CS3425-FINALPROJECT- AmmineniPuja

1. Create a Small Data Set (20 points)

To help you understand the domain, please design a small but complete set of the testing data based on the requirement. Your database should work with any data set. It should NOT be designed such that it only works with the sample data set.

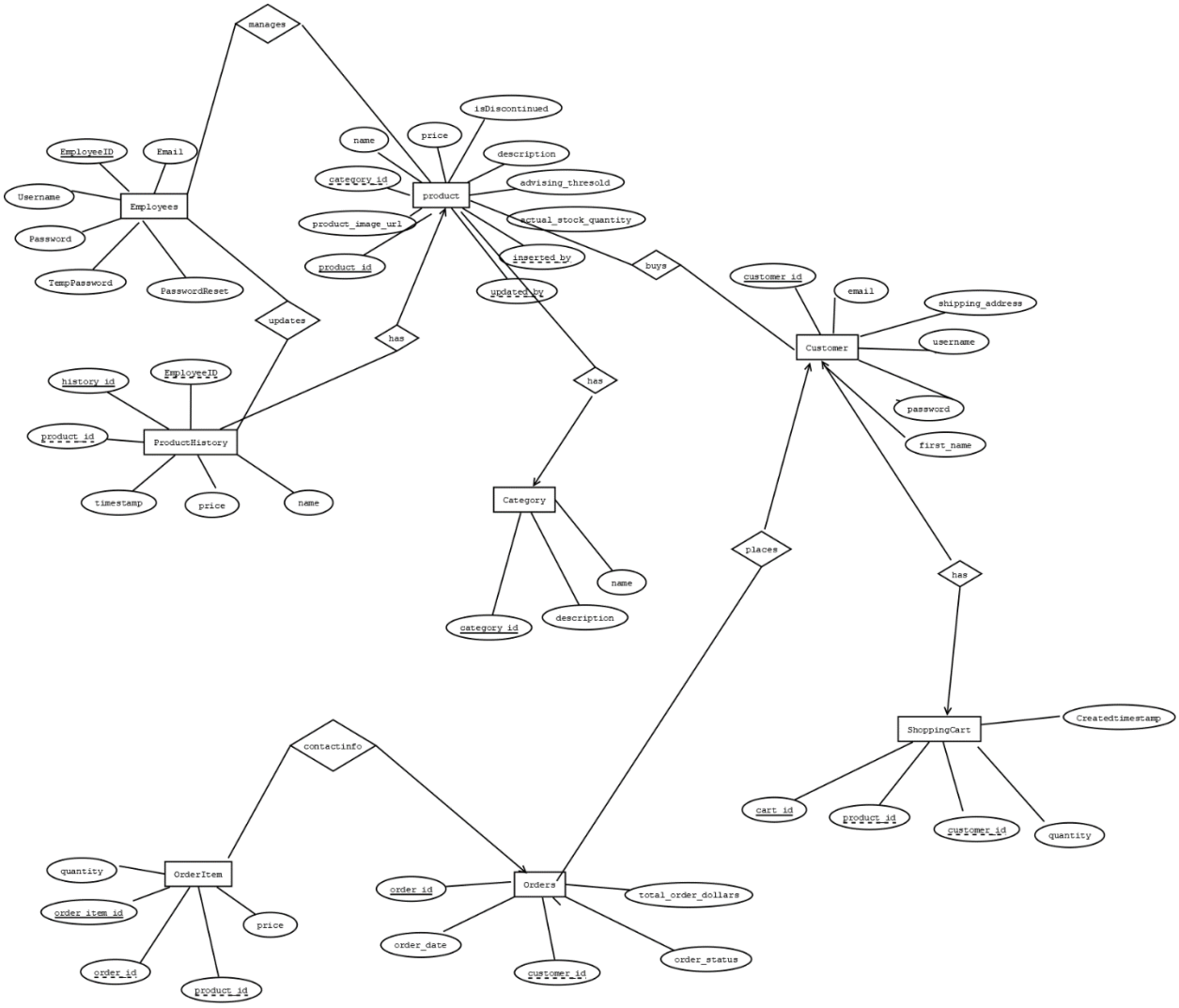
2 employees	‘PUJA’,’puja@gmail.com’,’Puja123’ ‘devi’,’devi@gmail.com’,’devi@123’
3 customers	1,'puja', 'pass1', 'pujaa', 'ammineni', 'ammineni@gmail.com', '111 Main St' 2,'rk', 'pass2', 'rkp', 'pagidimarri', 'pagidimarri@gmail.com', '222 Elm St' 3,'mahi', 'pass3', 'mahesh', 'gajjala', 'gajjala@gmail.com', '333 Oak St'
4 categories	'Beauty Products', 'Cosmetics and beauty-related items' 'clothes', 'boys and girls' 'Decor', ’Home Décor’ 'Mobiles', 'All Type available'
10 products (2-3 product in each category)	('Moisturizing Cream', 'Hydrating cream for daily use', 1, 29.99, 50, 100, 1, 1, M.jpg, 0); ('Anti-Aging Serum', 'Reduces fine lines and wrinkles', 1, 49.99, 30, 75, 2, 2, A.jpg, 0); ('Matte Lipstick - Red', 'Vibrant red lipstick', 2, 14.99, 40, 90, 1, 1, M.jpg, 0); ('Volumizing Mascara', 'Adds volume to lashes', 2, 19.99, 25, 60, 3, 3, V.jpg, 0); ('Rose Water Toner', 'Hydrating facial toner', 1, 24.99, 35, 80, 4, 4,R.jpg, 0); ('Eyeshadow Palette', 'Neutral tones eyeshadow palette', 2, 39.99, 20, 50, 5, 5, E.jpg, 0); ('Hydrating Face Mask', 'Moisturizing face mask', 1, 9.99, 60, 120, 6, 6, H.jpg, 0); ('Facial Cleanser', 'Gentle facial cleanser for daily use', 1, 19.99, 45, 100, 7, 7, F.jpg, 0); ('Hair Serum', 'Nourishing serum for hair', 3, 27.99, 50, 110, 8, 8, H.jpg, 0); (‘Hair oil’,’serum’,1,23,44,77,1,1,I.jpg,0);
Price changing history	2023-11-08 22:47:01 Moisturing Cream $29.99->$100
2 shopping carts	1 2023-11-09 10:00:00 1 1 2 2 2023-11-09 11:00:00 2 3 1
10 orders	Ex: Execute: > select * from Orders + ------------ + --------------- + -------------- + ----------------- + ---------- + | order_id | customer_id | order_date | order_status | total_order_dollars | + ------------ + --------------- + -------------- + ----------------- + ---------- + | 1 | 1 | 2023-11-09 12:00:00 | Pending | 119.97 | | 2 | 2 | 2023-11-09 12:30:00 | Processing | 240.50 | | 3 | 3 | 2023-11-09 13:00:00 | Shipped | 75.25 | | NULL | NULL | NULL | NULL | NULL | + ------------ + --------------- + -------------- + ----------------- + ---------- + 4 rows Execute:

```
> select * from OrderItem

+ ------------------ + ------------- + --------------- + ------------- + ----------
+
| order_item_id      | order_id      | product_id      | quantity      | price
|
+ ------------------ + ------------- + --------------- + ------------- + ----------
+
| 1                  | 1             | 1               | 3             | 49.99
|
| 2                  | 2             | 3               | 2             | 39.98
|
| 3                  | 3             | 2               | 1             | 19.99
|
| NULL               | NULL          | NULL            | NULL          | NULL
|
+ ------------------ + ------------- + --------------- + ------------- + ----------
+
4 rows
```

2. E-R Model and Relational Schema (100 points)

a) (30p) Construct an E-R diagram representing the conceptual design of the database. Be sure to identify primary keys, relationship, cardinalities, etc. (Refer to Assignment 2)


manages
EmployeeID
Email
Username
Employees
Password
TempPassword
PasswordReset
updates
history_id
EmployeeID
product_id
ProductHistory
timestamp
price
name
has
isDiscontinued
name
price
description
category_id
advising_thresold
product
product_image_url
actual_stock_quantity
product_id
inserted_by
updated_by
has
buys
Category
category_id
description
name
customer_id
email
shipping_address
username
Customer
password
first_name
places
has
ShoppingCart
Createdtimestamp
cart_id
product_id
customer_id
quantity
contactinfo
quantity
OrderItem
order_item_id
price
order_id
product_id
order_id
Orders
total_order_dollars
order_date
customer_id
order_status

b) Relational Schema

1- Employees Table

Attributes:
EmployeeID (Primary Key)
Username
Email
Password
TempPassword
PasswordReset

2-Category Table

Attributes:
category_id(Primary Key)
name
description

3-product Table

Attributes:
product_id (Primary Key)
name
description
Price
advising_threshold
actual_stock_quantity
inserted_by(Foreign Key referencing Employees)
updated_by(Foreign Key referencing Employees)
product_image_url
isDiscontinued
CategoryID (Foreign Key referencing Category)

4-Customers Table

Attributes:
customer_id (Primary Key)
username
password
first_name
last_name
email
shipping_address

5-ShoppingCart Table

Attributes:
cart_id (Primary Key)
quantity
Createdtimestamp
customer_id (Foreign Key referencing Customer)
product_id (Foreign Key referencing product)

6-Orders Table
Attributes:
order_id(Primary Key)
customer_id(Foreign Key referencing Customer)
order_date
order_status
total_order_dollars

7-OrderItem Table
Attributes:
order_item_id (Primary Key)
order_id (Foreign Key referencing ShoppingCarts)
product_id (Foreign Key referencing Products)
quantity
price

8-ProductHistory Table
Attributes:
history_id (Primary Key)
price
name
EmployeeID (Foreign Key referencing employee)
product_id (Foreign Key referencing Products)
timestamp DATETIME

c)

```
drop table Employees;

CREATE TABLE Employees(
 EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
 Username VARCHAR(255) NOT NULL,
 Email VARCHAR(255) NOT NULL,
 Password CHAR(64) NOT NULL,
 TempPassword CHAR(64) ,
 PasswordReset INT DEFAULT 1);

drop table Cateogory
 CREATE TABLE Category (
   category_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   description VARCHAR(255)
  );
drop table product

  CREATE TABLE product (
   product_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(255),
   description TEXT,
   category_id INT,
   price DECIMAL(10, 2),
   advising_threshold INT,
   actual_stock_quantity INT,
   inserted_by INT,
   updated_by INT,
   product_image_url BLOB,
   isDiscontinued BOOLEAN,
   FOREIGN KEY (category_id) REFERENCES Category (category_id),
   FOREIGN KEY (inserted_by) REFERENCES Employee (EmployeeID),
   FOREIGN KEY (updated_by) REFERENCES Employee (EmployeeID)
);

drop table Customer;
CREATE TABLE Customer (
   customer_id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(255)  NOT NULL,
   password VARCHAR(60) NOT NULL,
   first_name VARCHAR(255),
   last_name VARCHAR(255),
   email VARCHAR(255) NOT NULL,
   shipping_address VARCHAR(255)
);
```

```
drop table Shoppingcart;

CREATE TABLE ShoppingCart (
        cart_id INT AUTO_INCREMENT PRIMARY KEY,
   CreatedTimestamp TIMESTAMP,
   customer_id INT,
   product_id INT,
   quantity INT,
   FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
   FOREIGN KEY (product_id) REFERENCES product(product_id)
);

drop table orders;

CREATE TABLE Orders (
   order_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   customer_id INT NOT NULL,
   order_date DATETIME NOT NULL,
   order_status VARCHAR(50) NOT NULL,
   total_order_dollars DECIMAL(10, 2) NOT NULL,
   FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);


drop table OrderItem;

CREATE TABLE OrderItem (
   order_item_id INT AUTO_INCREMENT PRIMARY KEY,
   order_id INT,
   product_id INT,
   quantity INT NOT NULL,
   price DECIMAL(10, 2),
   FOREIGN KEY (order_id) REFERENCES Orders(order_id),
   FOREIGN KEY (product_id) REFERENCES product(product_id)
);




drop table ProductHistory;
```

```
CREATE TABLE ProductHistory (
   history_id INT AUTO_INCREMENT PRIMARY KEY,
   EmployeeID INT,
   product_id INT,
    price INT,
    name varchar(20),
   timestamp DATETIME,
   FOREIGN KEY (product_id) REFERENCES product(product_id),
   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID));
```

Outputs:
Execute:
> select * from Employees

```
+ --------------- + ------------- + ---------- + ------------- + ----------------- + ------------
------ +
| EmployeeID      | Username      | Email      | Password      | TempPassword      |
PasswordReset      |
+ --------------- + ------------- + ---------- + ------------- + ----------------- + ------------
------ +
| 1               | PUJA          | Puja@gmail.com |
e80714e9c9059cf8f341dc4489e1c8a7f376db64406f09a30a664adabd2a96a0 |
Temporary1        | 1                 |
| 2               | DEVI          | Devi@gmail.com |
5fb1fafc04cac831a83b8e816eda3a05eca0434cf5511f7fea3eeb2309a6901f |
Temporary1        | 1                 |
| NULL            | NULL          | NULL       | NULL          | NULL              | NULL
|
+ --------------- + ------------- + ---------- + ------------- + ----------------- + ------------
------ +
3 rows
```

Execute:
> select * from Category

```
+ ----------------- + ---------- + ----------------- +
| category_id       | name       | description       |
+ ----------------- + ---------- + ----------------- +
| 1                 | Beauty     | All brands available |
| 2                 | clothes    | boys and girls    |
| 3                 | decaration | All Home Decors available |
| 4                 | Mobiles    | All Type available |
| NULL              | NULL       | NULL              |
+ ----------------- + ---------- + ----------------- +
5 rows
```

Execute:
> select * from product

```
+ --------------- + --------- + ---------------- + ---------------- + ---------- + ---------------
--------- + ------------------------ + ---------------- + --------------- + ---------------------
-- + ------------------- +
| product_id      | name      | description      | category_id      | price      |
advising_threshold      | actual_stock_quantity      | inserted_by      | updated_by
| product_image_url      | isDiscontinued      |
+ --------------- + --------- + ---------------- + ---------------- + ---------- + ---------------
--------- + ------------------------ + ---------------- + --------------- + ---------------------
-- + ------------------- +
| 1             | Moisturizing Cream | Hydrating cream for daily use | 1               |
29.99      | 50                  | 100                    | 1              | 1              | ...
| 0                |
| 2             | Anti-Aging Serum | Reduces fine lines and wrinkles | 1               |
49.99      | 30                  | 75                     | 2              | 2              | ...
| 0                |
| 3             | Matte Lipstick - Red | Vibrant red lipstick | 2               | 14.99      | 40
| 90                     | 1              | 1              | ...                | 0                |
| 4             | Volumizing Mascara | Adds volume to lashes | 2               | 19.99      |
25                   | 60                     | 2              | 1              | ...                  | 0
|
| 5             | Rose Water Toner | Hydrating facial toner | 1               | 24.99      | 35
| 80                     | 2              | 2              | ...                | 0                |
| 6             | Eyeshadow Palette | Neutral tones eyeshadow palette | 2               |
39.99      | 20                  | 50                     | 2              | 1              | ...
| 0                |
| 7             | Hydrating Face Mask | Moisturizing face mask | 1               | 9.99
| 60                   | 120                    | 1              | 2              | ...                  | 0
|
| 8             | Facial Cleanser | Gentle facial cleanser for daily use | 1               |
19.99      | 45                  | 100                    | 2              | 1              | ...
| 0                |
| NULL          | NULL      | NULL           | NULL           | NULL       | NULL
| NULL                   | NULL           | NULL           | NULL                 | NULL
|
```

+ --------------- + --------- + ---------------- + ---------------- + ---------- + --------------
--------- + ------------------------ + ---------------- + --------------- + --------------------
-- + ------------------- +
9 rows

Execute:
> select * from Customer

+ ---------------- + ------------ + ------------ + --------------- + -------------- + ---------
- + --------------------- +
| customer_id | username | password | first_name | last_name |
email | shipping_address |
+ ---------------- + ------------ + ------------ + --------------- + -------------- + ---------
- + --------------------- +
| 1 | puja | pass1 | pujaa | ammineni |
ammineni@gmail.com | 111 Main St |
| 2 | rk | pass2 | rkp | pagidimarri |
pagidimarri@gmail.com | 222 Elm St |
| 3 | mahi | pass3 | mahesh | gajjala |
gajjala@gmail.com | 333 Oak St |
| NULL | NULL | NULL | NULL | NULL | NULL
| NULL |
+ ---------------- + ------------ + ------------ + --------------- + -------------- + ---------
- + --------------------- +
4 rows

Execute:
> select * from ShoppingCart

+ ------------ + --------------------- + ---------------- + --------------- + ------------- +
| cart_id | CreatedTimestamp | customer_id | product_id | quantity |
+ ------------ + --------------------- + ---------------- + --------------- + ------------- +
1	2023-11-09 10:00:00	1	1	2
2	2023-11-09 11:00:00	2	3	1
NULL	NULL	NULL	NULL	NULL
+ ------------ + --------------------- + ---------------- + --------------- + ------------- +
3 rows

Execute:
> select * from Orders

```
+ ------------ + ---------------- + --------------- + ----------------- + ----------------------
- +
| order_id      | customer_id      | order_date      | order_status      |
total_order_dollars      |
+ ------------ + ---------------- + --------------- + ----------------- + ----------------------
- +
| 1             | 1                | 2023-11-09 12:00:00 | Pending           | 119.97                 |
| 2             | 2                | 2023-11-09 12:30:00 | Processing        | 240.50
|
| 3             | 3                | 2023-11-09 13:00:00 | Shipped           | 75.25                  |
| NULL          | NULL             | NULL            | NULL              | NULL
|
+ ------------ + ---------------- + --------------- + ----------------- + ----------------------
- +
4 rows
```

Execute:
> select * from OrderItem

```
+ ------------------ + ------------- + ---------------- + ------------- + ---------- +
| order_item_id      | order_id      | product_id       | quantity      | price      |
+ ------------------ + ------------- + ---------------- + ------------- + ---------- +
| 1                  | 1             | 1                | 3             | 49.99      |
| 2                  | 2             | 3                | 2             | 39.98      |
| 3                  | 3             | 2                | 1             | 19.99      |
| NULL               | NULL          | NULL             | NULL          | NULL       |
+ ------------------ + ------------- + ---------------- + ------------- + ---------- +
4 rows
```

Execute:
> select * from ProductHistory

```
+ --------------- + --------------- + --------------- + ---------- + --------- + -------------- +
| history_id      | EmployeeID      | product_id      | price      | name      | timestamp
|
+ --------------- + --------------- + --------------- + ---------- + --------- + -------------- +
| 1               | 1               | 1               | 50         | Product Updated | 2023-11-09
14:00:00 |
| 2               | 2               | 2               | 75         | Price Change | 2023-11-09 15:00:00
|
| 4               | 1               | 2               | 100        | Name Change | 2023-11-09
16:00:00 |
| 5               | 2               | 1               | 80         | Description Modified | 2023-11-09
18:00:00 |
| NULL            | NULL            | NULL            | NULL       | NULL      | NULL
|
+ --------------- + --------------- + --------------- + ---------- + --------- + -------------- +
5 rows
```

3. Stored Procedures and Triggers (40 points)

Tip: Keep all SQL statements in one file **createPSM.sql.** Make sure you keep the statements in the right order and include the drop statements so the scripts can be run multiple times to recreate all the triggers and procedures.

a)

1) Procedure restock_product (): update the product's quantity with the new restocking amount.

```
drop procedure CreateEmployee;

DELIMITER //
CREATE PROCEDURE CreateEmployee(
 username1 VARCHAR(255),
 email1 VARCHAR(255),
 password1 VARCHAR(255)
)
BEGIN
 SET @temppassword = 'Temporary1';
 SET @hashedpassword = SHA2(password1, 256);

 INSERT INTO Employees (Username, Email, Password, TempPassword)
 VALUES (username1, email1, @hashedpassword, @temppassword);
 SET @new_employee_id = LAST_INSERT_ID();

UPDATE Employees
 SET PasswordReset = 1
 WHERE EmployeeID = @new_employee_id;

END;//
DELIMITER;
```

2)

```
drop procedure insert_category
DELIMITER //
CREATE PROCEDURE insert_category(
   IN categoryName VARCHAR(100),
   IN categoryDescription VARCHAR(255)
)
BEGIN
   INSERT INTO Category (name, description)
   VALUES (categoryName, categoryDescription);
END //
DELIMITER ;
```

3)

```
drop procedure insert_product;
DELIMITER //
CREATE PROCEDURE insert_product(
   IN p_name VARCHAR(255),
   IN p_description TEXT,
   IN p_category_id INT,
   IN p_price DECIMAL(10, 2),
   IN p_advising_threshold INT,
   IN p_actual_stock_quantity INT,
   IN p_inserted_by INT,
        IN p_updated_by INT,
   IN p_product_image_url BLOB,
   IN p_isDiscontinued BOOLEAN
)
BEGIN
   INSERT INTO product (name, description, category_id, price,
advising_threshold, actual_stock_quantity, inserted_by,updated_by,
product_image_url, isDiscontinued)
   VALUES (p_name, p_description, p_category_id, p_price,
p_advising_threshold, p_actual_stock_quantity, p_inserted_by,p_updated_by,
p_product_image_url, p_isDiscontinued);
END//
DELIMITER ;
```

4)

```
drop procedure update_product_price;

DELIMITER //
CREATE PROCEDURE update_product_price(
   IN productID INT,
   IN newPrice DECIMAL(10, 2)
)
BEGIN
   UPDATE product
   SET price = newPrice
   WHERE product_id = id;
END //
DELIMITER ;
```

5)

```
drop procedure restock_product;

DELIMITER //
CREATE PROCEDURE restock_product(
   IN id INT,
   IN restockAmount INT
)
BEGIN
   UPDATE product
   SET actual_stock_quantity = actual_stock_quantity + restockAmount
   WHERE product_id = id;
END //
DELIMITER ;
```

b) (10p) Create functions to insert order and order items

1) Function insert_order(): create a new order and return the new order id
2) Function insert_order_item(): which will insert a new row in the exist order, and update the remaining stock accordingly. Please note: in Phase1, for simplicity we are assuming there are enough stocking. In the second phase, we will check and won't allow the order be placed if the product ran out of stock.

1)

```
drop function insert_order;
DELIMITER //

CREATE FUNCTION insert_order(
    custID INT,
   oDate DATETIME,
   oStatus VARCHAR(50),
   oTotal DECIMAL(10, 2)
)
RETURNS INT
BEGIN
   DECLARE newOrderID INT;

   INSERT INTO Orders (customer_id, order_date, order_status,
total_order_dollars)
   VALUES (custID, oDate, oStatus, oTotal);

   SET newOrderID = LAST_INSERT_ID();

   RETURN newOrderID;
END //

DELIMITER ;
```

2)

```
drop function add_order_item_custom;
DELIMITER //


CREATE FUNCTION add_order_item_custom(
   orderID_custom INT,
   productID_custom INT,
   itemQuantity_custom INT,
   itemPrice_custom DECIMAL(10, 2)
)
RETURNS INT
BEGIN
   DECLARE rowsAffected_custom INT;

   INSERT INTO OrderItem (order_id, product_id, quantity, price)
   VALUES (orderID_custom, productID_custom, itemQuantity_custom,
itemPrice_custom);
SET rowsAffected_custom = ROW_COUNT();

   -- Adjust the stock by reducing the purchased quantity.
   UPDATE product
   SET actual_stock_quantity = actual_stock_quantity - itemQuantity_custom
   WHERE product_id = productID_custom;

   RETURN rowsAffected_custom;
END //
```

b) (10p) Create triggers to record product update history.

1) After insert, insert the action in the product history

```
drop trigger product_insert_history;
DELIMITER //
CREATE TRIGGER product_insert_history
AFTER INSERT ON product
FOR EACH ROW
BEGIN
INSERT INTO ProductHistory(EmployeeID, product_id, price, name, timestamp)
VALUES (NEW.updated_by, NEW.product_id, NEW.price, NEW.name,
NOW());
END;
//
```

2) After update, insert the action in the product history

```
DROP TRIGGER IF EXISTS restock_product;
DROP TRIGGER IF EXISTS price;

CREATE TRIGGER restock_product
AFTER UPDATE ON product
FOR EACH ROW
BEGIN
  IF OLD.actual_stock_quantity != NEW.actual_stock_quantity THEN
    INSERT INTO ProductHistory (product_id, name, price, quantity, timestamp)
    VALUES (NEW.product_id, NEW.name, NEW.price, NEW.actual_stock_quantity, NOW());
  END IF;
END;
//
CREATE TRIGGER price
AFTER UPDATE ON product
FOR EACH ROW
BEGIN
  IF NEW.price != OLD.price THEN
    INSERT INTO ProductHistory (product_id, name, price, quantity, ChangeTimestamp)
    VALUES (NEW.product_id, NEW.name, NEW.price, NEW.actual_stock_quantity,
NOW());
  END IF;
END;
//
```

3) Before update, raise SQL error to reject the update if prod id is changed. Ex: Use the recommend 45000 as the user defined state, but assign a relevant a message SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = ' The prod id is not allowed to be changed';

```
drop trigger reject_id_change;
DELIMITER //
CREATE TRIGGER reject_id_change
 BEFORE UPDATE ON product
 FOR EACH ROW
 BEGIN
    IF OLD.product_id != NEW.product_id THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product ID modification is not
            allowed';
 END IF;
END;
//
```

4.

```
Drop trigger before_product_deletion;
DELIMITER //
CREATE TRIGGER before_product_deletion
BEFORE DELETE ON product
FOR EACH ROW
BEGIN
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product deletion is prohibited';
END;
//
```

4. Insert the Small Set of Data (40 points)

.

a)

```
call CreateEmployee('PUJA','Puja@gmail.com','Puja555');
call CreateEmployee('DEVI','Devi@gmail.com','devi777');
select * from Employees;
```

```
call insert_category('Beauty','All brands available');
call insert_category('clothes','boys and girls');
call insert_category('decaration','All Home Decors available');
call insert_category('Mobiles','All Type available');
select * from Category;
```

```
call insert_product('Moisturizing Cream', 'Hydrating cream for daily use', 1, 29.99, 50,
100, 1, 1, 'imp.jpg', 0);
call insert_product('Anti-Aging Serum', 'Reduces fine lines and wrinkles', 1, 49.99, 30,
75, 2, 2, 'inp.jpg', 0);
call insert_product('Matte Lipstick - Red', 'Vibrant red lipstick', 2, 14.99, 40, 90, 1, 1,
'lip.jpg', 0);
call insert_product('Volumizing Mascara', 'Adds volume to lashes', 2, 19.99, 25, 60, 2, 1,
'mascara.jpg', 0);
call insert_product('Rose Water Toner', 'Hydrating facial toner', 1, 24.99, 35, 80, 2,2,
'hyd.jpg', 0);
call insert_product('Eyeshadow Palette', 'Neutral tones eyeshadow palette', 2, 39.99,
20, 50, 2, 1, 'ton.jpg', 0);
call insert_product('Hydrating Face Mask', 'Moisturizing face mask', 1, 9.99, 60, 120, 1,2,
'face.jpg', 0);
call insert_product('Facial Cleanser', 'Gentle facial cleanser for daily use', 1, 19.99, 45,
100, 2,1, 'cleanser.jpg', 0);
select * from product;
```

b) Password should be stored as hash value using **sha-256 algorithm**.

Execute:

> select * from Employees

```
+ --------------- + ------------- + ---------- + ------------- + ----------------- + ------------------ +
| EmployeeID      | Username      | Email      | Password      | TempPassword      |
PasswordReset      |
+ --------------- + ------------- + ---------- + ------------- + ----------------- + ------------------ +
| 1               | PUJA          | Puja@gmail.com |
e80714e9c9059cf8f341dc4489e1c8a7f376db64406f09a30a664adabd2a96a0 | Temporary1
| 1               |
| 2               | DEVI          | Devi@gmail.com |
5fb1fafc04cac831a83b8e816eda3a05eca0434cf5511f7fea3eeb2309a6901f | Temporary1
| 1               |
| NULL            | NULL          | NULL       | NULL          | NULL              | NULL              |
+ --------------- + ------------- + ---------- + ------------- + ----------------- + ------------------ +
3 rows
```

C)When the order is placed, the product stock quantity should be updated with an update statement. Be sure to include the update statements.

```
1 11 Moisturizing Cream 29.99 2023-11-08 21:50:46
2 12 Hair Conditioner 19.99 2023-11-08 21:50:46
3 13 Lip Balm 14.99 2023-11-08 21:50:46
4 14 Eyeshadow Palette 22.99 2023-11-08 21:50:46
5 15 Facial Serum 27.99 2023-11-08 21:50:46
6 16 Nail Polish Set 19.99 2023-11-08 21:50:47
7 16 Cleansing Oil 32.99 2023-11-08 21:50:47
8 18 Body Lotion 26.99 2023-11-08 21:50:47
9 19 Sunscreen SPF 50 18.99 2023-11-08 21:50:47
10 20 Makeup Brush Set 23.99 2023-11-08 21:50:47
 11 11 Moisturizing Cream 29.99 2023-11-08 21:50:46
  12 16 Cleansing Oil 32.99 2023-11-08 21:50:47
```

5. Generate the Report: **genReport.sql** (20 points)

Write the SQL statements to generate the reports.
See section "Functions for employee 4) Genereate Reports"

a) View the historic prices for a given product.

```
   select name, price, Createdtimestamp
 from ProductHistory where productID = 1
order by Createdtimestamp
```

Execute:

```
> select name, price, timestamp
from ProductHistory
where product_id = 1
order by timestamp
```

```
+ --------- + ---------- + -------------- +
| name      | price      | timestamp      |
+ --------- + ---------- + -------------- +
| Product Updated | 50         | 2023-11-09 14:00:00 |
| Description Modified | 80         | 2023-11-09 18:00:00 |
+ --------- + ---------- + -------------- +
2 rows
```

b) Find the highest and lowest price within a given period of time.

```
SELECT MAX(price) AS maximum, MIN(price) AS minimum
FROM ProductHistory
WHERE timestamp >= '2023-11-09 14.00.00' AND timestamp <= '2023-11-09 18:00:00';
```

Execute:

```
> SELECT MAX(price) AS maximum_price, MIN(price) AS minimum_price
FROM ProductHistory
WHERE timestamp >= '2023-11-09 14.00.00' AND timestamp <= '2023-11-09 18:00:00';
```

```
+ ------------------ + ------------------ +
| maximum_price      | minimum_price      |
+ ------------------ + ------------------ +
| 100                | 50                 |
+ ------------------ + ------------------ +
1 rows
```

c) List how many qualities sold for each product within a specified time frame. You may ignore the ones that has not be sold.
Execute:

```
> SELECT
   p.product_id,
   p.name AS name,
   COUNT(oi.product_id) AS quantity_sold
FROM product p
LEFT JOIN OrderItem oi ON p.product_id = oi.product_id
LEFT JOIN Orders o ON oi.order_id = o.order_id
WHERE o.order_date >= '2023-11-09 12:00:00' AND o.order_date <= '2023-11-09 13:00:00'
GROUP BY p.product_id, p.name
HAVING quantity_sold > 0
LIMIT 0, 10
```

```
+ --------------- + --------- + ------------------ +
| product_id      | name      | quantity_sold      |
+ --------------- + --------- + ------------------ +
| 1               | Moisturizing Cream | 1                |
| 3               | Matte Lipstick - Red | 1                |
| 2               | Anti-Aging Serum | 1                |
+ --------------- + --------- + ------------------ +
3 rows
```

d) Identify products below the restocking threshold and the quantity needed to reach the threshold.

```
SELECT
   p.product_id,
   p.name AS ProductName,
   p.advising_threshold AS RestockingThreshold,
   (p.advising_threshold - p.actual_stock_quantity) AS QuantityNeeded
FROM product p
WHERE p.actual_stock_quantity < p.advising_threshold;
```

```
+ --------------- + ---------------- + ------------------------ + ------------------- +
1 rows
```

//ALL PRODUCTS ARE STOCKED

createtable.sql

```
drop table Employees;
CREATE TABLE Employees(
 EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
 Username VARCHAR(255) NOT NULL,
 Email VARCHAR(255) NOT NULL,
 Password CHAR(64) NOT NULL,
 TempPassword CHAR(64) ,
 PasswordReset INT DEFAULT 1);

drop table Category;
 CREATE TABLE Category (
    category_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description VARCHAR(255)
  );

  drop table product;
   CREATE TABLE product (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    description TEXT,
    category_id INT,
    price DECIMAL(10, 2),
    advising_threshold INT,
    actual_stock_quantity INT,
    inserted_by INT,
    updated_by INT,
    product_image_url BLOB,
    isDiscontinued BOOLEAN,
    FOREIGN KEY (category_id) REFERENCES Category (category_id),
    FOREIGN KEY (inserted_by) REFERENCES Employees(EmployeeID),
    FOREIGN KEY (updated_by) REFERENCES Employees(EmployeeID)
);

drop table Customer;
CREATE TABLE Customer (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255)  NOT NULL,
    password VARCHAR(60) NOT NULL,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
```

```
    email VARCHAR(255) NOT NULL,
    shipping_address VARCHAR(255)
);

drop table ShoppingCart;
CREATE TABLE ShoppingCart (
            cart_id INT AUTO_INCREMENT PRIMARY KEY,
    CreatedTimestamp TIMESTAMP,
    customer_id INT,
    product_id INT,
    quantity INT,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (product_id) REFERENCES product(product_id)
);

drop table Orders;
CREATE TABLE Orders (
    order_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date DATETIME NOT NULL,
    order_status VARCHAR(50) NOT NULL,
    total_order_dollars DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

drop table OrderItem;
CREATE TABLE OrderItem (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT NOT NULL,
    price DECIMAL(10, 2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES product(product_id)
);


drop table ProductHistory;
CREATE TABLE ProductHistory (
    history_id INT AUTO_INCREMENT PRIMARY KEY,
    EmployeeID INT,
    product_id INT,
     price INT,
     name varchar(20),
    timestamp DATETIME,
    FOREIGN KEY (product_id) REFERENCES product(product_id),
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID));
```

CreatePSM.SQL

```
drop procedure CreateEmployee;

DELIMITER //
CREATE PROCEDURE CreateEmployee(
 username1 VARCHAR(255),
 email1 VARCHAR(255),
 password1 VARCHAR(255)
)
BEGIN
 SET @temppassword = 'Temporary1';
 SET @hashedpassword = SHA2(password1, 256);

 INSERT INTO Employees (Username, Email, Password, TempPassword)
 VALUES (username1, email1, @hashedpassword, @temppassword);
 SET @new_employee_id = LAST_INSERT_ID();

UPDATE Employees
 SET PasswordReset = 1
 WHERE EmployeeID = @new_employee_id;
END;//
DELIMITER;

drop procedure insert_category;
DELIMITER //
CREATE PROCEDURE insert_category(
   IN categoryName VARCHAR(100),
   IN categoryDescription VARCHAR(255)
)
BEGIN
   INSERT INTO Category (name, description)
   VALUES (categoryName, categoryDescription);
END //
DELIMITER ;
```

```
drop procedure insert_product;
    DELIMITER //
    CREATE PROCEDURE insert_product(
      IN p_name VARCHAR(255),
      IN p_description TEXT,
      IN p_category_id INT,
      IN p_price DECIMAL(10, 2),
      IN p_advising_threshold INT,
      IN p_actual_stock_quantity INT,
      IN p_inserted_by INT,
        IN p_updated_by INT,
      IN p_product_image_url BLOB,
      IN p_isDiscontinued BOOLEAN
    )
    BEGIN
      INSERT INTO product (name, description, category_id, price, advising_threshold,
actual_stock_quantity, inserted_by,updated_by, product_image_url, isDiscontinued)
      VALUES (p_name, p_description, p_category_id, p_price, p_advising_threshold,
p_actual_stock_quantity, p_inserted_by,p_updated_by, p_product_image_url, p_isDiscontinued);
    END//
    DELIMITER ;

    drop procedure insert_customer;
    DELIMITER //

    CREATE PROCEDURE insert_customer(
      IN custUsername VARCHAR(255),
      IN custPassword VARCHAR(60),
      IN custFirstName VARCHAR(255),
      IN custLastName VARCHAR(255),
      IN custEmail VARCHAR(255),
      IN custShippingAddress VARCHAR(255)
    )

      drop procedure CreateEmployee;

    DELIMITER //
    CREATE PROCEDURE CreateEmployee(
     username1 VARCHAR(255),
     email1 VARCHAR(255),
     password1 VARCHAR(255)
    )
    BEGIN
     SET @temppassword = 'Temporary1';
     SET @hashedpassword = SHA2(password1, 256);
```

```
 INSERT INTO Employees (Username, Email, Password, TempPassword)
 VALUES (username1, email1, @hashedpassword, @temppassword);
 SET @new_employee_id = LAST_INSERT_ID();

UPDATE Employees
 SET PasswordReset = 1
 WHERE EmployeeID = @new_employee_id;

END;//
DELIMITER;
drop procedure insert_category
DELIMITER //
CREATE PROCEDURE insert_category(
   IN categoryName VARCHAR(100),
   IN categoryDescription VARCHAR(255)
)
BEGIN
   INSERT INTO Category (name, description)
   VALUES (categoryName, categoryDescription);
END //
DELIMITER ;

drop procedure insert_product;
DELIMITER //
CREATE PROCEDURE insert_product(
   IN p_name VARCHAR(255),
   IN p_description TEXT,
   IN p_category_id INT,
   IN p_price DECIMAL(10, 2),
   IN p_advising_threshold INT,
   IN p_actual_stock_quantity INT,
   IN p_inserted_by INT,
     IN p_updated_by INT,
   IN p_product_image_url BLOB,
   IN p_isDiscontinued BOOLEAN
)
BEGIN
   INSERT INTO product (name, description, category_id, price, advising_threshold,
actual_stock_quantity, inserted_by,updated_by, product_image_url, isDiscontinued)
   VALUES (p_name, p_description, p_category_id, p_price, p_advising_threshold,
p_actual_stock_quantity, p_inserted_by,p_updated_by, p_product_image_url,
p_isDiscontinued);
END//
DELIMITER ;

drop procedure update_product_price;
```

```
DELIMITER //
CREATE PROCEDURE update_product_price(
    IN productID INT,
    IN newPrice DECIMAL(10, 2)
)
BEGIN
    UPDATE product
    SET price = newPrice
    WHERE product_id = id;
END //
DELIMITER ;

drop procedure restock_product;

DELIMITER //
CREATE PROCEDURE restock_product(
    IN id INT,
    IN restockAmount INT
)
BEGIN
    UPDATE product
    SET actual_stock_quantity = actual_stock_quantity + restockAmount
    WHERE product_id = id;
END //
DELIMITER ;




 drop function insert_order;
DELIMITER //

CREATE FUNCTION insert_order(
     custID INT,
    oDate DATETIME,
    oStatus VARCHAR(50),
    oTotal DECIMAL(10, 2)
)
RETURNS INT
BEGIN
    DECLARE newOrderID INT;
```

```
    INSERT INTO Orders (customer_id, order_date, order_status, total_order_dollars)
    VALUES (custID, oDate, oStatus, oTotal);

    SET newOrderID = LAST_INSERT_ID();

    RETURN newOrderID;
  END //

  DELIMITER ;


            drop function add_order_item_custom;
  DELIMITER //


  CREATE FUNCTION add_order_item_custom(
    orderID_custom INT,
    productID_custom INT,
    itemQuantity_custom INT,
    itemPrice_custom DECIMAL(10, 2)
  )
  RETURNS INT
  BEGIN
    DECLARE rowsAffected_custom INT;

              INSERT INTO OrderItem (order_id, product_id, quantity, price)
              VALUES (orderID_custom, productID_custom, itemQuantity_custom,
            itemPrice_custom);
            SET rowsAffected_custom = ROW_COUNT();

              -- Adjust the stock by reducing the purchased quantity.
              UPDATE product
              SET actual_stock_quantity = actual_stock_quantity - itemQuantity_custom
              WHERE product_id = productID_custom;

              RETURN rowsAffected_custom;
            END //



drop trigger product_insert_history;
  DELIMITER //
  CREATE TRIGGER product_insert_history
 AFTER INSERT ON product
 FOR EACH ROW
 BEGIN
```

```
    INSERT INTO ProductHistory(EmployeeID, product_id, price, name, timestamp)
    VALUES (NEW.updated_by, NEW.product_id, NEW.price, NEW.name,
     NOW());
     END;
      //

      DROP TRIGGER IF EXISTS restock_product;
     DROP TRIGGER IF EXISTS price;


     CREATE TRIGGER restock_product
     AFTER UPDATE ON product
     FOR EACH ROW
     BEGIN
        IF OLD.actual_stock_quantity != NEW.actual_stock_quantity THEN
           INSERT INTO ProductHistory (product_id, name, price, quantity, timestamp)
           VALUES (NEW.product_id, NEW.name, NEW.price, NEW.actual_stock_quantity, NOW());
        END IF;
     END;
     //
     CREATE TRIGGER price
     AFTER UPDATE ON product
     FOR EACH ROW
     BEGIN
        IF NEW.price != OLD.price THEN
           INSERT INTO ProductHistory (product_id, name, price, quantity, ChangeTimestamp)
           VALUES (NEW.product_id, NEW.name, NEW.price, NEW.actual_stock_quantity,
NOW());
        END IF;
     END;
     //



   drop trigger reject_id_change;
   DELIMITER //
   CREATE TRIGGER reject_id_change
    BEFORE UPDATE ON product
    FOR EACH ROW
    BEGIN
       IF OLD.product_id != NEW.product_id THEN
          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product ID modification is not
               allowed';
    END IF;
END;
//
```

```
Drop trigger before_product_deletion;
DELIMITER //
CREATE TRIGGER before_product_deletion
BEFORE DELETE ON product
FOR EACH ROW
BEGIN
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product deletion is prohibited';
END;
//




CREATE PROCEDURE insert_shopping_cart_item(
   IN cartCustomerID INT,
   IN cartProductID INT,
   IN cartQuantity INT,
   IN cartCreatedTimestamp TIMESTAMP
)
BEGIN
   INSERT INTO ShoppingCart (customer_id, product_id, quantity, CreatedTimestamp)
   VALUES (cartCustomerID, cartProductID, cartQuantity, cartCreatedTimestamp);
END //
DELIMITER //

CREATE PROCEDURE insert_order(
   IN orderCustomerID INT,
   IN orderDate DATETIME,
   IN orderStatus VARCHAR(50),
   IN totalOrderAmount DECIMAL(10, 2)
)
BEGIN
   INSERT INTO Orders (customer_id, order_date, order_status, total_order_dollars)
   VALUES (orderCustomerID, orderDate, orderStatus, totalOrderAmount);
END //

DELIMITER ;
DELIMITER //

CREATE PROCEDURE insert_order_item(
   IN orderId INT,
```

```
    IN productId INT,
    IN itemQuantity INT,
    IN itemPrice DECIMAL(10, 2)
)
BEGIN
    INSERT INTO OrderItem (order_id, product_id, quantity, price)
    VALUES (orderId, productId, itemQuantity, itemPrice);
END //

DELIMITER ;




DELIMITER //

CREATE PROCEDURE insert_product_history(
    IN empID INT,
    IN prodID INT,
    IN prodPrice INT,
    IN prodName VARCHAR(20),
    IN timestamp DATETIME
)
BEGIN
    INSERT INTO ProductHistory (EmployeeID, product_id, price, name, timestamp)
    VALUES (empID, prodID, prodPrice, prodName, timestamp);
END //
DELIMETER;//
```

Insertdata.sql

```
call CreateEmployee('PUJA','Puja@gmail.com','Puja555');
call CreateEmployee('DEVI','Devi@gmail.com','devi777');
select * from Employees;

call insert_category('Beauty','All brands available');
call insert_category('clothes','boys and girls');
call insert_category('decaration','All Home Decors available');
call insert_category('Mobiles','All Type available');
select * from Category;
call insert_product('Moisturizing Cream', 'Hydrating cream for daily use', 1, 29.99,
50, 100, 1, 1, 'imp.jpg', 0);
call insert_product('Anti-Aging Serum', 'Reduces fine lines and wrinkles', 1, 49.99,
30, 75, 2, 2, 'inp.jpg', 0);
call insert_product('Matte Lipstick - Red', 'Vibrant red lipstick', 2, 14.99, 40, 90, 1, 1,
'lip.jpg', 0);
call insert_product('Volumizing Mascara', 'Adds volume to lashes', 2, 19.99, 25, 60, 2,
1, 'mascara.jpg', 0);
call insert_product('Rose Water Toner', 'Hydrating facial toner', 1, 24.99, 35, 80, 2,2,
'hyd.jpg', 0);
call insert_product('Eyeshadow Palette', 'Neutral tones eyeshadow palette', 2, 39.99,
20, 50, 2, 1, 'ton.jpg', 0);
call insert_product('Hydrating Face Mask', 'Moisturizing face mask', 1, 9.99, 60, 120,
1,2, 'face.jpg', 0);
call insert_product('Facial Cleanser', 'Gentle facial cleanser for daily use', 1, 19.99,
45, 100, 2,1, 'cleanser.jpg', 0);
select * from product;
INSERT INTO Customer (username, password, first_name, last_name, email,
shipping_address)
 VALUES
 ('puja', 'pass1', 'pujaa', 'ammineni', 'ammineni@gmail.com', '111 Main St'),
 ('rk', 'pass2', 'rkp', 'pagidimarri', 'pagidimarri@gmail.com', '222 Elm St'),
 ('mahi', 'pass3', 'mahesh', 'gajjala', 'gajjala@gmail.com', '333 Oak St');
 select * from Customer;
```

```
 CALL insert_shopping_cart_item(1, 1, 2, '2023-11-09 10:00:00');
CALL insert_shopping_cart_item(2, 3, 1, '2023-11-09 11:00:00');
select * from ShoppingCart;
CALL insert_order(1, '2023-11-09 12:00:00', 'Pending', 119.97);
CALL insert_order(2, '2023-11-09 12:30:00', 'Processing', 240.50);
CALL insert_order(3, '2023-11-09 13:00:00', 'Shipped', 75.25);
CALL insert_order_item(1, 1, 3, 49.99);
CALL insert_order_item(2, 3, 2, 39.98);
CALL insert_order_item(3, 2, 1, 19.99);

select * from OrderItem;
CALL insert_product_history(1, 1, 50, 'Product Updated', '2023-11-09 14:00:00');
CALL insert_product_history(2, 2, 75, 'Price Change', '2023-11-09 15:00:00');
CALL insert_product_history(1, 2, 100, 'Name Change', '2023-11-09 16:00:00');
CALL insert_product_history(1, 3, 60, 'Stock Update', '2023-11-09 17:00:00');
CALL insert_product_history(2, 1, 80, 'Description Modified', '2023-11-09 18:00:00');
select * from ProductHistory;
select * from Employees;
select * from Category;
 select * from Products; select * from Customers;
 select * from ShoppingCarts;
select * from OrderS;
select * from OrderItem;
select * from ProductHistory;
SELECT MAX(price) AS maximum, MIN(price) AS minimum
FROM ProductHistory
WHERE timestamp >= '2023-11-09 14.00.00' AND timestamp <= '2023-11-09
18:00:00';
SELECT MAX(price) AS maximum_price, MIN(price) AS minimum_price
FROM ProductHistory;
SELECT
   p.product_id,
   p.name AS name,
   COUNT(oi.product_id) AS quantity_sold
FROM product p
LEFT JOIN OrderItem oi ON p.product_id = oi.product_id
LEFT JOIN Orders o ON oi.order_id = o.order_id
WHERE o.order_date >= '2023-11-09 12:00:00' AND o.order_date <= '2023-11-09
13:00:00'
GROUP BY p.product_id, p.name
HAVING quantity_sold > 0
LIMIT 0, 10;

SELECT
   p.product_id,
   p.name AS ProductName,
```

```
    p.advising_threshold AS RestockingThreshold,
    (p.advising_threshold - p.actual_stock_quantity) AS QuantityNeeded
FROM product p
WHERE p.actual_stock_quantity < p.advising_threshold;
select*from product;
```