Trigger:

```
DELIMITER $$
CREATE TRIGGER after_comment_delete
AFTER DELETE ON Comments
FOR EACH ROW
BEGIN
   -- Decrement NumberOfReviews in Products table when a comment is deleted
   UPDATE Products
   SET NumberOfReviews = NumberOfReviews - 1
   WHERE ProductId = OLD.ProductId;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER after_comment_insert
AFTER INSERT ON Comments
FOR EACH ROW
BEGIN
   -- Increment NumberOfReviews in Products table when a comment is added
   UPDATE Products
   SET NumberOfReviews = NumberOfReviews + 1
   WHERE ProductId = NEW.ProductId;
END$$
DELIMITER ;
```

Stored Procedure:

```
DELIMITER $$

CREATE PROCEDURE search_product_procedure(
  IN p_search VARCHAR(255),
  IN p_priceRange VARCHAR(50),
  IN p_brand VARCHAR(255),
  IN p_productType VARCHAR(50),
  IN p_rating DECIMAL(2,1),
  IN p_sortBy VARCHAR(50),
  IN p_sortOrder VARCHAR(10)
)
BEGIN
  DECLARE v_query TEXT;
  DECLARE v_order_field VARCHAR(50);
  DECLARE v_order_dir VARCHAR(10);

  -- Base query
  SET v_query = 'SELECT Products.*, Brands.BrandName
                FROM Products
                JOIN Brands ON Products.BrandId = Brands.BrandId
                WHERE 1=1';

  -- Conditions based on parameters
  IF p_search IS NOT NULL AND p_search <> '' THEN
    SET v_query = CONCAT(v_query, ' AND Products.ProductName LIKE "%', p_search, '%"');
  END IF;

  IF p_brand IS NOT NULL AND p_brand <> '' THEN
    SET v_query = CONCAT(v_query, ' AND Brands.BrandName = "', p_brand, '"');
  END IF;

  IF p_priceRange = 'under25' THEN
    SET v_query = CONCAT(v_query, ' AND Price < 25');
  ELSEIF p_priceRange = '25to50' THEN
    SET v_query = CONCAT(v_query, ' AND Price BETWEEN 25 AND 50');
  ELSEIF p_priceRange = '50to100' THEN
    SET v_query = CONCAT(v_query, ' AND Price BETWEEN 50 AND 100');
  ELSEIF p_priceRange = '100andAbove' THEN
    SET v_query = CONCAT(v_query, ' AND Price > 100');
  END IF;

  IF p_productType IS NOT NULL AND p_productType <> '' THEN
    SET v_query = CONCAT(v_query, ' AND Category = "', p_productType, '"');
```

```
  END IF;

  IF p_rating > 0 THEN
    SET v_query = CONCAT(v_query, ' AND Rating >= ', p_rating);
  END IF;

  IF p_sortBy = 'Price' THEN
    SET v_order_field = 'Price';
  ELSEIF p_sortBy = 'Rating' THEN
    SET v_order_field = 'Rating';
  ELSE
    SET v_order_field = NULL;
  END IF;

  IF p_sortOrder = 'desc' THEN
    SET v_order_dir = 'DESC';
  ELSE
    SET v_order_dir = 'ASC';
  END IF;

  IF v_order_field IS NOT NULL THEN
    SET v_query = CONCAT(v_query, ' ORDER BY ', v_order_field, ' ', v_order_dir);
  END IF;

  SET @q = v_query;
  -- Prepare and execute the dynamic query
  PREPARE stmt FROM @q;
  EXECUTE stmt;
  DEALLOCATE PREPARE stmt;

  -- Second query:
  -- Count how many products matched the previous criteria.
  SET v_query = REPLACE(v_query, 'SELECT Products.*, Brands.BrandName', 'SELECT
COUNT(*) AS total_products');

  SET v_query = SUBSTRING_INDEX(v_query, ' ORDER BY ', 1);

  SET @q = v_query;
  -- Prepare and execute the dynamic query
  PREPARE stmt FROM @q;
  EXECUTE stmt;
  DEALLOCATE PREPARE stmt;

END$$
```

```
DELIMITER ;
```

CONSTRAINT:

```
CREATE TABLE `WishListItem` (
  `RecordId` int NOT NULL AUTO_INCREMENT,
  `UserId` varchar(50) DEFAULT NULL,
  `ProductId` int DEFAULT NULL,
  PRIMARY KEY (`RecordId`),
  KEY `ProductId` (`ProductId`),
  KEY `FK_WishListItem_UserId` (`UserId`),
  CONSTRAINT `FK_WishListItem_UserId` FOREIGN KEY (`UserId`) REFERENCES `Users`
(`UserId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `WishListItem_ibfk_1` FOREIGN KEY (`UserId`) REFERENCES `Users` (`UserId`),
  CONSTRAINT `WishListItem_ibfk_2` FOREIGN KEY (`ProductId`) REFERENCES `Products`
(`ProductId`)

CREATE TABLE `Video` (
  `VideoId` int NOT NULL AUTO_INCREMENT,
  `ProductId` int DEFAULT NULL,
  `VideoLink` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`VideoId`),
  KEY `ProductId` (`ProductId`),
  CONSTRAINT `Video_ibfk_1` FOREIGN KEY (`ProductId`) REFERENCES `Products`
(`ProductId`)
)

CREATE TABLE `Users` (
  `UserId` varchar(50) NOT NULL,
  `UserName` varchar(50) NOT NULL,
  `Email` varchar(100) NOT NULL,
  `Age` int DEFAULT NULL,
  `Gender` varchar(20) DEFAULT NULL,
  `Password` varchar(255) NOT NULL,
  `SkinType` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`UserId`)
)

CREATE TABLE `Products` (
  `ProductId` int NOT NULL,
  `ProductName` varchar(255) NOT NULL,
  `Category` varchar(50) DEFAULT NULL,
  `Price` decimal(8,2) DEFAULT NULL,
  `BrandId` int DEFAULT NULL,
  `UsageFrequency` varchar(30) DEFAULT NULL,
  `SkinType` varchar(20) DEFAULT NULL,
```

```
  `NumberOfReviews` int DEFAULT NULL,
  `Rating` decimal(3,2) DEFAULT NULL,
  `GenderTarget` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`ProductId`),
  KEY `BrandId` (`BrandId`),
  CONSTRAINT `Products_ibfk_1` FOREIGN KEY (`BrandId`) REFERENCES `Brands` (`BrandId`)
)

CREATE TABLE `Comments` (
  `CommentId` int NOT NULL AUTO_INCREMENT,
  `ProductId` int DEFAULT NULL,
  `UserId` varchar(50) DEFAULT NULL,
  `Date` date DEFAULT NULL,
  `Rating` decimal(3,2) DEFAULT NULL,
  `CommentContent` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`CommentId`),
  KEY `idx_comments_product_id` (`ProductId`),
  KEY `FK_Comments_UserId` (`UserId`),
  CONSTRAINT `Comments_ibfk_1` FOREIGN KEY (`ProductId`) REFERENCES `Products`
(`ProductId`),
  CONSTRAINT `Comments_ibfk_2` FOREIGN KEY (`UserId`) REFERENCES `Users` (`UserId`),
  CONSTRAINT `FK_Comments_UserId` FOREIGN KEY (`UserId`) REFERENCES `Users`
(`UserId`) ON DELETE CASCADE ON UPDATE CASCADE
)


CREATE TABLE `Bundles` (
  `BundledId` int NOT NULL AUTO_INCREMENT,
  `UserId` varchar(50) DEFAULT NULL,
  `BundleName` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`BundledId`),
  KEY `FK_Bundles_UserId` (`UserId`),
  CONSTRAINT `FK_Bundles_UserId` FOREIGN KEY (`UserId`) REFERENCES `Users` (`UserId`)
ON DELETE CASCADE ON UPDATE CASCADE
)
CREATE TABLE `Bundle` (
  `RecordId` int NOT NULL AUTO_INCREMENT,
  `UserId` varchar(50) DEFAULT NULL,
  `ProductId` int DEFAULT NULL,
  `BundledId` int DEFAULT NULL,
  PRIMARY KEY (`RecordId`),
  KEY `UserId` (`UserId`),
  KEY `ProductId` (`ProductId`),
  CONSTRAINT `Bundle_ibfk_1` FOREIGN KEY (`UserId`) REFERENCES `Users` (`UserId`),
  CONSTRAINT `Bundle_ibfk_2` FOREIGN KEY (`ProductId`) REFERENCES `Products`
(`ProductId`)
```

```
)


CREATE TABLE `BundleProducts` (
  `BundleProductId` int NOT NULL AUTO_INCREMENT,
  `BundledId` int DEFAULT NULL,
  `ProductId` int DEFAULT NULL,
  PRIMARY KEY (`BundleProductId`),
  KEY `BundledId` (`BundledId`),
  KEY `ProductId` (`ProductId`),
  CONSTRAINT `BundleProducts_ibfk_1` FOREIGN KEY (`BundledId`) REFERENCES `Bundles`
(`BundledId`),
  CONSTRAINT `BundleProducts_ibfk_2` FOREIGN KEY (`ProductId`) REFERENCES `Products`
(`ProductId`)
)

CREATE TABLE `Brands` (
  `BrandId` int NOT NULL,
  `BrandCountry` varchar(255) DEFAULT NULL,
  `BrandName` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`BrandId`)
)
```

Transaction:

In backend code:
GitHub/fa24-cs411-team081-QueryMasters/myproject/cosmetics-backend/bundle_back.js

```
 await executeQuery(connection, `SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE
READ`);
```