AMAZON ONLINE SHOPPING

PROJECT TITLE:	**AMAZON (RETAIL)**
COURE NUMBER AND SECTION:	**6360.002**
TEAM NUMBER:	**AMAZON-2**
TEAM MEMBERS:	**MEGHNA SINGHAL (MXS180155)** **RAHUL CHOUDHARY PAVALUR BALAKRISHNA (RXP190029)** **SAAHITH HEGDE SAAHITH HEGDE (SXS190147)**

DATA REQUIREMENTS:

- A new user can create an account on the Amazon's website.
- Customers can update their details.
- Customers can see the details of all the products available in the inventory as well as search for the products.
- The products can be supplied by multiple suppliers which are stored in the warehouses which are in-turn stocked to the Inventory.
- Customers can filter the products based on their preferences.
- Customers can add/delete items to their wish-list.
- A customer can create multiple wish-lists.
- Customers can add/delete products (cart-items) to their cart.
- Customers can place order on the items present in their cart.
- Items in an order can be delivered from multiple warehouse locations.
- Order items are delivered to the customer through a shipper.
- Customers can see the status of their orders.
- Customers can view their order history.
- Sellers can add new products to the inventory.
- Sellers can see the number of items sold on a particular date from their listings.
- Administrators can control products being sold by the seller according to Amazon's standards.
- Administrators can see the number of purchases made by customers.
- Administrators can put up special offers/discounts.

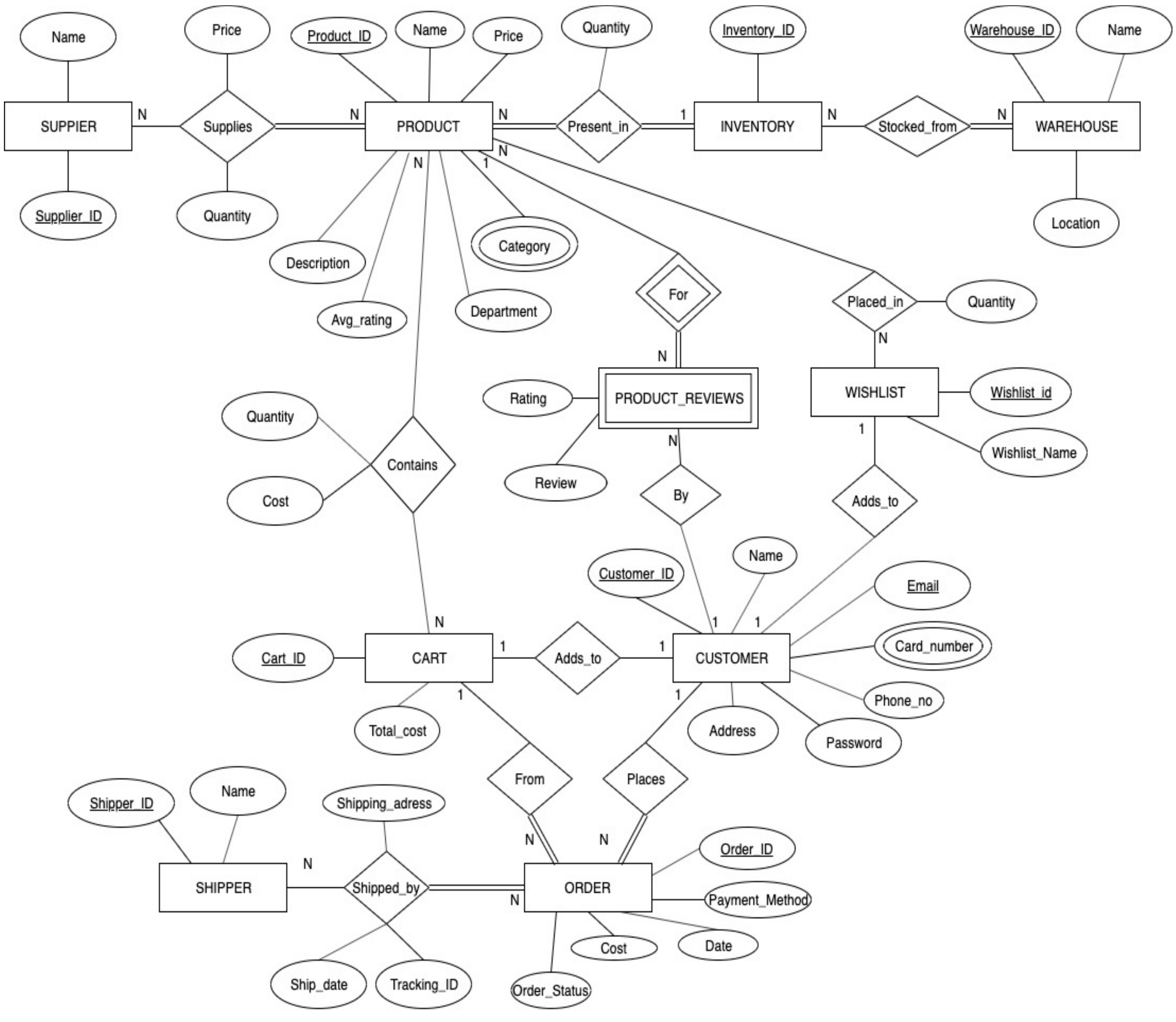
ER DIAGRAM:

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.

Assumptions for Amazon database–

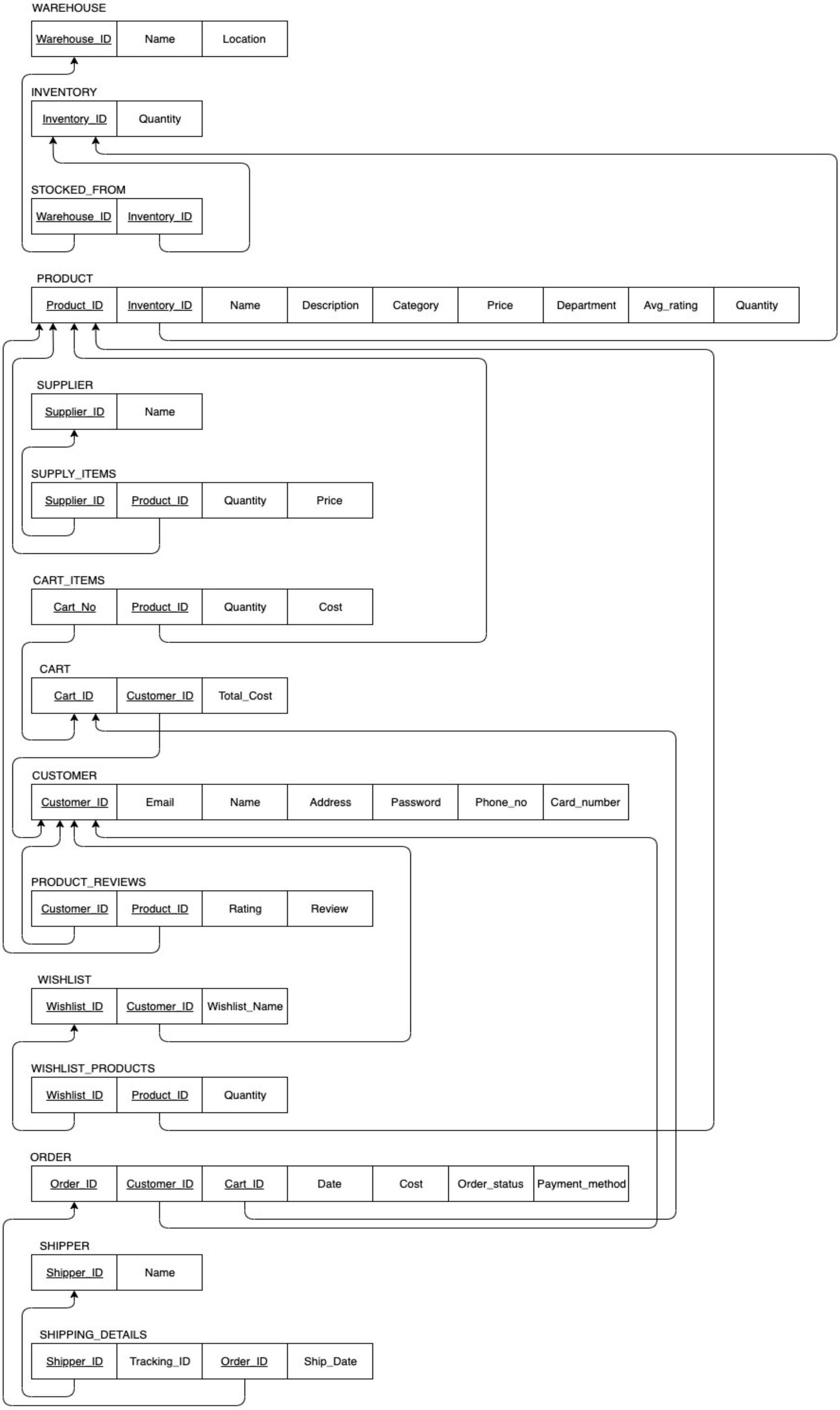
- The products in the inventory can only be stocked from the warehouse. Suppliers supply products to the warehouse.
- Customers can only place orders on the items present in the cart.

- Multiple orders can be places from the items in the cart.
- Customers can create multiple wish-lists which may or may not contain products.
- For a given order, only one customer can place it using only one payment method.
- A single shipper can ship multiple orders. Multiple parts of the order can be shipped by multiple shippers.
- A customer may or may not give a review of a product. But a single customer can give multiple reviews to multiple products.
- The products that are visible to the customers on the website, must be available at the inventory.



Amazon Online Shopping ER Diagram

MAPPING THE ER DIAGRAM TO RELATIONAL SCHEMA:



WAREHOUSE

Warehouse_ID	Name	Location

INVENTORY

Inventory_ID	Quantity

STOCKED_FROM

Warehouse_ID	Inventory_ID

PRODUCT

Product_ID	Inventory_ID	Name	Description	Category	Price	Department	Avg_rating	Quantity

SUPPLIER

Supplier_ID	Name

SUPPLY_ITEMS

Supplier_ID	Product_ID	Quantity	Price

CART_ITEMS

Cart_No	Product_ID	Quantity	Cost

CART

Cart_ID	Customer_ID	Total_Cost

CUSTOMER

Customer_ID	Email	Name	Address	Password	Phone_no	Card_number

PRODUCT_REVIEWS

Customer_ID	Product_ID	Rating	Review

WISHLIST

Wishlist_ID	Customer_ID	Wishlist_Name

WISHLIST_PRODUCTS

Wishlist_ID	Product_ID	Quantity

ORDER

Order_ID	Customer_ID	Cart_ID	Date	Cost	Order_status	Payment_method

SHIPPER

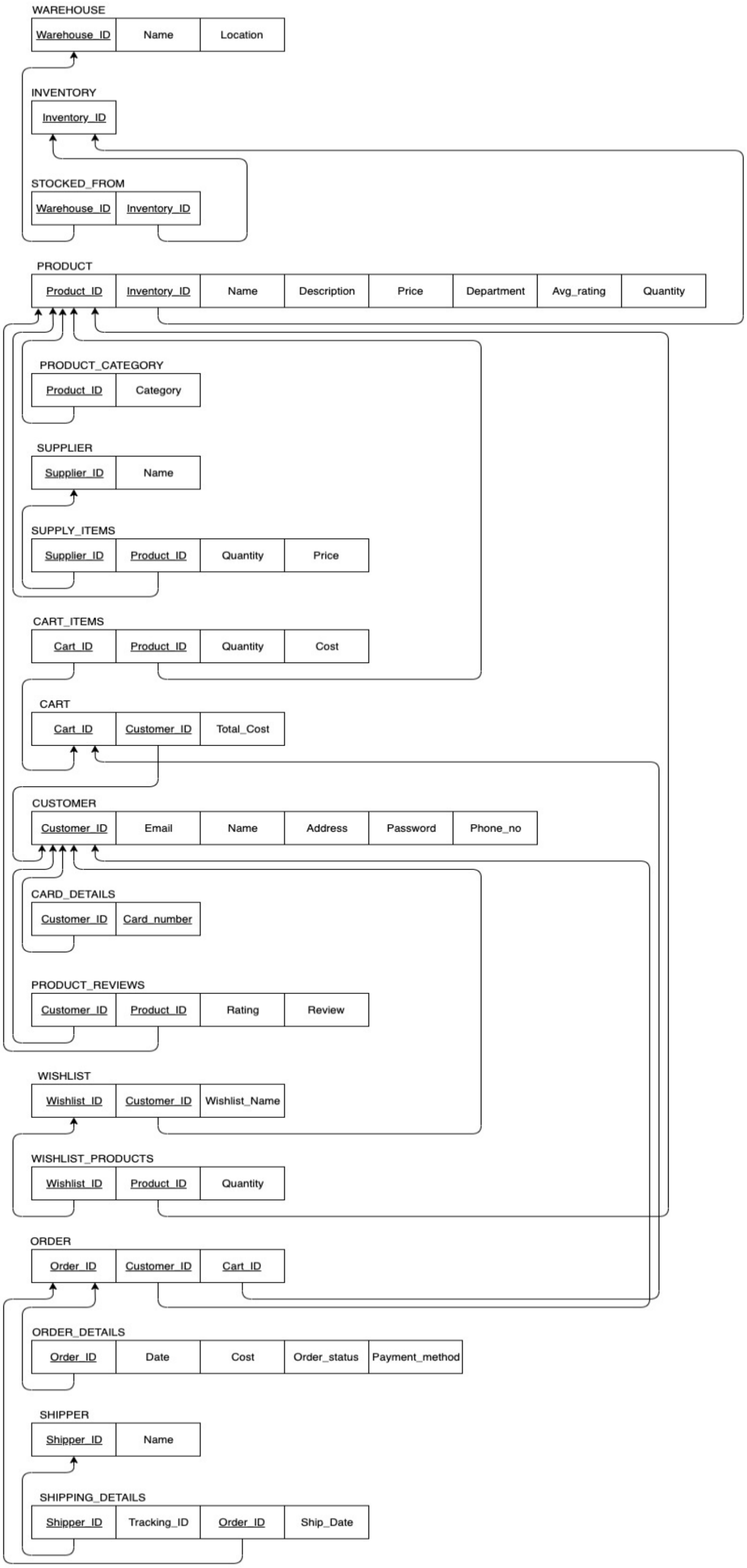
Shipper_ID	Name

SHIPPING_DETAILS

Shipper_ID	Tracking_ID	Order_ID	Ship_Date

DATABASE NORMALIZATION RULES:

- Normalization - The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations.
- Normal Form - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form.
- The mapped relational schema violates two instances of the 1st Normal Form and one instance of the 2nd Normal Form.
- 1st Normal Form Violations:
  - The 1st Normal Form disallows composite attributes, multivalued attributes and nested relations.
  - The 'category' attribute in the 'PRODUCT' relation is a multi-valued attribute.
  - To normalize this, a new relation is created for the multi-valued attribute (category) along with the primary key (product_id) of the 'PRODUCT' relation.
  - The 'card_number' attribute in the 'CUSTOMER' relation is also a multi-valued attribute. Hence, a new relation called 'CARD_DETAILS' is created.
  - This brings all the relations to the 1st Normal Form as no further violations are present.
- 2nd Normal Form Violations:
  - The 2nd Normal Form disallows a non-key attribute to be functionally determined by a part of the primary key.
  - In the 'ORDER' relation, given an Order_id, the details about the order such as the date, cost, order_status and the payment_method can be retrieved. This violates the 2nd Normal Form.
  - To normalize this, a new realtion is created with all the card details (date, cost, order_status and payment_method) along with Order_id.
  - Upon further inspection, no other relations having multiple prime attributes exist where the non-prime attribute only depend on a part of the primary key. Such as, we need both 'Product_id' and 'Cart_id' to determine the 'quantity' of that product and the 'cost' in the 'CART_ITEMS' relation. Both 'Shipper_id' and 'Order_id' is required to get the 'Tracking_id' and the 'Ship_date'.
  - This brings all the relations to the 2nd Normal Form.
- 3rd Normal Form Violations:
  - The 3rd Normal Form disallows a non-key attribute to be functionally determined by another non-key attribute (there should not be a transitive dependency).
  - In no relations, a non-key attribute can be determined by another non-key attribute.
  - This brings all the relations to the 3rd Normal Form.
- The final relational schema after normalization to 3NF is as follows:



WAREHOUSE

Warehouse_ID	Name	Location

INVENTORY

Inventory_ID

STOCKED_FROM

Warehouse_ID	Inventory_ID

PRODUCT

Product_ID	Inventory_ID	Name	Description	Price	Department	Avg_rating	Quantity

PRODUCT_CATEGORY

Product_ID	Category

SUPPLIER

Supplier_ID	Name

SUPPLY_ITEMS

Supplier_ID	Product_ID	Quantity	Price

CART_ITEMS

Cart_ID	Product_ID	Quantity	Cost

CART

Cart_ID	Customer_ID	Total_Cost

CUSTOMER

Customer_ID	Email	Name	Address	Password	Phone_no

CARD_DETAILS

Customer_ID	Card_number

PRODUCT_REVIEWS

Customer_ID	Product_ID	Rating	Review

WISHLIST

Wishlist_ID	Customer_ID	Wishlist_Name

WISHLIST_PRODUCTS

Wishlist_ID	Product_ID	Quantity

ORDER

Order_ID	Customer_ID	Cart_ID

ORDER_DETAILS

Order_ID	Date	Cost	Order_status	Payment_method

SHIPPER

Shipper_ID	Name

SHIPPING_DETAILS

Shipper_ID	Tracking_ID	Order_ID	Ship_Date

CREATING THE TABLES:

```
CREATE TABLE Warehouse (
Warehouse_id char(10) primary key,
Name varchar(50),
Location varchar(50)
);

CREATE TABLE Inventory (
Inventory_id char(10) primary key
);

CREATE TABLE Stocked_from (
Warehouse_id char(10),
Inventory_id char(10),
primary key(Warehouse_id, Inventory_id)
);

CREATE TABLE Product (
Product_id char(10) primary key,
Inventory_id char(10),
Name varchar(50),
Description varchar(100),
Price float(15),
Department char(20),
Avg_rating float(2) default 2.5,
Quantity int
);

CREATE TABLE Product_category (
Product_id char(10) primary key,
Category char(50)
);

CREATE TABLE Supplier (
Supplier_id char(10) primary key,
Name varchar(50)
);

CREATE TABLE Supply_items (
Product_id char(10),
Supplier_id char(10),
Price float(15),
Quantity int,
primary key(Product_id, Supplier_id)
);

CREATE TABLE Customer (
Customer_id char(10) primary key,
Email varchar(50),
Name varchar(50),
Address varchar(50),
Password varchar(50),
Phone_no numeric(10)
);

CREATE TABLE Cart (
Cart_id char(10),
Customer_id char(10),
Total_cost float(15),
Primary key(Cart_id, Customer_id)
);

CREATE TABLE Cart_items (
Cart_id char(10),
Product_id char(10),
Quantity int,
Cost float(15),
primary key(Cart_id, Product_id)
);

CREATE TABLE Card_details (
Customer_id char(10),
Card_number numeric(16),
primary key(Customer_id, Card_number)
);

CREATE TABLE Product_reviews (
Customer_id char(10),
Product_id char(10),
Rating float(2),
Review varchar(200),
primary key(Customer_id, Product_id)
);

CREATE TABLE Wishlist (
Wishlist_id char(10),
Customer_id char(10),
Wishlist_name varchar(50),
primary key(Wishlist_id, Customer_id)
);

CREATE TABLE Wishlist_products (
Wishlist_id char(10),
Product_id char(10),
Quantity int,
primary key(Wishlist_id, Product_id)
);

CREATE TABLE Order_tab (
Order_id char(10),
Customer_id char(10),
Cart_id char(10),
primary key(Order_id, Customer_id, Cart_id)
);

CREATE TABLE Order_details (
Order_id char(10) primary key,
Date Date,
Cost float(15),
Order_status char(20),
Payment_method char(20)
);

CREATE TABLE Shipper (
Shipper_id char(10) primary key,
Name varchar(50)
);

CREATE TABLE Shipping_details (
Shipper_id char(10),
Order_id char(10),
Tracking_id char(10),
Ship_Date Date default null,
primary key(Shipper_id, Order_id)
);
```

TRIGGERED ACTIONS ON FOREIGN KEYS:

```
ALTER TABLE Stocked_from ADD CONSTRAINT stk FOREIGN KEY(Warehouse_id) REFERENCES
Warehouse(Warehouse_id) ON DELETE CASCADE;
ALTER TABLE Stocked_from ADD CONSTRAINT stk2 FOREIGN KEY(Inventory_id) REFERENCES
Inventory(Inventory_id) ON DELETE CASCADE;

ALTER TABLE Product ADD CONSTRAINT pdt FOREIGN KEY(Inventory_id) REFERENCES
Inventory(Inventory_id) ON DELETE CASCADE;

ALTER TABLE Product_category ADD CONSTRAINT pdtc FOREIGN KEY(Product_id) REFERENCES
Product(Product_id) ON DELETE CASCADE;

ALTER TABLE Supply_items ADD CONSTRAINT sup FOREIGN KEY(Product_id) REFERENCES
Product(Product_id) ON DELETE CASCADE;
ALTER TABLE Supply_items ADD CONSTRAINT sup2 FOREIGN KEY(Supplier_id) REFERENCES
Supplier(Supplier_id) ON DELETE CASCADE;

ALTER TABLE Cart_items ADD CONSTRAINT crt FOREIGN KEY(Cart_id) REFERENCES Cart(Cart_id)
ON DELETE CASCADE;
ALTER TABLE Cart_items ADD CONSTRAINT crt2 FOREIGN KEY(Product_id) REFERENCES
Product(Product_id) ON DELETE CASCADE;

ALTER TABLE Cart ADD CONSTRAINT crt3 FOREIGN KEY(Customer_id) REFERENCES
Customer(Customer_id) ON DELETE CASCADE;

ALTER TABLE Card_details ADD CONSTRAINT crd FOREIGN KEY(Customer_id) REFERENCES
Customer(Customer_id) ON DELETE CASCADE;

ALTER TABLE Product_reviews ADD CONSTRAINT pdtr FOREIGN KEY(Customer_id) REFERENCES
Customer(Customer_id) ON DELETE CASCADE;
ALTER TABLE Product_reviews ADD CONSTRAINT pdtr2 FOREIGN KEY(Product_id) REFERENCES
Product(Product_id) ON DELETE CASCADE;

ALTER TABLE Wishlist ADD CONSTRAINT wsh FOREIGN KEY(Customer_id) REFERENCES
Customer(Customer_id) ON DELETE CASCADE;
```

```
ALTER TABLE Wishlist_products ADD CONSTRAINT wsh2 FOREIGN KEY(Wishlist_id) REFERENCES
Wishlist(Wishlist_id) ON DELETE CASCADE;
ALTER TABLE Wishlist_products ADD CONSTRAINT wsh3 FOREIGN KEY(Product_id) REFERENCES
Product(Product_id) ON DELETE CASCADE;

ALTER TABLE Order_tab ADD CONSTRAINT ord1 FOREIGN KEY(Customer_id) REFERENCES
Customer(Customer_id) ON DELETE CASCADE;
ALTER TABLE Order_tab ADD CONSTRAINT ord2 FOREIGN KEY(Cart_id) REFERENCES Cart(Cart_id)
ON DELETE CASCADE;

ALTER TABLE Order_details ADD CONSTRAINT ord3 FOREIGN KEY(Order_id) REFERENCES
Order_tab(Order_id) ON DELETE CASCADE;

ALTER TABLE Shipping_details ADD CONSTRAINT shp FOREIGN KEY(Shipper_id) REFERENCES
Shipper(Shipper_id) ON DELETE CASCADE;
ALTER TABLE Shipping_details ADD CONSTRAINT shp2 FOREIGN KEY(Order_id) REFERENCES
Order_tab(Order_id) ON DELETE CASCADE;
```

TRIGGERS AND STORED PROCEDURES:

STORED PROCEDURE #1:

A procedure to filter the products based on the user's preferences:

The input from the user such as the price range, the minimum ratings and the type (category) of the product is taken as the input and all the products that satisfy these conditions are displayed.

```
Create or replace procedure am_filtered_data
 (
  in_low_price in NUMBER,
  in_high_price in NUMBER,
  in_rating in FLOAT,
  in_product_type in VARCHAR
)
AS
id  product.product_id%TYPE;
name product.Name%TYPE;
price product.Price%TYPE;
desc product.Description%TYPE;
rating product.avg_rating%TYPE;
cursor c_product_details is
   select p.product_id, p.name, p.price, p.Description, p.avg_rating
from product p, product_category pc where
p.product_id= pc.product_id and
 pc.category = in_product_type
```

```
and p.price between in_low_price and in_high_price
and p.avg_rating >= in_rating;

BEGIN
OPEN c_product_details;
LOOP
FETCH c_product_details into id, name, price, desc, rating
EXIT WHEN(c_product_details%NOTFOUND);
dbms_output.put_line( “Product with product id:” || id || “ ,name:” || name
|| “ ,price:” || price || “ ,description:” || desc || “ and rating:” || rating || “have been
filtered”);
 END LOOP;
CLOSE c_product_details;
EXCEPTION
WHEN NO_DATA_FOUND THEN
dbms_output.put_line(“No products found”);
END am_filtered_data;
```

STORED PROCEDURE #2:

A procedure to be used by the administrators to get the details of all the products bought by customers:

The procedure takes a product category as an input and displays the information such as all the customer name, customer_id’s, product name, quantity, etc., which gives the administrators a summary of all the products bought by the customers.

```
Create or replace procedure am_product_summary
 (
  in_product_type in VARCHAR
)
AS
product_name product.name;
no_of_products  product.quantity;
product_type product_category.category%TYPE;
customer_id customer.customer_id%TYPE;
customer_name customer.name%TYPE;
order_id order.order_id%TYPE;
cursor c_products is
   select p.name, pc.category, COUNT(p.product_id) as product_count,
c.customer_id, c.customer_name, o.order_id
from product p, product_reviews pr, product_category pc, customer c , order o where
p.product_id=pr.product_id
and pr.customer_id=c.customer_id
and o.customer_id=c.customer_id and
p.product_id= pc.product_id and
 pc.category = in_product_type;

BEGIN
```

```
OPEN c_products;
LOOP
FETCH c_products into product_name, product_type, no_of_products, customer_id,
customer_name, order_id;
EXIT WHEN(c_products%NOTFOUND);
dbms_output.put_line(“Customer:” || customer_name || “customer_id:”||
“customer_id” || “order_id:” || order_id  || “bought product:” || name || “ category:”||
product_type || “in” || no_of_products || “quantity” );
 END LOOP;
CLOSE c_products;
EXCEPTION
WHEN NO_DATA_FOUND THEN
dbms_output.put_line(“No products found”);
END am_product_summary;
```

TRIGGER #1:

A trigger to calculate the total price of an order placed by the customer:

```
create or replace TRIGGER before_order
BEFORE INSERT ON
ORDER
FOR EACH ROW
BEGIN
Insert into Order_details (order_id, cost)
select o.order_id, SUM(p.price) from cart_items c, product p, order_details o
where p.product_id=c.product_id
and o.cart_id=c.cart_id and c.cart_id=NEW.cart_id;
END;
```

TRIGGER #2:

A trigger to calculate the average ratings of a product whenever a user gives a rating to that product:

```
create or replace TRIGGER avg_rating
AFTER INSERT ON
PRODUCT_REVIEWS
FOR EACH ROW
DECLARE
average product.avg_rating%TYPE;
product product.product_id%TYPE;
BEGIN
select pr.product_id, AVG(pr.rating)
into product, average
from product_reviews pr
where pr.product_id=NEW.product_id;

Update product set avg_rating= average where product_id = product;
```

```
END;
```

TRIGGER #3:

A trigger to reduce the total quantity of a product when the customer places an order:

```
CREATE or REPLACE TRIGGER suppliers
AFTER INSERT ON Order
FOR EACH ROW
DECLARE
product_quant INT
BEGIN
SELECT ci.quantity INTO product_count
FROM cart_items ci, cart c, order o, product p
WHERE o.order_status = “delivered” AND p.product_id = ci.product_id AND
ci.cart_id = c.cart_id AND c.cart_id = o.cart_id AND o.order_id = NEW.order_id;
UPDATE Product p2 set p2.quantity = p2.quantity – product_quant
WHERE p2.product_id = ci.product_id;
```