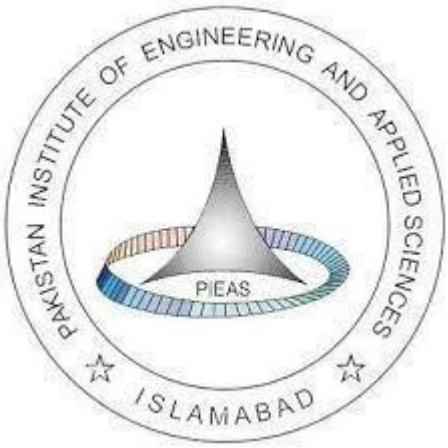


Project: Application for Supply Chain management(SCM)

Course: Internet Application Development(IAD)

Name: Muhammad Musa

Roll number: 0-3-31-036-2022

Instructor: Mr. Irfan Hameed

Pakistan Institute of Engineering & Applied Sciences(PIEAS)

Task 1

Project Proposal

The SCM Portal is a data-driven web application developed using ASP.NET (VB) to manage supply chain operations, providing role-based interfaces for Suppliers, Admins, and Customers. The system enables Suppliers to manage materials, Admins to monitor system data, and Customers to interact with products, ensuring a secure, validated, and efficient platform.

Project Objectives:

- Deliver a web application supporting material management, data oversight, and product ordering.
- Implement functionalities per Supplier.aspx (material actions, bio editing) and Admin.aspx (viewing lists).
- Ensure data-driven operations, state management, input validation, and robust security.

Requirements:

- **Data-Driven Architecture**: The SCM Portal uses a SQL Server database to store materials (materialId, name, quantity, supplierId), users (userId, username, role), products (productId, name, price, stock), and orders (orderId, customerId, productId). ADO.NET handles database operations, supporting Supplier actions (e.g., insert_material for btnAddMaterial_Click) and Admin queries (e.g., get_customers for rblOptions_SelectedIndexChanged).
- **State Management**: ASP.NET Session State tracks user sessions (e.g., Session["SupplierId"] for Supplier.aspx access) and ViewState preserves UI state (e.g., RadioButtonList selections in Admin.aspx), ensuring seamless navigation across actions like listing materials or viewing supplier lists.
- **Validation**: Client-side ASP.NET controls (e.g., RequiredFieldValidator for material names, RangeValidator for quantities) and server-side .vb checks validate inputs, preventing errors in Supplier actions (e.g., adding materials) and login credentials.
- **Data Security**: HTTPS secures data transmission, bcrypt encrypts passwords, and ASP.NET Identity enforces role-based access, restricting Supplier.aspx to Suppliers (for btnAddMaterial, btnEditBio) and Admin.aspx to Admins (for rblOptions).
- **User and Role-Based Functionalities**:
    - **Suppliers**: Via Supplier.aspx, add materials (btnAddMaterial), edit bio (btnEditBio), modify materials (btnModifyMaterial), list materials (btnListMaterials), and remove materials (btnRemoveMaterial), with feedback via lblMessage.
    - **Admins**: Via Admin.aspx, view customer lists (customerId, name), supplier lists (supplierId, name, bio), and product lists (productId, name, price) using RadioButtonList, with feedback via lblMessage.

- **Customers**: Browse products and place orders (assumed CustomerPortal), storing orders in the database.

Technical Approach:

- **Frontend**: ASP.NET Web Forms with HTML/CSS for interfaces like Supplier.aspx and Admin.aspx.
- **Backend**: .vb code for server-side logic, using ADO.NET for database connectivity.
- **Development**: Agile methodology with 2-week sprints, testing via unit and integration tests.
- **Security**: Role-based authorization and encrypted data storage.

Task 2

Software requirements specification

Languages: HTML, CSS, VB.NET, SQL

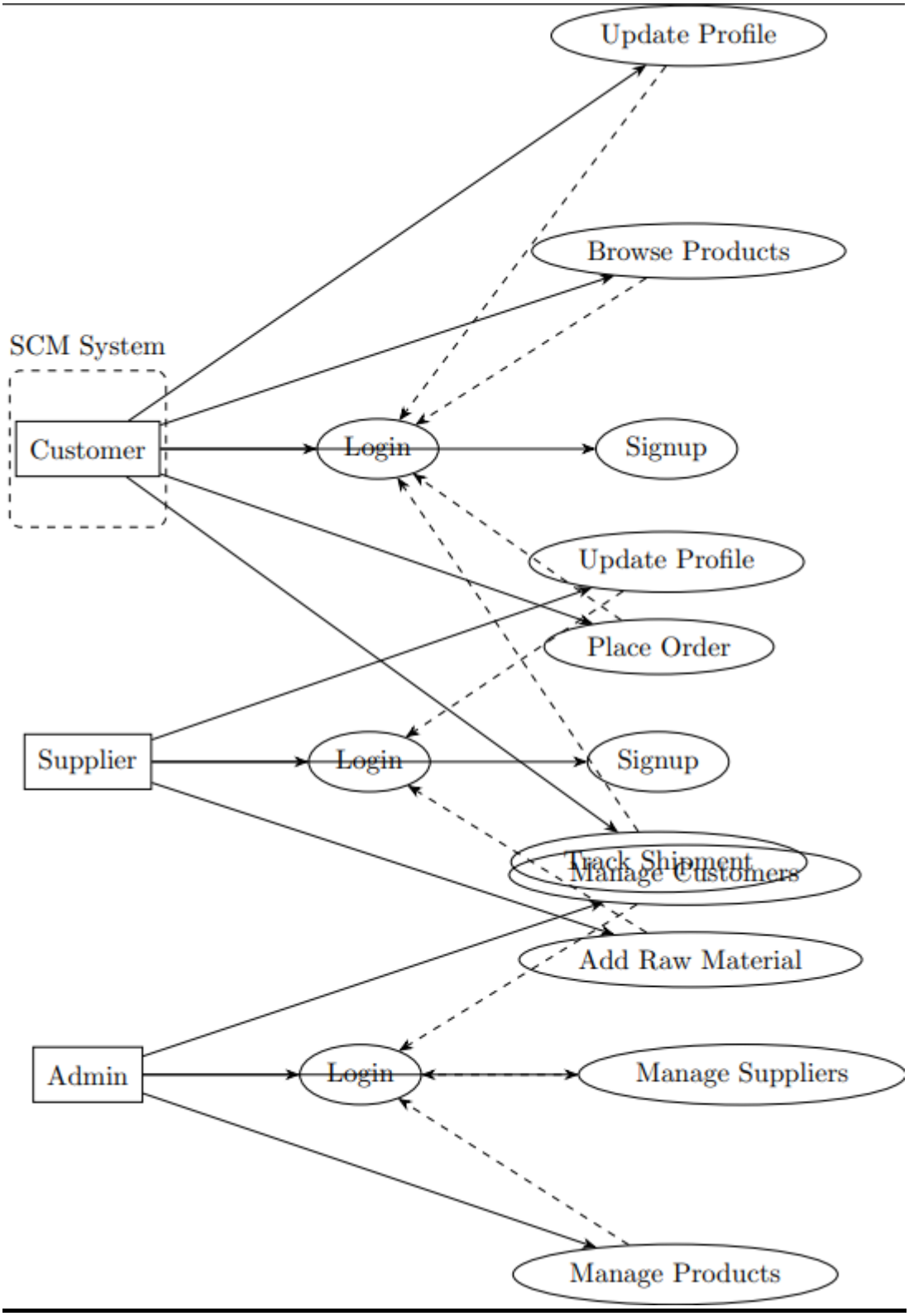
Platforms/Tools: Visual Studio, SQL Server Management Studio, Microsoft SQL Server IDE: Visual Studio (for ASP.NET Web Forms)

Database: Microsoft SQL Server

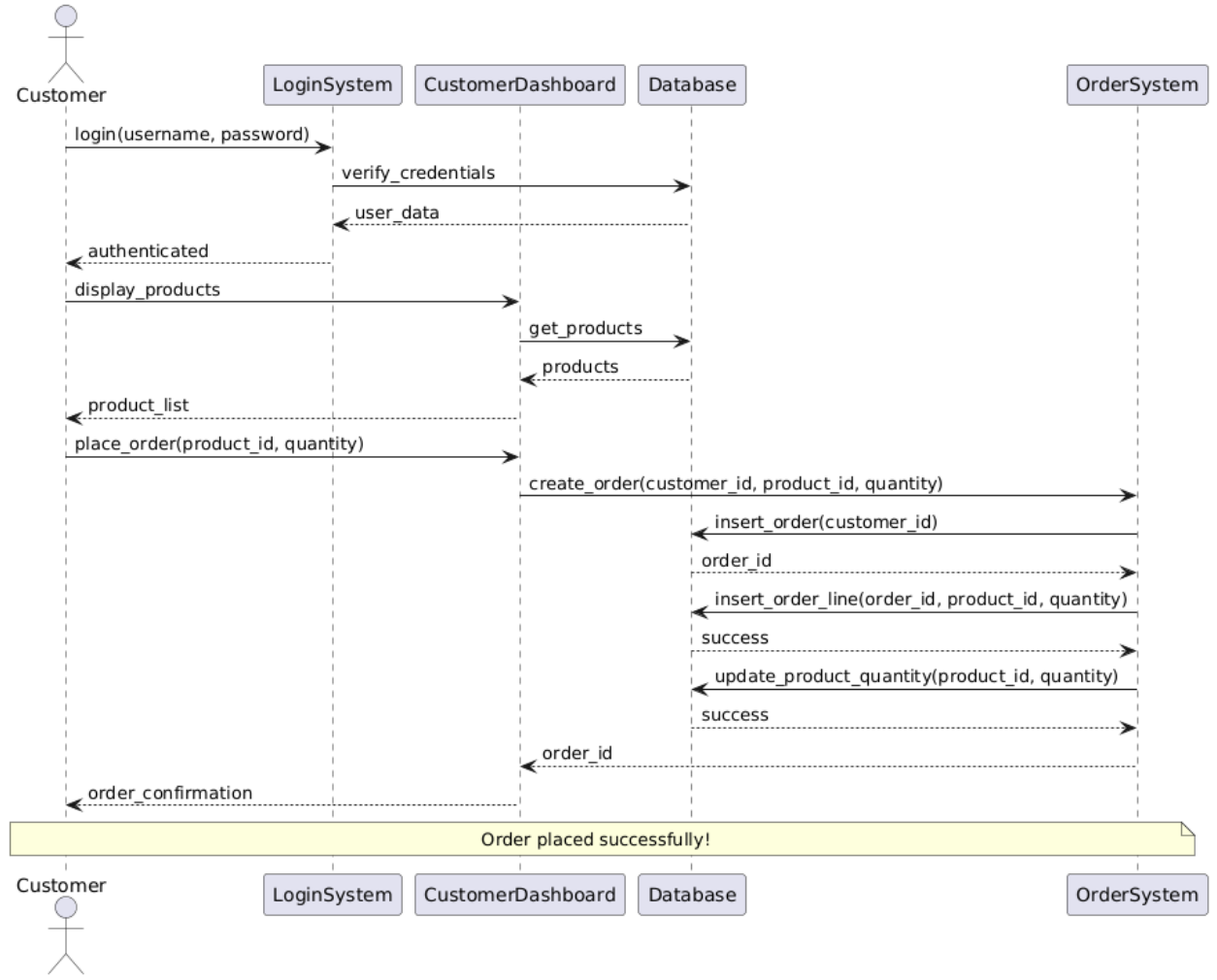
Hosting: Somee.com

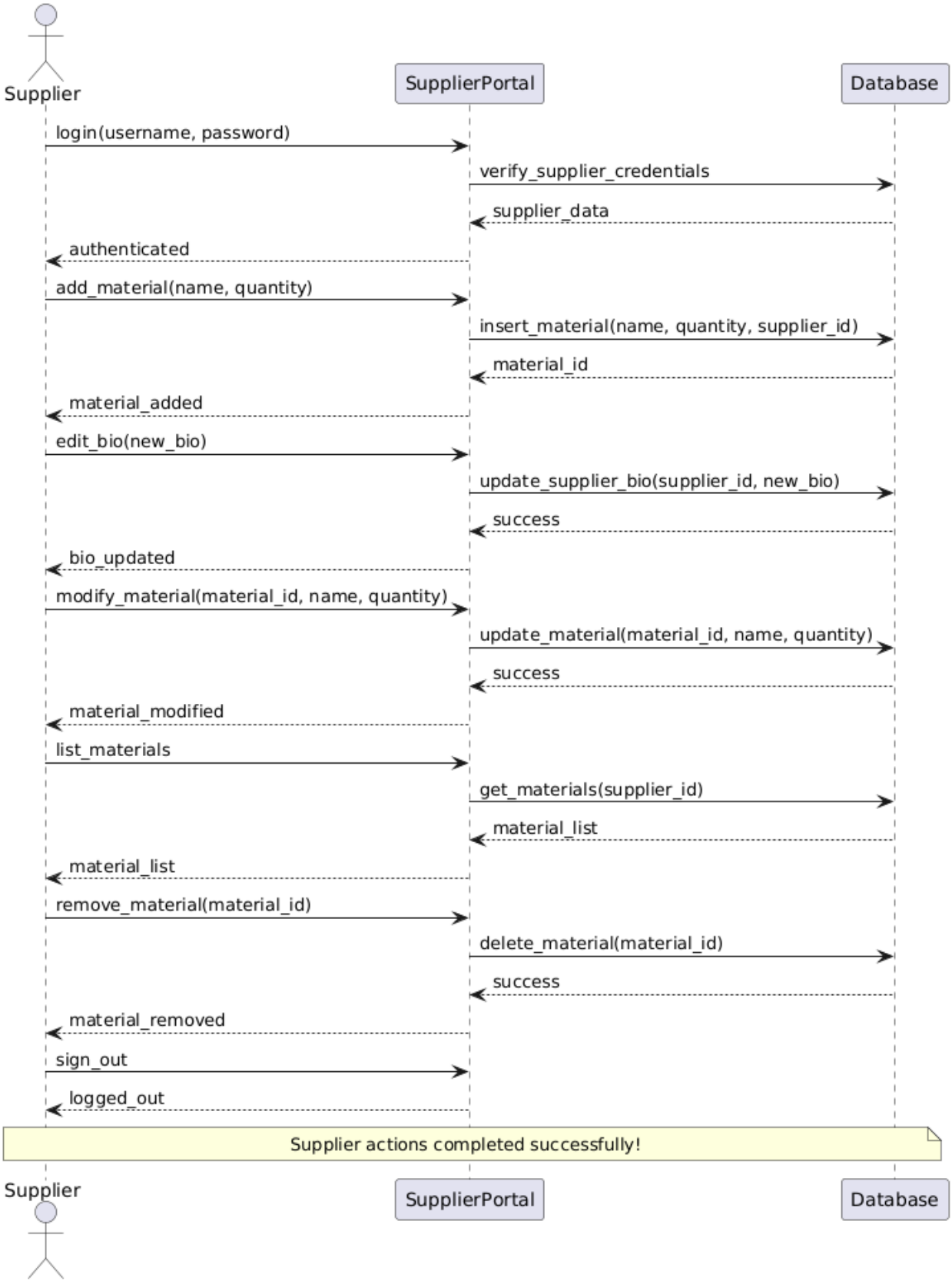
Task 3

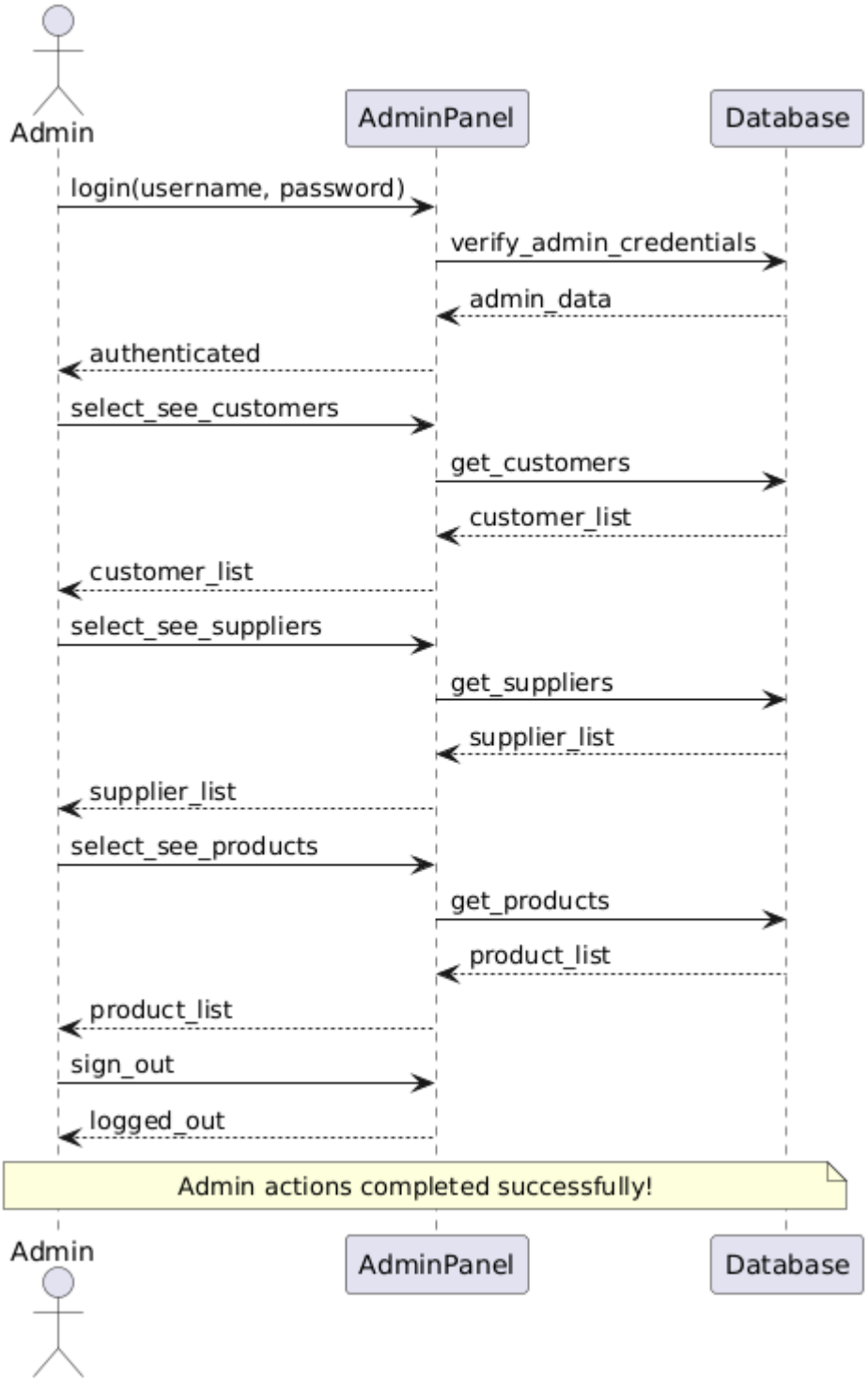
Use Case model:



Sequence diagrams

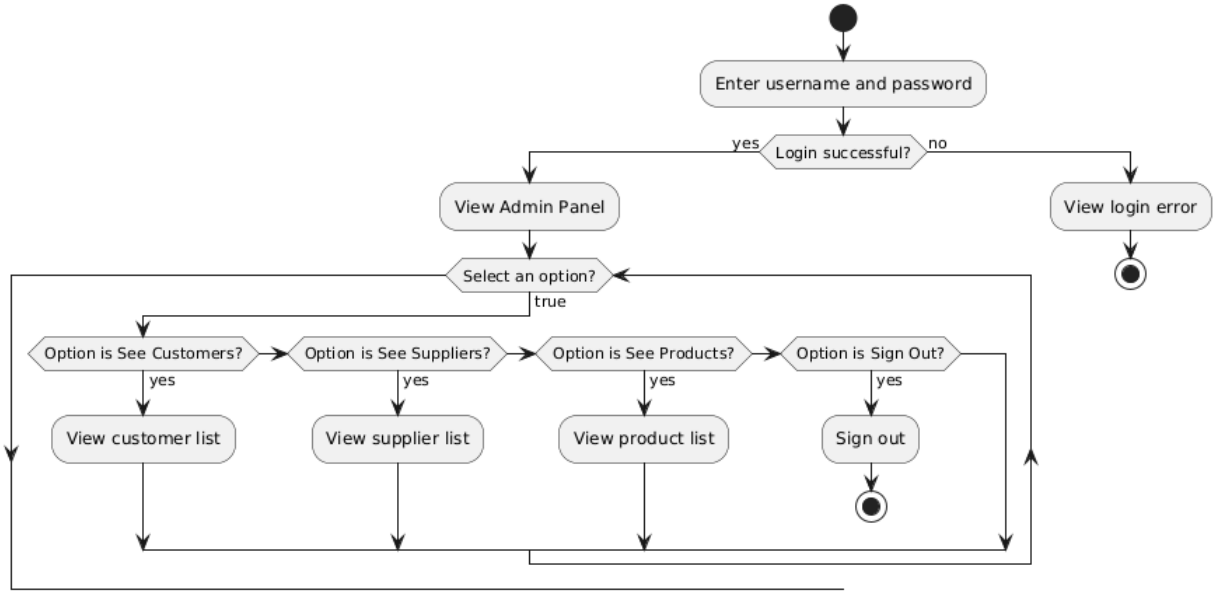



Supplier
SupplierPortal
Database
login(username, password)
verify_supplier_credentials
supplier_data
authenticated
add_material(name, quantity)
insert_material(name, quantity, supplier_id)
material_id
material_added
edit_bio(new_bio)
update_supplier_bio(supplier_id, new_bio)
success
bio_updated
modify_material(material_id, name, quantity)
update_material(material_id, name, quantity)
success
material_modified
list_materials
get_materials(supplier_id)
material_list
material_list
remove_material(material_id)
delete_material(material_id)
success
material_removed
sign_out
logged_out
Supplier actions completed successfully!
Supplier
SupplierPortal
Database

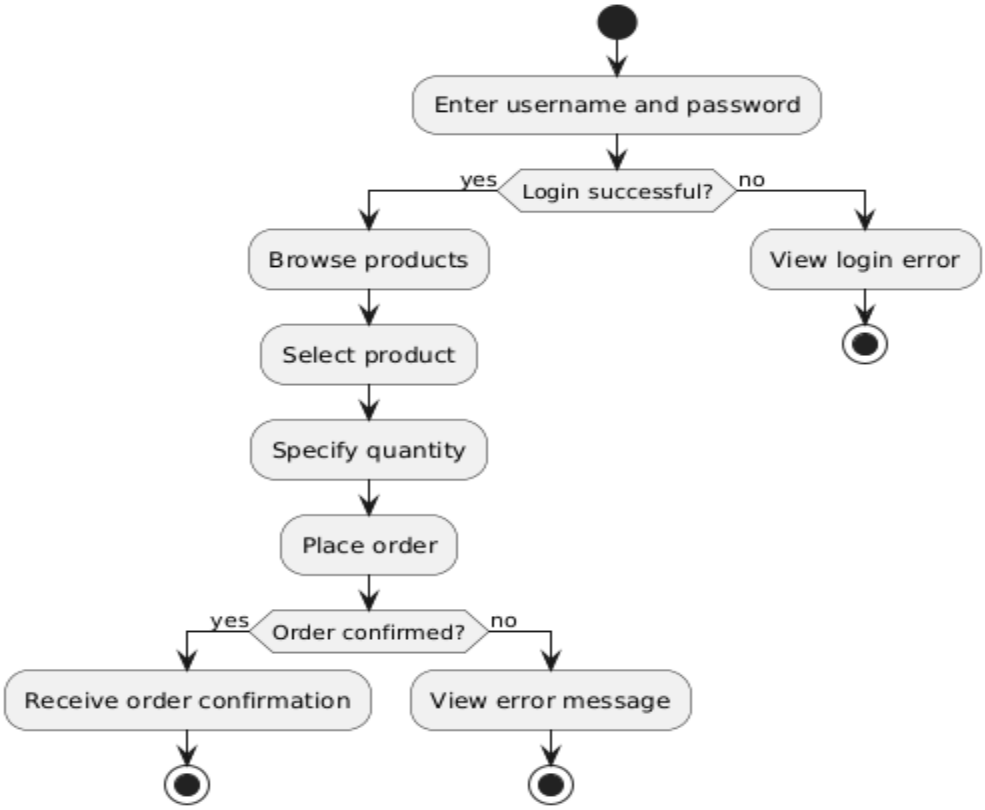

Admin
AdminPanel
Database
login(username, password)
verify_admin_credentials
admin_data
authenticated
select_see_customers
get_customers
customer_list
customer_list
select_see_suppliers
get_suppliers
supplier_list
supplier_list
select_see_products
get_products
product_list
product_list
sign_out
logged_out
Admin actions completed successfully!
Admin
AdminPanel
Database

Activity diagrams

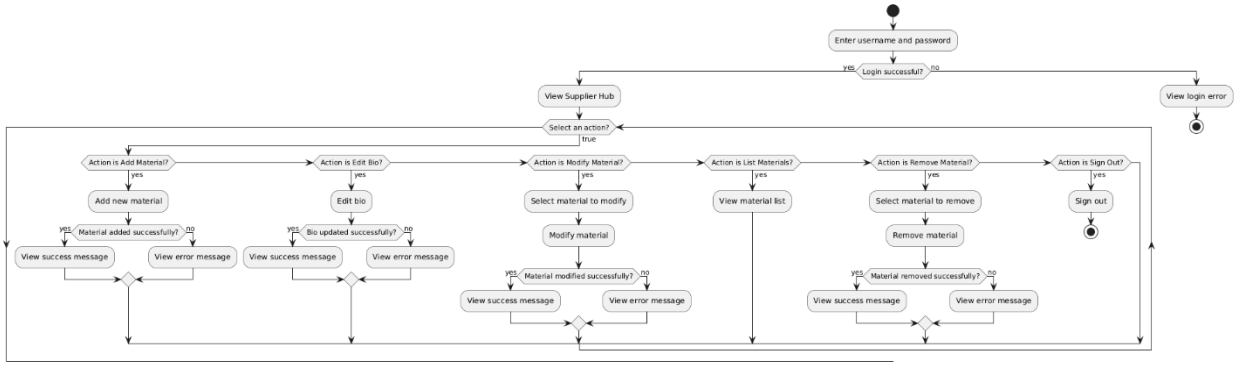
Admin



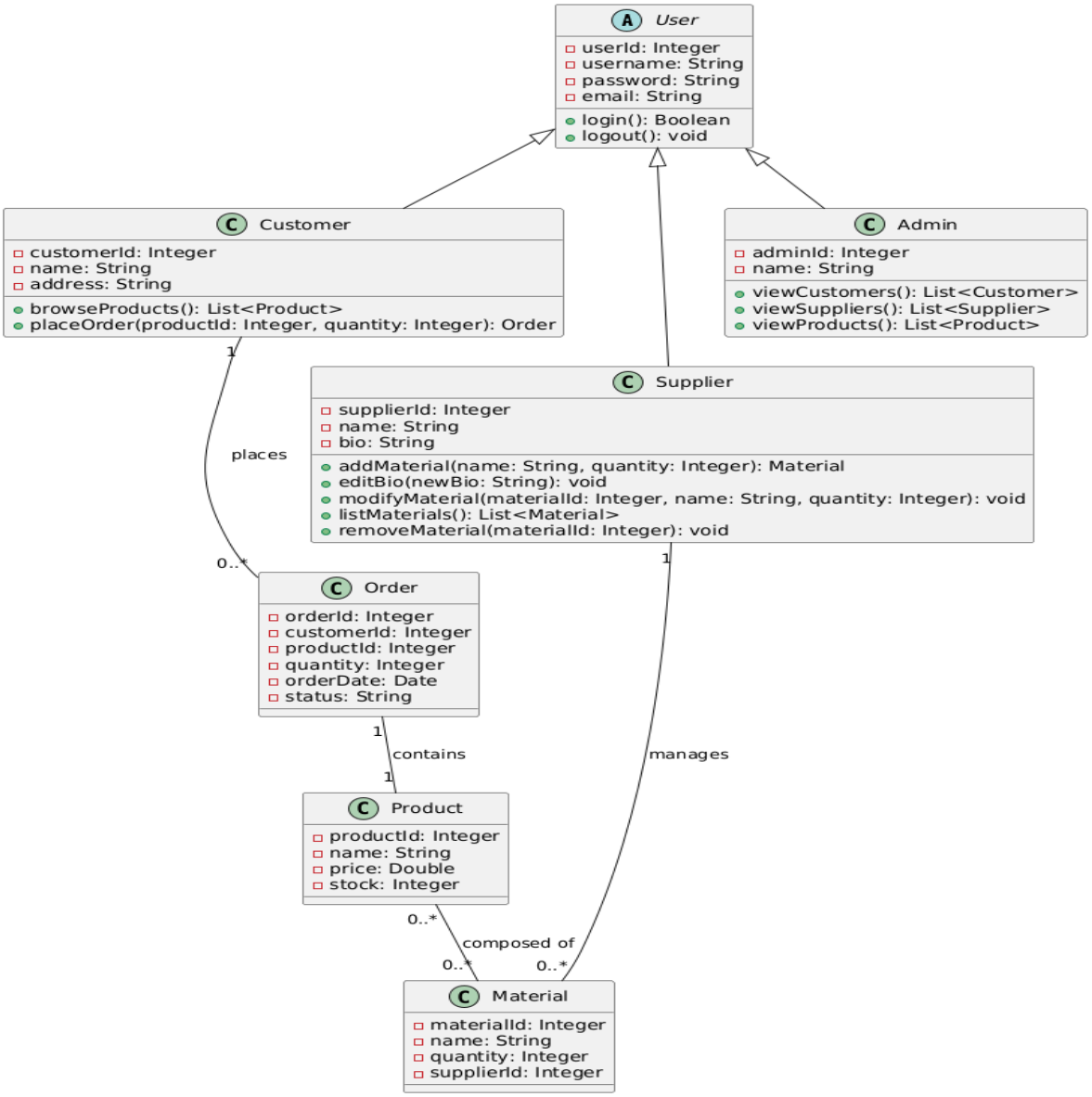
Customer



Supplier:

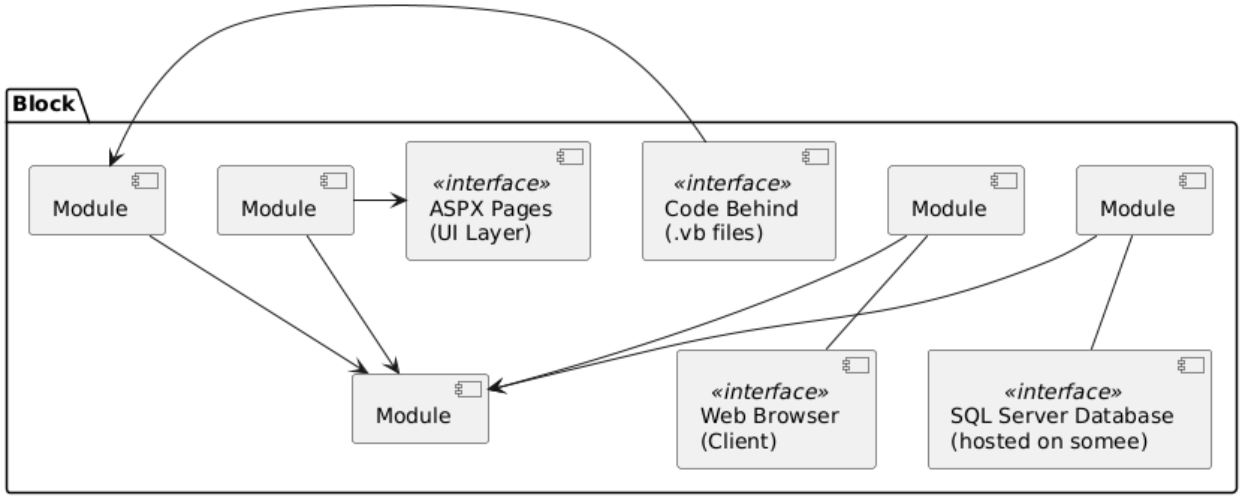


Class diagram

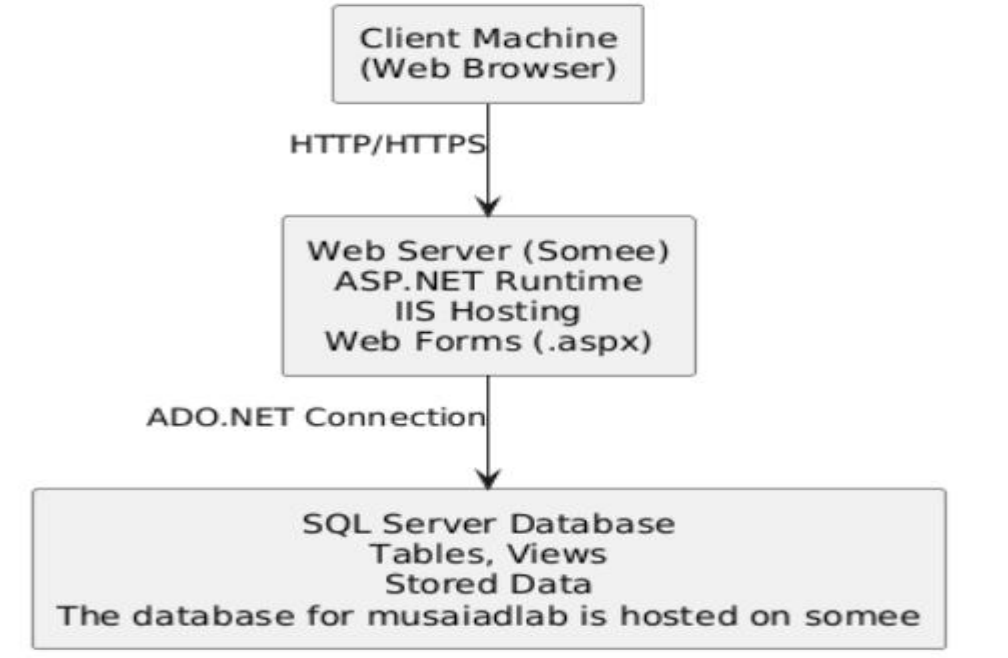


Task 3a

Component Model



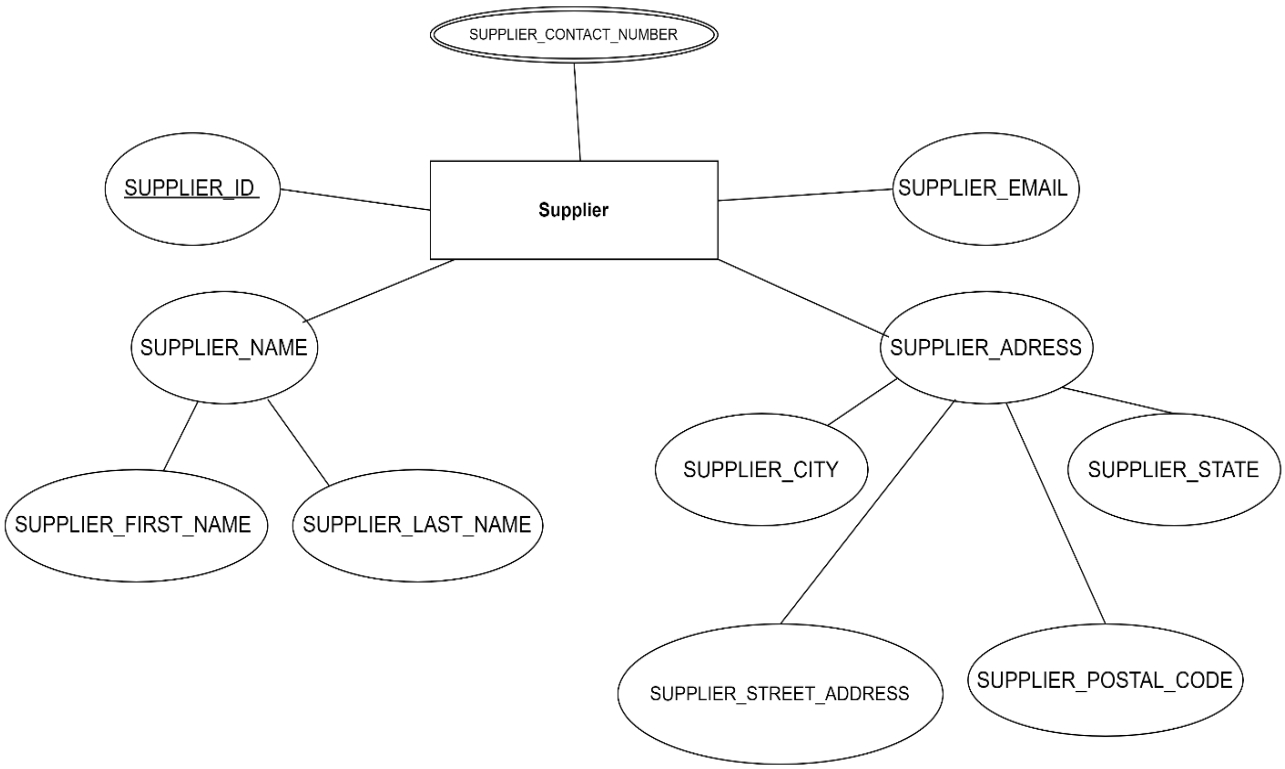
Deployment Model



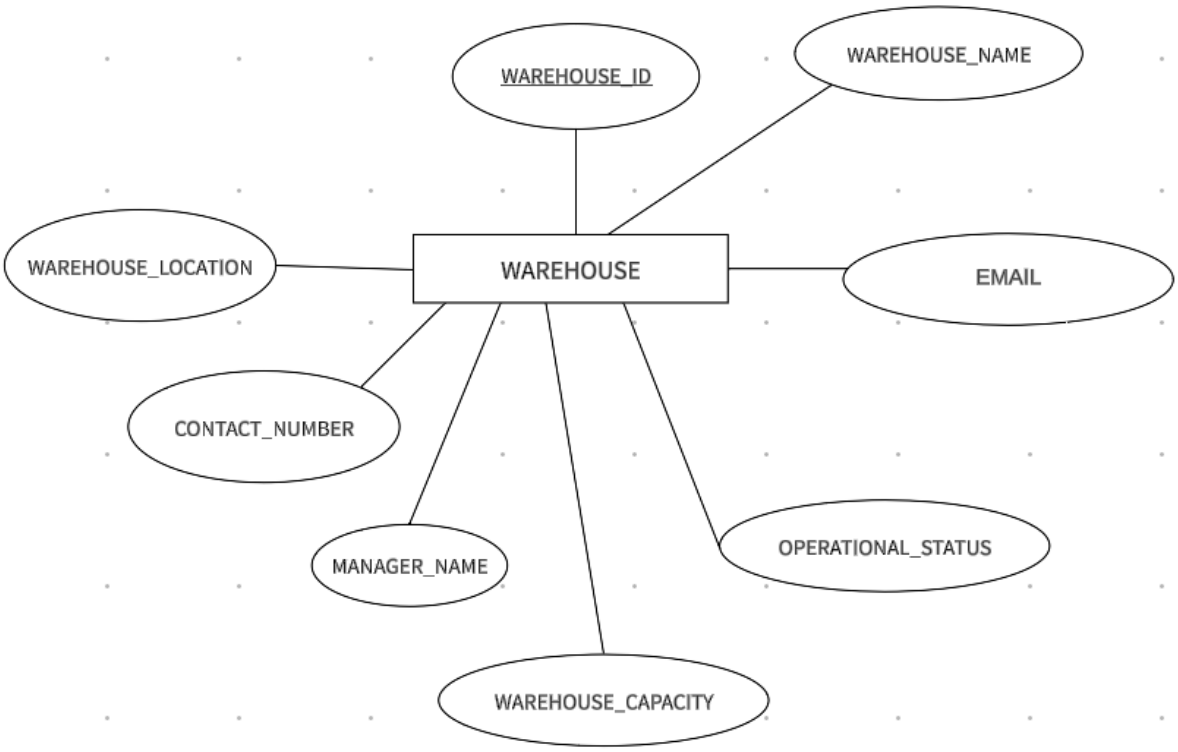
Task 4

ER DIAGRAM

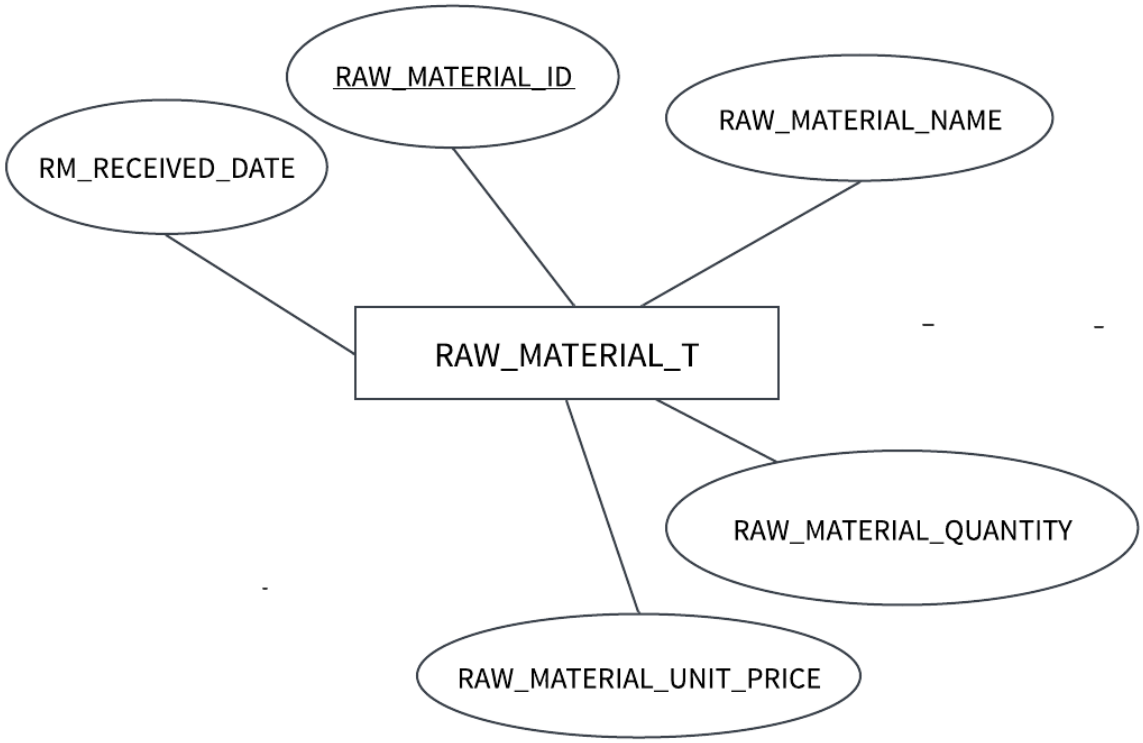
SUPPLIER_T:



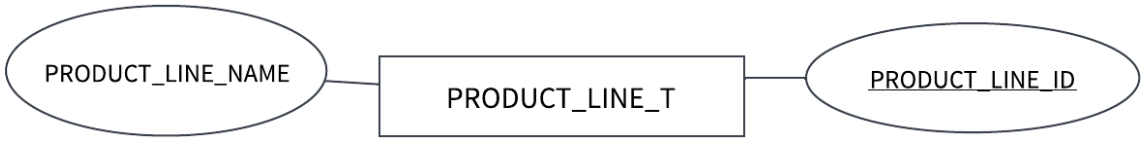
WAREHOUSE_T:



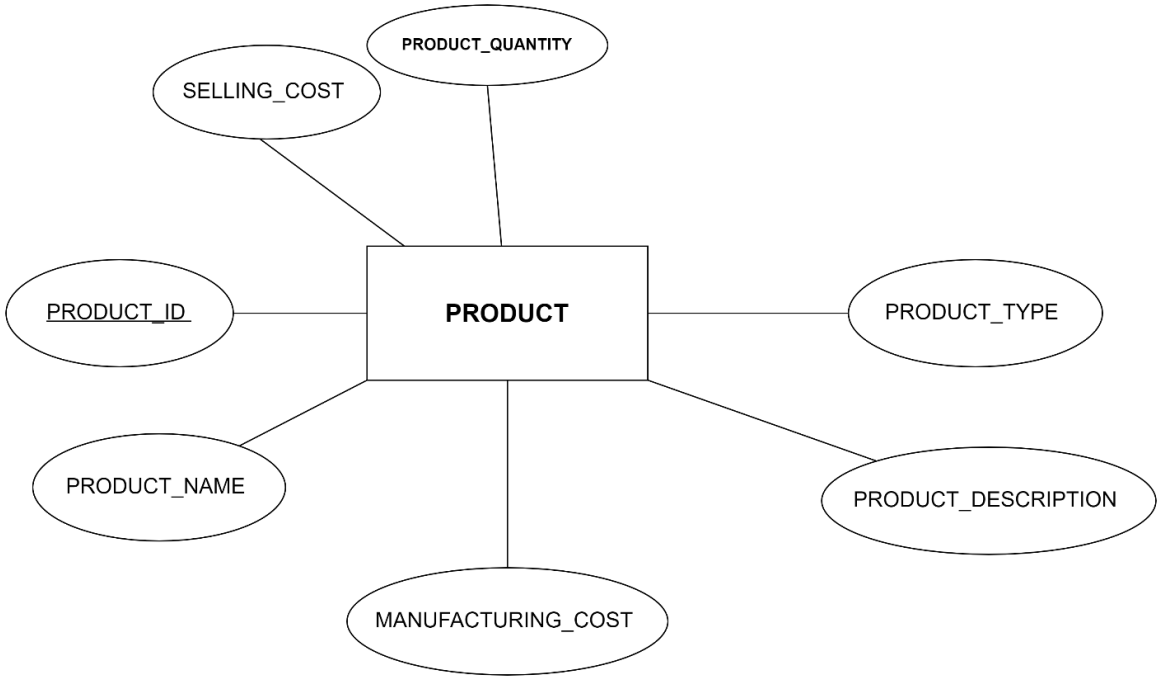
RAW_MATERIAL_T:



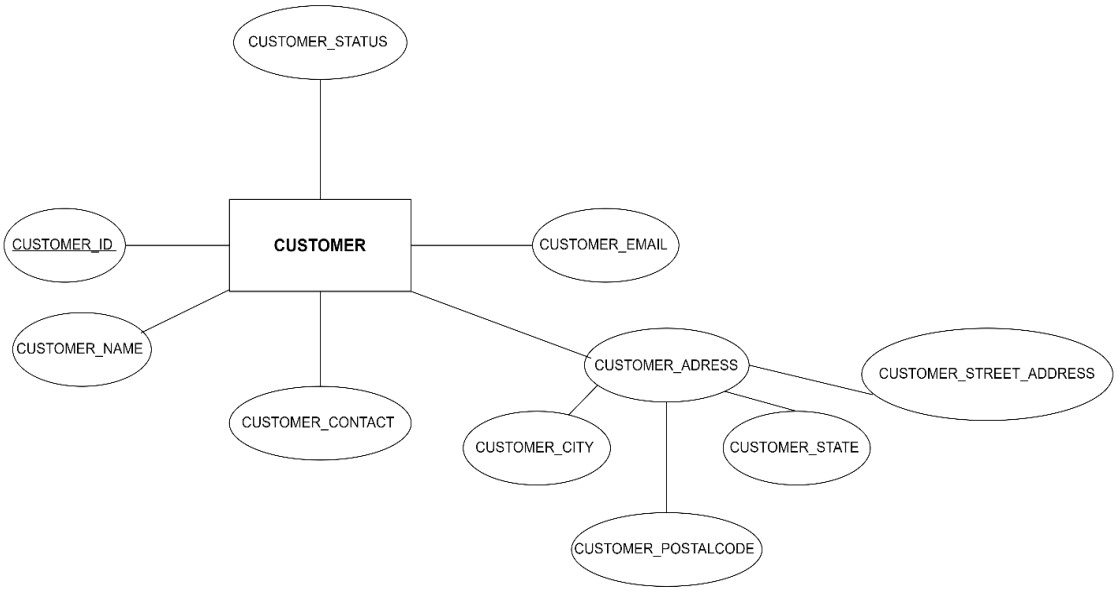
PRODUCT_LINE_T:



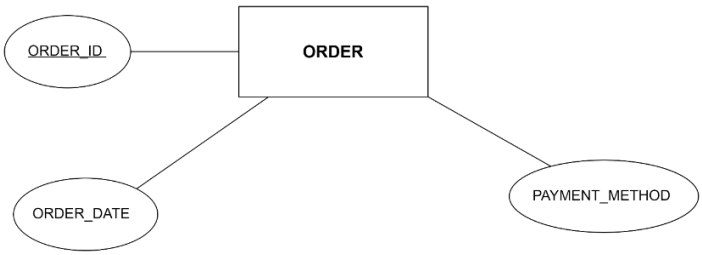
PRODUCT_T:



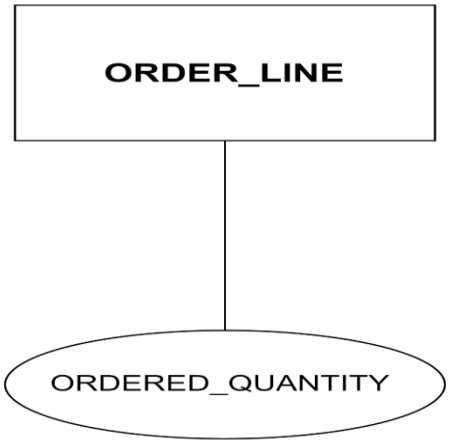
CUSTOMER_T:



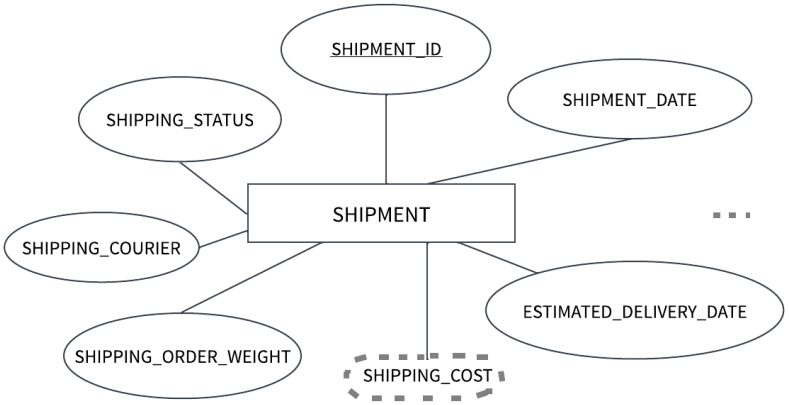
ORDER_T:



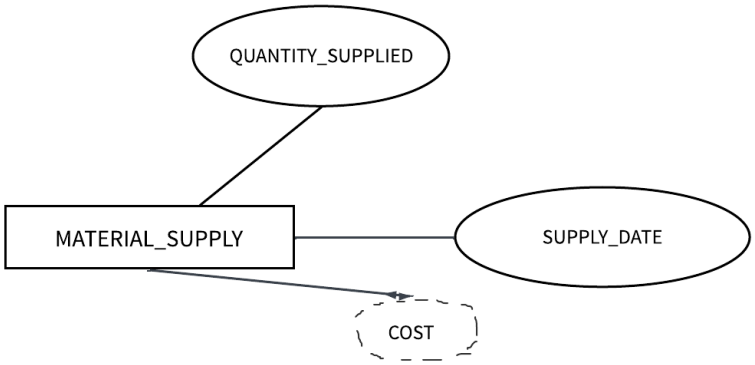
ORDER_LINE_T:



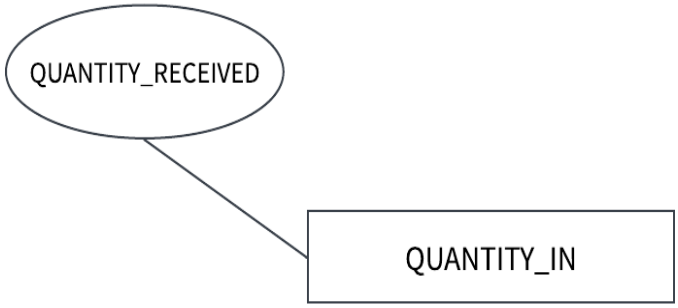
SHIPMENT:



MATERIAL_SUPPLY:

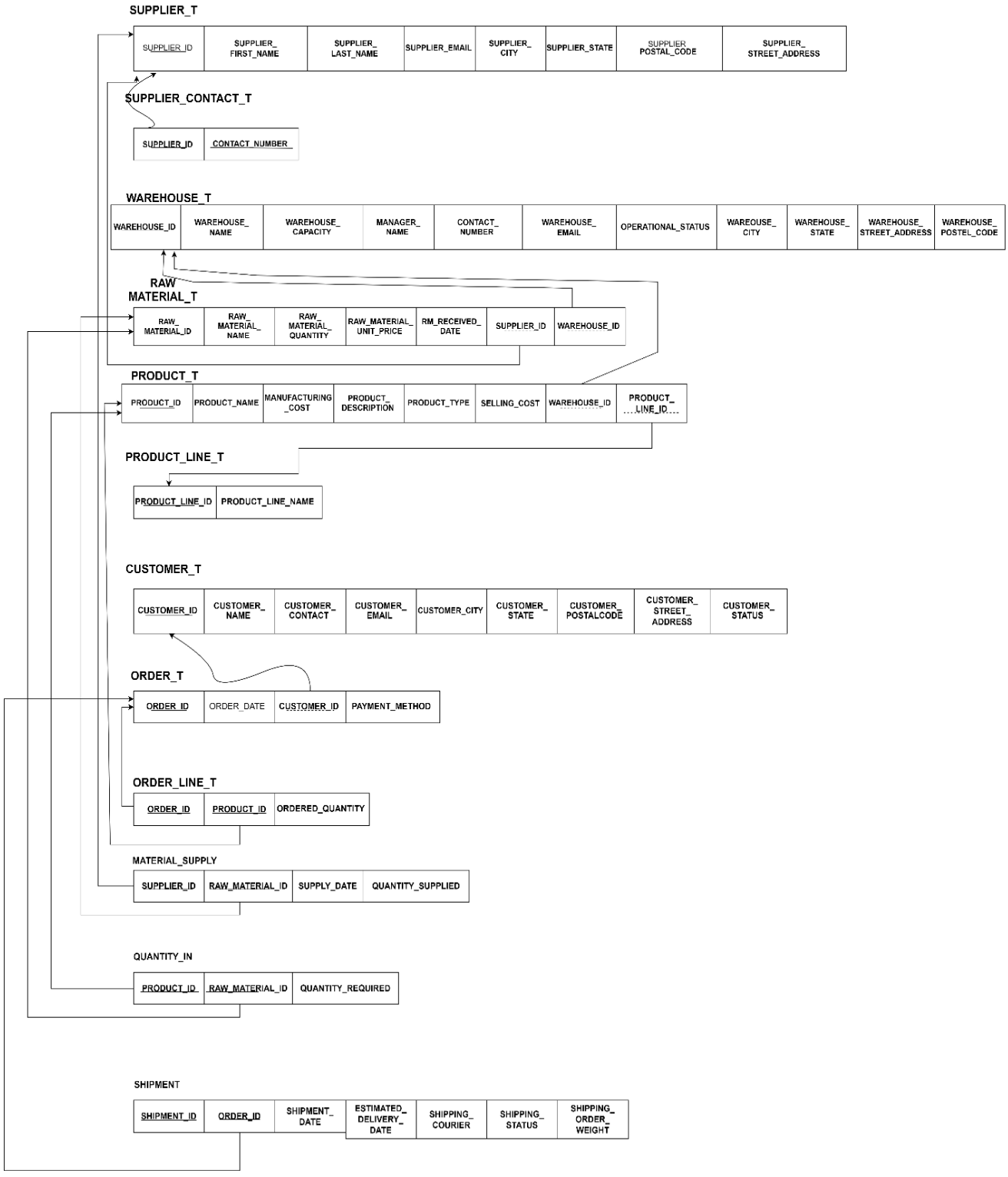


QUANTITY_IN:



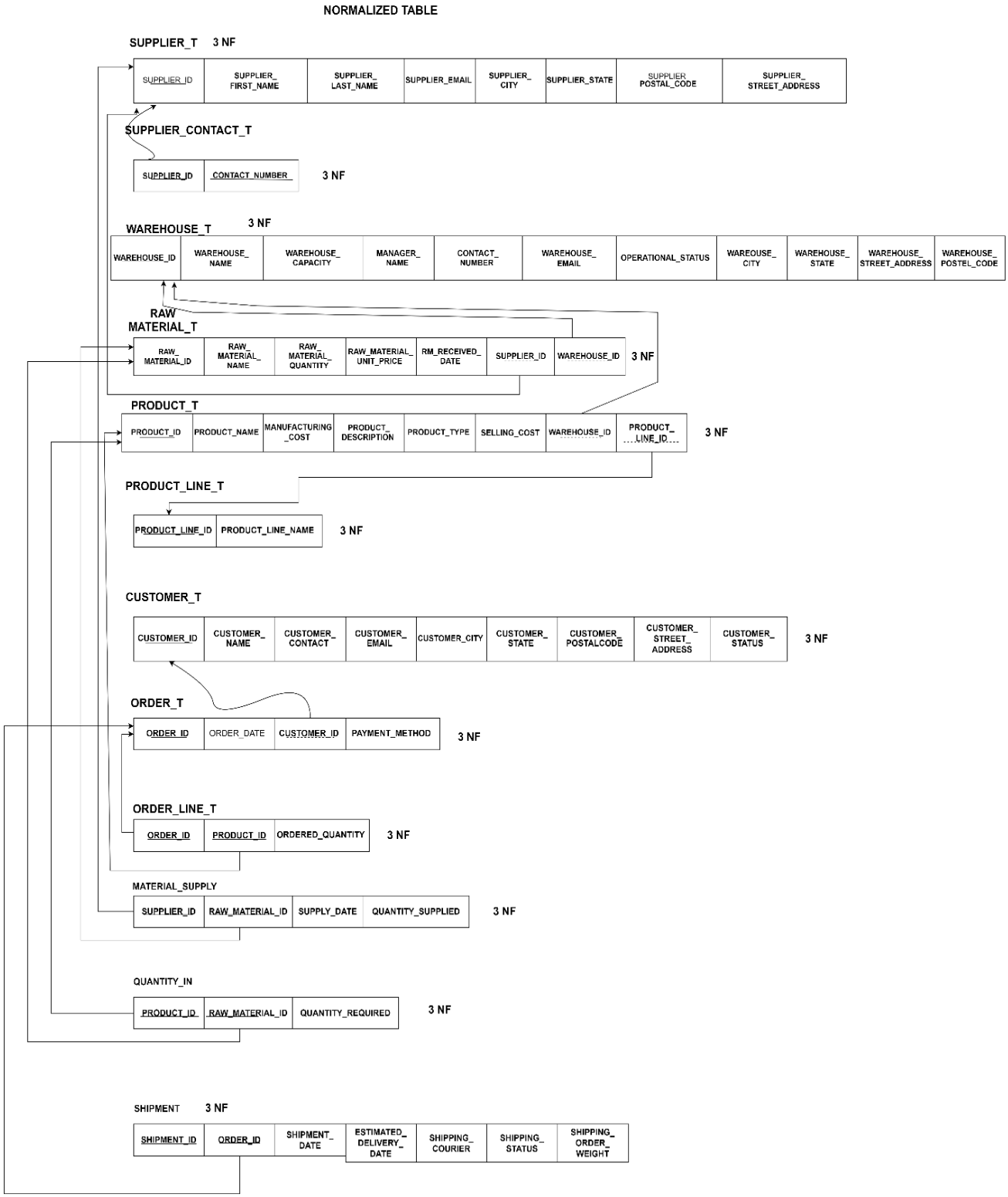
RELATIONAL MODEL

RELATIONAL SCHEMA



NORMALIZED TABLE

Table is already in 1NF,2NF so I show only 3NF HERE:



Physical Model

1. SUPPLIER_T

- **Attributes and Sizes**:
  - SUPPLIER_ID: 11 bytes
  - SUPPLIER_FIRST_NAME: 20 bytes
  - SUPPLIER_LAST_NAME: 20 bytes
  - SUPPLIER_EMAIL: 35 bytes
  - SUPPLIER_CITY: 20 bytes
  - SUPPLIER_STATE: 15 bytes
  - SUPPLIER_POSTAL_CODE: 7 bytes
  - SUPPLIER_STREET_ADDRESS: 30 bytes
- **Row Size**: 11 + 20 + 20 + 35 + 20 + 15 + 7 + 30 = **158 bytes**
- **Total for 1500 Rows**: 158 × 1500 = **237,000 bytes ≈ 231.64 KB**

---

2. SUPPLIER_CONTACT

- **Attributes and Sizes**:
  - SUPPLIER_ID: 11 bytes
  - CONTACT_NUMBER: 14 bytes
- **Row Size**: 11 + 14 = **25 bytes**
- **Total for 1500 Rows**: 25 × 1500 = **37,500 bytes ≈ 36.62 KB**

---

3. WAREHOUSE_T

- **Attributes and Sizes**:
  - WAREHOUSE_ID: 10 bytes
  - WAREHOUSE_NAME: 35 bytes
  - WAREHOUSE_LOCATION: 150 bytes
  - WAREHOUSE_CAPACITY: 10 bytes
  - MANAGER_NAME: 35 bytes
  - CONTACT_NUMBER: 15 bytes

- EMAIL: 100 bytes
- OPERATIONAL_STATUS: 25 bytes

- **Row Size**: 10 + 35 + 150 + 10 + 35 + 15 + 100 + 25 = **380 bytes**
- **Total for 1500 Rows**: 380 × 1500 = **570,000 bytes ≈ 556.64 KB**

---

4. RAW_MATERIAL_T

- **Attributes and Sizes**:
  - WAREHOUSE_ID: 10 bytes
  - RAW_MATERIAL_ID: 11 bytes
  - RAW_MATERIAL_NAME: 50 bytes
  - RAW_MATERIAL_QUANTITY: 8 bytes
  - RAW_MATERIAL_UNIT_PRICE: 6 bytes
  - SUPPLIER_ID: 11 bytes
  - RM_RECEIVED_DATE: 7 bytes
- **Row Size**: 10 + 11 + 50 + 8 + 6 + 11 + 7 = **103 bytes**
- **Total for 1500 Rows**: 103 × 1500 = **154,500 bytes ≈ 150.97 KB**

---

5. MATERIAL_SUPPLY

- **Attributes and Sizes**:
  - SUPPLIER_ID: 11 bytes
  - RAW_MATERIAL_ID: 11 bytes
  - SUPPLY_DATE: 7 bytes
  - QUANTITY_SUPPLIED: 6 bytes
- **Row Size**: 11 + 11 + 7 + 6 = **35 bytes**
- **Total for 1500 Rows**: 35 × 1500 = **52,500 bytes ≈ 51.27 KB**

---

6. PRODUCT_LINE_T

- **Attributes and Sizes**:
  - PRODUCT_LINE_ID: 11 bytes
  - PRODUCT_LINE_NAME: 100 bytes

- **Row Size**: 11 + 100 = **111 bytes**
- **Total for 1500 Rows**: 111 × 1500 = **166,500 bytes ≈ 162.50 KB**

---

7. PRODUCT_T

- **Attributes and Sizes**:
    - WAREHOUSE_ID: 10 bytes
    - PRODUCT_ID: 11 bytes
    - PRODUCT_NAME: 100 bytes
    - MANUFACTURING_COST: 6 bytes
    - PRODUCT_DESCRIPTION: 200 bytes
    - PRODUCT_TYPE: 20 bytes
    - SELLING_COST: 6 bytes
    - PRODUCT_LINE_ID: 11 bytes
    - PRODUCT_QUANTITY: 8 bytes
- **Row Size**: 10 + 11 + 100 + 6 + 200 + 20 + 6 + 11 + 8 = **372 bytes**
- **Total for 1500 Rows**: 372 × 1500 = **558,000 bytes ≈ 544.92 KB**

---

8. QUANTITY_IN

- **Attributes and Sizes**:
    - PRODUCT_ID: 11 bytes
    - RAW_MATERIAL_ID: 11 bytes
    - QUANTITY_REQUIRED: 8 bytes
- **Row Size**: 11 + 11 + 8 = **30 bytes**
- **Total for 1500 Rows**: 30 × 1500 = **45,000 bytes ≈ 43.95 KB**

---

9. CUSTOMER_T

- **Attributes and Sizes**:
    - CUSTOMER_ID: 11 bytes
    - CUSTOMER_NAME: 40 bytes
    - CUSTOMER_CONTACT: 14 bytes
    - CUSTOMER_EMAIL: 35 bytes

- CUSTOMER_CITY: 20 bytes
- CUSTOMER_STATE: 50 bytes
- CUSTOMER_POSTALCODE: 10 bytes
- CUSTOMER_STREET_ADDRESS: 30 bytes
- CUSTOMER_STATUS: 10 bytes

- **Row Size**: 11 + 40 + 14 + 35 + 20 + 50 + 10 + 30 + 10 = **220 bytes**
- **Total for 1500 Rows**: 220 × 1500 = **330,000 bytes ≈ 322.27 KB**

---

10. ORDER_T

- **Attributes and Sizes**:
  - ORDER_ID: 10 bytes
  - ORDER_DATE: 7 bytes
  - CUSTOMER_ID: 11 bytes
  - PAYMENT_METHOD: 10 bytes
- **Row Size**: 10 + 7 + 11 + 10 = **38 bytes**
- **Total for 1500 Rows**: 38 × 1500 = **57,000 bytes ≈ 55.66 KB**

---

11. ORDER_LINE_T

- **Attributes and Sizes**:
  - ORDER_ID: 10 bytes
  - PRODUCT_ID: 10 bytes
  - ORDERED_QUANTITY: 10 bytes
- **Row Size**: 10 + 10 + 10 = **30 bytes**
- **Total for 1500 Rows**: 30 × 1500 = **45,000 bytes ≈ 43.95 KB**

---

12. SHIPMENT

- **Attributes and Sizes**:
  - SHIPMENT_ID: 20 bytes
  - ORDER_ID: 10 bytes
  - SHIPMENT_DATE: 7 bytes
  - ESTIMATED_DELIVERY_DATE: 7 bytes

- SHIPPING_COURIER: 30 bytes
- SHIPPING_STATUS: 20 bytes
- SHIPPING_ORDER_WEIGHT: 10 bytes

- **Row Size**: 20 + 10 + 7 + 7 + 30 + 20 + 10 = **104 bytes**
- **Total for 1500 Rows**: 104 × 1500 = **156,000 bytes ≈ 152.34 KB**

Total Database Size

- Total for all 1500 rows across tables =
  231.64 KB + 36.62 KB + 556.64 KB + 150.97 KB + 51.27 KB + 162.50 KB + 544.92 KB + 43.95 KB + 322.27 KB + 55.66 KB + 43.95 KB + 152.34 KB ≈ 2352.73 KB ≈ 2.297 MB

SQL IMPEMENTATION

SUPPLER_T

```
CREATE TABLE SUPPLIER_T
(
   SUPPLIER_ID NUMBER(11, 0) NOT NULL,
   SUPPLIER_FIRST_NAME CHAR(20) NOT NULL,
   SUPPLIER_LAST_NAME CHAR(20) NOT NULL,
   SUPPLIER_EMAIL VARCHAR2(35),
   SUPPLIER_CITY CHAR(20),
   SUPPLIER_STATE CHAR(15),
   SUPPLIER_POSTAL_CODE NUMBER(7),
  SUPPLIER_STREET_ADDRESS VARCHAR2(30),
   CONSTRAINT SUPPLIER_PK PRIMARY KEY (SUPPLIER_ID)
);
```

SUPPLIER_CONTACT:

```
CREATE TABLE SUPPLIER_CONTACT
```

```
(
  SUPPLIER_ID NUMBER(11, 0) NOT NULL,
  CONTACT_NUMBER NUMBER(14),
  CONSTRAINT SUPPLIER_CONTACT_PK PRIMARY KEY (SUPPLIER_ID, CONTACT_NUMBER),
  CONSTRAINT SUPPLIER_CONTACT_FK FOREIGN KEY (SUPPLIER_ID)
  REFERENCES SUPPLIER_T (SUPPLIER_ID) ON DELETE CASCADE
);
```

WAREHOUSE_T:

```
CREATE TABLE WAREHOUSE_T (
  WAREHOUSE_ID NUMBER(10),          -- Unique identifier for the warehouse
  WAREHOUSE_NAME VARCHAR2(35),          -- Name of the warehouse
  WAREHOUSE_LOCATION VARCHAR2(150),          -- Physical address or location of the warehouse
  WAREHOUSE_CAPACITY NUMBER(10),          -- Capacity of the warehouse (e.g., in square feet or units)
  MANAGER_NAME CHAR(35),          -- Name of the warehouse manager
  CONTACT_NUMBER NUMBER(15),          -- Contact number for the warehouse manager
  EMAIL VARCHAR2(100),          -- Email address of the warehouse
  OPERATIONAL_STATUS CHAR(25),          -- Current operational status (e.g., "Active", "Under Maintenance")
CONSTRAINT WAREHOUSE_PK PRIMARY KEY (WAREHOUSE_ID)
);
```

RAW MATERIAL TABLE:

```
CREATE TABLE RAW_MATERIAL_T
(
  WAREHOUSE_ID NUMBER(10), --FOREIGN KEY
  RAW_MATERIAL_ID NUMBER(11, 0) NOT NULL, -- Primary Key
  RAW_MATERIAL_NAME VARCHAR2(50) NOT NULL, -- Name of the raw material
  RAW_MATERIAL_QUANTITY NUMBER(8, 0) NOT NULL, -- Quantity with up to 2 decimal places
  RAW_MATERIAL_UNIT_PRICE NUMBER(6, 2) NOT NULL,
  SUPPLIER_ID NUMBER(11, 0), -- Foreign Key to Supplier_t
  RM_RECEIVED_DATE DATE DEFAULT SYSDATE, -- Date when raw material was received
```

```
  CONSTRAINT RAW_MATERIAL_PK PRIMARY KEY (RAW_MATERIAL_ID),
  CONSTRAINT SUPPLIER_ID_FK_RM FOREIGN KEY (SUPPLIER_ID)
  REFERENCES SUPPLIER_T (SUPPLIER_ID) ON DELETE SET NULL,
CONSTRAINT WAREHOUSE_ID_FK_RM FOREIGN KEY (WAREHOUSE_ID)
REFERENCES WAREHOUSE_T (WAREHOUSE_ID) ON DELETE SET NULL
);
```

MATERIAL_SUPPLY:

```
CREATE TABLE MATERIAL_SUPPLY (
  SUPPLIER_ID NUMBER(11, 0), -- Foreign key from SUPPLIER_T
  RAW_MATERIAL_ID NUMBER(11, 0), -- Foreign Key from RAW_MATERIAL_T
  SUPPLY_DATE DATE DEFAULT SYSDATE,
  QUANTITY_SUPPLIED NUMBER(6, 0),

  -- Composite primary key for SUPPLIER_ID and RAW_MATERIAL_ID
  CONSTRAINT MATERIAL_SUPPLY_PK PRIMARY KEY (SUPPLIER_ID, RAW_MATERIAL_ID),

  -- Foreign key constraint for SUPPLIER_ID
  CONSTRAINT SUPPLIER_ID_FK FOREIGN KEY (SUPPLIER_ID)
    REFERENCES SUPPLIER_T (SUPPLIER_ID) ON DELETE SET NULL,

  -- Foreign key constraint for RAW_MATERIAL_ID
  CONSTRAINT RAW_MATERIAL_ID_FK FOREIGN KEY (RAW_MATERIAL_ID)
    REFERENCES RAW_MATERIAL_T (RAW_MATERIAL_ID)
);
```

PRODUCT_LINE_T:

```
CREATE TABLE PRODUCT_LINE_T
(
  PRODUCT_LINE_ID NUMBER(11, 0) NOT NULL, -- Primary Key
```

```
  PRODUCT_LINE_NAME VARCHAR2(100) NOT NULL, -- Name of the product line
  CONSTRAINT PRODUCT_LINE_PK PRIMARY KEY (PRODUCT_LINE_ID)
);
```

PRODUCT_T:

```
CREATE TABLE PRODUCT_T
(
WAREHOUSE_ID NUMBER(10), --Foreign key
  PRODUCT_ID NUMBER(11, 0) NOT NULL, -- Primary Key
  PRODUCT_NAME VARCHAR2(100) NOT NULL, -- Name of the product
  MANUFACTURING_COST NUMBER(6, 2) NOT NULL, -- Manufacturing cost with up to 2 decimal places
PRODUCT_DESCRIPTION VARCHAR2(200),
PRODUCT_TYPE CHAR(20),
  SELLING_COST NUMBER(6, 2) NOT NULL, -- Selling cost with up to 2 decimal places
PRODUCT_LINE_ID NUMBER(11, 0), -- Referencing PRODUCT_LINE_T
PRODUCT_QUANTITY NUMBER(8, 0),
  CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID),
CONSTRAINT WAREHOUSE_ID_FK_PRODUCT FOREIGN KEY (WAREHOUSE_ID)
REFERENCES WAREHOUSE_T (WAREHOUSE_ID) ON DELETE SET NULL,
CONSTRAINT PRODUCT_LINE_ID_FK FOREIGN KEY (PRODUCT_LINE_ID)
REFERENCES PRODUCT_LINE_T (PRODUCT_LINE_ID) ON DELETE SET NULL
);
```

QUANTITY_IN:

```
CREATE TABLE QUANTITY_IN(
PRODUCT_ID NUMBER(11, 0),
RAW_MATERIAL_ID NUMBER(11, 0),
QUANTITY_REQUIRED NUMBER(8, 0),
CONSTRAINT PRODUCT_ID_FK_Q_IN FOREIGN KEY (PRODUCT_ID)
REFERENCES PRODUCT_T (PRODUCT_ID),
CONSTRAINT RAW_MATERIAL_ID_FK_Q_IN FOREIGN KEY (RAW_MATERIAL_ID)
```

```
REFERENCES RAW_MATERIAL_T(RAW_MATERIAL_ID)
);
```

CUSTOMER_T:

```
CREATE TABLE CUSTOMER_T
(
   CUSTOMER_ID NUMBER(11, 0) NOT NULL, -- Primary Key
   CUSTOMER_NAME CHAR(40) NOT NULL, -- Name of the customer
   CUSTOMER_CONTACT NUMBER(14) NOT NULL, -- Contact number of the customer
   CUSTOMER_EMAIL VARCHAR2(35), -- Email of the customer
   CUSTOMER_CITY CHAR(20) NOT NULL, -- City of the customer
   CUSTOMER_STATE VARCHAR2(50) NOT NULL, -- State of the customer
   CUSTOMER_POSTALCODE NUMBER(10) NOT NULL, -- Postal code of the customer's location
CUSTOMER_STREET_ADDRESS VARCHAR2(30),
CUSTOMER_STATUS CHAR(10), --CHECK WHETHER CUSTOMER HAS LEFT OR NOT
   CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID)
);
```

ORDER_T:

```
CREATE TABLE ORDER_T(
ORDER_ID NUMBER(10) NOT NULL,        -- Order_Id as primary key
ORDER_DATE DATE DEFAULT SYSDATE, -- Order date
CUSTOMER_ID NUMBER(11, 0),  -- Foreign Key from
PAYMENT_METHOD VARCHAR2(10),         -- Payment method (assuming a maximum of 10 characters)
CONSTRAINT CUSTOMER_ID_FK FOREIGN KEY (CUSTOMER_ID)
REFERENCES CUSTOMER_T (CUSTOMER_ID),
CONSTRAINT ORDER_ID_PK PRIMARY KEY (ORDER_ID)
);
```

ORDER_LINE_T:

```
CREATE TABLE ORDER_LINE_T(
   ORDER_ID NUMBER(10),                -- Order_Id (foreign key referencing Order_t)
```

```
  PRODUCT_ID NUMBER(10),            -- Product_Id (foreign key referencing Product_t)
  ORDERED_QUANTITY NUMBER(10),      -- Ordered_Quantity (numeric value)
 PRIMARY KEY(ORDER_ID, PRODUCT_ID),   -- Composite Primary Key on (Order_Id, Product_Id)
CONSTRAINT ORDER_ID_FK FOREIGN KEY (ORDER_ID) REFERENCES ORDER_T(ORDER_ID),  -- Foreign Key linking to Order_t
CONSTRAINT PRODUCT_ID_FK FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID)
);
```

SHIPMENT:

```
CREATE TABLE SHIPMENT (
  SHIPMENT_ID  NUMBER(20),        -- Tracking Number as primary key
  ORDER_ID NUMBER(10, 0),                -- Foreign key referencing Order_Id in Order_t
  SHIPMENT_DATE DATE,                    -- Shipment Date (DATE data type to store date and time)
  ESTIMATED_DELIEVERY_DATE DATE,              -- Estimated Delivery Date (DATE data type to store date and time)
SHIPPING_COURIER VARCHAR2(30),            -- Shipping Courier (VARCHAR2 data type to store courier name)
  SHIPPING_STATUS CHAR(20),               -- Shipping Status (VARCHAR2 data type to store shipping status)
SHIPPING_ORDER_WEIGHT  NUMBER(10),

 CONSTRAINT ORDER_T_FK FOREIGN KEY (ORDER_ID) REFERENCES ORDER_T (ORDER_ID), -- Foreign Key linking to Order_t
CONSTRAINT SHIPMENT_PK PRIMARY KEY (SHIPMENT_ID, ORDER_ID)
);
```

DATA INSERTION

I am just showing the data insertion by taking some record of inserting value in SQL.

SUPPLIER_T:

```
INSERT INTO SUPPLIER_T (SUPPLIER_ID, SUPPLIER_FIRST_NAME, SUPPLIER_LAST_NAME, SUPPLIER_EMAIL, SUPPLIER_CITY, SUPPLIER_STATE, SUPPLIER_POSTAL_CODE, SUPPLIER_STREET_ADDRESS) VALUES (3221, 'Larry', 'Chavez', 'hollyanderson@anderson.com', 'Roseland', 'FL', 96240, '898 Harris Branch');
INSERT INTO SUPPLIER_T (SUPPLIER_ID, SUPPLIER_FIRST_NAME, SUPPLIER_LAST_NAME, SUPPLIER_EMAIL, SUPPLIER_CITY, SUPPLIER_STATE, SUPPLIER_POSTAL_CODE, SUPPLIER_STREET_ADDRESS) VALUES (3223, 'Lisa', 'White', 'trevor52@gonzalez-hubbard.com', 'South Donnabury', 'MO', 63345, '664 Cody Greens Apt. 644');
```

SUPPLIER_ID	SUPPLIER_FIRST_NAME	SUPPLIER_LAST_NAME	SUPPLIER_EMAIL	SUPPLIER_CITY	SUPPLIER_STATE	SUPPLIER_POSTAL_CODE	SUPPLIER_STREET_ADDRESS
3259	Keith	Johnson	john66@hughes-miller.org	Robertsborough	OR	34368	9483 Mccoy Burg Apt. 825
3261	Margaret	Barnes	johnsonann@graves.net	Lambmouth	HI	2345	993 Kim Ville
3263	Megan	Mason	vfarley@gmail.com	Mccannport	IN	58457	2595 Dustin Rapid Apt. 768
3265	Kathryn	Brown	destinyarnold@patton.org	South Marychester	WV	51235	89242 Koch Canyon
3267	Benjamin	Martin	heather44@bishop.info	Port Angelahaven	MN	99643	4357 Thomas Mill Suite 585
3269	Christopher	Peterson	michael02@reed-mills.com	West Richardhaven	AR	68887	3202 Kevin Shore Suite 898
3273	Doris	Hernandez	troymacias@erickson.com	New Christinafort	NC	28666	230 Williams Court
3275	Rebekah	Day	kristinpatterson@gmail.com	Kristiborough	NM	19686	1499 Denise Neck
3277	Christopher	Mcintyre	wallaceshannon@hotmail.com	Snowview	ID	92678	9742 Perry Summit Apt. 625
3279	Ruth	Little	xbell@hotmail.com	Nataliestad	DE	78425	758 Sean Branch Apt. 330

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.06 seconds CSV Export

SUPPLIER_CONTACT:

```
INSERT INTO SUPPLIER_CONTACT (SUPPLIER_ID, CONTACT_NUMBER) VALUES (3221, 18005551234);
INSERT INTO SUPPLIER_CONTACT (SUPPLIER_ID, CONTACT_NUMBER) VALUES (3221, 14155550987);
```

SUPPLIER_ID	CONTACT_NUMBER
3221	14155550987
3221	18005551234
3223	14155554563
3223	18885552345
3225	14155553456
3225	18005554567
3227	14155555638
3227	14155555678
3229	18005556789
3229	18005558637

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds CSV Export

WAREHOUSE_T:

```
INSERT INTO WAREHOUSE_T (
    WAREHOUSE_ID, WAREHOUSE_NAME, WAREHOUSE_LOCATION,
    WAREHOUSE_CAPACITY, MANAGER_NAME, CONTACT_NUMBER,
    EMAIL, OPERATIONAL_STATUS
) VALUES (
    4001, 'Central Distribution Center', '1234 Logistics Parkway, Chicago, IL 60601',
    85000, 'Michael Rodriguez', 18005551234,
    'central.warehouse@globaldist.com', 'Active'
);

INSERT INTO WAREHOUSE_T (
    WAREHOUSE_ID, WAREHOUSE_NAME, WAREHOUSE_LOCATION,
    WAREHOUSE_CAPACITY, MANAGER_NAME, CONTACT_NUMBER,
    EMAIL, OPERATIONAL_STATUS
) VALUES (
    4003, 'West Coast Fulfillment Hub', '5678 Ocean Drive, Los Angeles, CA 90001',
    92000, 'Sarah Chen', 14155552345,
    'westcoast.warehouse@globaldist.com', 'Active'
);
```

WAREHOUSE_ID	WAREHOUSE_NAME	WAREHOUSE_LOCATION	WAREHOUSE_CAPACITY	MANAGER_NAME	CONTACT_NUMBER	EMAIL	OPERATIONAL_STATUS
4001	Central Distribution Center	1234 Logistics Parkway, Chicago, IL 60601	85000	Michael Rodriguez	18005551234	central.warehouse@globaldist.com	Active
4003	West Coast Fulfillment Hub	5678 Ocean Drive, Los Angeles, CA 90001	92000	Sarah Chen	14155552345	westcoast.warehouse@globaldist.com	Active
4576	Northeast Logistics Center	9012 Industrial Road, Newark, NJ 07102	65000	David Kim	18885553456	northeast.warehouse@globaldist.com	Under Maintenance
4823	Southern Regional Warehouse	3456 Sunbelt Avenue, Atlanta, GA 30303	78000	Jennifer Martinez	14155554567	southern.warehouse@globaldist.com	Active
4659	Midwest Supply Depot	7890 Prairie Lane, Kansas City, MO 64101	55000	Thomas Anderson	18005555678	midwest.warehouse@globaldist.com	Active
4544	Pacific Northwest Warehouse	2345 Evergreen Terrace, Seattle, WA 98101	68000	Emily Wong	14155556789	pnw.warehouse@globaldist.com	Active
4444	Southwest Distribution Center	6789 Desert Road, Phoenix, AZ 85001	72000	Robert Garcia	18885557890	southwest.warehouse@globaldist.com	Under Maintenance
4331	East Coast Mega Warehouse	4321 Harbor Street, Miami, FL 33101	95000	Amanda Lee	14155558901	eastcoast.warehouse@globaldist.com	Active
4868	Rocky Mountain Storage Facility	8765 Mountain View Drive, Denver, CO 80202	62000	Steven Thompson	18005559012	rockymountain.warehouse@globaldist.com	Active
4978	Texas Logistics Complex	5432 Big Sky Boulevard, Dallas, TX 75201	88000	Maria Rodriguez	14155550123	texas.warehouse@globaldist.com	Active

10 rows returned in 0.02 seconds CSV Export

RAW_MATERIAL_T:

```
INSERT INTO RAW_MATERIAL_T (WAREHOUSE_ID, RAW_MATERIAL_ID, RAW_MATERIAL_NAME,
RAW_MATERIAL_QUANTITY, RAW_MATERIAL_UNIT_PRICE, SUPPLIER_ID)
VALUES (4001, 5010, 'Copper Wire', 5000, 0.25, 3221);
INSERT INTO RAW_MATERIAL_T (WAREHOUSE_ID, RAW_MATERIAL_ID, RAW_MATERIAL_NAME,
RAW_MATERIAL_QUANTITY, RAW_MATERIAL_UNIT_PRICE, SUPPLIER_ID)
VALUES (4978, 5020, 'Keyboard PCB', 2000, 2.25, 3241);
```

WAREHOUSE_ID	RAW_MATERIAL_ID	RAW_MATERIAL_NAME	RAW_MATERIAL_QUANTITY	RAW_MATERIAL_UNIT_PRICE	SUPPLIER_ID	RM_RECEIVED_DATE
4001	5010	Copper Wire	5000	55	3221	15-DEC-24
4001	5110	PCB Board	2500	1.5	3275	15-DEC-24
4823	5210	USB Connector Plastic	3000	.35	3221	15-DEC-24
4659	5310	Silicon Chip	1500	3.25	3227	15-DEC-24
4544	5410	Gold-Plated Contacts	750	2.75	3229	15-DEC-24
4544	5510	Optical Sensor	2000	1.8	3231	15-DEC-24
4331	5610	Mouse Button Switches	4000	.45	3233	15-DEC-24
4868	5710	Mouse Scroll Wheel	3500	.65	3235	15-DEC-24
4868	5810	Mouse Shell Plastic	5000	.3	3237	15-DEC-24
4978	5910	Rubber Mouse Feet	6000	.15	3237	15-DEC-24

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds CSV Export

MATERIAL_Supply:

```
INSERT INTO MATERIAL_SUPPLY (SUPPLIER_ID, RAW_MATERIAL_ID, QUANTITY_SUPPLIED)
VALUES (3221, 5210, 3200);
INSERT INTO MATERIAL_SUPPLY (SUPPLIER_ID, RAW_MATERIAL_ID, QUANTITY_SUPPLIED)
VALUES (3227, 5310, 1400);
```

SUPPLIER_ID	RAW_MATERIAL_ID	SUPPLY_DATE	QUANTITY_SUPPLIED
3251	5710	15-DEC-24	3700
3253	5220	15-DEC-24	1400
3255	5910	15-DEC-24	2800
3267	5420	15-DEC-24	1900
3271	5320	15-DEC-24	2000

PRODUCT_T:

```
INSERT INTO PRODUCT_T (WAREHOUSE_ID, PRODUCT_ID, PRODUCT_NAME, MANUFACTURING_COST,
PRODUCT_DESCRIPTION, PRODUCT_TYPE, SELLING_COST, PRODUCT_LINE_ID, PRODUCT_QUANTITY)
VALUES (4003, 7331, 'NanoStore Elite', 8.75, 'Compact and durable ultra-slim USB drive with
military-grade encryption and waterproof coating', 'USB', 24.50, 6502, 600);
INSERT INTO PRODUCT_T (WAREHOUSE_ID, PRODUCT_ID, PRODUCT_NAME, MANUFACTURING_COST,
PRODUCT_DESCRIPTION, PRODUCT_TYPE, SELLING_COST, PRODUCT_LINE_ID, PRODUCT_QUANTITY)
VALUES (4331, 7833, 'SilentGlide Precision', 22.75, 'Ergonomic wireless mouse with ultra-quiet
click mechanism and adjustable DPI settings', 'Mouse', 79.99, 6147, 1500);
```

WAREHOUSE_ID	PRODUCT_ID	PRODUCT_NAME	MANUFACTURING_COST	PRODUCT_DESCRIPTION	PRODUCT_TYPE	SELLING_COST	PRODUCT_LINE_ID	PRODUCT_QUANTITY
4001	7222	SpeedFlash Pro	80	High-speed USB 3.1 flash drive with sleek metallic design and enhanced data transfer capabilities	USB	100	6023	700
4003	7331	NanoStore Elite	8.75	Compact and durable ultra-slim USB drive with military-grade encryption and waterproof coating	USB	24.5	6502	600
4331	7833	SilentGlide Precision	22.75	Ergonomic wireless mouse with ultra-quiet click mechanism and adjustable DPI settings	Mouse	79.99	6147	1500
4868	7651	QuantumClick Elite	18.5	Gaming mouse with RGB lighting, 16000 DPI sensor, and customizable weight system	Mouse	89.5	6147	1900
4544	7301	MechaStrike Tournament	45.75	Mechanical gaming keyboard with Cherry MX switches, full RGB lighting, and macro programmability	Keyboard	129.99	6147	1600
4978	7434	SilentFlow Professional	35.5	Low-noise mechanical keyboard with ergonomic wrist rest and custom keycap design	Keyboard	99.5	6381	1500

6 rows returned in 0.00 seconds CSV Export

QUANTITY_IN:

```
INSERT INTO QUANTITY_IN (PRODUCT_ID, RAW_MATERIAL_ID, QUANTITY_REQUIRED)
VALUES (7331, 5410 , 600);
INSERT INTO QUANTITY_IN (PRODUCT_ID, RAW_MATERIAL_ID, QUANTITY_REQUIRED)
VALUES (7833, 5510 , 1500);
```

PRODUCT_ID	RAW_MATERIAL_ID	QUANTITY_REQUIRED
7222	5410	100
7222	5010	700
7222	5110	700
7222	5210	700
7222	5310	700
7222	5410	700
7331	5010	600
7331	5110	600
7331	5210	600
7331	5310	600

More than 10 rows available. Increase rows selector to view more rows.

PRODUCT_LINE_T:

```
INSERT INTO PRODUCT_LINE_T (PRODUCT_LINE_ID, PRODUCT_LINE_NAME)
VALUES (6502, 'High-Performance Storage Systems');
INSERT INTO PRODUCT_LINE_T (PRODUCT_LINE_ID, PRODUCT_LINE_NAME)
VALUES (6634, 'Mobile Computing Accessories');
```

PRODUCT_LINE_ID	PRODUCT_LINE_NAME
6023	Professional Desktop Accessories
6147	Gaming Peripherals
6259	Wireless Communication Devices
6381	Ergonomic Computer Solutions
6502	High-Performance Storage Systems
6634	Mobile Computing Accessories
6755	Professional Audio-Visual Equipment
6876	Enterprise Networking Solutions
6992	Home Office Technology
6415	Advanced Peripheral Innovations

ORDER_T:

```
INSERT INTO ORDER_T (ORDER_ID, CUSTOMER_ID, PAYMENT_METHOD)
VALUES (1003, 8050, 'DEBIT');
INSERT INTO ORDER_T (ORDER_ID, CUSTOMER_ID, PAYMENT_METHOD)
VALUES (1005, 8080, 'CASH');
```

ORDER_ID	ORDER_DATE	CUSTOMER_ID	PAYMENT_METHOD
1001	15-DEC-24	8010	CREDIT
1002	15-DEC-24	8010	PAYPAL
1003	15-DEC-24	8050	DEBIT
1005	15-DEC-24	8080	CASH
1006	15-DEC-24	8090	PAYPAL

5 rows returned in 0.02 seconds CSV Export

ORDER_LINE:

```
INSERT INTO ORDER_LINE_T (ORDER_ID, PRODUCT_ID, ORDERED_QUANTITY)
```

```
VALUES (1002, 7651, 20);
INSERT INTO ORDER_LINE_T (ORDER_ID, PRODUCT_ID, ORDERED_QUANTITY)
VALUES (1003, 7222, 30);
```

ORDER_ID	PRODUCT_ID	ORDERED_QUANTITY
1001	7331	6
1001	7434	10
1002	7651	20
1003	7222	30
1003	7301	50
1006	7434	15
1005	7651	9
1006	7222	8
1002	7301	6
1005	7331	3

10 rows returned in 0.01 seconds CSV Export

CUSTOMER_T:

```
INSERT INTO CUSTOMER_T
(CUSTOMER_ID, CUSTOMER_NAME, CUSTOMER_CONTACT, CUSTOMER_EMAIL,
 CUSTOMER_CITY, CUSTOMER_STATE, CUSTOMER_POSTALCODE, CUSTOMER_STREET_ADDRESS, CUSTOMER_STATUS)
VALUES
(8020, 'Emily Johnson', 15559876543, 'emily.j@gmail.com',
 'Chicago', 'Illinois', 60601, '456 Michigan Street', 'ACTIVE');
INSERT INTO CUSTOMER_T
(CUSTOMER_ID, CUSTOMER_NAME, CUSTOMER_CONTACT, CUSTOMER_EMAIL,
 CUSTOMER_CITY, CUSTOMER_STATE, CUSTOMER_POSTALCODE, CUSTOMER_STREET_ADDRESS, CUSTOMER_STATUS)
VALUES
(8030, 'Michael Rodriguez', 15552223333, 'michael.r@hotmail.com',
 'Los Angeles', 'California', 90001, '789 Sunset Boulevard', 'ACTIVE');
```

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_CONTACT	CUSTOMER_EMAIL	CUSTOMER_CITY	CUSTOMER_STATE	CUSTOMER_POSTALCODE	CUSTOMER_STREET_ADDRESS	CUSTOMER_STATUS
8010	John Smith	15551234567	john.smith@email.com	New York	New York	10001	123 Broadway Ave	ACTIVE
8020	Emily Johnson	15559876543	emily.j@gmail.com	Chicago	Illinois	60601	456 Michigan Street	ACTIVE
8030	Michael Rodriguez	15552223333	michael.r@hotmail.com	Los Angeles	California	90001	789 Sunset Boulevard	ACTIVE
8040	Sarah Williams	15554445555	sarah.w@yahoo.com	Houston	Texas	77001	212 Main Street	INACTIVE
8050	David Lee	15556667777	david.lee@outlook.com	Phoenix	Arizona	85001	345 Desert Road	ACTIVE
8060	Jennifer Martinez	15558889999	jennifer.m@gmail.com	Miami	Florida	33101	678 Ocean Drive	ACTIVE
8070	Robert Thompson	15550001111	robert.t@email.com	Seattle	Washington	98101	901 Pine Street	INACTIVE
8080	Amanda Baker	15552226666	amanda.b@yahoo.com	Denver	Colorado	80201	234 Mountain View	ACTIVE
8090	Christopher Davis	15554442222	chris.davis@hotmail.com	Atlanta	Georgia	30301	567 Peachtree Street	ACTIVE
8100	Jessica Wilson	15556664444	jessica.w@outlook.com	Boston	Massachusetts	2101	890 Beacon Hill	ACTIVE

10 rows returned in 0.01 seconds CSV Export

SHIPMENT_T:

```
INSERT INTO SHIPMENT
(SHIPMENT_ID, ORDER_ID, SHIPMENT_DATE, ESTIMATED_DELIEVERY_DATE, SHIPPING_COURIER,
SHIPPING_STATUS, SHIPPING_ORDER_WEIGHT)
VALUES
(78901234561, 1002, TO_DATE('2024-03-16', 'YYYY-MM-DD'),
 TO_DATE('2024-03-22', 'YYYY-MM-DD'), 'UPS', 'PROCESSING', 33);
INSERT INTO SHIPMENT
(SHIPMENT_ID, ORDER_ID, SHIPMENT_DATE, ESTIMATED_DELIEVERY_DATE, SHIPPING_COURIER,
SHIPPING_STATUS, SHIPPING_ORDER_WEIGHT)
VALUES
(78901234232, 1003, TO_DATE('2024-03-17', 'YYYY-MM-DD'),
 TO_DATE('2024-03-23', 'YYYY-MM-DD'), 'USPS', 'DELIVERED', 22);
```

SHIPMENT_ID	ORDER_ID	SHIPMENT_DATE	ESTIMATED_DELIEVERY_DATE	SHIPPING_COURIER	SHIPPING_STATUS	SHIPPING_ORDER_WEIGHT
78901234561	1001	15-MAR-24	20-MAR-24	FedEx	IN_TRANSIT	15
78901234561	1002	16-MAR-24	22-MAR-24	UPS	PROCESSING	33
78901234232	1003	17-MAR-24	23-MAR-24	USPS	DELIVERED	22
78901234234	1003	18-MAR-24	25-MAR-24	DHL	IN_TRANSIT	16
78901234234	1005	19-MAR-24	26-MAR-24	FedEx	PROCESSING	24
78901234254	1006	20-MAR-24	27-MAR-24	UPS	IN_TRANSIT	13
78901234254	1002	16-MAR-24	28-MAR-24	USPS	DELIVERED	25
78901234266	1002	16-MAR-24	29-MAR-24	DHL	PROCESSING	14

8 rows returned in 0.01 seconds CSV Export

QURIES FOR ANALYSIS

1. Top Selling(REVENUE) Product by Product Type:

```
SELECT PRODUCT_TYPE, TOTAL_REVENUE
FROM (
    SELECT
        P.PRODUCT_TYPE,
        SUM(OL.ORDERED_QUANTITY * P.SELLING_COST) AS TOTAL_REVENUE
    FROM
        ORDER_LINE_T OL
    JOIN
        PRODUCT_T P ON OL.PRODUCT_ID = P.PRODUCT_ID
    GROUP BY
        P.PRODUCT_TYPE
    ORDER BY
        TOTAL_REVENUE DESC
)
WHERE ROWNUM = 1;
```

PRODUCT_TYPE	TOTAL_REVENUE
Keyboard	9766.94

1 rows returned in 0.05 seconds CSV Export

2. Average Manufacturing Cost per Product

AVERAGE_MANUFACTURING_COST
35.208333333333333333333333333333333333

1 rows returned in 0.00 seconds CSV Export

TRIGGERS:

Trigger on RAW_MATERIAL_PRICE UPDATE:

```
CREATE TABLE RAW_MATERIAL_PRICE_AUDIT (
  RAW_MATERIAL_ID NUMBER(11, 0) NOT NULL, -- ID of the raw material
  OLD_UNIT_PRICE NUMBER(6, 2), -- Price before the update
  NEW_UNIT_PRICE NUMBER(6, 2), -- Price after the update
  UPDATED_ON DATE DEFAULT SYSDATE -- Timestamp of the update
);


CREATE OR REPLACE TRIGGER TRG_RAW_MATERIAL_PRICE_AUDIT
AFTER UPDATE OF RAW_MATERIAL_UNIT_PRICE ON RAW_MATERIAL_T
FOR EACH ROW
BEGIN
```

```
  -- Check if the unit price has actually changed
  IF :OLD.RAW_MATERIAL_UNIT_PRICE != :NEW.RAW_MATERIAL_UNIT_PRICE THEN
    -- Insert the change into the audit table
    INSERT INTO RAW_MATERIAL_PRICE_AUDIT (
      RAW_MATERIAL_ID, OLD_UNIT_PRICE, NEW_UNIT_PRICE, UPDATED_ON
    ) VALUES (
      :OLD.RAW_MATERIAL_ID,       -- The ID of the raw material being updated
      :OLD.RAW_MATERIAL_UNIT_PRICE, -- The old price before the update
      :NEW.RAW_MATERIAL_UNIT_PRICE, -- The new price after the update
      SYSDATE                  -- The current timestamp
    );
  END IF;
END;
/
```

EXAMPLE:

```
SELECT * FROM RAW_MATERIAL_T;
UPDATE RAW_MATERIAL_T
SET RAW_MATERIAL_UNIT_PRICE = 55.00
WHERE RAW_MATERIAL_ID = 5010;
SELECT * FROM RAW_MATERIAL_PRICE_AUDIT;
```

RAW_MATERIAL_ID	OLD_UNIT_PRICE	NEW_UNIT_PRICE	UPDATED_ON
5010	.25	55	16-DEC-24

1 rows returned in 0.00 seconds CSV Export

```
UPDATE RAW_MATERIAL_T
SET RAW_MATERIAL_UNIT_PRICE = 85
WHERE RAW_MATERIAL_ID = 5010;
SELECT * FROM RAW_MATERIAL_PRICE_AUDIT;
```

RAW_MATERIAL_ID	OLD_UNIT_PRICE	NEW_UNIT_PRICE	UPDATED_ON
5010	.25	55	16-DEC-24
5010	55	85	16-DEC-24

2 rows returned in 0.01 seconds CSV Export

Check if the required quantity exceeds the available quantity:

```
CREATE OR REPLACE TRIGGER TRG_QUANTITY_CHECK
BEFORE INSERT OR UPDATE ON QUANTITY_IN
FOR EACH ROW
DECLARE
  v_available_quantity NUMBER(8, 0); -- Variable to store raw material quantity
BEGIN
  -- Fetch the available quantity of the raw material from RAW_MATERIAL_T
  SELECT RAW_MATERIAL_QUANTITY
  INTO v_available_quantity
  FROM RAW_MATERIAL_T
  WHERE RAW_MATERIAL_ID = :NEW.RAW_MATERIAL_ID;

  -- Check if the required quantity exceeds the available quantity
  IF :NEW.QUANTITY_REQUIRED > v_available_quantity THEN
    -- Raise an error if the required quantity exceeds the available quantity
    RAISE_APPLICATION_ERROR(-20002, 'Error: Required quantity exceeds available raw material quantity.');
  END IF; -- Ensure this line ends with a proper semicolon and END IF formatting
END;
/
```

EXAMPLE:

```
SELECT * FROM RAW_MATERIAL_T;
SELECT * FROM PRODUCT_T;
SELECT * FROM QUANTITY_IN;
INSERT INTO QUANTITY_IN (
```

```
  PRODUCT_ID, RAW_MATERIAL_ID, QUANTITY_REQUIRED
) VALUES (
  7222, 5410, 1250
);
```

```
ORA-20002: Error: Required quantity exceeds available raw material quantity.
ORA-06512: at "SCM.TRG_QUANTITY_CHECK", line 13
ORA-04088: error during execution of trigger 'SCM.TRG_QUANTITY_CHECK'
1. INSERT INTO QUANTITY_IN (
2.     PRODUCT_ID, RAW_MATERIAL_ID, QUANTITY_REQUIRED
3. ) VALUES (
```

Create a trigger on manufacturing cost and selling cost such that the manufacturing cost is always less than selling cost. Also whenever the manufacturing cost or selling cost is updated, they both should not be same i.e old manufacturing cost should not be equal to new manufacturing cost, similarly, old selling cost should not be equal new selling cost. Also make a table containing product_id along with its manufacturing and selling cost with the respective time stamp in which both is updated...

```
CREATE TABLE PRODUCT_COST_AUDIT (
  PRODUCT_ID NUMBER(11, 0) NOT NULL, -- ID of the product
  OLD_MANUFACTURING_COST NUMBER(6, 2), -- Old manufacturing cost
  NEW_MANUFACTURING_COST NUMBER(6, 2), -- New manufacturing cost
  OLD_SELLING_COST NUMBER(6, 2), -- Old selling cost
  NEW_SELLING_COST NUMBER(6, 2), -- New selling cost
  UPDATED_ON DATE DEFAULT SYSDATE -- Timestamp of the update
);
CREATE OR REPLACE TRIGGER TRG_PRODUCT_COST_VALIDATION
BEFORE UPDATE OF MANUFACTURING_COST, SELLING_COST ON PRODUCT_T
FOR EACH ROW
BEGIN
  -- Ensure manufacturing cost is less than selling cost
```

```
  IF :NEW.MANUFACTURING_COST >= :NEW.SELLING_COST THEN
    RAISE_APPLICATION_ERROR(-20003, 'Error: Manufacturing cost must be less than selling cost.');
  END IF;

  -- Ensure manufacturing cost changes
  IF :OLD.MANUFACTURING_COST = :NEW.MANUFACTURING_COST THEN
    RAISE_APPLICATION_ERROR(-20004, 'Error: Manufacturing cost must be updated to a new value.');
  END IF;

  -- Ensure selling cost changes
  IF :OLD.SELLING_COST = :NEW.SELLING_COST THEN
    RAISE_APPLICATION_ERROR(-20005, 'Error: Selling cost must be updated to a new value.');
  END IF;

  -- Log the update into the audit table
  INSERT INTO PRODUCT_COST_AUDIT (
    PRODUCT_ID,
    OLD_MANUFACTURING_COST, NEW_MANUFACTURING_COST,
    OLD_SELLING_COST, NEW_SELLING_COST,
    UPDATED_ON
  ) VALUES (
    :NEW.PRODUCT_ID,
    :OLD.MANUFACTURING_COST, :NEW.MANUFACTURING_COST,
    :OLD.SELLING_COST, :NEW.SELLING_COST,
    SYSDATE
  );
END;
/
```

EXAMPLE

```
SELECT * FROM PRODUCT_T;

UPDATE PRODUCT_T
  SET MANUFACTURING_COST = 80.00, SELLING_COST = 80.00
  WHERE PRODUCT_ID = 7222;
```

```
ORA-20003: Error: Manufacturing cost must be less than selling cost.
ORA-06512: at "SCM.TRG_PRODUCT_COST_VALIDATION", line 4
ORA-04088: error during execution of trigger 'SCM.TRG_PRODUCT_COST_VALIDATION'
2.      SET MANUFACTURING_COST = 80.00, SELLING_COST = 80.00
3.      WHERE PRODUCT_ID = 7222;
```

EXAMPLE

```
SELECT * FROM PRODUCT_T;

UPDATE PRODUCT_T
  SET MANUFACTURING_COST = 80.00, SELLING_COST = 100
  WHERE PRODUCT_ID = 7222;

SELECT * FROM PRODUCT_COST_AUDIT;
```

PRODUCT_ID	OLD_MANUFACTURING_COST	NEW_MANUFACTURING_COST	OLD_SELLING_COST	NEW_SELLING_COST	UPDATED_ON
7222	12.5	80	29.99	100	16-DEC-24

```
SELECT * FROM PRODUCT_T;

UPDATE PRODUCT_T
SET MANUFACTURING_COST = 100, SELLING_COST = 105
WHERE PRODUCT_ID = 7222;

SELECT * FROM PRODUCT_COST_AUDIT;
```

PRODUCT_ID	OLD_MANUFACTURING_COST	NEW_MANUFACTURING_COST	OLD_SELLING_COST	NEW_SELLING_COST	UPDATED_ON
7222	80	100	100	105	16-DEC-24
7222	12.5	80	29.99	100	16-DEC-24

2 rows returned in 0.02 seconds CSV Export

Create a trigger which checks that ESTIMATED_DELIEVERY_DATE is always greater than SHIPMENT_DATE. If it is, then the data in all the columns should be allowed to insert (or update)

```
CREATE OR REPLACE TRIGGER TRG_SHIPMENT_DATE_VALIDATION

BEFORE INSERT OR UPDATE ON SHIPMENT
```

```
FOR EACH ROW
BEGIN
  -- Check if the Estimated Delivery Date is greater than the Shipment Date
  IF :NEW.ESTIMATED_DELIEVERY_DATE <= :NEW.SHIPMENT_DATE THEN
    -- Raise an error if the condition is not met
    RAISE_APPLICATION_ERROR(-20006, 'Error: Estimated Delivery Date must be greater than Shipment Date.');
  END IF;
END;
/
```

EXAMPLE:

```
SELECT * FROM SHIPMENT;
UPDATE SHIPMENT
SET ESTIMATED_DELIEVERY_DATE = TO_DATE('2021-12-15', 'YYYY-MM-DD')
WHERE SHIPMENT_ID = 78901234561;
```

```
ORA-20006: Error: Estimated Delivery Date must be greater than Shipment Date.
ORA-06512: at "SCM.TRG_SHIPMENT_DATE_VALIDATION", line 5
ORA-04088: error during execution of trigger 'SCM.TRG_SHIPMENT_DATE_VALIDATION'
3. WHERE SHIPMENT_ID = 78901234561;
```

ROLES

ADMIN Role

- **Privileges:**
  - **SELECT**, **INSERT**, **UPDATE**, **DELETE** on all tables (SUPPLIER_T, RAW_MATERIAL_T, PRODUCT_T, CUSTOMER_T, ORDER_T, etc.). This role has full control over all tables and records in the system.
- **SCRIPT**

```
CREATE ROLE ADMIN;
```

GRANT SELECT, INSERT, UPDATE, DELETE ON SUPPLIER_T TO ADMIN;

GRANT SELECT, INSERT, UPDATE, DELETE ON RAW_MATERIAL_T TO ADMIN;

GRANT SELECT, INSERT, UPDATE, DELETE ON PRODUCT_T TO ADMIN;

GRANT SELECT, INSERT, UPDATE, DELETE ON CUSTOMER_T TO ADMIN;

GRANT SELECT, INSERT, UPDATE, DELETE ON ORDER_T TO ADMIN;

GRANT SELECT, INSERT, UPDATE, DELETE ON ORDER_LINE_T TO ADMIN;

GRANT SELECT, INSERT, UPDATE, DELETE ON SHIPMENT TO ADMIN;

GRANT SELECT, INSERT, UPDATE, DELETE ON WAREHOUSE_T TO ADMIN;

SUPPLIER_MANAGER Role

- **Privileges:**
    - **SELECT**, **INSERT**, **UPDATE, DELETE** on the SUPPLIER_T and SUPPLIER_CONTACT tables. This allows Supplier Managers to manage suppliers' data.
    - **SELECT** on RAW_MATERIAL_T for viewing raw material details linked to suppliers.
- **SCRIPT:**

  -- Create the Supplier Manager role

  CREATE ROLE SUPPLIER_MANAGER;

  -- Grant SELECT, INSERT, UPDATE, DELETE privileges on SUPPLIER_T table to SUPPLIER_MANAGER

  GRANT SELECT, INSERT, UPDATE, DELETE ON SUPPLIER_T TO SUPPLIER_MANAGER;

  -- Grant SELECT, INSERT, UPDATE, DELETE privileges on SUPPLIER_CONTACT table to SUPPLIER_MANAGER

  GRANT SELECT, INSERT, UPDATE, DELETE ON SUPPLIER_CONTACT TO SUPPLIER_MANAGER;

  -- Grant SELECT privilege on RAW_MATERIAL_T table to SUPPLIER_MANAGER

  GRANT SELECT ON RAW_MATERIAL_T TO SUPPLIER_MANAGER;

WAREHOUSE_MANAGER Role

- **Privileges:**

- **SELECT**, **INSERT**, **UPDATE,DELETE** on the WAREHOUSE_T and RAW_MATERIAL_T tables. Warehouse managers can manage warehouse details and stock quantities.
- **SELECT** on PRODUCT_T to view product details.

- **SCRIPT:**

-- Create the Warehouse Manager role

```
CREATE ROLE WAREHOUSE_MANAGER;

-- Grant SELECT, INSERT, UPDATE, DELETE privileges on WAREHOUSE_T table to WAREHOUSE_MANAGER

GRANT SELECT, INSERT, UPDATE, DELETE ON WAREHOUSE_T TO WAREHOUSE_MANAGER;

-- Grant SELECT, INSERT, UPDATE, DELETE privileges on RAW_MATERIAL_T table to WAREHOUSE_MANAGER

GRANT SELECT, INSERT, UPDATE, DELETE ON RAW_MATERIAL_T TO WAREHOUSE_MANAGER;

-- Grant SELECT privilege on PRODUCT_T table to WAREHOUSE_MANAGER

GRANT SELECT ON PRODUCT_T TO WAREHOUSE_MANAGER;
```

PRODUCT_MANAGER Role

- **Privileges:**
  - **SELECT**, **INSERT**, **UPDATE** on PRODUCT_T and PRODUCT_LINE_T tables to manage product data.
  - **SELECT** on QUANTITY_IN to review required raw material quantities.
- **Restrictions:**
  - No delete permissions, ensuring that product-related records cannot be removed by the PRODUCT_MANAGER
- **SCRIPT:**

```
CREATE ROLE PRODUCT_MANAGER;

GRANT SELECT, INSERT, UPDATE ON PRODUCT_T TO PRODUCT_MANAGER;

GRANT SELECT, INSERT, UPDATE ON PRODUCT_LINE_T TO PRODUCT_MANAGER;

GRANT SELECT ON QUANTITY_IN TO PRODUCT_MANAGER;

REVOKE DELETE ON PRODUCT_T FROM PRODUCT_MANAGER;
```

ORDER_MANAGER Role (with deletion restrictions)

- **Privileges:**

- **SELECT**, **INSERT**, **UPDATE** on the ORDER_T, ORDER_LINE_T, CUSTOMER_T, and SHIPMENT tables. This allows the ORDER_MANAGER to manage orders, customer information, and shipments.
- **No DELETE** on ORDER_T and CUSTOMER_T. The ORDER_MANAGER is prevented from deleting any customer or order records.

- **SCRIPT:**

```
CREATE ROLE ORDER_MANAGER;

GRANT SELECT, INSERT, UPDATE ON ORDER_T TO ORDER_MANAGER;

GRANT SELECT, INSERT, UPDATE ON ORDER_LINE_T TO ORDER_MANAGER;

GRANT SELECT, INSERT, UPDATE ON CUSTOMER_T TO ORDER_MANAGER;

GRANT SELECT, INSERT, UPDATE ON SHIPMENT TO ORDER_MANAGER;

-- Restrict deletion of Customer and Order records

REVOKE DELETE ON ORDER_T FROM ORDER_MANAGER;

REVOKE DELETE ON CUSTOMER_T FROM ORDER_MANAGER;
```