Introduction

This database is designed to manage essential data for a wholesale or retail business. Its core functionalities include:

- **Inventory Tracking:** The 'product' and 'warehouse' tables enable effective inventory management with stock levels, product descriptions, and warehouse locations.
- **Supplier Management:** The 'supplier' table stores comprehensive supplier details, facilitating procurement and supply chain optimization.
- **Customer Relationship Management (CRM):** The 'customer' table provides a centralized repository for customer information, supporting sales and marketing efforts.
- **Sales and Order Processing:** The 'order_table' and 'order_details' tables enable streamlined management of orders, including order dates, items, quantities, and total amounts.
- **Employee Management:** The 'employees' table stores key employee data such as names, positions, contact details, and hire dates.

Module Description

1. Inventory Management

- **Description:** Controls product information, stock updates, warehouse locations, and reorder management
- **Functions:**
    - Add new products
    - Update product descriptions and pricing
    - Adjust stock quantities (incoming shipments, sales deductions)
    - Generate low-stock alerts
    - Track inventory across multiple warehouses

2. Supplier Management

- **Description:** Manages supplier information, sourcing, and contact management.
- **Functions:**
    - Add/edit supplier records
    - Record contact information and communication history
    - Track supplier performance (lead times, product quality)
    - Link suppliers to specific products

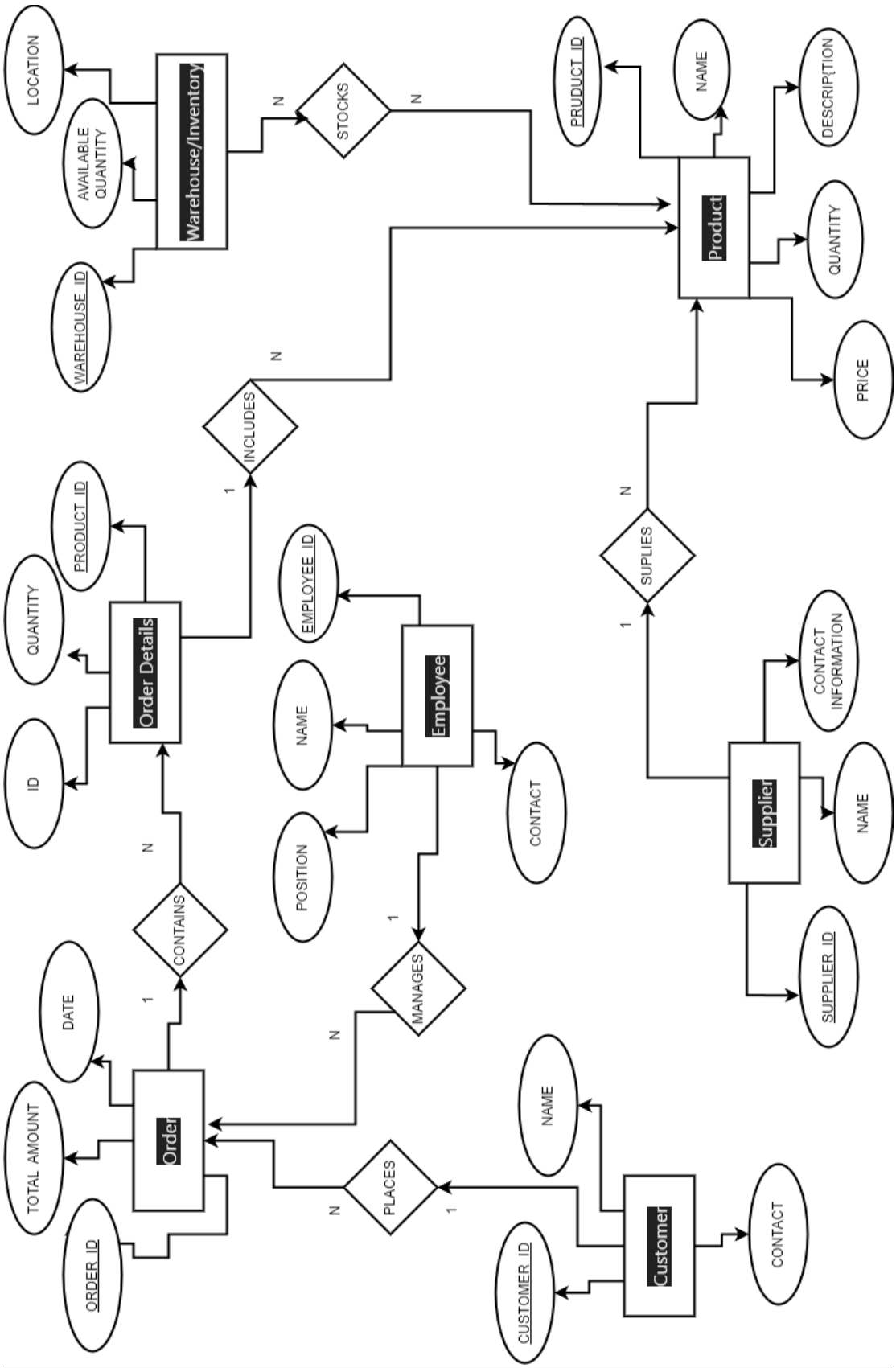
3. Customer Management (CRM)

- **Description:** Centralizes customer data and interactions.
- **Functions:**
    - Add/edit customer profiles
    - Track purchase history
    - Generate customer order reports
    - Support targeted marketing or loyalty programs

4. Employee Management

- **Description:** Maintains employee records and potentially sales performance tracking.
- **Functions:**
    - Store employee data (names, positions, hire dates, etc.)
    - Track sales performance by employee (if applicable)
    - Manage employee permissions within the system (if you have a user interface)

Entity-Relationship Diagram:



Data Dictionary

1. Employee Table:

Attribute name	Data type	Max field size	Description
Employee_id	int	10	Employee Identity
Position	varchar	20	Position of emp.
Fname	varchar	25	First Name
Lname	varchar	25	Last Name
Contact	int	10	Mobile number

2. Order table:

Attribute name	Data type	Max field size	Description
Order_id	int	10	Order id
Total_amount	int	20	Total amount of product
Date	date	-	Order date

3. Order details:

Attribute name	Data type	Max field size	Description
id	int	10	Order id
quantity	int	25	quantity
Product_id	int	10	Product id

4. Customer:

Attribute name	Data type	Max field size	Description
Name	varchar	25	Name of customer
Contact	int	10	Mobile of customer
Customer_id	int	10	Customer id

5. Supplier:

Attribute name	Data type	Max field size	Description
Supplier_id	int	10	Supplier id
Name	varchar	25	Name of supplier
Contact	int	10	Mobile number of supplier

6. Product:

Attribute name	**Data type**	**Max field size**	**Description**
Name	varchar	25	Product name
Quantity	int	25	Product quantity
description	varchar	50	Product description
price	int	10	price

7. Warehouse/inventory:

Attribute name	**Data type**	**Max field size**	**Description**
location	varchar	25	Warehouse location
Warehouse_id	int	10	Warehouse id
Available_quantity	int	10	Quantity available

DDL for table creation

1. Employees

```
CREATE TABLE employees (
  Employee_id INT(10) PRIMARY KEY,
  Position VARCHAR(20),
  Fname VARCHAR(25),
  Lname VARCHAR(25),
  Contact INT(10)
);
```

2. Customer

```
CREATE TABLE customer (
Customer_id INT(10) PRIMARY KEY,
  Name VARCHAR(25),
  Contact INT(10)
);
```

3. Product

```
CREATE TABLE product (
  Product_id INT PRIMARY KEY,
  Name VARCHAR(25),
  Quantity INT(25),
  description VARCHAR(50),
  price INT(10),
  Supplier_id INT,
  FOREIGN KEY (Supplier_id) REFERENCES supplier(Supplier_id)
);
```

4. Order details

```
CREATE TABLE order_details (
  Order_id INT(10) PRIMARY KEY,
  Total_amount INT(20),
  Date DATE
);
```

5. order_table

```
CREATE TABLE order_table (
  Order_id INT PRIMARY KEY,
  Date DATETIME, -- Stores both date and time of the order
  Total_amount DECIMAL(10,2), -- Adjust precision if needed
  Customer_id INT,
  Employee_id INT,
  FOREIGN KEY (Customer_id) REFERENCES customer(Customer_id),
  FOREIGN KEY (Employee_id) REFERENCES employees(Employee_id)
);
```

6. Warehouse

```
CREATE TABLE warehouse (
   Warehouse_id INT(10) PRIMARY KEY,
   location VARCHAR(25),
   Available_quantity INT(10),
   Product_id INT, -- Optional foreign key to link to the 'product' table
   FOREIGN KEY (Product_id) REFERENCES product(Product_id)
```

7. Supplier

```
CREATE TABLE supplier (
  Supplier_id INT(10) PRIMARY KEY,
   Name VARCHAR(25),
   Contact INT(10)
);
```

Simple Query

Query : List all products with a price over $500

```
mysql> SELECT Name, price
    -> FROM product
    -> WHERE price > 500;
+------------------+-------+
| Name             | price |
+------------------+-------+
| Laptop           |   800 |
| Smartphone       |   550 |
| Desktop Computer |  1200 |
| Camera           |  1500 |
+------------------+-------+
4 rows in set (0.00 sec)
```

Query : Get the total quantity of 'Desktop Computers' in stock across all warehouses

```
mysql> SELECT SUM(Available_quantity) AS total_desktop_stock
    -> FROM warehouse
    -> JOIN product ON warehouse.Product_id = product.Product_id
    -> WHERE product.Name = 'Desktop Computer';
+--------------------+
| total_desktop_stock |
+--------------------+
|               NULL |
+--------------------+
1 row in set (0.01 sec)
```

Query : Find the supplier who provides 'Smartphones'

```
mysql> SELECT supplier.Name
    -> FROM supplier
    -> JOIN product ON supplier.Supplier_id = product.Supplier_id
    -> WHERE product.Name = 'Smartphone';
Empty set (0.00 sec)
```

Query : List the customers with names starting with the letter 'E'

```
mysql> SELECT Name
    -> FROM customer
    -> WHERE Name LIKE 'E%';
+-------------+
| Name        |
+-------------+
| Emily Davis |
+-------------+
1 row in set (0.01 sec)
```

Query : write a query to select all warehouses where available_quantity is above 500

```
mysql> SELECT *
    -> FROM warehouse
    -> WHERE Available_quantity > 500;
+--------------+-------------+--------------------+------------+
| Warehouse_id | location    | Available_quantity | Product_id |
+--------------+-------------+--------------------+------------+
|            2 | Los Angeles |                800 |       NULL |
|            3 | Chicago     |                650 |       NULL |
|            5 | Dallas      |                700 |       NULL |
|            7 | Miami       |                550 |       NULL |
|            8 | Houston     |                600 |       NULL |
+--------------+-------------+--------------------+------------+
5 rows in set (0.00 sec)
```

Query : see all attributes of table.

```
mysql> describe employees;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Employee_id | int         | NO   | PRI | NULL    |       |
| Position    | varchar(20) | YES  |     | NULL    |       |
| Fname       | varchar(25) | YES  |     | NULL    |       |
| Lname       | varchar(25) | YES  |     | NULL    |       |
| Contact     | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

Query: insert data into employee.

```
mysql> INSERT INTO employees (Employee_id, Position, Fname, Lname, Contact)
    -> VALUES
    ->     (1, 'Sales Manager', 'Sarah', 'Johnson', 1234567890),
    ->     (2, 'Software Engineer', 'Michael', 'Lee', 9876543210),
    ->     (3, 'Marketing Associate', 'Emily', 'Davis', 3331234567),
    ->     (4, 'Accountant', 'David', 'Wilson', 4445678901),
    ->     (5, 'Customer Support', 'Olivia', 'Miller', 5556789012),
    ->     (6, 'HR Manager', 'Thomas', 'Anderson', 6667890123),
    ->     (7, 'Data Analyst', 'Sophia', 'Martinez', 7778901234),
    ->     (8, 'Product Designer', 'Joshua', 'Thompson', 8889012345),
    ->     (9, 'Sales Representative', 'Isabella', 'Garcia', 9990123456),
    ->     (10, 'IT Technician', 'Noah', 'Williams', 2223456789);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Query: insert data into order_table.

```
mysql> INSERT INTO order_table (Order_id, Total_amount, Date)
    -> VALUES
    ->     (1, 500, '2023-12-12'),
    ->     (2, 320, '2023-12-15'),
    ->     (3, 850, '2023-12-18'),
    ->     (4, 120, '2023-12-20'),
    ->     (5, 955, '2023-12-22'),
    ->     (6, 480, '2023-12-23'),
    ->     (7, 275, '2023-12-24'),
    ->     (8, 640, '2023-12-25'),
    ->     (9, 1100, '2023-12-28'),
    ->     (10, 790, '2023-12-30');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> CREATE TABLE order_details (
    ->     id INT(10) PRIMARY KEY,
    ->     quantity INT(25),
    ->     Product_id INT(10),
    ->     Order_id INT(10), -- Assuming you eventually want to link to an 'order_table'
    ->     FOREIGN KEY (Product_id) REFERENCES product(Product_id),
    ->     FOREIGN KEY (Order_id) REFERENCES order_table(Order_id)
    -> );
Query OK, 0 rows affected, 4 warnings (0.06 sec)
```

```
mysql> INSERT INTO product (Product_id, Name, Quantity, description, price)
    -> VALUES
    ->     (1, 'Laptop', 20, 'High-performance laptop', 800),
    ->     (2, 'Smartphone', 50, 'Latest generation smartphone', 550),
    ->     (3, 'Desktop Computer', 15, 'Powerful desktop PC', 1200),
    ->     (4, 'Wireless Headphones', 30, 'Noise-cancelling headphones', 150),
    ->     (5, 'Smartwatch', 40, 'Fitness and health tracker', 200),
    ->     (6, 'Tablet', 25, 'Versatile tablet for work and play', 400),
    ->     (7, 'Gaming Console', 10, 'Next-gen gaming console', 500),
    ->     (8, 'Camera', 8, 'Professional-grade camera', 1500),
    ->     (9, 'Smart Speaker', 60, 'Voice-controlled smart home hub', 80),
    ->     (10, 'Ebook Reader', 35, 'Dedicated e-reader', 120);
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO order_details (id, quantity, Product_id, Order_id)
    -> VALUES
    ->     (1, 2, 1, 1),
    ->     (2, 1, 5, 2),
    ->     (3, 3, 3, 1),
    ->     (4, 5, 2, 3),
    ->     (5, 1, 4, 2),
    ->     (6, 2, 1, 3),
    ->     (7, 4, 5, 1),
    ->     (8, 1, 2, 2),
    ->     (9, 2, 4, 3),
    ->     (10, 3, 3, 2);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Query: change size of int to bigint.

```
mysql> ALTER TABLE customer
    ->
    -> MODIFY COLUMN Contact bigint;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO customer (Customer_id, Name, Contact)
    -> VALUES
    ->     (1, 'Sarah Johnson', 1234567890),
    ->     (2, 'Michael Lee', 9876543210),
    ->     (3, 'Emily Davis', 3331234567),
    ->     (4, 'David Wilson', 4445678901),
    ->     (5, 'Olivia Miller', 5556789012),
    ->     (6, 'Thomas Anderson', 6667890123),
    ->     (7, 'Sophia Martinez', 7778901234),
    ->     (8, 'Joshua Thompson', 8889012345),
    ->     (9, 'Isabella Garcia', 9990123456),
    ->     (10, 'Noah Williams', 2223456789);
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT Fname, Lname, Position
    -> FROM employees
    -> WHERE Position LIKE 'Sales%';
+----------+---------+----------------------+
| Fname    | Lname   | Position             |
+----------+---------+----------------------+
| Sarah    | Johnson | Sales Manager        |
| Isabella | Garcia  | Sales Representative |
+----------+---------+----------------------+
2 rows in set (0.01 sec)
```

```
mysql> SELECT *
    -> FROM product
    -> WHERE price BETWEEN 300 AND 1000;
+------------+----------------+----------+-----------------------------------+-------+-------------+
| Product_id | Name           | Quantity | description                       | price | Supplier_id |
+------------+----------------+----------+-----------------------------------+-------+-------------+
|          1 | Laptop         |       20 | High-performance laptop           |   800 |        NULL |
|          2 | Smartphone     |       50 | Latest generation smartphone      |   550 |        NULL |
|          6 | Tablet         |       25 | Versatile tablet for work and play |   400 |        NULL |
|          7 | Gaming Console |       10 | Next-gen gaming console           |   500 |        NULL |
+------------+----------------+----------+-----------------------------------+-------+-------------+
4 rows in set (0.01 sec)
```

Query : Count how many customers have names starting with the letter 'S'.

```
mysql> SELECT COUNT(*) AS customers_starting_with_S
    -> FROM customer
    -> WHERE Name LIKE 'S%';
+---------------------------+
| customers_starting_with_S |
+---------------------------+
|                         2 |
+---------------------------+
1 row in set (0.02 sec)
```

Query : List all order details that include a 'Smartwatch'.

```
mysql> SELECT *
    -> FROM order_details
    -> JOIN product ON order_details.Product_id = product.Product_id
    -> WHERE product.Name = 'Smartwatch';
+----+----------+------------+----------+------------+------------+----------+---------------------------+-------+-------------+
| id | quantity | Product_id | Order_id | Product_id | Name       | Quantity | description               | price | Supplier_id |
+----+----------+------------+----------+------------+------------+----------+---------------------------+-------+-------------+
|  2 |        1 |          5 |        2 |          5 | Smartwatch |       40 | Fitness and health tracker |   200 |           5 |
|  7 |        4 |          5 |        1 |          5 | Smartwatch |       40 | Fitness and health tracker |   200 |           5 |
+----+----------+------------+----------+------------+------------+----------+---------------------------+-------+-------------+
2 rows in set (0.03 sec)
```

Query : Get the name of the warehouse with the lowest 'Available_quantity'.

```
mysql> SELECT location, Available_quantity
    -> FROM warehouse
    -> ORDER BY Available_quantity ASC
    -> LIMIT 1;
+----------+--------------------+
| location | Available_quantity |
+----------+--------------------+
| Boston   |                300 |
+----------+--------------------+
1 row in set (0.00 sec)
```

Query : Calculate the total value of the 'Laptop' inventory (price * quantity).

```
mysql> SELECT (price * Quantity) AS total_laptop_value
    -> FROM product
    -> WHERE Name = 'Laptop';
+--------------------+
| total_laptop_value |
+--------------------+
|              16000 |
+--------------------+
1 row in set (0.01 sec)
```

Query : List the 3 most expensive products.

```
mysql> SELECT Name, price
    -> FROM product
    -> ORDER BY price DESC
    -> LIMIT 3;
+------------------+-------+
| Name             | price |
+------------------+-------+
| Camera           |  1500 |
| Desktop Computer |  1200 |
| Laptop           |   800 |
+------------------+-------+
3 rows in set (0.00 sec)
```

Query : Count the number of employees with the last name 'Johnson'.

```
mysql> SELECT COUNT(*) AS johnson_count
    -> FROM employees
    -> WHERE Lname = 'Johnson';
+---------------+
| johnson_count |
+---------------+
|             1 |
+---------------+
1 row in set (0.00 sec)
```

Complex query

Query- adding values to a column which all together.

Existing table...

```
mysql> select * from warehouse;
+--------------+---------------+--------------------+------------+
| Warehouse_id | location      | Available_quantity | Product_id |
+--------------+---------------+--------------------+------------+
|            1 | New York City |                500 |       NULL |
|            2 | Los Angeles   |                800 |       NULL |
|            3 | Chicago       |                650 |       NULL |
|            4 | Atlanta       |                400 |       NULL |
|            5 | Dallas        |                700 |       NULL |
|            6 | Seattle       |                350 |       NULL |
|            7 | Miami         |                550 |       NULL |
|            8 | Houston       |                600 |       NULL |
|            9 | Boston        |                300 |       NULL |
|           10 | Denver        |                450 |       NULL |
+--------------+---------------+--------------------+------------+
10 rows in set (0.00 sec)
```

Creating a new temporary table

```
mysql> CREATE TEMPORARY TABLE warehouse_product_mapping (
    ->     Warehouse_id INT,
    ->     Product_id INT
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Adding the column in the temporary table

```
mysql> INSERT INTO warehouse_product_mapping (Warehouse_id, Product_id)
    -> VALUES
    ->     (1, 1),
    ->     (2, 2),
    -> (3,3),
    -> (4,4),
    -> (5,5),
    -> (6,6),
    -> (7,7),
    -> (8,8),
    -> (9,9),
    -> (10,10)
    -> ;
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from warehouse_product_mapping;
+--------------+------------+
| Warehouse_id | Product_id |
+--------------+------------+
|            1 |          1 |
|            2 |          2 |
|            3 |          3 |
|            4 |          4 |
|            5 |          5 |
|            6 |          6 |
|            7 |          7 |
|            8 |          8 |
|            9 |          9 |
|           10 |         10 |
+--------------+------------+
10 rows in set (0.00 sec)
```

Query : Joining the temporary table with the warehouse table...

```
mysql> UPDATE warehouse w
    -> JOIN warehouse_product_mapping map ON w.Warehouse_id = map.Warehouse_id
    -> SET w.Product_id = map.Product_id;
Query OK, 10 rows affected (0.02 sec)
Rows matched: 10  Changed: 10  Warnings: 0

mysql> select * from warehouse;
+--------------+---------------+--------------------+------------+
| Warehouse_id | location      | Available_quantity | Product_id |
+--------------+---------------+--------------------+------------+
|            1 | New York City |                500 |          1 |
|            2 | Los Angeles   |                800 |          2 |
|            3 | Chicago       |                650 |          3 |
|            4 | Atlanta       |                400 |          4 |
|            5 | Dallas        |                700 |          5 |
|            6 | Seattle       |                350 |          6 |
|            7 | Miami         |                550 |          7 |
|            8 | Houston       |                600 |          8 |
|            9 | Boston        |                300 |          9 |
|           10 | Denver        |                450 |         10 |
+--------------+---------------+--------------------+------------+
10 rows in set (0.00 sec)
```

Query : Find the total quantity of 'Desktop Computer' available across all warehouses.

```
mysql> SELECT SUM(Available_quantity) AS total_desktop_quantity
    -> FROM warehouse
    -> JOIN product ON warehouse.Product_id = product.Product_id
    -> WHERE product.Name = 'Desktop Computer';
+-------------------------+
| total_desktop_quantity |
+-------------------------+
|                     650 |
+-------------------------+
1 row in set (0.01 sec)
```

Query : Get the total value of inventory held in warehouses located in California (assuming you add a region/state column to the warehouse table).

```
mysql> SELECT SUM(product.price * warehouse.Available_quantity) AS total_california_inventory_value
    -> FROM warehouse
    -> JOIN product ON warehouse.Product_id = product.Product_id
    -> WHERE warehouse.location LIKE '%denver%';
+---------------------------------+
| total_california_inventory_value |
+---------------------------------+
|                           54000 |
+---------------------------------+
1 row in set (0.00 sec)
```

Query : List the top 3 suppliers based on the total value of products they provide.

```
mysql> SELECT supplier.Name, SUM(product.price * quantity) AS total_value
    -> FROM supplier
    -> JOIN product ON supplier.Supplier_id = product.Supplier_id
    -> GROUP BY supplier.Name
    -> ORDER BY total_value DESC
    -> LIMIT 3;
+---------------------+-------------+
| Name                | total_value |
+---------------------+-------------+
| Gadget Supplies     |       27500 |
| Component Solutions |       18000 |
| Tech Giant          |       16000 |
+---------------------+-------------+
3 rows in set (0.00 sec)
```

Query : Find customers who have ordered both a 'Laptop' and a 'Smartwatch'.

```
mysql> SELECT supplier.Name, SUM(product.price * quantity) AS total_value
    -> FROM supplier
    -> JOIN product ON supplier.Supplier_id = product.Supplier_id
    -> GROUP BY supplier.Name
    -> ORDER BY total_value DESC
    -> LIMIT 3;
+---------------------+-------------+
| Name                | total_value |
+---------------------+-------------+
| Gadget Supplies     |       27500 |
| Component Solutions |       18000 |
| Tech Giant          |       16000 |
+---------------------+-------------+
3 rows in set (0.00 sec)
```

Query : Calculate the average order value for each month

```
mysql> SELECT MONTH(Date) AS order_month, AVG(Total_amount) AS avg_order_value
    -> FROM order_table
    -> GROUP BY MONTH(Date);
+-------------+-----------------+
| order_month | avg_order_value |
+-------------+-----------------+
|           9 |        500.0000 |
|          10 |        585.0000 |
|          12 |        622.8571 |
+-------------+-----------------+
3 rows in set (0.00 sec)
```

Query : Get a list of products that have never been ordered.

```
mysql> SELECT product.Name
    -> FROM product
    -> LEFT JOIN order_details ON product.Product_id = order_details.Product_id
    -> WHERE order_details.Product_id IS NULL;
+----------------+
| Name           |
+----------------+
| Tablet         |
| Gaming Console |
| Camera         |
| Smart Speaker  |
| Ebook Reader   |
+----------------+
5 rows in set (0.01 sec)
```

Query : Find employees hired before the year 2023 along with the total number of years they've been employed

```
mysql> SELECT Fname, Lname, YEAR(CURDATE()) - YEAR(hire_date) AS years_employed
    -> FROM employees
    -> WHERE YEAR(hire_date) < 2023;
+---------+---------+----------------+
| Fname   | Lname   | years_employed |
+---------+---------+----------------+
| Sarah   | Johnson |             14 |
| Michael | Lee     |             12 |
| Emily   | Davis   |             11 |
| David   | Wilson  |              9 |
| Olivia  | Miller  |              8 |
| Thomas  | Anderson|              7 |
| Sophia  | Martinez|              6 |
| Joshua  | Thompson|              5 |
| Isabella| Garcia  |              5 |
| Noah    | Williams|              2 |
+---------+---------+----------------+
10 rows in set (0.01 sec)
```

Query : List warehouses where the quantity of 'Desktop Computer' exceeds the average 'Desktop Computer' quantity across all warehouses.

```
mysql> SELECT location
    -> FROM warehouse
    -> WHERE Available_quantity > (
    ->     SELECT AVG(Available_quantity)
    ->     FROM warehouse
    ->     JOIN product ON warehouse.Product_id = product.Product_id
    ->     WHERE product.Name = 'Desktop Computer'
    -> );
+-------------+
| location    |
+-------------+
| Los Angeles |
| Dallas      |
+-------------+
2 rows in set (0.00 sec)
```

Query: procedure.

```
mysql> DELIMITER &&
mysql> CREATE PROCEDURE calculate_years_employed()
    -> BEGIN
    ->     SELECT Fname, Lname, YEAR(CURDATE()) - YEAR(hire_date) AS years_employed
    ->     FROM employees
    ->     WHERE YEAR(hire_date) < 2023;
    -> END &&
Query OK, 0 rows affected (0.03 sec)
```

OuterJoin

Query : List all products, including those without any orders.

```
mysql> use wholesale_system;
Database changed
mysql> SELECT product.Name, product.Price, SUM(order_details.quantity) AS total_ordered
    -> FROM product
    -> LEFT JOIN order_details ON product.Product_id = order_details.Product_id
    -> GROUP BY product.Name, product.Price;
+--------------------+-------+---------------+
| Name               | Price | total_ordered |
+--------------------+-------+---------------+
| Laptop             |   800 |             4 |
| Smartphone         |   550 |             6 |
| Desktop Computer   |  1200 |             6 |
| Wireless Headphones |   150 |             3 |
| Smartwatch         |   200 |             5 |
| Tablet             |   400 |          NULL |
| Gaming Console     |   500 |          NULL |
| Camera             |  1500 |          NULL |
| Smart Speaker      |    80 |          NULL |
| Ebook Reader       |   120 |          NULL |
+--------------------+-------+---------------+
10 rows in set (0.00 sec)
```

Query 3: Find warehouses that haven't stocked a particular product (let's say Product ID 5).

```
mysql> SELECT warehouse.location, warehouse.Available_quantity
    -> FROM warehouse
    -> RIGHT JOIN product ON warehouse.Product_id = product.Product_id
    -> WHERE product.Product_id = 5;
+----------+--------------------+
| location | Available_quantity |
+----------+--------------------+
| Dallas   |                700 |
+----------+--------------------+
1 row in set (0.00 sec)
```

Nested query

Query : Get the total order value for the most expensive product.

```
mysql> SELECT SUM(order_details.quantity * price) AS most_expensive_total
    -> FROM order_details
    -> JOIN product ON order_details.Product_id = product.Product_id
    -> WHERE product.price = (
    ->     SELECT MAX(price)
    ->     FROM product
    -> );
+----------------------+
| most_expensive_total |
+----------------------+
|                 NULL |
+----------------------+
1 row in set (0.01 sec)
```

Query : Find employees who work in warehouses with 'Laptop' in stock.

```
mysql> SELECT Fname, Lname
    -> FROM employees
    -> WHERE Employee_id IN (
    ->     SELECT Employee_id
    ->     FROM warehouse
    ->     JOIN product ON warehouse.Product_id = product.Product_id
    ->     WHERE product.Name = 'Laptop'
    -> );
+----------+----------+
| Fname    | Lname    |
+----------+----------+
| Sarah    | Johnson  |
| Michael  | Lee      |
| Emily    | Davis    |
| David    | Wilson   |
| Olivia   | Miller   |
| Thomas   | Anderson |
| Sophia   | Martinez |
| Joshua   | Thompson |
| Isabella | Garcia   |
| Noah     | Williams |
+----------+----------+
10 rows in set (0.01 sec)
```

Query : List the suppliers of products that have never been ordered.

```
mysql> SELECT supplier.Name
    -> FROM supplier
    -> JOIN product ON supplier.Supplier_id = product.Supplier_id
    -> WHERE product.Product_id NOT IN (
    ->     SELECT Product_id
    ->     FROM order_details
    -> );
+----------------------+
| Name                 |
+----------------------+
| Direct Distributors  |
| Innovative Tech      |
| Reliable Components  |
| Apex Electronics     |
| Future Tech Supplies |
+----------------------+
5 rows in set (0.00 sec)
```

Triggers

```
mysql> DELIMITER //
mysql> CREATE TRIGGER calculate_tenure
    -> BEFORE INSERT ON employees
    -> FOR EACH ROW
    -> BEGIN
    ->     SET NEW.years_of_service = DATEDIFF(CURDATE(), NEW.hire_date) / 365;
    -> END //
```

Query : Total sales per month for the past year

```
mysql> SELECT MONTH(order_table.Date) AS order_month, SUM(order_table.Total_amount) AS total_sales
    -> FROM order_table
    -> WHERE YEAR(order_table.Date) = YEAR(CURDATE()) - 1
    -> GROUP BY MONTH(order_table.Date);
+-------------+-------------+
| order_month | total_sales |
+-------------+-------------+
|           9 |         500 |
|          10 |        1170 |
|          12 |        4360 |
+-------------+-------------+
3 rows in set (0.00 sec)
```

Query : Products from suppliers who have had consistently increasing average order value over the past 3 months

```
mysql> SELECT product.Name, supplier.Name
    -> FROM product
    -> JOIN supplier ON product.Supplier_id = supplier.Supplier_id
    -> JOIN order_details ON product.Product_id = order_details.Product_id
    -> JOIN order_table ON order_details.Order_id = order_table.Order_id
    -> WHERE order_table.Date > DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    -> GROUP BY product.Name, supplier.Name
    -> HAVING (
    ->     SELECT AVG(Total_amount)
    ->     FROM order_table t1
    ->     WHERE t1.Date BETWEEN DATE_SUB(order_table.Date, INTERVAL 2 MONTH) AND DATE_SUB(order_table.Date, INTERVAL 1
MONTH)
    ->       AND t1.Customer_id = order_table.Customer_id
    ->    ) > (
    ->     SELECT AVG(Total_amount)
    ->     FROM order_table t2
    ->     WHERE t2.Date BETWEEN DATE_SUB(order_table.Date, INTERVAL 1 MONTH) AND order_table.Date
    ->       AND t2.Customer_id = order_table.Customer_id
    ->    );
ERROR 1054 (42S22): Unknown column 'wholesale_system.order_table.Date' in 'where clause'
```