

# **Database Systems**

## **LAB REPORT**

**Abdul Haseeb Ul Hasan**

**CIS 22-26**

**BS-22-IB-101577**

# Food Delivery System

The **Packaged Food Delivery Database System** or simply **Food Delivery System** is designed to manage the operations of a business delivering packaged food products, encompassing customers, orders, products, inventory, suppliers, warehouses, and deliveries. The system ensures efficient storage, tracking, and fulfillment of orders while maintaining data integrity and adhering to normalization principles (1NF, 2NF, and 3NF).

## Entities And Attributes

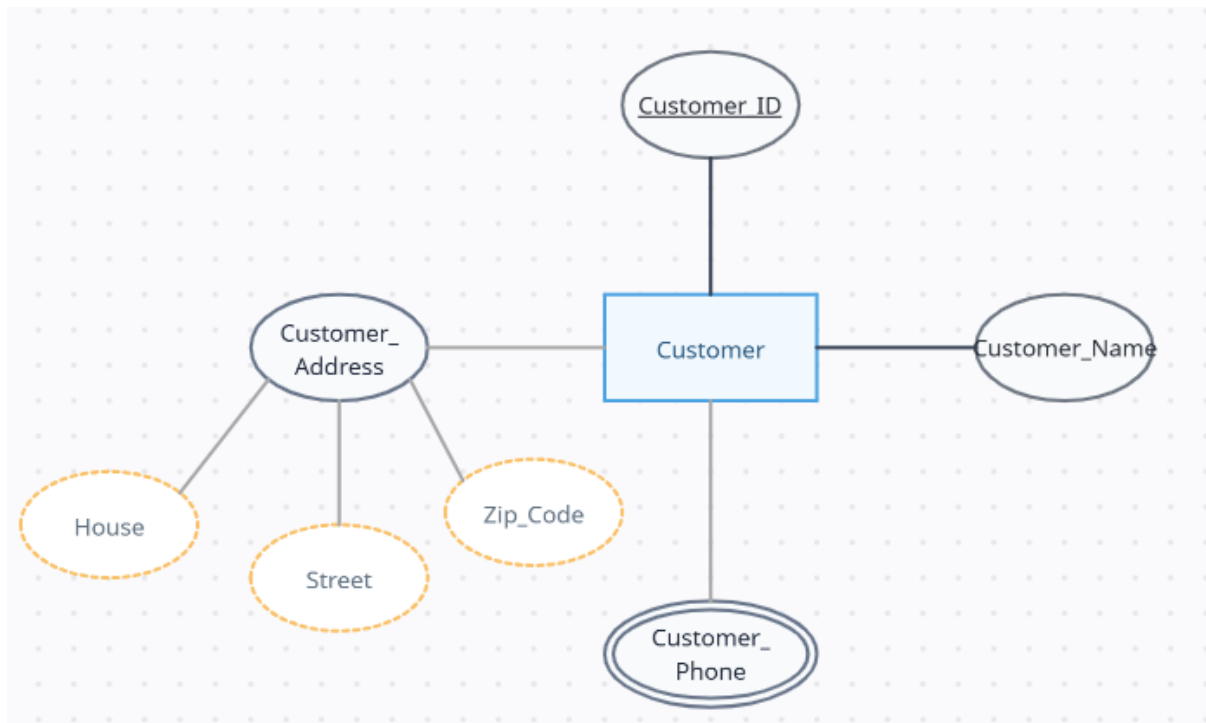
### 1. Customer:

**Description:** This entity helps identify and manage customer details, enabling personalized service and order tracking.

**Purpose:** Stores information about the customers who place orders.

**Attributes:**

- **Customer\_ID:** A unique identifier for each customer (Primary Key).
- **Customer\_Name:** The customer's name.
- **Customer\_Address:** The customer's address. Composed of House, Street and Zip\_Code
- **Customer\_Phone:** Customer's Phone No.



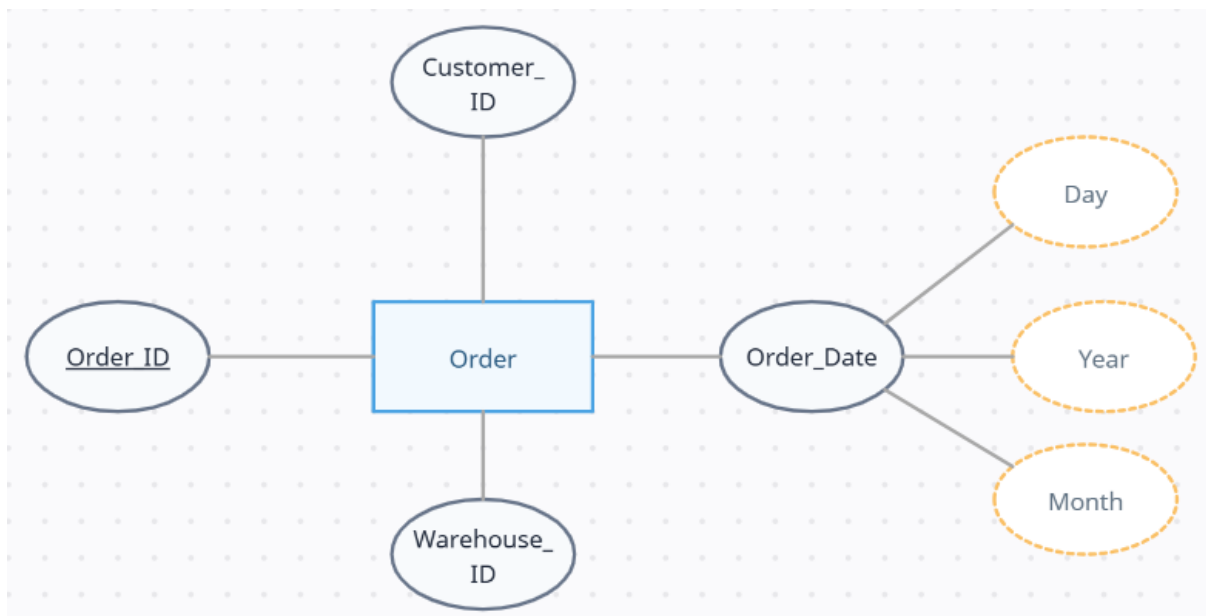
## 2. Order:

**Description:** Links customers to their orders and provides details like order value and the date of placement.

**Purpose:** Represents an order placed by a customer.

**Attributes:**

- **Order\_ID:** A unique identifier for each order (Primary Key).
- **Customer\_ID:** References the Customer table (Foreign Key).
- **Warehouse\_ID:** References the Warehouse table (Foreign Key).
- **Total\_Price:** The total price of all items in the order.
- **Order\_Date:** The date the order was placed. Composed of day, month and year.



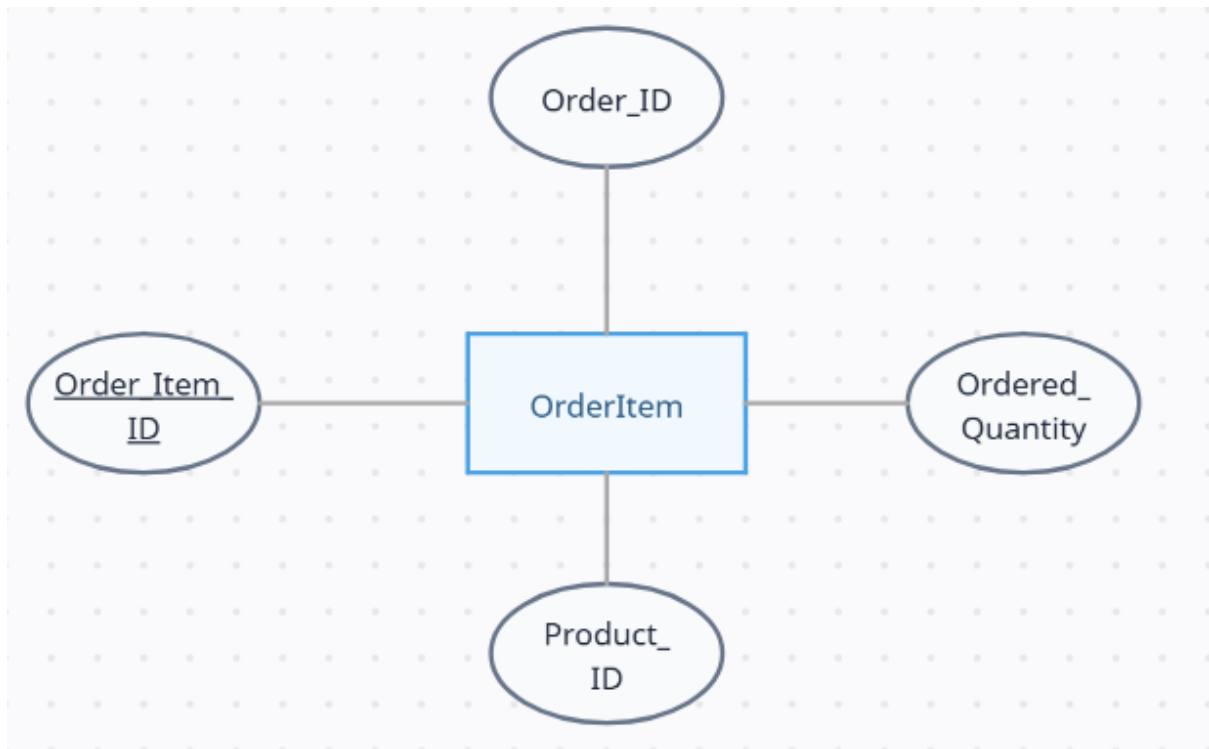
## 3. OrderItem:

**Description:** Facilitates the breakdown of an order into its component items, enabling detailed order management.

**Purpose:** Tracks individual items within an order.

**Attributes:**

- **Order\_Item\_ID:** A unique identifier for each order item (Primary Key).
- **Order\_ID:** References the Order table (Foreign Key).
- **Product\_ID:** References the Product table (Foreign Key).
- **Ordered\_Quantity:** The quantity of the product in the order.



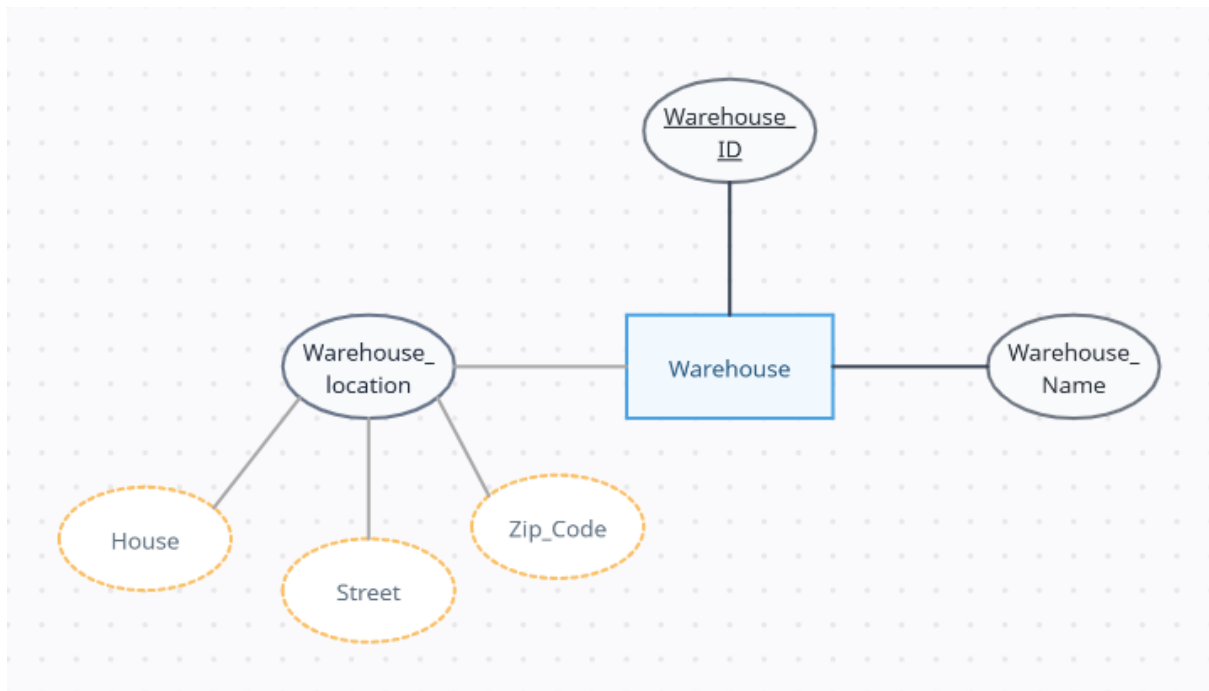
#### 4. Warehouse:

**Description:** Ensures products are available for order fulfillment by storing them efficiently.

**Purpose:** Represents storage facilities where products are housed.

**Attributes:**

- **Warehouse\_ID:** A unique identifier for each warehouse (Primary Key).
- **Warehouse\_Name:** The name of the warehouse.
- **Warehouse\_Location:** The physical address or location of the warehouse. Composed of House, Street and Zip\_Code.



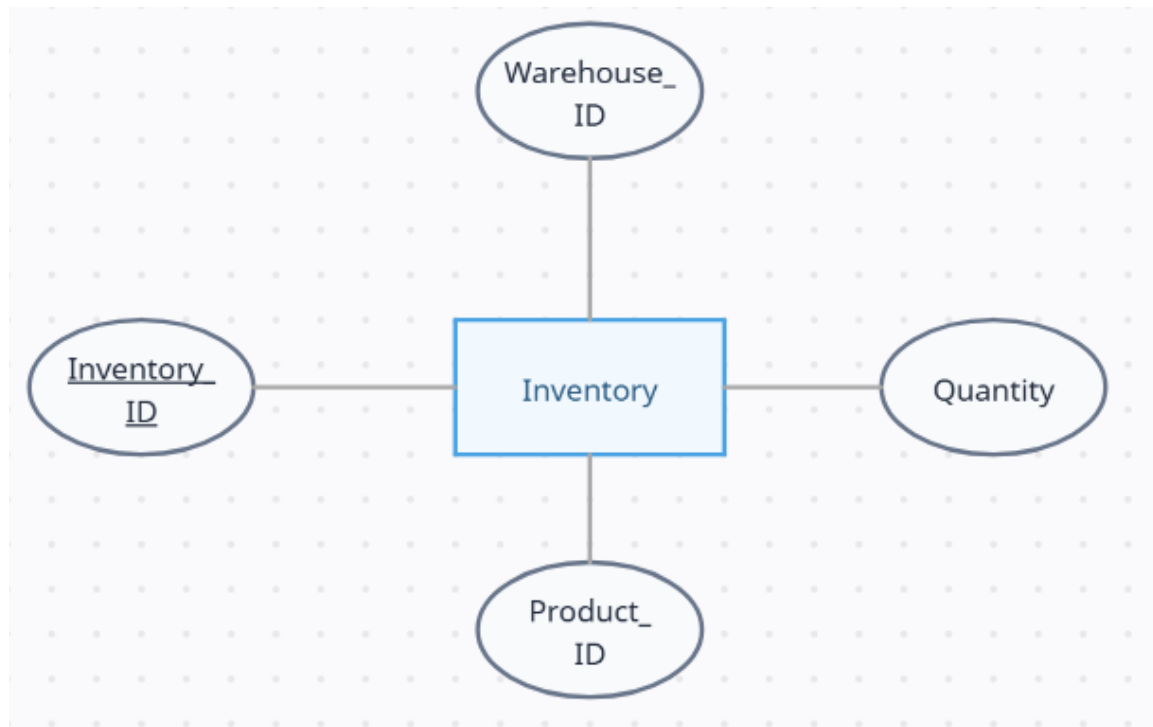
## 5. Inventory:

**Description:** Maintains up-to-date information about stock levels, ensuring the smooth operation of the supply chain.

**Purpose:** Tracks stock levels of products in warehouses.

**Attributes:**

- **Inventory\_ID:** A unique identifier for each inventory record (Primary Key).
- **Warehouse\_ID:** References the Warehouse table (Foreign Key).
- **Product\_ID:** References the Product table (Foreign Key).
- **Quantity:** The quantity of items in stock.



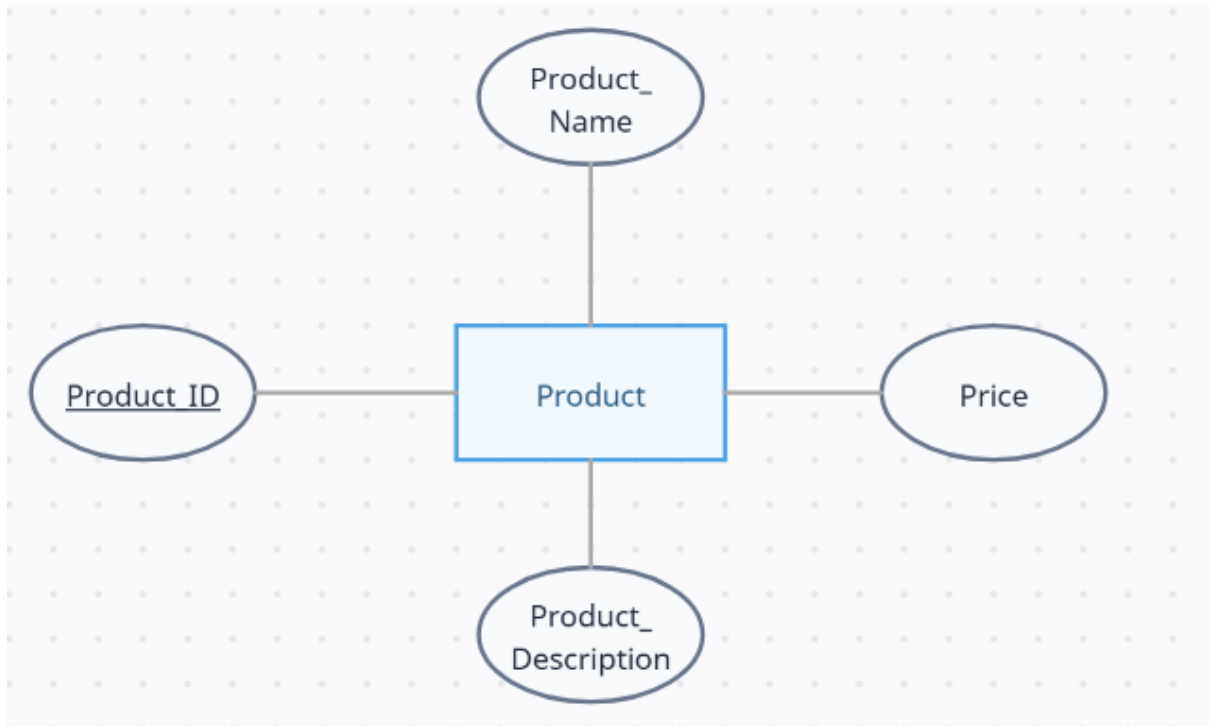
## 6. Product:

**Description:** Central to the system, this entity helps in product management, pricing, and categorization.

**Purpose:** Contains details of packaged food products.

**Attributes:**

- **Product\_ID:** A unique identifier for each product (Primary Key).
- **Product\_Name:** The name of the product.
- **Product\_Description:** The description of the product.
- **Price:** The price per unit of the product.



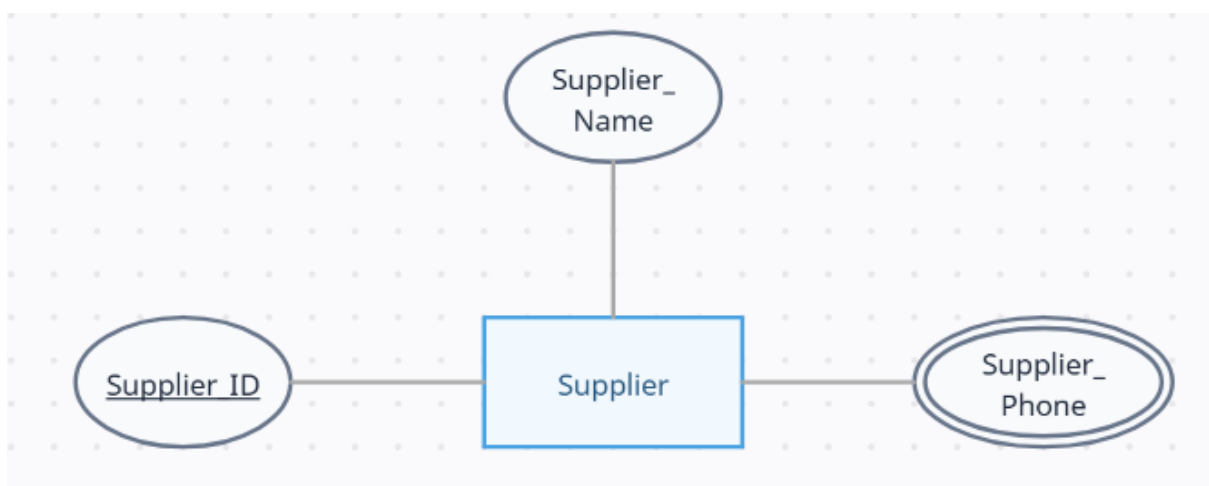
## 7. Supplier:

**Description:** Tracks suppliers to ensure timely restocking of inventory.

**Purpose:** Stores details about suppliers who provide products to warehouses.

**Attributes:**

- **Supplier\_ID:** A unique identifier for each supplier (Primary Key).
- **Supplier\_Name:** The name of the supplier.
- **Supplier\_Phone:** Contact details of the supplier.



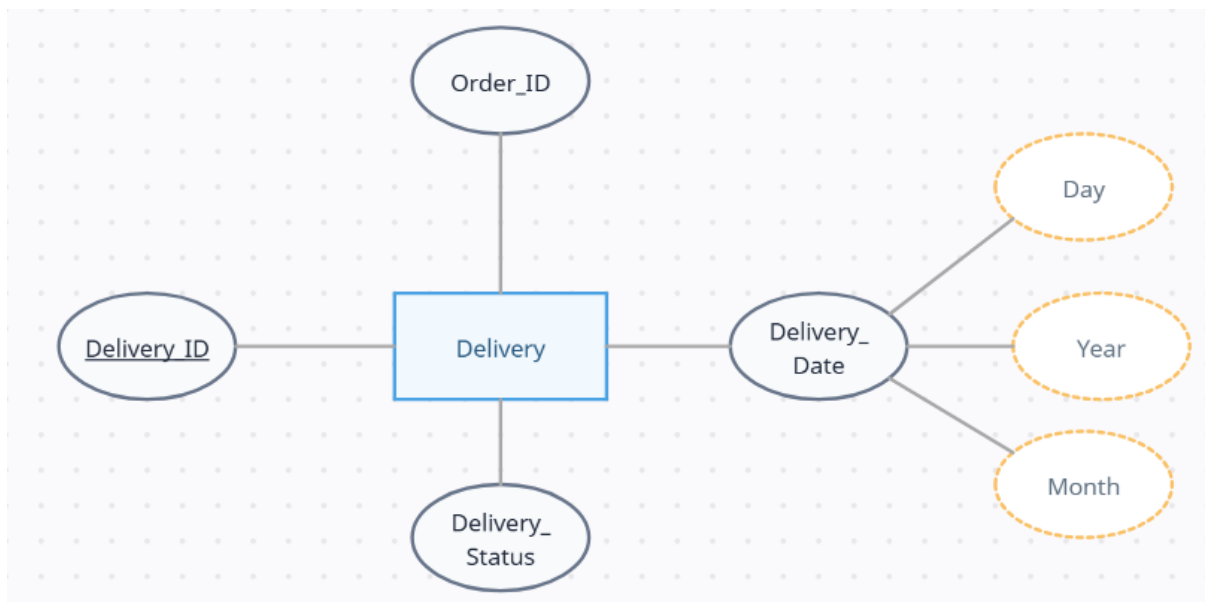
## 8. Delivery:

**Description:** Tracks the fulfillment process, ensuring timely and accurate deliveries to customers.

**Purpose:** Manages the process of delivering orders to customers.

**Attributes:**

- **Delivery\_ID:** A unique identifier for each delivery (Primary Key).
- **Order\_ID:** References the **Order** table (Foreign Key).
- **Delivery\_Date:** The date when the order is delivered. Composed of Day, Month and Year
- **Delivery\_Status:** The current status of the delivery (e.g., Pending, In Transit, Delivered).



### Entity Descriptions in the System

1. **Customer** connects the system to the individuals making purchases.
2. **Order** and **OrderItem** ensure comprehensive tracking of customer purchases and their details.
3. **Warehouse** and **Inventory** handle the backend logistics of stock management.
4. **Product** is the core item being delivered and sold.
5. **Supplier** manages the replenishment of stock in warehouses.
6. **Delivery** ensures orders are tracked through to their final fulfillment.

This setup enables a seamless workflow, from order placement and product availability to delivery.

# Business Rules

## 1. Customer Rules

1. Each **customer** must have a unique identifier (**Customer\_ID**).
2. A customer can place multiple **Orders**, but each order must belong to one and only one customer.
  - **Relationship: 1:M** between **Customer** and **Order**.

## 2. Order Rules

1. Each **order** must have a unique identifier (**Order\_ID**).
2. Each order is fulfilled by a single **warehouse**.
  - **Relationship: 1:M** between **Warehouse** and **Order**.
3. Each order must consist of at least one **order item**.
  - **Relationship: 1:M** between **Order** and **Order\_Item**.
4. The Total\_Price for an order is derived from the SUM of the prices and quantities of the associated order items.
5. Each order must have an associated delivery.

## 3. Order Item Rules

1. Each **order item** must reference an **order** and a **product**.
  - **Relationships:**
    - **1:M** between **Order** and **Order\_Item**.
    - **M:1** between **Order\_Item** and **Product**.
2. An order item must specify the Quantity of the product being ordered.

## 4. Product Rules

1. Each **product** must have a unique identifier (Product\_ID) and belong to a defined category (e.g., snacks, beverages).

2. A product can be provided by multiple **Suppliers**.
  - **Relationship: 1:M** between **Supplier** and **Product**.
3. A product can be stored in multiple **Warehouses**, and each warehouse can store multiple products.
  - **Relationship: M:M** between **Product** and **Warehouse** (resolved through the **Inventory** table).

## 5. Warehouse Rules

1. Each **warehouse** must have a unique identifier (Warehouse\_ID) and a physical location.
2. A warehouse can fulfill multiple **orders**.
  - **Relationship: 1:M** between **Warehouse** and **Order**.
3. A warehouse must track its stock through the **inventory**.

## 6. Inventory Rules

1. Each **inventory** record must link a specific **warehouse** and a specific **product**.
  - **Relationships:**
    - **1:M** between **Warehouse** and **Inventory**.
    - **1:M** between **Product** and **Inventory**.
2. The Quantity field must be updated whenever products are added or removed from stock.

## 7. Supplier Rules

1. Each **supplier** must have a unique identifier (Supplier\_ID) and contact information.
2. A supplier can provide multiple **products** but must only supply products they are listed for.

## 8. Delivery Rules

1. Each **delivery** must have a unique identifier (Delivery\_ID) and be associated with exactly one **order**.
  - **Relationship: 1:1** between **Order** and **Delivery**.

2. The Status field must indicate the progress of the delivery (e.g., Pending, In Transit, Delivered).
3. Deliveries must have a recorded Delivery\_Date upon completion.

## Relationships

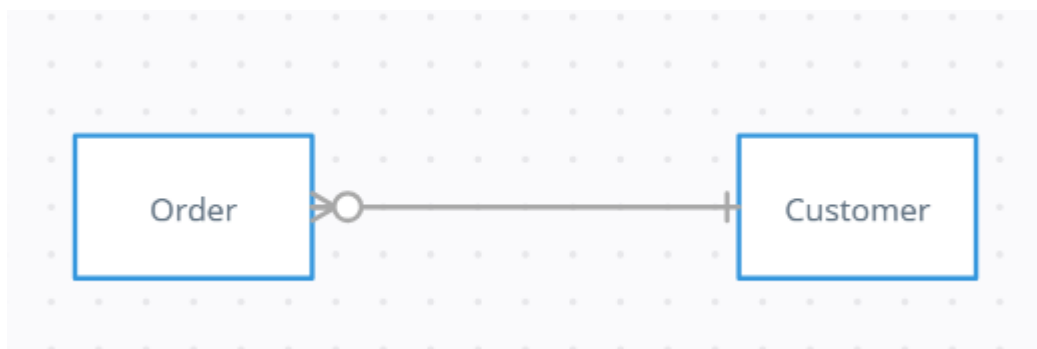
### 1. Unary Relationships:

There are no unary relationships in the system.

### 2. Binary Relationships:

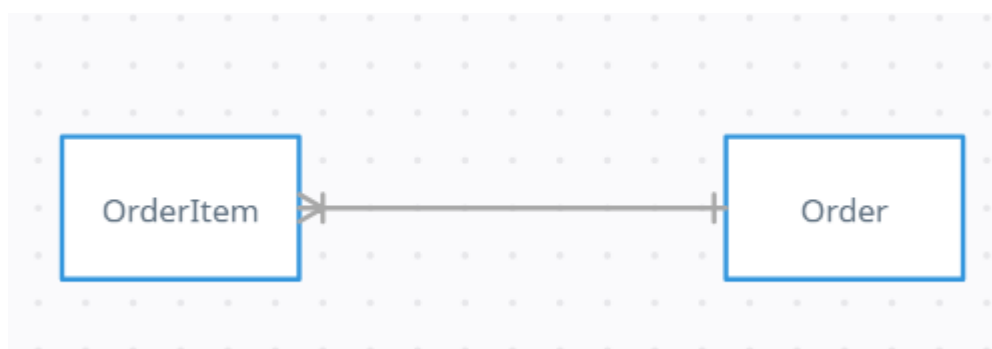
#### 1. Customer - Order

- **Relationship:** A customer places orders. **1:M**



#### 2. Order - OrderItem

- **Relationship:** An order contains multiple order items. **1:M**



#### 3. OrderItem - Product

- **Relationship:** An order item refers to a specific product. **1:M**



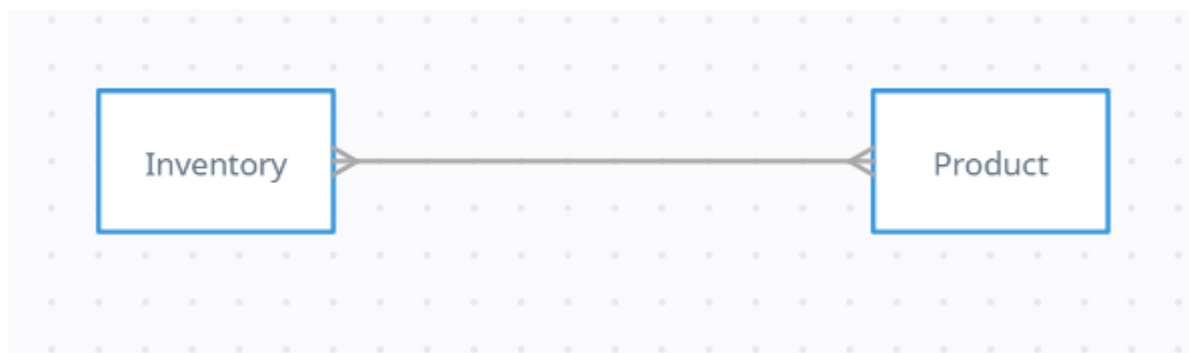
#### 4. Warehouse - Inventory

- Relationship: A warehouse has an inventory of products. **1:M**



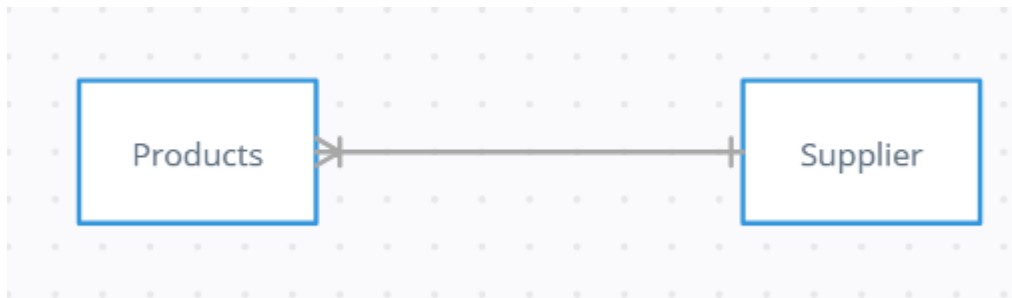
#### 5. Product - Inventory

- Relationship: Products are stored in inventories at warehouses. **M:M**



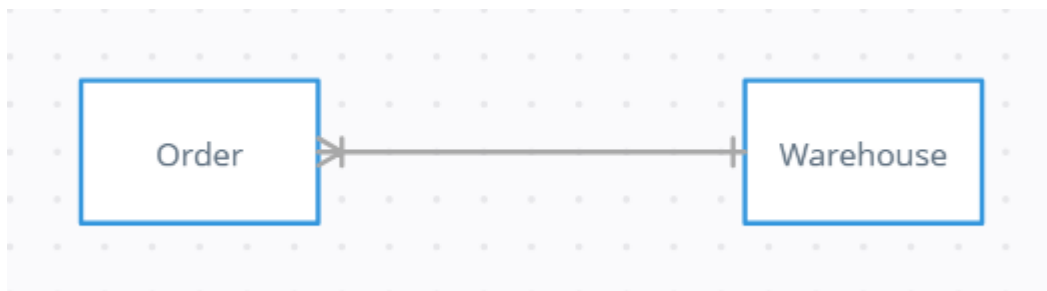
#### 6. Supplier - Product

- Relationship: A supplier provides products. **1:M**



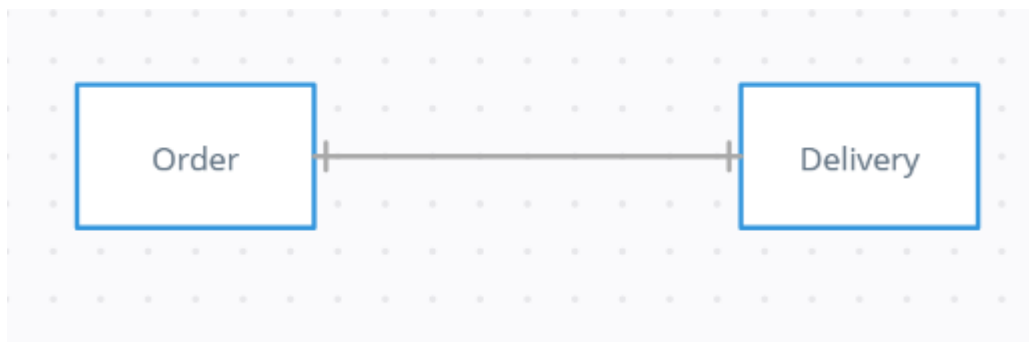
### 7. Warehouse - Order

- **Relationship:** A warehouse fulfills orders. **1:M**



### 8. Order - Delivery

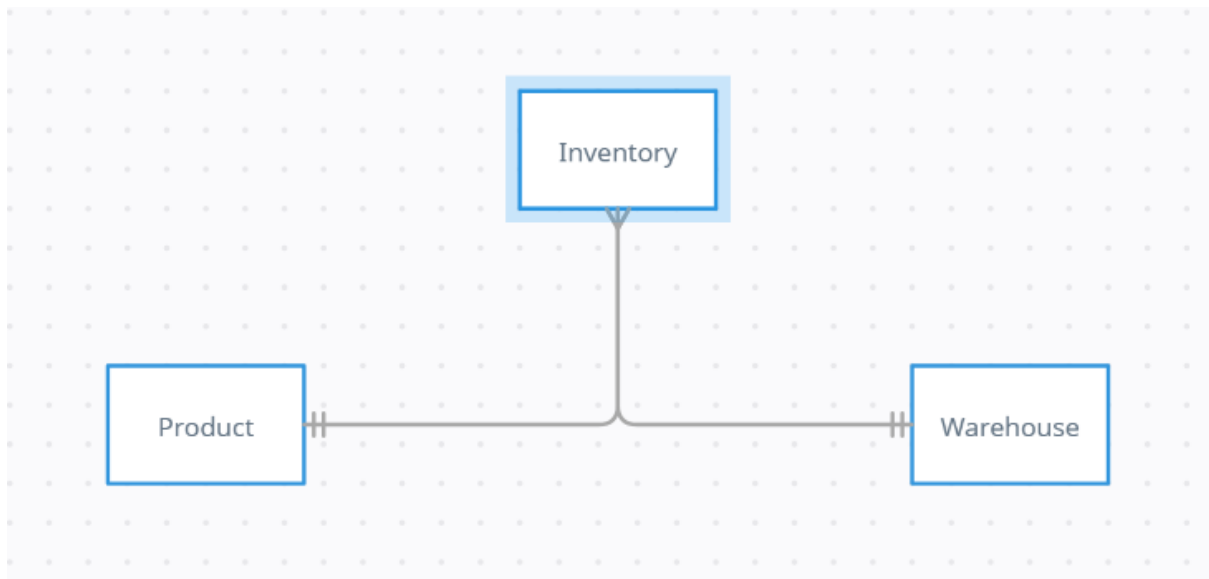
- **Relationship:** An order is associated with a delivery. **1:1**



## 3. Ternary Relationships:

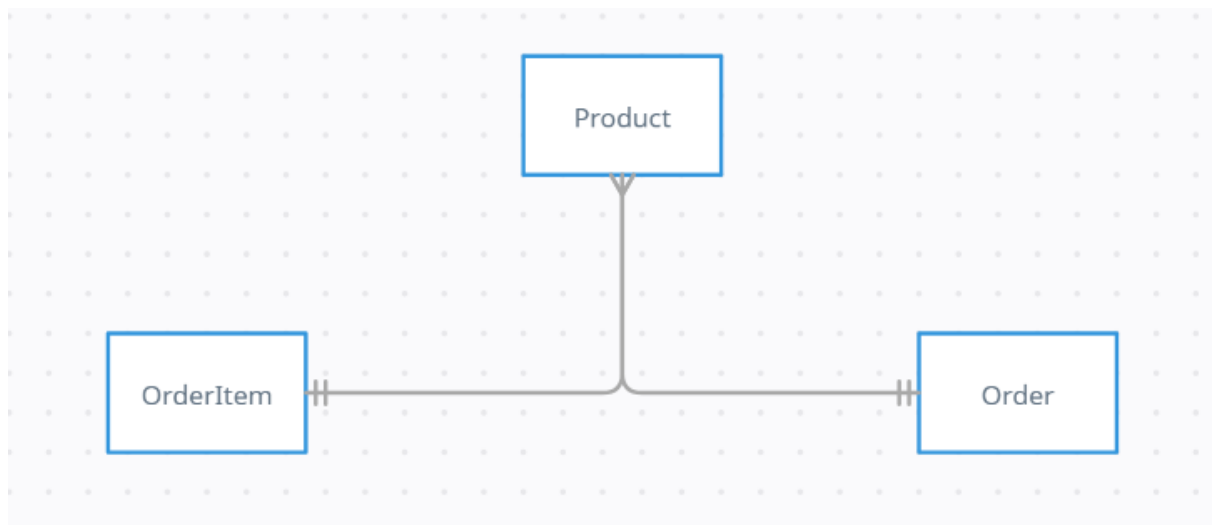
### 1. Warehouse - Product - Inventory

- **Relationship:** A specific warehouse stores a specific product in its inventory.
- **Explanation:** This ensures that the system tracks how much of a specific product is stored at a specific warehouse.

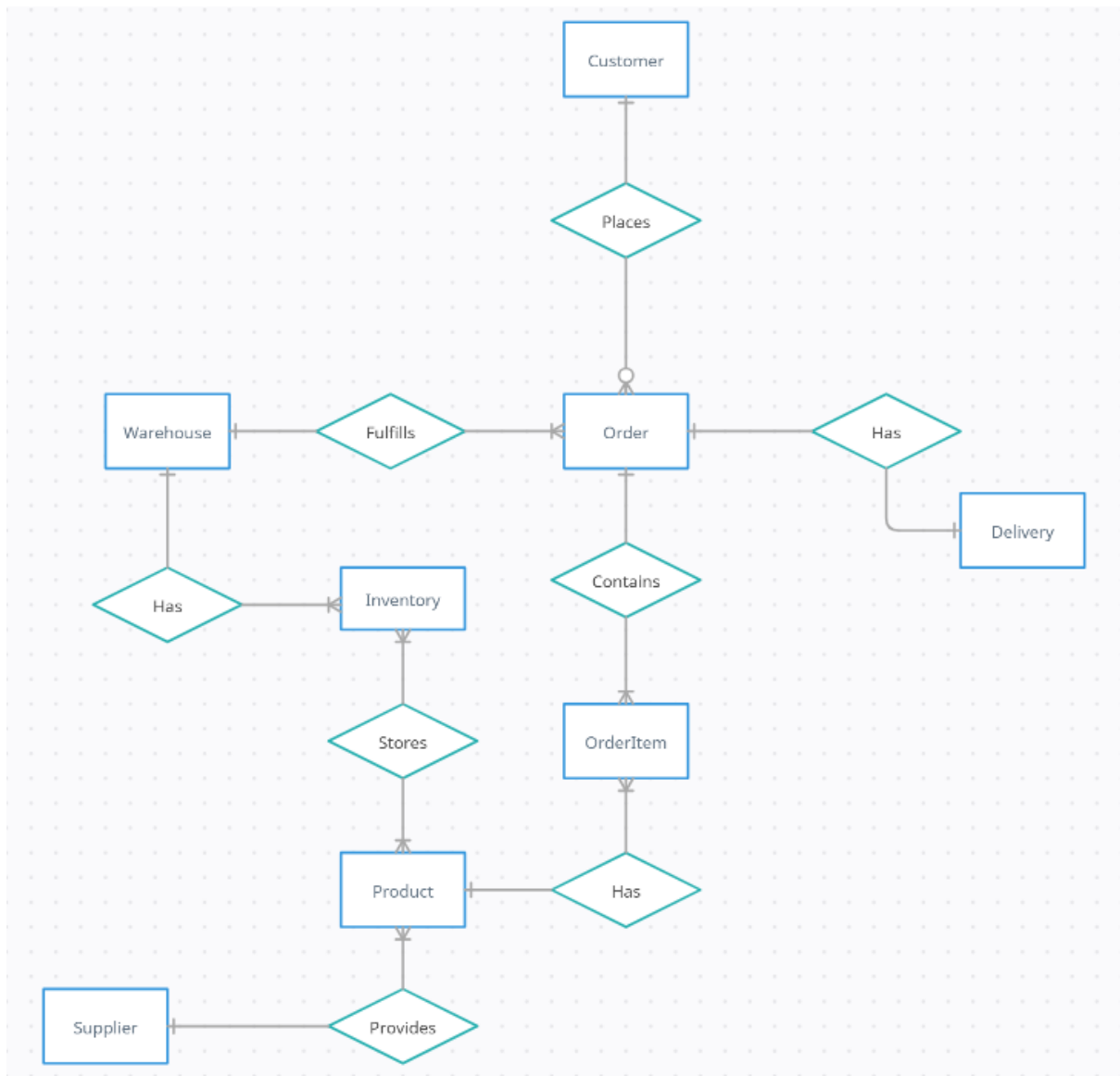


## 2. Order - Product - OrderItem

- **Relationship:** An order contains multiple products, each represented as an order item.
- **Explanation:** This allows detailed tracking of the quantity and price of individual products in an order.



## Complete ER Diagram



# RELATIONAL MODEL

## 1. Customer

<u>Customer_ID</u>	Customer_Name	House	Street	Zip_Code	Customer_Phone
--------------------	---------------	-------	--------	----------	----------------

## 2. Order

<u>Order_ID</u>	Customer_ID	Warehouse_ID	Day	Month	Year	Total_Price
-----------------	-------------	--------------	-----	-------	------	-------------

## 3. OrderItem

<u>Order_Item_ID</u>	Order_ID	Product_ID	Ordered_Quantity
----------------------	----------	------------	------------------

## 4. Product

<u>Product_ID</u>	Product_Name	Product_Description	Price
-------------------	--------------	---------------------	-------

## 5. Warehouse

<u>Warehouse_ID</u>	Warehouse_Name	House	Street	Zip_Code
---------------------	----------------	-------	--------	----------

## 6. Inventory

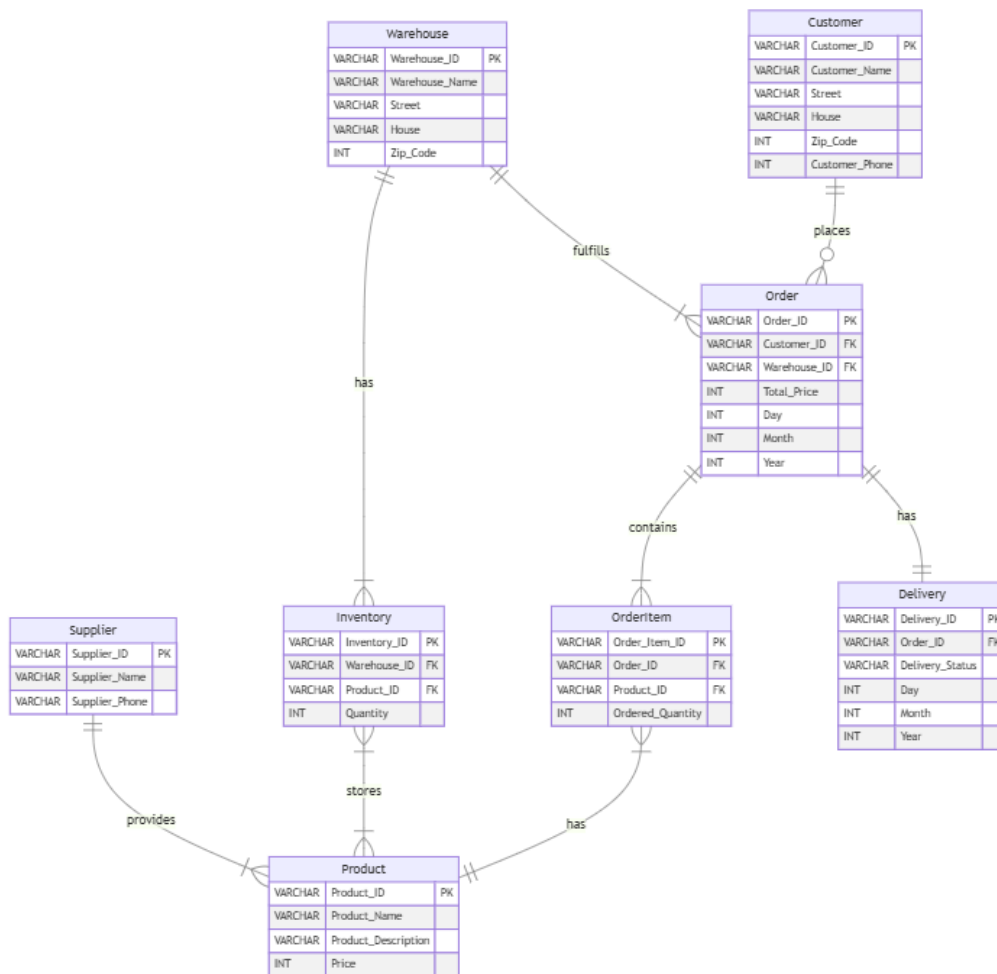
<u>Inventory_ID</u>	Warehouse_ID	Product_ID	Quantity
---------------------	--------------	------------	----------

## 7. Supplier

<u>Supplier_ID</u>	Supplier_Name	Supplier_Phone
--------------------	---------------	----------------

## 8. Delivery

<u>Delivery_ID</u>	Order_ID	Delivery_Status	Day	Month	Year
--------------------	----------	-----------------	-----	-------	------



## Entity Integrity

Table	Primary Key
Customer	Customer_ID
Order	Order_ID
OrderItem	Order_Item_ID
Warehouse	Warehouse_ID
Product	Product_ID
Inventory	Inventory_ID
Supplier	Supplier_ID
Delivery	Delivery_ID

## Referential Integrity

Table	Foreign Key	Reference
Customer	/	/
Order	Customer_ID, Warehouse_ID	Customer, Warehouse
OrderItem	Order_ID, Product_ID	Order, Product
Inventory	Warehouse_ID	Warehouse

Warehouse	/	/
Supplier	/	/
Delivery	Order_ID	Order
Product	/	/

## Domain Integrity

### 1. Customer

Customer_ID INT	Must be a unique, non-null integer (primary key).
Customer_Name VARCHAR(50)	Must be a string of up to 50 characters (e.g., letters, spaces).
House VARCHAR(10)	Must be a valid alphanumeric string (e.g., "123" or "12A").
Street VARCHAR(10)	Must be a valid alphanumeric string (e.g., "Street 5").
Zip_Code CHAR	Must be a 5-digit number.
Customer_Phone VARCHAR(15)	Must be a valid format for phone numbers.

### 2. Order

Order_ID INT	Must be a unique, non-null integer (primary key).
Customer_ID INT	Must reference a valid Customer_ID in the Customer table (foreign key).
Warehouse_ID INT	Must reference a valid Warehouse_ID in the Warehouse table (foreign key).
Total_Price DECIMAL(10, 2)	Must be a positive decimal (e.g., 0 or greater).
Day INT	Must be an integer between 1 and 31.
Month INT	Must be an integer between 1 and 12.
Year INT	Must be an integer in the range 2000–2100.

### 3. OrderItem

Order_Item_ID INT	Must be a unique, non-null integer (primary key).
Order_ID INT	Must reference a valid Order_ID in the Order table (foreign key).
Product_ID INT	Must reference a valid Product_ID in the Product table (foreign key).
Ordered_Quantity INT	Must be a positive integer (e.g., $\geq 1$ ).

### 4. Product

Product_ID INT	Must be a unique, non-null integer (primary key).
Product_Name VARCHAR(50)	Must be a unique string of up to 50 characters.
Product_Description VARCHAR(100)	Must be a string of up to 100 characters.

Product_Price DECIMAL(10, 2)	Must be a positive decimal value (e.g., $\geq 0$ ).
---------------------------------	---

## 5. Inventory

Inventory_ID INT	Must be a unique, non-null integer (primary key).
Warehouse_ID INT	Must reference a valid Warehouse_ID in the Warehouse table (foreign key).
Product_ID INT	Must reference a valid Product_ID in the Product table (foreign key).
Quantity INT	Must be a non-negative integer (e.g., $\geq 0$ ).

## 6. Warehouse

Warehouse_ID INT	Must be a unique, non-null integer (primary key).
Warehouse_Name VARCHAR(50)	Must be a string of up to 50 characters (e.g., unique names).
Street VARCHAR(10)	Must be a valid alphanumeric string (e.g., "Street 5").
House VARCHAR(10)	Must be a valid alphanumeric string (e.g., "123" or "12A").
Zip_Code CHAR(5)	Must be a 5-digit integer.

## 7. Supplier

Supplier_ID INT	Must be a unique, non-null integer (primary key).
Supplier_Name VARCHAR(50)	Must be a unique string of up to 50 characters.
Supplier_Phone VARCHAR(15)	Must follow a valid format for phone numbers.

## 8. Delivery

Delivery_ID INT	Must be a unique, non-null integer (primary key).
Order_ID INT	Must reference a valid Order_ID in the Order table (foreign key).
Day INT	Must be an integer between 1 and 31.
Month INT	Must be an integer between 1 and 12.
Year INT	Must be an integer in the range 2000–2100.
Delivery_Status VARCHAR(20)	Must belong to a predefined set of statuses (e.g., "Pending", "Shipped", "Delivered").

# Normalization

### 1NF:

**Atomicity:** All values are atomic (single-valued), as composite attributes like Customer\_Address and Order\_Date have been decomposed into their atomic components (Street, House, Zip\_Code, Day, Month, Year).

**Multivalued Attributes:** Attributes like Customer\_Phone and Supplier\_Phone are constrained to one value per record, ensuring compliance with 1NF.

### 2NF:

The system is already in 2NF. **No Partial Dependencies:** Since each table has a single-column primary key (e.g., Order\_ID in the Order table or Product\_ID in the Product table), there are no partial dependencies. Composite keys and their associated partial dependencies are avoided.

### 3NF:

The system is already in 3NF. **No Transitive Dependencies:** All attributes are dependent only on the primary key. For example, in the Customer table, attributes like Customer\_Name and Customer\_Phone depend only on Customer\_ID.

## Table Creation And Data Insertion

### 1. Customer\_T:

```
SQL> CREATE TABLE Customer (  
2     Customer_ID INT PRIMARY KEY,  
3     Customer_Name VARCHAR(50) NOT NULL,  
4     House VARCHAR(10) NOT NULL,  
5     Street VARCHAR(50) NOT NULL,  
6     Zip_Code CHAR(5) NOT NULL,  
7     Customer_Phone VARCHAR(15) NOT NULL  
8 );
```

```
SQL> ALTER TABLE Customer RENAME TO Customer_T;  
  
Table altered.
```

```
SQL> INSERT INTO Customer_T (Customer_ID, Customer_Name, House, Street, Zip_Code, Customer_Phone)  
2 VALUES  
3 (2, 'Kamala Harris', '456', 'Broadway Ave', '67890', '555-5678');  
  
1 row created.  
  
SQL> NSERT INTO Customer_T (Customer_ID, Customer_Name, House, Street, Zip_Code, Customer_Phone)  
SP2-0734: unknown command beginning "NSERT INTO..." - rest of line ignored.  
SQL> INSERT INTO Customer_T (Customer_ID, Customer_Name, House, Street, Zip_Code, Customer_Phone)  
2 VALUES  
3 (1, 'Donald Trump', '123', 'Main Street', '12345', '555-1234');  
  
1 row created.
```

## 2. Order\_T:

```
SQL> CREATE TABLE Order_T (  
 2   Order_ID INT PRIMARY KEY,  
 3       Customer_ID INT NOT NULL,  
 4       Warehouse_ID INT NOT NULL,  
 5       Day INT NOT NULL CHECK (Day BETWEEN 1 AND 31),  
 6       Month INT NOT NULL CHECK (Month BETWEEN 1 AND 12),  
 7       Year INT NOT NULL CHECK (Year BETWEEN 2000 AND 2100),  
 8       Total_Price DECIMAL(10, 2) NOT NULL,  
 9       FOREIGN KEY (Customer_ID) REFERENCES Customer_T(Customer_ID),  
10       FOREIGN KEY (Warehouse_ID) REFERENCES Warehouse_T(Warehouse_ID)  
11 );
```

Table created.

```
SQL> INSERT INTO Order_T (Order_ID, Customer_ID, Warehouse_ID, Day, Month, Year, Total_Price)  
 2 VALUES  
 3 (1, 1, 1, 15, 12, 2024, 150.50);
```

1 row created.

```
SQL> INSERT INTO Order_T (Order_ID, Customer_ID, Warehouse_ID, Day, Month, Year, Total_Price)  
 2 VALUES  
 3 (2, 2, 2, 16, 12, 2024, 200.75);
```

1 row created.

## 3. Order\_Item\_T:

```
SQL> CREATE TABLE Order_Item_T (  
 2   Order_Item_ID INT PRIMARY KEY,  
 3   Order_ID INT NOT NULL,  
 4   Product_ID INT NOT NULL,  
 5   Ordered_Quantity INT NOT NULL CHECK (Ordered_Quantity >= 1),  
 6   FOREIGN KEY (Order_ID) REFERENCES Order_T(Order_ID),  
 7   FOREIGN KEY (Product_ID) REFERENCES Product_T(Product_ID)  
 8 );
```

Table created.

```
SQL> INSERT INTO Order_Item_T (Order_Item_ID, Order_ID, Product_ID, Ordered_Quantity)  
 2 VALUES  
 3 (1, 1, 1, 3);
```

1 row created.

```
SQL> INSERT INTO Order_Item_T (Order_Item_ID, Order_ID, Product_ID, Ordered_Quantity)  
 2 VALUES  
 3 (2, 1, 2, 2);
```

1 row created.

```
SQL> INSERT INTO Order_Item_T (Order_Item_ID, Order_ID, Product_ID, Ordered_Quantity)  
 2 VALUES  
 3 (3, 2, 3, 5);
```

1 row created.

#### 4. Product\_T:

```
SQL> CREATE TABLE Product_T (  
2     Product_ID INT PRIMARY KEY,  
3     Product_Name VARCHAR(50) NOT NULL UNIQUE,  
4     Product_Description VARCHAR(100),  
5     Price DECIMAL(10, 2) NOT NULL CHECK (Price >= 0)  
6 );
```

Table created.

```
SQL> INSERT INTO Product_T (Product_ID, Product_Name, Product_Description, Price)  
2 VALUES  
3 (1, 'Pizza', 'A large cheese pizza', 10.00);
```

1 row created.

```
SQL> INSERT INTO Product_T (Product_ID, Product_Name, Product_Description, Price)  
2 VALUES  
3 (2, 'Burger', 'A juicy beef burger', 5.50);
```

1 row created.

```
SQL> INSERT INTO Product_T (Product_ID, Product_Name, Product_Description, Price)  
2 VALUES  
3 (3, 'Soda', 'A 500ml can of soda', 2.00);
```

1 row created.

#### 5. Warehouse\_T:

```
SQL> CREATE TABLE Warehouse_T (  
2     Warehouse_ID INT PRIMARY KEY,  
3     Warehouse_Name VARCHAR(50) NOT NULL UNIQUE,  
4     House VARCHAR(10) NOT NULL,  
5     Street VARCHAR(50) NOT NULL,  
6     Zip_Code CHAR(5) NOT NULL  
7 );
```

```
SQL> INSERT INTO Warehouse_T (Warehouse_ID, Warehouse_Name, House, Street, Zip_Code)  
2 VALUES  
3 (1, 'Warehouse A', '100', 'Warehouse Road', '12301');
```

1 row created.

```
SQL> INSERT INTO Warehouse_T (Warehouse_ID, Warehouse_Name, House, Street, Zip_Code)  
2 VALUES  
3 (2, 'Warehouse B', '200', 'Storage Lane', '67802');
```

1 row created.

## 6. Inventory\_T:

```
SQL> CREATE TABLE Inventory_T (  
  2     Inventory_ID INT PRIMARY KEY,  
  3     Warehouse_ID INT NOT NULL,  
  4     Product_ID INT NOT NULL,  
  5     Quantity INT NOT NULL CHECK (Quantity >= 0),  
  6     FOREIGN KEY (Warehouse_ID) REFERENCES Warehouse_T(Warehouse_ID),  
  7     FOREIGN KEY (Product_ID) REFERENCES Product_T(Product_ID)  
  8 );
```

Table created.

```
SQL> INSERT INTO Inventory_T (Inventory_ID, Warehouse_ID, Product_ID, Quantity)  
  2 VALUES  
  3 (1, 1, 1, 100);
```

1 row created.

```
SQL> INSERT INTO Inventory_T (Inventory_ID, Warehouse_ID, Product_ID, Quantity)  
  2 VALUES  
  3 (2, 1, 2, 50);
```

1 row created.

```
SQL> INSERT INTO Inventory_T (Inventory_ID, Warehouse_ID, Product_ID, Quantity)  
  2 VALUES  
  3 (3, 2, 3, 200);
```

1 row created.

## 7. Supplier\_T:

```
SQL> CREATE TABLE Supplier_T (  
  2     Supplier_ID INT PRIMARY KEY,  
  3     Supplier_Name VARCHAR(50) NOT NULL UNIQUE,  
  4     Supplier_Phone VARCHAR(15) NOT NULL  
  5 );
```

Table created.

```
SQL> INSERT INTO Supplier_T (Supplier_ID, Supplier_Name, Supplier_Phone)
  2 VALUES
  3 (1, 'Food Supplier Inc.', '555-1010');

1 row created.

SQL> INSERT INTO Supplier_T (Supplier_ID, Supplier_Name, Supplier_Phone)
  2 VALUES
  3 (2, 'Beverage Co.', '555-2020');

1 row created.
```

## 8. Delivery\_T:

```
SQL> CREATE TABLE Delivery_T (
  2 Delivery_ID INT PRIMARY KEY,
  3 Order_ID INT NOT NULL,
  4 Delivery_Status VARCHAR(20) NOT NULL CHECK (Delivery_Status IN ('Pending', 'Shipped', 'Delivered')),
  5 Day INT NOT NULL CHECK (Day BETWEEN 1 AND 31),
  6 Month INT NOT NULL CHECK (Month BETWEEN 1 AND 12),
  7 Year INT NOT NULL CHECK (Year BETWEEN 2000 AND 2100),
  8 FOREIGN KEY (Order_ID) REFERENCES Order_T(Order_ID)
  9 );

Table created.
```

```
SQL> INSERT INTO Delivery_T (Delivery_ID, Order_ID, Delivery_Status, Day, Month, Year)
  2 VALUES
  3 (1, 1, 'Shipped', 17, 12, 2024);

1 row created.

SQL> INSERT INTO Delivery_T (Delivery_ID, Order_ID, Delivery_Status, Day, Month, Year)
  2 VALUES
  3 (2, 2, 'Delivered', 18, 12, 2024);

1 row created.
```

## Querying and Testing

View all orders with Customer and Warehouse Details

```
SQL> SELECT o.Order_ID, o.Total_Price, c.Customer_Name, c.Customer_Phone, w.Warehouse_Name, w.Street, w.Zip_Code
  2 FROM Order_T o
  3 JOIN Customer_T c ON o.Customer_ID = c.Customer_ID
  4 JOIN Warehouse_T w ON o.Warehouse_ID = w.Warehouse_ID;

ORDER_ID TOTAL_PRICE CUSTOMER_NAME
-----
CUSTOMER_PHONE WAREHOUSE_NAME
-----
STREET ZIP_C
-----
1 150.5 Donald Trump
555-1234 Warehouse A
Warehouse Road 12301
2 200.75 Kamala Harris
555-5678 Warehouse B
Storage Lane 67802
```

Calculate total inventory for each warehouse

```
SQL> SELECT w.Warehouse_Name, SUM(i.Quantity * p.Price) AS Total_Inventory_Value
 2 FROM Inventory_T i
 3 JOIN Warehouse_T w ON i.Warehouse_ID = w.Warehouse_ID
 4 JOIN Product_T p ON i.Product_ID = p.Product_ID
 5 GROUP BY w.Warehouse_Name;
```

WAREHOUSE_NAME	TOTAL_INVENTORY_VALUE
Warehouse A	1275
Warehouse B	400

Check Orders for Warehouses with the prefix “Warehouse”

```
SQL> SELECT o.Order_ID, c.Customer_Name, o.Total_Price, o.Day, o.Month, o.Year
 2 FROM Order_T o
 3 JOIN Warehouse_T w ON o.Warehouse_ID = w.Warehouse_ID
 4 JOIN Customer_T c ON o.Customer_ID = c.Customer_ID
 5 WHERE w.Warehouse_Name LIKE 'Warehouse%';
```

ORDER_ID	CUSTOMER_NAME	TOTAL_PRICE
DAY	MONTH	YEAR
1	Donald Trump	150.5
15	12	2024
2	Kamala Harris	200.75
16	12	2024

## Updating/Altering Tables

Altering Customer\_Name to include only 30 characters

```
SQL> ALTER TABLE Customer_T
 2 MODIFY Customer_Name VARCHAR(30);

Table altered.

SQL> INSERT INTO Customer_T (Customer_ID, Customer_Name, House, Street, Zip_Code, Customer_Phone)
 2 VALUES
 3 (4, 'I am Abdul Haseeb Ul Hasan I am studying Computer Science From Pakistan Institute Of Engineering and Applied Sciences', '101', 'Main S
treet', 90210, '555-123-4567');
(4, 'I am Abdul Haseeb Ul Hasan I am studying Computer Science From Pakistan Institute Of Engineering and Applied Sciences', '101', 'Main Street
', 90210, '555-123-4567')

ERROR at line 3:
ORA-12899: value too large for column "REPORT"."CUSTOMER_T"."CUSTOMER_NAME"
(actual: 117, maximum: 30)
```

Update Customer\_Address

```
SQL> UPDATE Customer_T
 2 SET House = '111', Street = 'Broadway Street', Zip_Code = 10012
 3 WHERE Customer_ID = 1;

1 row updated.

SQL> SELECT House, Street, Zip_Code
 2 FROM Customer_T
 3 WHERE Customer_ID = 1;
```

HOUSE	STREET	ZIP_C
111	Broadway Street	10012