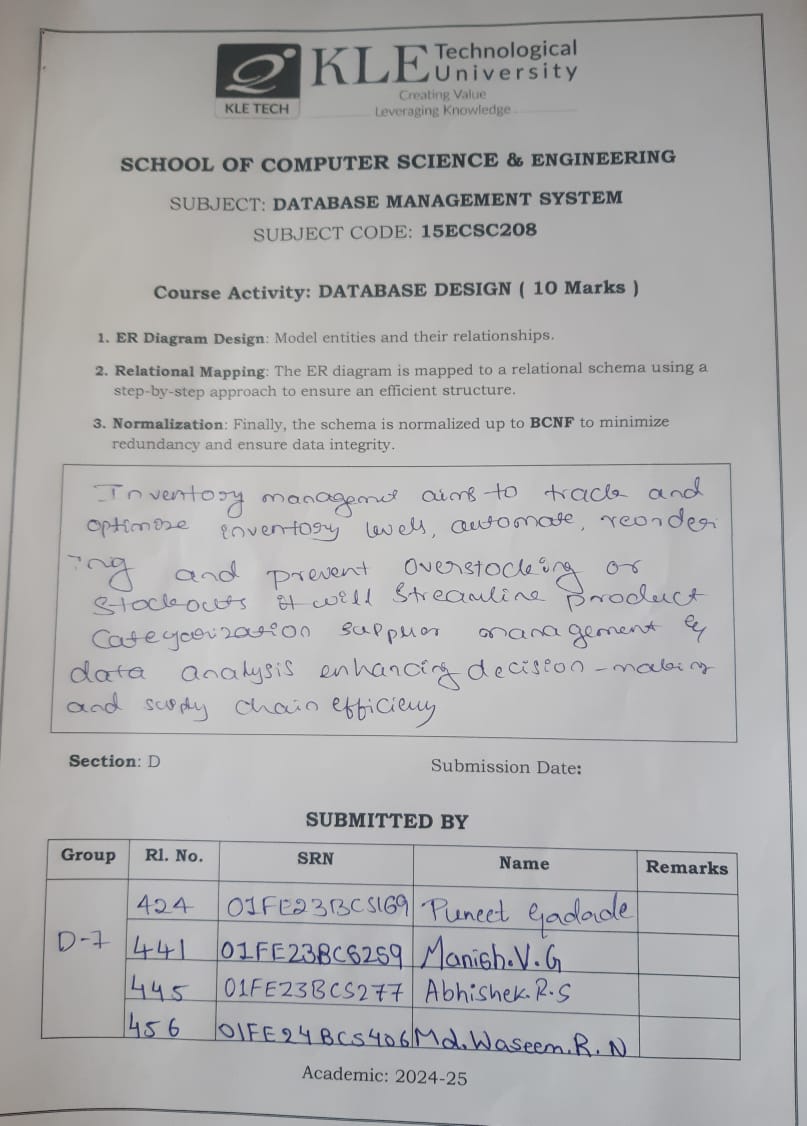
****

**DBMS Activity**

**Inventory management System**

**Course Title:** Database Management System

**Team Members:**

1. PUNEET GADADE

2. MANISH V G

3. ABHISHEK R S

4. MD.WASEEM N

**Submitted To:**

- Instructor Name: Manohar Madgi

**MINI-WORLD DISCRIPTION OF INVENTORY MANAGEMENT::**

Due to the digitalization and increasing demand for efficient stock control, inventory management systems have become vital for businesses to track and manage their stock efficiently. The data generated by these systems can be stored and managed using a Database Management System.

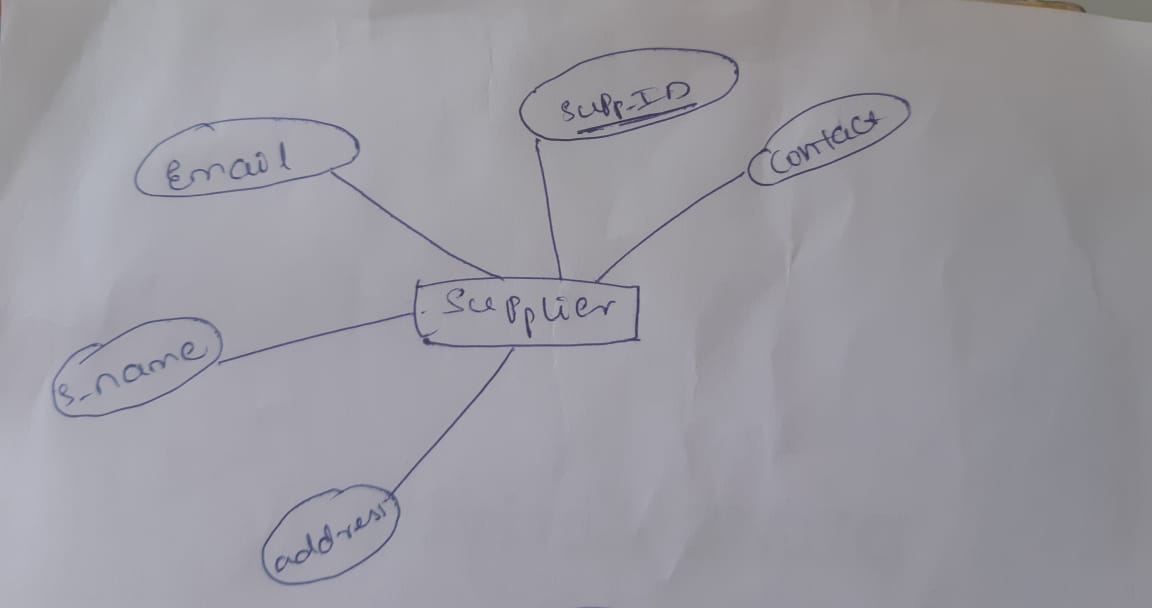
Our database keeps track of products, suppliers, warehouses, employees, and customers involved in inventory operations. It also records transactions, orders, and stock availability for smooth business operations.

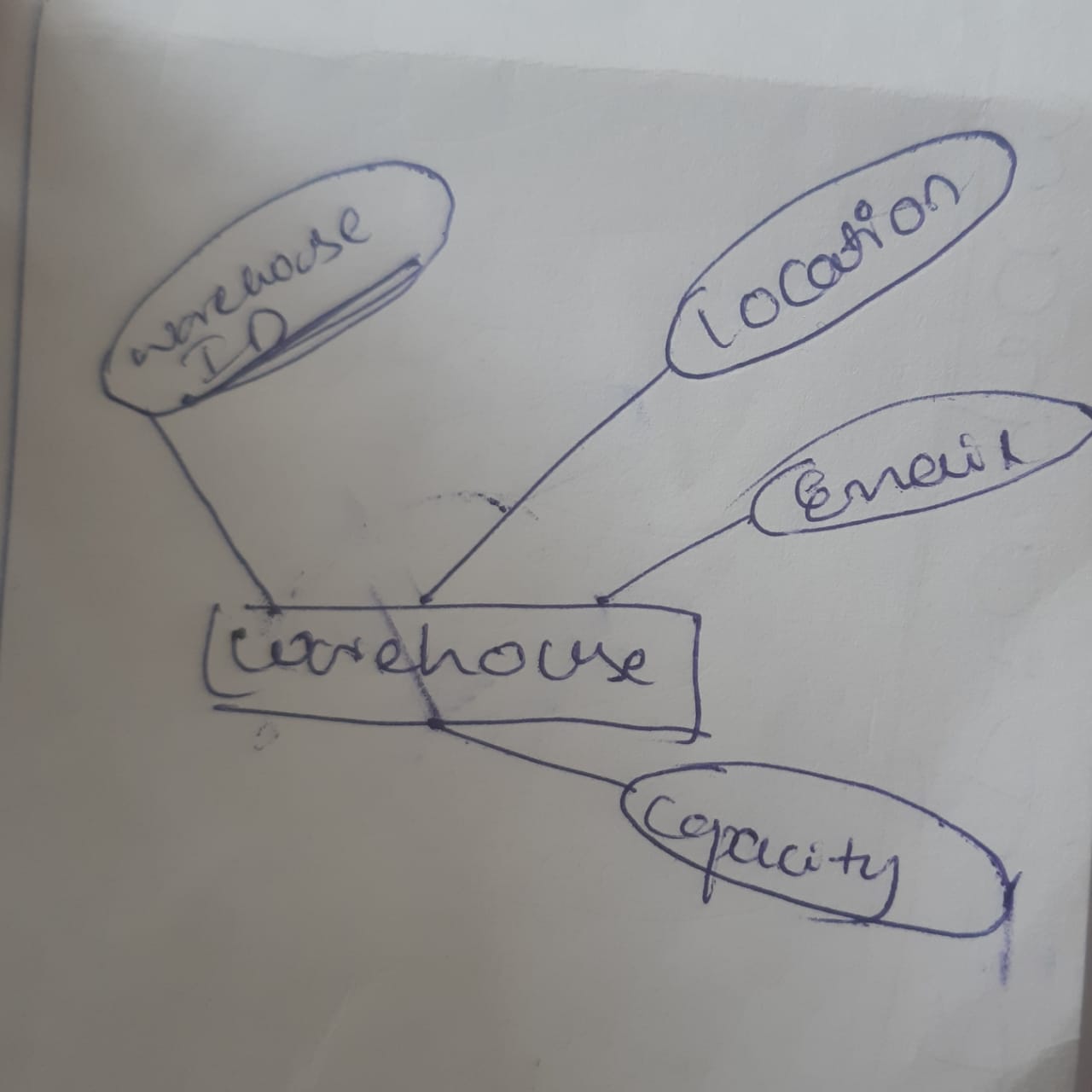
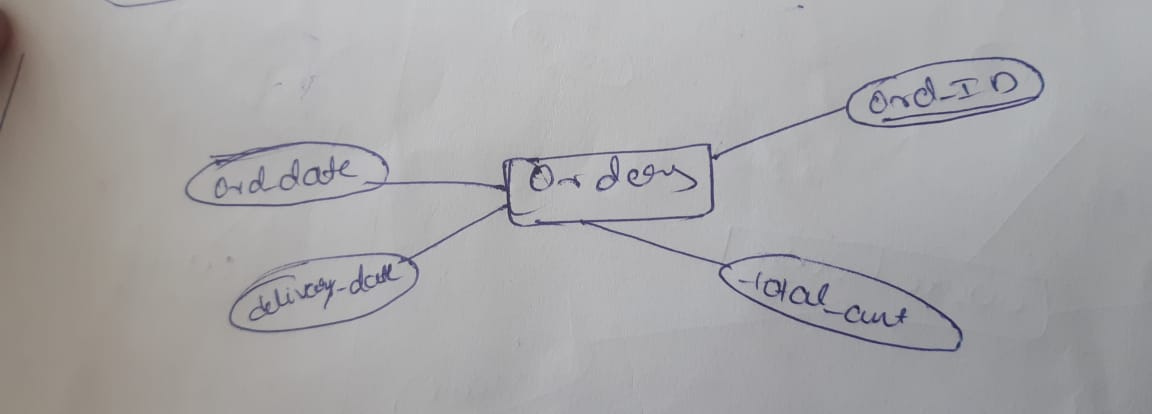
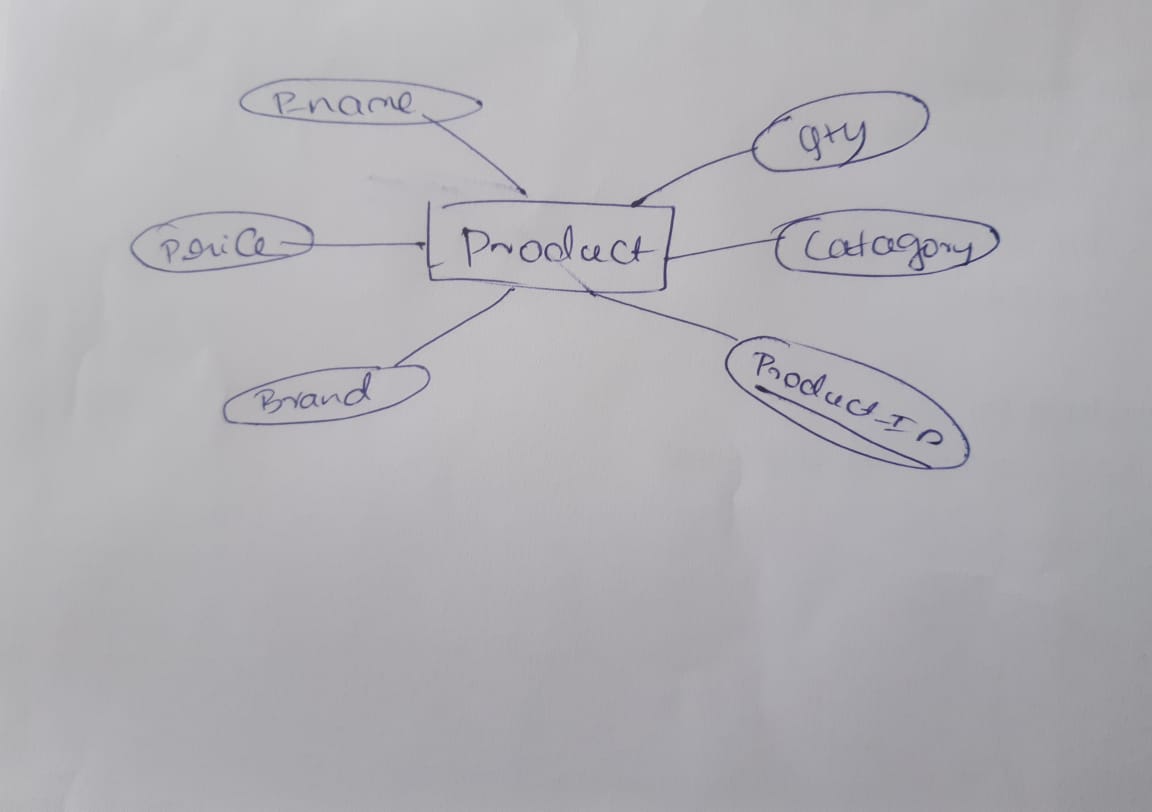
* Every product has a unique product\_id, along with attributes such as product\_name, category, unit\_price and brand
* Suppliers associated with the inventory system have a unique supplier\_id and provide details like supplier\_name, contact\_number, email, and address.
* Warehouses have a unique warehouse\_id, warehouse\_name, and location,address eamail. Each warehouse has a defined capacity and a manager assigned to oversee its operations.
* Customers interacting with the system are given a unique customer\_id and store details like c\_name, address, email,pin ,city, and phone\_number.
* Orders placed by customers are tracked in the database with a unique order\_id. The table records customer\_id, order\_date, total\_amount, and payment\_mode.
* Each order consists of one or more products, recorded in the Order\_Details table with attributes like order\_id, product\_id, quantity, and price.

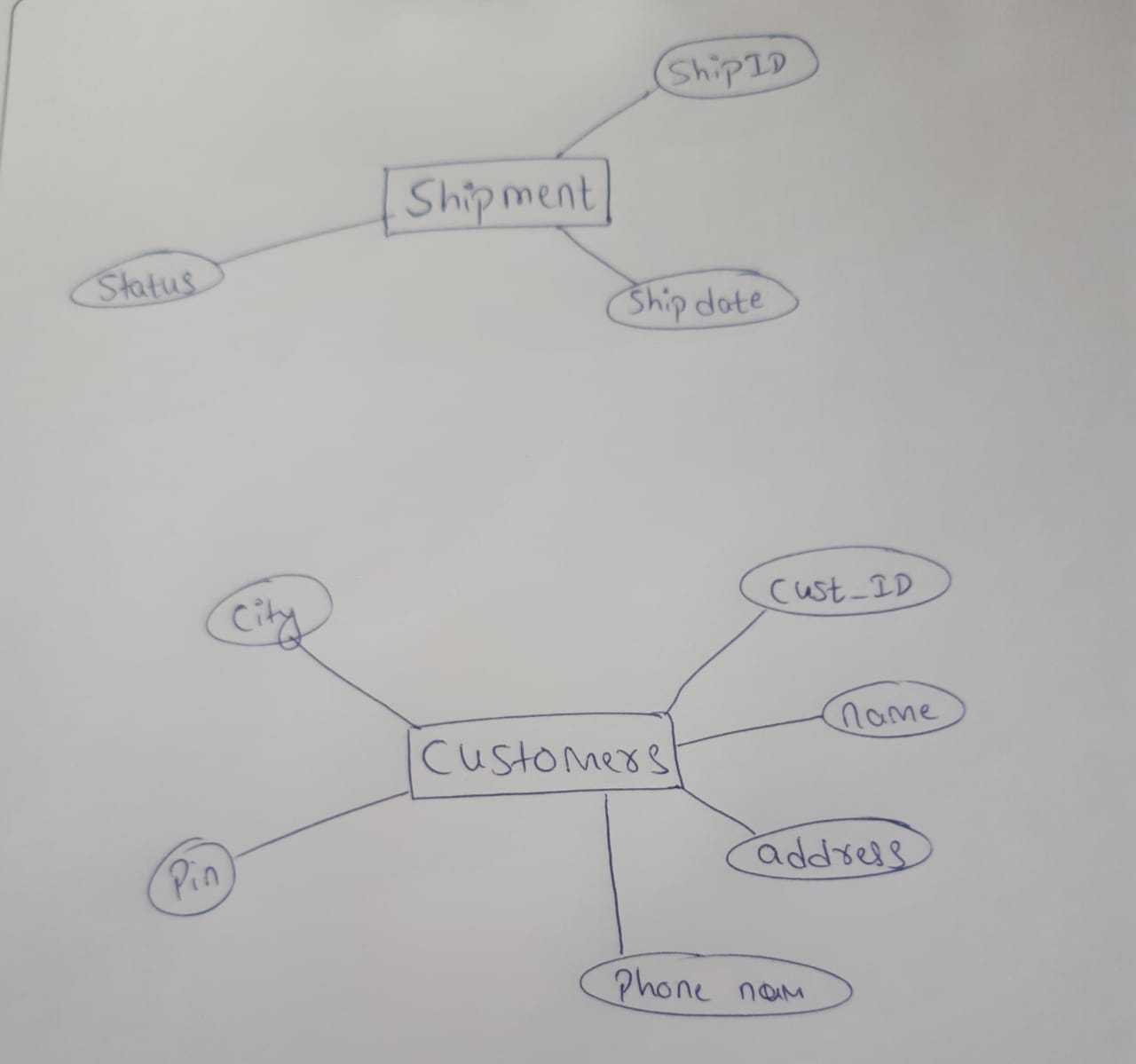
ENTITIES

**Total Six Entities**

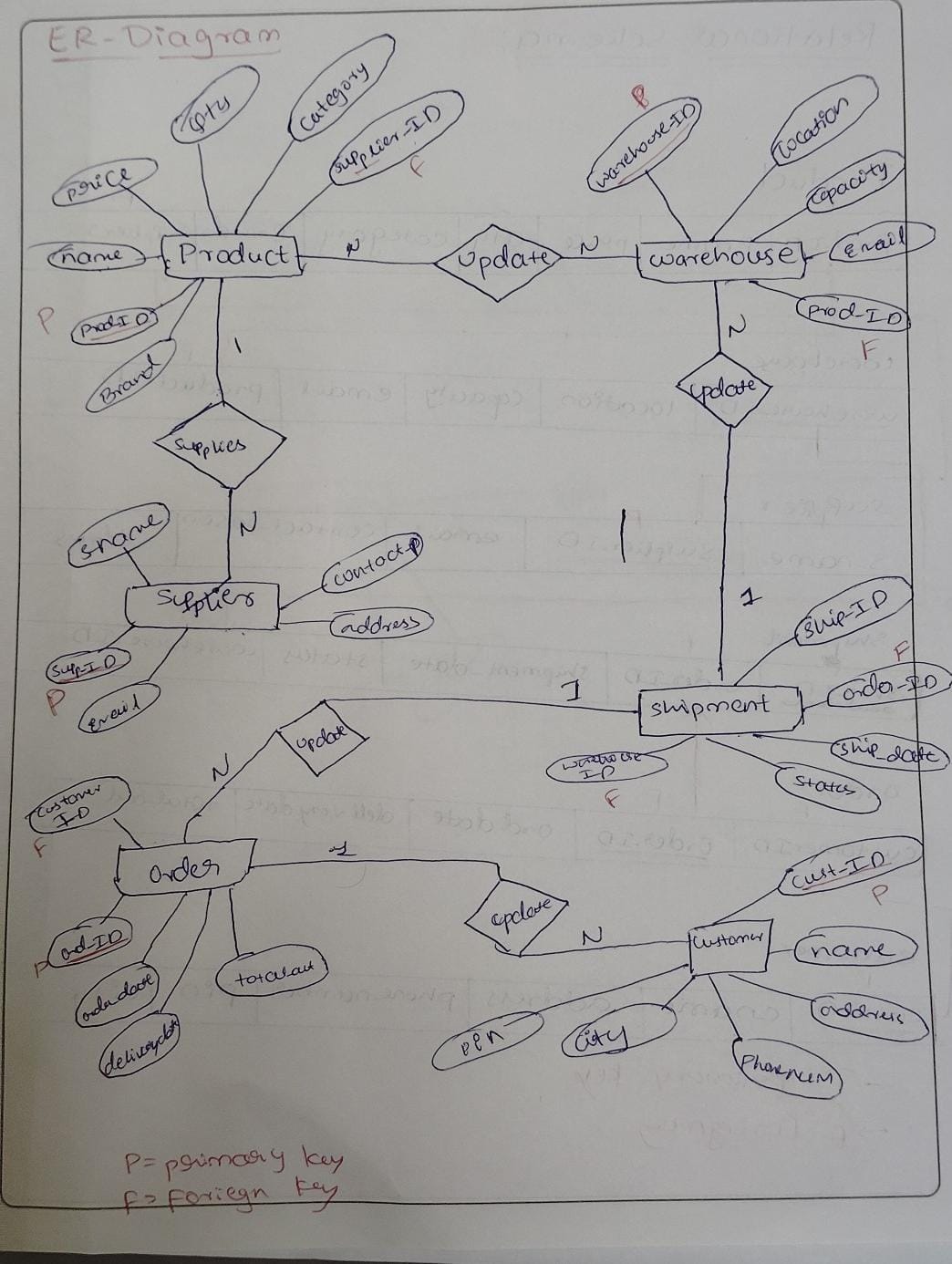
1. **products**
2. **warehouses**
3. **Supplier**
4. **Shipment**
5. **order**
6. **customer**

****

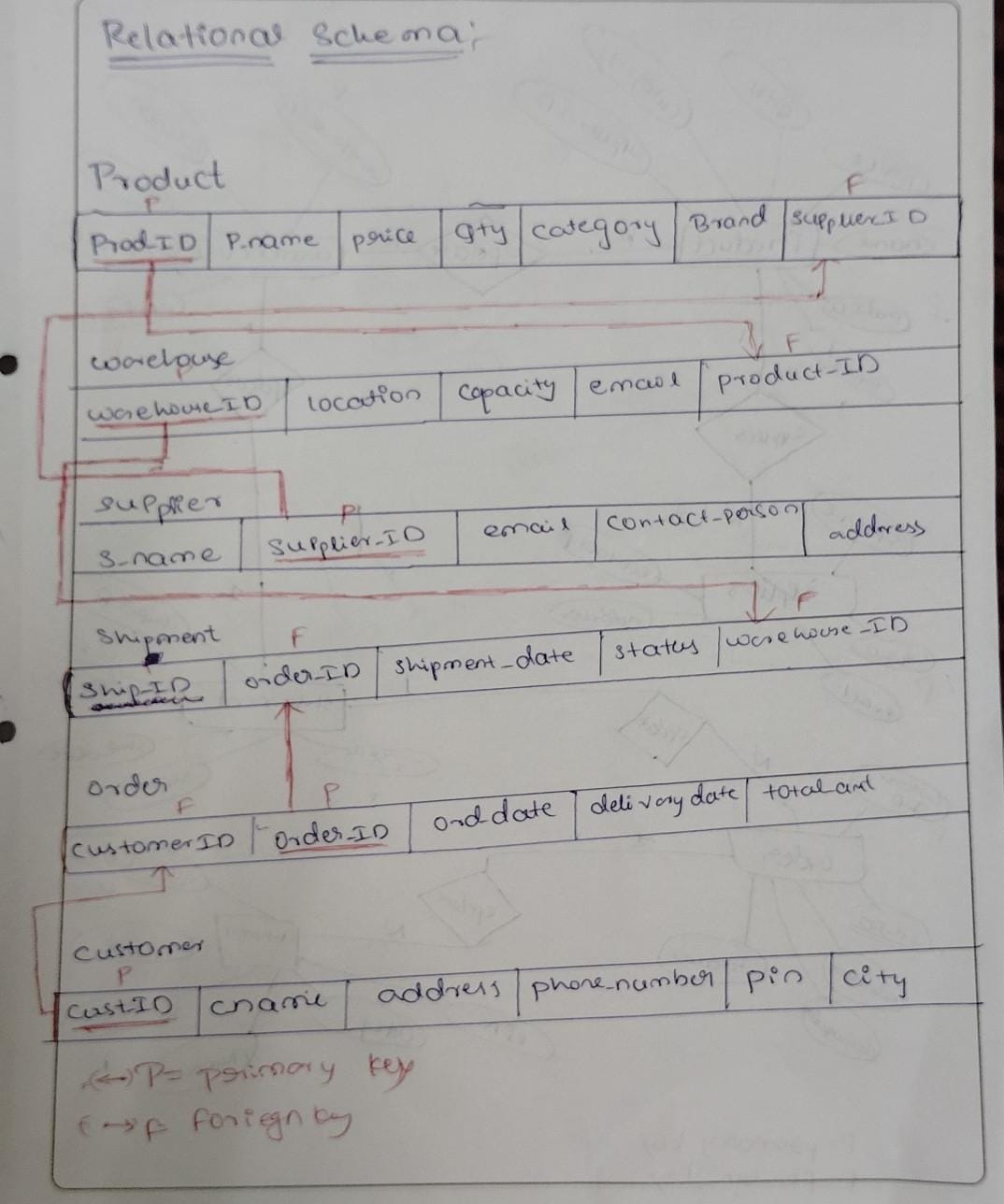
****

****

**ER-Diagram**

****

**Relational Schema**

****

**NORMALIZATION**

*Step 1:* inventory management (Unnormalized)

**products**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prod\_id** | **P\_name** | **price** | **qty** | **category** | **Sup\_id** | **brand** |

**warehouse**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| W\_id | location | capacity | email | Prod\_id |

**supplier**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sup\_id | S\_name | email | Contact\_p | address |

**shipment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ship\_id | Ordre\_id | Ship\_date | status | W\_d |

**order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C\_id | ord\_id | Ord\_date | Deli\_date | Ttal\_amt |

**customer**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C\_id | C\_name | address | Phone\_num | pin | city |

*Step 2:* First Normal Form (1NF)

* **Requirement:** Each table must have atomic (indivisible) values, and there should be no repeating groups.

#### ****Before Normalization (Issues):****

* If any attribute contains a list of values (e.g., multiple products in one field or multiple suppliers for a product), it violates 1NF.

#### ****Resolution for 1NF:****

* All attributes in the provided schema are already atomic (e.g., P\_name, price, etc.).
* There are no repeating groups within the schema.

### *Step 3:* Second Normal Form (2NF)

* **Requirement:** The table must be in 1NF, and all non-prime attributes must be fully functionally dependent on the primary key.

**Potential Violations:**

1. **WarehouseProduct Table:**
   * The composite key (WarehouseID, ProductID) ensures that there is no partial dependency.
   * This table is already in **2NF**.
2. **Product Table:**
   * Attributes like Brand and SupplierID may not fully depend on ProdID. For example, multiple suppliers can provide the same product with different brands, creating partial dependencies.

**Resolution for 2NF:**

* Decompose Product into:
  1. **Product Table**: (ProdID, P\_name, price, qty, category)
  2. **ProductSupplier Table**: (ProdID, SupplierID, Brand)

*Step 4:* Third Normal Form (3NF)

* **Requirement:** The table must be in 2NF, and there should be no transitive dependencies (a non-prime attribute depending on another non-prime attribute).

**Potential Violations:**

1. **Customer Table:**
   * Attributes like PIN and city are transitively dependent on the address field. For example, the PIN code determines the city.
2. **Shipment Table:**
   * WarehouseID could be transitively dependent on OrderID via other relationships in the database.

**Resolution for 3NF:**

* Decompose further:
  + **Customer Table**: (CustID, Cname, phone\_number)
  + **CustomerAddress Table**: (CustID, address, PIN, city)
* Validate the Shipment table, ensuring WarehouseID is directly associated with ShipID.

### *Step 5:* Boyce-Codd Normal Form (BCNF)

* **Requirement:** Every determinant in the table must be a candidate key (i.e., no dependency should exist where a non-candidate key determines another attribute).

**Potential Violations:**

1. **Shipment Table:**
   * If OrderID determines WarehouseID, it may violate BCNF.

**Resolution for BCNF:**

* Ensure WarehouseID is fully dependent only on ShipID or decompose further:
  + **Shipment Table**: (ShipID, OrderID, shipment\_date, status)
  + Any dependency issues are resolved by restructuring relationships

**Inventory management System(after BCNF)**

**products**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prod\_id** | **P\_name** | **price** | **qty** | **catagory** |

**ProductSupplier**

|  |  |  |
| --- | --- | --- |
| **Prod\_id** | **Sup\_id** | **brand** |

**warehouse**

|  |  |  |  |
| --- | --- | --- | --- |
| W\_id | email | capacity | location |

**Warehouse product**

|  |  |
| --- | --- |
| W\_id | Prod\_id |

**supplier**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S\_id | S\_name | C\_p | email | adress |

**shipments**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ship\_id | Ord\_id | Ship\_date | W\_id | status |

**Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ord\_id | C\_id | Ord\_date | Deli\_date | Total\_amt |

**customer**

|  |  |  |
| --- | --- | --- |
| C\_id | C\_name | Phone\_num |

**Customeraddress**

|  |  |  |  |
| --- | --- | --- | --- |
| address | c\_id | pin | city |

**SQL Code:**

DROP TABLE PRODUCTS;

DROP TABLE WAREHOUSE;

DROP TABLE SUPPLIERS;

DROP TABLE SHIPMENTS;

DROP TABLE IORDER;

DROP TABLE CUSTOMER;

CREATE TABLE SUPPLIERS

(

S\_NAME VARCHAR(20),

SUPP\_ID INT PRIMARY KEY,

EMAIL VARCHAR(20),

PROD\_ID INT

);

DESC SUPPLIERS;

SELECT\*FROM SUPPLIERS;

INSERT INTO SUPPLIERS VALUES('Supplier A', 1, 'A@example.com', 101);

INSERT INTO SUPPLIERS VALUES('Supplier B', 2, 'B@example.com', 102);

INSERT INTO SUPPLIERS VALUES('Supplier C', 3, 'C@example.com', 103);

INSERT INTO SUPPLIERS VALUES('Supplier D', 4, 'D@example.com', 104);

INSERT INTO SUPPLIERS VALUES('Supplier E', 5, 'E@example.com', 105);

INSERT INTO SUPPLIERS VALUES('Supplier F', 6, 'F@example.com', 106);

INSERT INTO SUPPLIERS VALUES('Supplier G', 7, 'G@example.com', 107);

CREATE TABLE PRODUCTS

(

PROD\_ID INT PRIMARY KEY,

P\_NAME VARCHAR(20),

PRICE INT,

QTY INT,

CATAGORY VARCHAR(20),

BRAND VARCHAR(20),

SUPP\_ID INT,

foreign key(SUPP\_ID)references SUPPLIERS(SUPP\_ID)

);

DESC PRODUCTS;

SELECT\* FROM PRODUCTS;

INSERT INTO PRODUCTS VALUES(101, 'Product A', 100, 50, 'Electronics', 'Brand X', 1);

INSERT INTO PRODUCTS VALUES(102, 'Product B', 200, 30, 'Clothing', 'Brand Y', 2);

INSERT INTO PRODUCTS VALUES(103, 'Product C', 150, 20, 'Kitchenware', 'Brand Z', 3);

INSERT INTO PRODUCTS VALUES(104, 'Product D', 300, 10, 'Furniture', 'Brand X', 4);

INSERT INTO PRODUCTS VALUES(105, 'Product E', 250, 40, 'Electronics', 'Brand Y', 5);

INSERT INTO PRODUCTS VALUES(106, 'Product F', 350, 60, 'Clothing', 'Brand Z', 6);

INSERT INTO PRODUCTS VALUES(107, 'Product G', 400, 80, 'Kitchenware', 'Brand X', 7);

CREATE TABLE WAREHOUSE

(

WAREHOUSE\_ID INT PRIMARY KEY,

LOCATION VARCHAR(20),

CAPACITY VARCHAR(20),

EMAIL VARCHAR(20),

PROD\_ID INT,

FOREIGN KEY(PROD\_ID) REFERENCES PRODUCTS(PROD\_ID)

);

DESC WAREHOUSE;

SELECT \* FROM WAREHOUSE;

INSERT INTO WAREHOUSE VALUES(1, 'New York', '5000', 'ny@example.com', 101);

INSERT INTO WAREHOUSE VALUES(2, 'Los Angeles', '4000', 'lae@example.com', 102);

INSERT INTO WAREHOUSE VALUES(3, 'Chicago', '6000', 'chic@example.com', 103);

INSERT INTO WAREHOUSE VALUES(4, 'Houston', '7000', 'hous@example.com', 104);

INSERT INTO WAREHOUSE VALUES(5, 'Phoenix', '4500', 'phoe@example.com', 105);

INSERT INTO WAREHOUSE VALUES(6, 'Philadelphia', '3500', 'phil@example.com', 106);

INSERT INTO WAREHOUSE VALUES(7, 'San Francisco', '5500', 'sf@example.com', 107);

CREATE TABLE IORDER

(

ORD\_ID INT PRIMARY KEY,

ORD\_DATE DATE,

DELIVARY\_DATE DATE,

TOTAL\_AMT INT,

CUST\_ID INT,

FOREIGN KEY(CUST\_ID) REFERENCES CUSTOMER(CUST\_ID)

);

DESC IORDER;

SELECT \* FROM IORDER;

INSERT INTO IORDER VALUES(1001, TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2024-01-10', 'YYYY-MM-DD'), 500, 1);

INSERT INTO IORDER VALUES(1002, TO\_DATE('2024-01-02', 'YYYY-MM-DD'), TO\_DATE('2024-01-11', 'YYYY-MM-DD'), 800, 2);

INSERT INTO IORDER VALUES(1003, TO\_DATE('2024-01-03', 'YYYY-MM-DD'), TO\_DATE('2024-01-12', 'YYYY-MM-DD'), 1200, 3);

INSERT INTO IORDER VALUES(1004, TO\_DATE('2024-01-04', 'YYYY-MM-DD'), TO\_DATE('2024-01-13', 'YYYY-MM-DD'), 400, 4);

INSERT INTO IORDER VALUES(1005, TO\_DATE('2024-01-05', 'YYYY-MM-DD'), TO\_DATE('2024-01-14', 'YYYY-MM-DD'), 1500, 5);

INSERT INTO IORDER VALUES(1006, TO\_DATE('2024-01-06', 'YYYY-MM-DD'), TO\_DATE('2024-01-15', 'YYYY-MM-DD'), 950, 6);

INSERT INTO IORDER VALUES(1007, TO\_DATE('2024-01-07', 'YYYY-MM-DD'), TO\_DATE('2024-01-16', 'YYYY-MM-DD'), 1100, 7);

CREATE TABLE SHIPMENTS

(

SHIP\_ID INT,

SHIP\_DATE DATE,

STATUS VARCHAR(20),

ORD\_ID INT,

WAREHOUSE\_ID INT,

FOREIGN KEY(ORD\_ID) REFERENCES IORDER(ORD\_ID),

FOREIGN KEY(WAREHOUSE\_ID) REFERENCES WAREHOUSE(WAREHOUSE\_ID)

);

DESC SHIPMENTS;

SELECT \* FROM SHIPMENTS;

INSERT INTO SHIPMENTS VALUES(1, TO\_DATE('2024-01-02', 'YYYY-MM-DD'), 'Shipped', 1001, 1);

INSERT INTO SHIPMENTS VALUES(2, TO\_DATE('2024-01-03', 'YYYY-MM-DD'), 'Shipped', 1002, 2);

INSERT INTO SHIPMENTS VALUES(3, TO\_DATE('2024-01-04', 'YYYY-MM-DD'), 'In Transit', 1003, 3);

INSERT INTO SHIPMENTS VALUES(4, TO\_DATE('2024-01-05', 'YYYY-MM-DD'), 'Delivered', 1004, 4);

INSERT INTO SHIPMENTS VALUES(5, TO\_DATE('2024-01-06', 'YYYY-MM-DD'), 'Shipped', 1005, 5);

INSERT INTO SHIPMENTS VALUES(6, TO\_DATE('2024-01-07', 'YYYY-MM-DD'), 'Shipped', 1006, 6);

INSERT INTO SHIPMENTS VALUES(7, TO\_DATE('2024-01-08', 'YYYY-MM-DD'), 'In Transit', 1007, 7);

CREATE TABLE CUSTOMER

(

CUST\_ID INT PRIMARY KEY,

C\_NAME VARCHAR(20),

ADDRESS VARCHAR(20),

PHONE\_NO INT,

PIN INT,

CITY VARCHAR(20)

);

DESC CUSTOMER;

SELECT \* FROM CUSTOMER;

INSERT INTO CUSTOMER VALUES(1, 'John Doe', '123 Main St', 1234567890, 10001, 'New York');

INSERT INTO CUSTOMER VALUES(2, 'Jane Smith', '456 Oak St', 2345678901, 20002, 'Los Angeles');

INSERT INTO CUSTOMER VALUES(3, 'Alice Johnson', '789 Pine St', 3456789012, 30003, 'Chicago');

INSERT INTO CUSTOMER VALUES(4, 'Bob Brown', '101 Maple St', 4567890123, 40004, 'Houston');

INSERT INTO CUSTOMER VALUES(5, 'Charlie Davis', '202 Birch St', 5678901234, 50005, 'Phoenix');

INSERT INTO CUSTOMER VALUES(6, 'Dana Miller', '303 Cedar St', 6789012345, 60006, 'Philadelphia');

INSERT INTO CUSTOMER VALUES(7, 'Eve Wilson', '404 Elm St', 7890123456, 70007, 'San Francisco');

**Queries:**

1. Retrieve all products along with their supplier details.
2. Find the total number of products stored in each warehouse.
3. List all orders with their shipment status and delivery date.
4. Get customer details for all orders placed within a specific date range.
5. Find the top 3 most expensive products.

1. Retrieve all products along with their supplier details.

SELECT P.PROD\_ID, P.P\_NAME, P.PRICE, P.QTY, P.CATAGORY, P.BRAND, S.S\_NAME, S.EMAIL

FROM PRODUCTS P, SUPPLIERS S

WHERE P.SUPP\_ID = S.SUPP\_ID;

2. Find the total number of products stored in each warehouse.

SELECT W.WAREHOUSE\_ID, W.LOCATION, COUNT(P.PROD\_ID) AS TOTAL\_PRODUCTS

FROM WAREHOUSE W, PRODUCTS P

WHERE W.PROD\_ID = P.PROD\_ID

GROUP BY W.WAREHOUSE\_ID, W.LOCATION

3. List all orders with their shipment status and delivery date.

SELECT O.ORD\_ID, O.ORD\_DATE, O.DELIVARY\_DATE, S.STATUS

FROM IORDER O, SHIPMENTS S

WHERE O.ORD\_ID = S.ORD\_ID;

4. Get customer details for all orders placed within a specific date range.

SELECT C.CUST\_ID, C.C\_NAME, C.ADDRESS, C.PHONE\_NO, O.ORD\_ID, O.ORD\_DATE

FROM CUSTOMER C, IORDER O

WHERE C.CUST\_ID = O.CUST\_ID

AND O.ORD\_DATE >= TO\_DATE('2024-01-01', 'YYYY-MM-DD')

AND O.ORD\_DATE <= TO\_DATE('2024-01-31', 'YYYY-MM-DD');

5. Find the top 3 most expensive products.

SELECT PROD\_ID, P\_NAME, PRICE

FROM (

SELECT PROD\_ID, P\_NAME, PRICE

FROM PRODUCTS

ORDER BY PRICE DESC

)

WHERE ROWNUM <= 3;