with Sale_Type and subtypes
CREATE TABLE IF NOT EXISTS `TargetDB`.`Sales` (
  `Sale_ID` INT NOT NULL,
  `Quantity` INT NULL,
  `Total_Sales` FLOAT NULL,
  `Product_Category` VARCHAR(45) NULL,
  `Product_ID` INT NOT NULL,
  `Discount_ID` INT NOT NULL,
  `Sale_Type` ENUM('Online', 'Instore') NOT NULL,
  PRIMARY KEY (`Sale_ID`),
  INDEX `fk_Sales_Product_idx` (`Product_ID` ASC) VISIBLE,
  INDEX `fk_Sales_Discount_idx` (`Discount_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Sales_Product`
    FOREIGN KEY (`Product_ID`)
    REFERENCES `TargetDB`.`Product` (`Product_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sales_Discount`
    FOREIGN KEY (`Discount_ID`)
    REFERENCES `TargetDB`.`Discount` (`Discount_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TargetDB`.`Orders` (
  `Order_ID` INT NOT NULL,
  `Order_Date` VARCHAR(45) NULL,
  `Ship_Date` VARCHAR(45) NULL,
  `Customer_ID` INT NOT NULL,
  `Product_ID` INT NOT NULL,
  `Store_ID` INT NOT NULL,
  PRIMARY KEY (`Order_ID`),
  INDEX `fk_Orders_Customer_idx` (`Customer_ID` ASC) VISIBLE,
  INDEX `fk_Orders_Product_idx` (`Product_ID` ASC) VISIBLE,
  INDEX `fk_Orders_Store_idx` (`Store_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Orders_Customer`
    FOREIGN KEY (`Customer_ID`)
    REFERENCES `TargetDB`.`Customer` (`Customer_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Orders_Product`
    FOREIGN KEY (`Product_ID`)
    REFERENCES `TargetDB`.`Product` (`Product_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Orders_Store`
    FOREIGN KEY (`Store_ID`)
    REFERENCES `TargetDB`.`Store` (`Store_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
) ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `TargetDB`.`Product_Supplier`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TargetDB`.`Product_Supplier` (
  `Product_ID` INT NOT NULL,
  `Supplier_ID` INT NOT NULL,
  PRIMARY KEY (`Product_ID`, `Supplier_ID`),
  INDEX `fk_Product_Supplier_Product_idx` (`Product_ID` ASC) VISIBLE,
  INDEX `fk_Product_Supplier_Supplier_idx` (`Supplier_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Product_Supplier_Product`
```

```
  FOREIGN KEY (`Product_ID`)
  REFERENCES `TargetDB`.`Product` (`Product_ID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_Product_Supplier_Supplier`
  FOREIGN KEY (`Supplier_ID`)
  REFERENCES `TargetDB`.`Supplier` (`Supplier_ID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION
) ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `TargetDB`.`Product_Inventory`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TargetDB`.`Product_Inventory` (
 `Product_ID` INT NOT NULL,
 `Inventory_ID` INT NOT NULL,
 PRIMARY KEY (`Product_ID`, `Inventory_ID`),
 INDEX `fk_Product_Inventory_Product_idx` (`Product_ID` ASC) VISIBLE,
 INDEX `fk_Product_Inventory_Inventory_idx` (`Inventory_ID` ASC) VISIBLE,
 CONSTRAINT `fk_Product_Inventory_Product`
  FOREIGN KEY (`Product_ID`)
  REFERENCES `TargetDB`.`Product` (`Product_ID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
 CONSTRAINT `fk_Product_Inventory_Inventory`
  FOREIGN KEY (`Inventory_ID`)
  REFERENCES `TargetDB`.`Inventory` (`Inventory_ID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE
) ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `TargetDB`.`Store_Product`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `TargetDB`.`Store_Product` (
 `Store_ID` INT NOT NULL,
 `Product_ID` INT NOT NULL,
 PRIMARY KEY (`Store_ID`, `Product_ID`),
 INDEX `fk_Store_Product_Store_idx` (`Store_ID` ASC) VISIBLE,
 INDEX `fk_Store_Product_Product_idx` (`Product_ID` ASC) VISIBLE,
 CONSTRAINT `fk_Store_Product_Store`
  FOREIGN KEY (`Store_ID`)
  REFERENCES `TargetDB`.`Store` (`Store_ID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
 CONSTRAINT `fk_Store_Product_Product`
  FOREIGN KEY (`Product_ID`)
  REFERENCES `TargetDB`.`Product` (`Product_ID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE
) ENGINE = InnoDB;
```

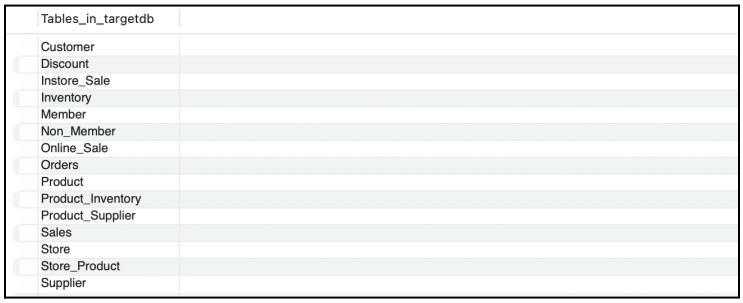
9. LOAD Statements

```
-- Insert data into Discount table
INSERT INTO TargetDB.Discount (Discount_ID, Discount_Name, Discount_Type, Discount_Amount, Applicable_Products, Start_Date,
End_Date, Product_ID)
VALUES (1, 'Discount 1', 'Percentage', '10%', 'All', '2023-01-01', '2023-01-31',1),
      (2, 'Discount 2', 'Percentage', '20%', 'All', '2023-02-01', '2023-02-28', 2),
      (3, 'Discount 3', 'Percentage', '15%', 'All', '2023-03-01', '2023-03-31',3),
      (4, 'Discount 4', 'Percentage', '25%', 'All', '2023-04-01', '2023-04-30',4),
      (5, 'Discount 5', 'Percentage', '30%', 'All', '2023-05-01', '2023-05-31',5),
      (6, 'Discount 6', 'Percentage', '12%', 'All', '2023-06-01', '2023-06-30',6),
```

```
    (7, 'Discount 7', 'Percentage', '18%', 'All', '2023-07-01', '2023-07-31',7),
    (8, 'Discount 8', 'Percentage', '22%', 'All', '2023-08-01', '2023-08-31',8),
    (9, 'Discount 9', 'Percentage', '28%', 'All', '2023-09-01', '2023-09-30',9),
    (10, 'Discount 10', 'Percentage', '15%', 'All', '2023-10-01', '2023-10-31',10);

-- Insert data into Store table
INSERT INTO TargetDB.Store (Store_ID, Store_Name, Store_Address, Store_City, State, Zip_Code, Store_Contact,Inventory_ID)
VALUES (1, 'Store1', '123 Main St', 'City1', 'State1', 12345, '987-654-3210',1),
    (2, 'Store2', '456 Oak St', 'City2', 'State2', 67890, '123-456-7890',2),
    (3, 'Store3', '789 Pine St', 'City3', 'State3', 10111, '345-678-9012',3),
    (4, 'Store4', '111 Elm St', 'City4', 'State4', 20222, '567-890-1234',4),
    (5, 'Store5', '222 Maple St', 'City5', 'State5', 30333, '789-012-3456',5),
    (6, 'Store6', '333 Birch St', 'City6', 'State6', 40444, '890-123-4567',6),
    (7, 'Store7', '444 Cedar St', 'City7', 'State7', 50555, '901-234-5678',7),
    (8, 'Store8', '555 Pine St', 'City8', 'State8', 60666, '234-567-8901',8),
    (9, 'Store9', '666 Oak St', 'City9', 'State9', 70777, '345-678-9012',9),
    (10, 'Store10', '777 Maple St', 'City10', 'State10', 80888, '456-789-0123',10);

-- Insert data into Supplier table
INSERT INTO TargetDB.Supplier (Supplier_ID, Supplier_Name, Contact_Name, Contact_Email, Contact_Phone)
VALUES (1, 'Supplier X', 'John Smith', 'john@example.com', '123-456-7890'),
    (2, 'Supplier Y', 'Jane Doe', 'jane@example.com', '987-654-3210'),
    (3, 'Supplier Z', 'Alice Johnson', 'alice@example.com', '234-567-8901'),
    (4, 'Supplier W', 'Bob Williams', 'bob@example.com', '345-678-9012'),
    (5, 'Supplier P', 'Eva Brown', 'eva@example.com', '456-789-0123'),
    (6, 'Supplier Q', 'Charlie Lee', 'charlie@example.com', '567-890-1234'),
    (7, 'Supplier R', 'Grace Martin', 'grace@example.com', '678-901-2345'),
    (8, 'Supplier S', 'David Jones', 'david@example.com', '789-012-3456'),
    (9, 'Supplier T', 'Sophia Miller', 'sophia@example.com', '890-123-4567'),
    (10, 'Supplier U', 'Samuel Taylor', 'samuel@example.com', '901-234-5678');

-- Insert data into Product table
INSERT INTO TargetDB.Product (Product_ID, Product_Name, Category, Price, Description)
VALUES (1, 'Laptop', 'Electronics', 1000.00, 'Large'),
    (2, 'T-Shirt', 'Clothing', 20.00, 'Medium' ),
    (3, 'Smartphone', 'Electronics', 800.00, 'Medium'),
    (4, 'Jeans', 'Clothing', 50.00, 'Large'),
    (5, 'Headphones', 'Electronics', 80.00, 'Small'),
    (6, 'Dress', 'Clothing', 40.00, 'Medium'),
    (7, 'Tablet', 'Electronics', 300.00, 'Medium'),
    (8, 'Sweater', 'Clothing', 35.00, 'Large'),
    (9, 'Camera', 'Electronics', 500.00, 'Small'),
    (10, 'Shoes', 'Clothing', 60.00, 'Medium');

-- Insert data into Inventory table
INSERT INTO TargetDB.Inventory (Inventory_ID, QTY_in_Stock, Reorder_Level, Last_Restocked_Date)
VALUES (1, 100, 'Low', '2022-12-01'),
    (2, 50, 'Medium', '2022-12-01'),
    (3, 80, 'High', '2022-12-01'),
    (4, 30, 'Low', '2022-12-01'),
    (5, 70, 'Medium', '2022-12-01'),
    (6, 40, 'High', '2022-12-01'),
    (7, 60, 'Medium', '2022-12-01'),
    (8, 25, 'Low', '2022-12-01'),
    (9, 45, 'Medium', '2022-12-01'),
    (10, 55, 'High', '2022-12-01');

-- Insert data into Customer table
INSERT INTO TargetDB.Customer (Customer_ID, Customer_Type)
VALUES (1, 'Member'),
    (2, 'Non-Member'),
    (3, 'Member'),
    (4, 'Non-Member'),
    (5, 'Member'),
    (6, 'Non-Member'),
    (7, 'Member'),
    (8, 'Non-Member'),
    (9, 'Member'),
    (10, 'Non-Member'),
    (11, 'Non-Member'),  -- Added missing entries for Non-Members
```

```
    (12, 'Non-Member'),
    (13, 'Non-Member'),
    (14, 'Non-Member'),
    (15, 'Non-Member'),
    (16, 'Non-Member'),
    (17, 'Non-Member'),
    (18, 'Non-Member'),
    (19, 'Non-Member'),
    (20, 'Non-Member');

-- Insert data into Member table
INSERT INTO TargetDB.Member (First_Name, Last_Name, Email, Customer_ID)
VALUES ('John', 'Doe', 'john@example.com', 1),
    ('Jane', 'Smith', 'jane@example.com', 3),
    ('Alice', 'Johnson', 'alice@example.com', 5),
    ('Bob', 'Williams', 'bob@example.com', 7),
    ('Eva', 'Brown', 'eva@example.com', 9),
    ('Charlie', 'Lee', 'charlie@example.com', 11),
    ('Grace', 'Martin', 'grace@example.com', 13),
    ('David', 'Jones', 'david@example.com', 15),
    ('Sophia', 'Miller', 'sophia@example.com', 17),
    ('Samuel', 'Taylor', 'samuel@example.com', 19);

-- Insert data into Non_Member table
INSERT INTO TargetDB.Non_Member (Contact_Number, Customer_ID)
VALUES (1234567890, 2),
    (9876543210, 4),
    (3456789012, 6),
    (5678901234, 8),
    (7890123456, 10),
    (2345678901, 12),
    (4567890123, 14),
    (6789012345, 16),
    (8901234567, 18),
    (9012345678, 20);

-- Insert data into Orders table
INSERT INTO TargetDB.Orders (Order_ID, Order_Date, Ship_Date, Customer_ID, Product_ID, Store_ID)
VALUES (1, '2023-01-15', '2023-01-20', 1, 1, 1),
    (2, '2023-02-01', '2023-02-05', 2, 2, 2),
    (3, '2023-03-12', '2023-03-18', 3, 3, 3),
    (4, '2023-04-05', '2023-04-10', 4, 4, 4),
    (5, '2023-05-20', '2023-05-25', 5, 5, 5),
    (6, '2023-06-08', '2023-06-13', 6, 6, 6),
    (7, '2023-07-19', '2023-07-24', 7, 7, 7),
    (8, '2023-08-03', '2023-08-08', 8, 8, 8),
    (9, '2023-09-22', '2023-09-27', 9, 9, 9),
    (10, '2023-10-10', '2023-10-15', 10, 10, 10);

-- Insert data into Sales table
INSERT INTO TargetDB.Sales (Sale_ID, Quantity, Total_Sales, Sale_Type, Product_Category, Product_ID, Discount_ID)
VALUES (1, 5, 5000.00, 'Online', 'Electronics', 1, 1),
    (2, 10, 300.00, 'Online', 'Clothing', 2, 2),
    (3, 8, 2400.00, 'Online', 'Electronics', 3, 3),
    (4, 15, 750.00, 'Online', 'Clothing', 4, 4),
    (5, 3, 240.00, 'Online', 'Electronics', 5, 5),
    (6, 12, 480.00, 'Instore', 'Clothing', 6, 6),
    (7, 7, 2100.00, 'Instore', 'Electronics', 7, 7),
    (8, 20, 700.00, 'Instore', 'Clothing', 8, 8),
    (9, 6, 3000.00, 'Instore', 'Electronics', 9, 9),
    (10, 18, 1080.00, 'Instore', 'Clothing', 10, 10);

-- Insert data into Instore_Sale table
INSERT INTO TargetDB.Instore_Sale (Sale_ID, Store_Location)
VALUES (6, 'New York, NY'),
    (7, 'Los Angeles, CA'),
    (8, 'Chicago, IL'),
    (9, 'Houston, TX'),
    (10, 'Phoenix, AZ');
```

```
-- Insert data into Online_Sale table
INSERT INTO TargetDB.Online_Sale (Sale_ID, Website)
VALUES (1, 'https://www.targettechmart.com'),
       (2, 'https://www.targetfashionhub.com'),
       (3, 'https://www.targetelectronicsdepot.com'),
       (4, 'https://www.targetstyleemporium.com'),
       (5, 'https://www.targetgadgetgalaxy.net');

-- Insert data into Product_Supplier table
INSERT INTO TargetDB.Product_Supplier (Product_ID, Supplier_ID)
VALUES
    (1, 1),
    (1, 2),
    (2, 1),
    (2, 2),
    (3, 1),
    (3, 2),
    (4, 1),
    (4, 2),
    (5, 1),
    (5, 2);

-- Insert into `Store_Product`
INSERT INTO `TargetDB`.`Store_Product` (`Store_ID`, `Product_ID`)
VALUES
  (1, 1),
  (2, 2),
  (3, 3),
  (4, 4),
  (5, 5),
  (6, 6),
  (7, 7),
  (8, 8),
  (9, 9),
  (10, 10);

-- Insert into `Product_Inventory`
INSERT INTO `TargetDB`.`Product_Inventory` (`Product_ID`, `Inventory_ID`)
VALUES
  (1, 1),
  (2, 2),
  (3, 3),
  (4, 4),
  (5, 5),
  (6, 6),
  (7, 7),
  (8, 8),
  (9, 9),
  (10, 10)
SHOW TABLES;
```



Tables_in_targetdb
Customer
Discount
Instore_Sale
Inventory
Member
Non_Member
Online_Sale
Orders
Product
Product_Inventory
Product_Supplier
Sales
Store
Store_Product
Supplier

Research Questions

-- Q1: Which customers have received discounts on products they purchased, and what is the total discount percent for each customer?

```
SELECT
   C.Customer_ID,
   CASE
      WHEN C.Customer_Type = 'Member' THEN M.First_Name
      WHEN C.Customer_Type = 'Non-Member' THEN N.Contact_Number
      ELSE 'Unknown'
   END AS Customer_Name,
   SUM(CASE
         WHEN D.Discount_Type = 'Percentage' THEN CAST(REPLACE(D.Discount_Amount, '%', '') AS
DECIMAL(5,2))
         ELSE 0
      END) AS Total_Discount_Percent
FROM
   TargetDB.Customer AS C
JOIN
   TargetDB.Orders AS O ON C.Customer_ID = O.Customer_ID
JOIN
   TargetDB.Discount AS D ON O.Product_ID = D.Product_ID
LEFT JOIN
   TargetDB.Member AS M ON C.Customer_ID = M.Customer_ID
LEFT JOIN
   TargetDB.Non_Member AS N ON C.Customer_ID = N.Customer_ID
GROUP BY
   C.Customer_ID, C.Customer_Type, M.First_Name, N.Contact_Number
ORDER BY
   C.Customer_ID;
```

Output:

Customer_ID	Customer_Name	Total_Discount_Percent
1	John	10.00
2	1234567890	20.00
3	Jane	15.00
4	9876543210	25.00
5	Alice	30.00
6	3456789012	12.00
7	Bob	18.00
8	5678901234	22.00
9	Eva	28.00
10	7890123456	15.00

Purpose of the question: The query looks for consumers in the Target database who have used product discounts and calculates the overall discount % for each client. This data provides useful insights into client purchase habits and can be used to develop customized marketing and customer relationship management strategies.

Q2: What are the contact details of suppliers who have products in low inventory (below the reorder level), and what products do they supply?

```
SELECT
   S.Supplier_ID,
   S.Supplier_Name,
   S.Contact_Name,
   S.Contact_Email,
   S.Contact_Phone,
   P.Product_ID,
   P.Product_Name,
   I.QTY_in_Stock,
   I.Reorder_Level
FROM
   TargetDB.Product_Supplier PS
```

```
JOIN
   TargetDB.Supplier S ON PS.Supplier_ID = S.Supplier_ID
JOIN
   TargetDB.Product P ON PS.Product_ID = P.Product_ID
JOIN
   TargetDB.Product_Inventory PI ON P.Product_ID = PI.Product_ID
JOIN
   TargetDB.Inventory I ON PI.Inventory_ID = I.Inventory_ID
WHERE
   I.Reorder_Level='Low';
```

Output:

Supplier_ID	Supplier_Name	Contact_Name	Contact_Email	Contact_Pho...	Product_ID	Product_Name	QTY_in_Stock	Reorder_Level
1	Supplier X	John Smith	john@example.com	123-456-7890	1	Laptop	100	Low
2	Supplier Y	Jane Doe	jane@example.com	987-654-3210	1	Laptop	100	Low
1	Supplier X	John Smith	john@example.com	123-456-7890	4	Jeans	30	Low
2	Supplier Y	Jane Doe	jane@example.com	987-654-3210	4	Jeans	30	Low

Purpose of the question: This Target database query seeks the contact information of suppliers whose products are now below the reorder level. It also inquires about the specific products supplied by these providers with limited inventory. This information is useful for inventory management and enables timely engagement with suppliers to alleviate stock shortages and optimize supply chain operations.

Q3: What is the total value of discounts applied to products in the “Electronics” category, and which customers have benefited the most from these discounts?

```
SELECT
   C.Customer_ID,
   CASE
      WHEN C.Customer_Type = 'Member' THEN M.First_Name
      WHEN C.Customer_Type = 'Non-Member' THEN N.Contact_Number
      ELSE 'Unknown'
   END AS Customer_Name,
   SUM(CASE
         WHEN D.Discount_Type = 'Percentage' THEN
            (P.Price * S.Quantity * CAST(REPLACE(D.Discount_Amount, '%', '') AS DECIMAL(5,2))) / 100
         ELSE 0
      END) AS Total_Discount_Value
FROM
   TargetDB.Customer AS C
JOIN
   TargetDB.Orders AS O ON C.Customer_ID = O.Customer_ID
JOIN
   TargetDB.Product AS P ON O.Product_ID = P.Product_ID
JOIN
   TargetDB.Sales AS S ON P.Product_ID = S.Product_ID
JOIN
   TargetDB.Discount AS D ON S.Discount_ID = D.Discount_ID
LEFT JOIN
   TargetDB.Member AS M ON C.Customer_ID = M.Customer_ID
LEFT JOIN
   TargetDB.Non_Member AS N ON C.Customer_ID = N.Customer_ID
WHERE
   P.Category = 'Electronics'
```

```
GROUP BY
   C.Customer_ID, C.Customer_Type, M.First_Name, N.Contact_Number
ORDER BY
   Total_Discount_Value DESC;
```

Output:

Customer_ID	Customer_Name	Total_Discount_Value
3	Jane	960
9	Eva	840
1	John	500
7	Bob	378
5	Alice	72

Purpose of the question: It determines the effectiveness of the discount campaigns in the "Electronics" category and adjust the future promotions based on the insights gained from customer behavior in response to the discounts.

Q4: How many online and in-store sales have been made for each product category?

```
SELECT
   P.Category,
   COUNT(CASE WHEN S.Sale_Type = 'Online' THEN 1 END) AS Online_Sales,
   COUNT(CASE WHEN S.Sale_Type = 'Instore' THEN 1 END) AS Instore_Sales
FROM
   TargetDB.Product AS P
LEFT JOIN
   TargetDB.Sales AS S ON P.Product_ID = S.Product_ID
GROUP BY
   P.Category;
```

Output:

Category	Online_Sales	Instore_Sales
Electronics	3	2
Clothing	2	3

Purpose of the question: It determines the distribution of sales for each product category, distinguishing between online and in-store sales. The query provides a count of online and in-store sales for each product category, offering insights into the preferred sales channel for different types of products.

Conclusion:

To sum up, we as a group gained experience in designing a relational database schema, establishing relationships between tables, and defining foreign keys to maintain data integrity. We learnt that querying an already existing database was an easier challenge to tackle then that of building our own database.We leveraged customer-specific discount insights to create personalized marketing strategies along with trying to ensure that data inserted into different tables is consistent, and foreign key relationships are maintained to avoid referential integrity issues. In the end, through this project, we gained valuable insights into effectively presenting data in real-world scenarios, highlighting the project's practical relevance and its significant impact on our learning experience.