CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(255),

SupplierID INT,

CategoryID INT,

Unit VARCHAR(255),

Price DECIMAL(10, 2)

);

INSERT INTO Products

(ProductID, ProductName, SupplierID, CategoryID, Unit, Price)

VALUES

(1, 'Chais', 1, 1, '10 boxes x 20 bags', 18.00),

(2, 'Chang', 1, 1, '24 - 12 oz bottles', 19.00),

(3, 'Aniseed Syrup', 1, 2, '12 - 550 ml bottles', 10.00),

(4, 'Chef Anton''s Cajun Seasoning', 2, 2, '48 - 6 oz jars', 22.00),

(5, 'Chef Anton''s Gumbo Mix', 2, 2, '36 boxes', 21.35),

(6, 'Grandma''s Boysenberry Spread', 3, 2, '12 - 8 oz jars', 25.00),

(7, 'Uncle Bob''s Organic Dried Pears', 3, 7, '12 - 1 lb pkgs.', 30.00),

(8, 'Northwoods Cranberry Sauce', 3, 2, '12 - 12 oz jars', 40.00),

(9, 'Mishi Kobe Niku', 4, 6, '18 - 500 g pkgs.', 97.00),

(10, 'Ikura', 4, 8, '12 - 200 ml jars', 31.00),

(11, 'Queso Cabrales', 5, 4, '1 kg pkg.', 21.00),

(12, 'Queso Manchego La Pastora', 5, 4, '10 - 500 g pkgs.', 38.00),

(13, 'Konbu', 6, 8, '2 kg box', 6.00),

(14, 'Tofu', 6, 7, '40 - 100 g pkgs.', 23.25),

(15, 'Genen Shouyu', 6, 2, '24 - 250 ml bottles', 15.50),

(16, 'Pavlova', 7, 3, '32 - 500 g boxes', 17.45),

(17, 'Alice Mutton', 7, 6, '20 - 1 kg tins', 39.00),

(18, 'Carnarvon Tigers', 7, 8, '16 kg pkg.', 62.50),

(19, 'Teatime Chocolate Biscuits', 8, 3, '10 boxes x 12 pieces', 9.20),

(20, 'Sir Rodney''s Marmalade', 8, 3, '30 gift boxes', 81.00),

(21, 'Sir Rodney''s Scones', 8, 3, '24 pkgs. x 4 pieces', 10.00),

(22, 'Gustaf''s Knäckebröd', 9, 5, '24 - 500 g pkgs.', 21.00),

(23, 'Tunnbröd', 9, 5, '12 - 250 g pkgs.', 9.00),

(24, 'Guaraná Fantástica', 10, 1, '12 - 355 ml cans', 4.50),

(25, 'NuNuCa Nuß-Nougat-Creme', 11, 3, '20 - 450 g glasses', 14.00),

(26, 'Gumbär Gummibärchen', 11, 3, '100 - 250 g bags', 31.23),

(27, 'Schoggi Schokolade', 11, 3, '100 - 100 g pieces', 43.90),

(28, 'Rössle Sauerkraut', 12, 7, '25 - 825 g cans', 45.60),

(29, 'Thüringer Rostbratwurst', 12, 6, '50 bags x 30 sausages', 123.79),

(30, 'Nord-Ost Matjeshering', 13, 8, '10 - 200 g glasses', 25.89),

(31, 'Gorgonzola Telino', 14, 4, '12 - 100 g pkgs.', 12.50),

(32, 'Mascarpone Fabioli', 14, 4, '24 - 200 g pkgs.', 32.00),

(33, 'Geitost', 15, 4, '500 g', 2.50),

(34, 'Sasquatch Ale', 16, 1, '24 - 12 oz bottles', 14.00),

(35, 'Steeleye Stout', 16, 1, '24 - 12 oz bottles', 18.00),

(36, 'Inlagd Sill', 17, 8, '24 - 250 g jars', 19.00),

(37, 'Gravad lax', 17, 8, '12 - 500 g pkgs.', 26.00),

(38, 'Côte de Blaye', 18, 1, '12 - 75 cl bottles', 263.50),

(39, 'Chartreuse verte', 18, 1, '750 cc per bottle', 18.00),

(40, 'Boston Crab Meat', 19, 8, '24 - 4 oz tins', 18.40),

(41, 'Jack''s New England Clam Chowder', 19, 8, '12 - 12 oz cans', 9.65),

(42, 'Singaporean Hokkien Fried Mee', 20, 5, '32 - 1 kg pkgs.', 14.00),

(43, 'Ipoh Coffee', 20, 1, '16 - 500 g tins', 46.00),

(44, 'Gula Malacca', 20, 2, '20 - 2 kg bags', 19.45),

(45, 'Røgede sild', 21, 8, '1k pkg.', 9.50),

(46, 'Spegesild', 21, 8, '4 - 450 g glasses', 12.00),

(47, 'Zaanse koeken', 22, 3, '10 - 4 oz boxes', 9.50),

(48, 'Chocolade', 22, 3, '10 pkgs.', 12.75),

(49, 'Maxilaku', 23, 3, '24 - 50 g pkgs.', 20.00),

(50, 'Valkoinen suklaa', 23, 3, '12 - 100 g bars', 16.25),

(51, 'Manjimup Dried Apples', 24, 7, '50 - 300 g pkgs.', 53.00),

(52, 'Filo Mix', 24, 5, '16 - 2 kg boxes', 7.00),

(53, 'Perth Pasties', 24, 6, '48 pieces', 32.80),

(54, 'Tourtière', 25, 6, '16 pies', 7.45),

(55, 'Pâté chinois', 25, 6, '24 boxes x 2 pies', 24.00),

(56, 'Gnocchi di nonna Alice', 26, 5, '24 - 250 g pkgs.', 38.00),

(57, 'Ravioli Angelo', 26, 5, '24 - 250 g pkgs.', 19.50),

(58, 'Escargots de Bourgogne', 27, 8, '24 pieces', 13.25),

(59, 'Raclette Courdavault', 28, 4, '5 kg pkg.', 55.00),

(60, 'Camembert Pierrot', 28, 4, '15 - 300 g rounds', 34.00),

(61, 'Sirop d''érable', 29, 2, '24 - 500 ml bottles', 28.50),

(62, 'Tarte au sucre', 29, 3, '48 pies', 49.30),

(63, 'Vegie-spread', 7, 2, '15 - 625 g jars', 43.90),

(64, 'Wimmers gute Semmelknödel', 12, 5, '20 bags x 4 pieces', 33.25),

(65, 'Louisiana Fiery Hot Pepper Sauce', 2, 2, '32 - 8 oz bottles', 21.05),

(66, 'Louisiana Hot Spiced Okra', 2, 2, '24 - 8 oz jars', 17.00),

(67, 'Laughing Lumberjack Lager', 16, 1, '24 - 12 oz bottles', 14.00),

(68, 'Scottish Longbreads', 8, 3, '10 boxes x 8 pieces', 12.50),

(69, 'Gudbrandsdalsost', 15, 4, '10 kg pkg.', 36.00),

(70, 'Outback Lager', 7, 1, '24 - 355 ml bottles', 15.00),

(71, 'Fløtemysost', 15, 4, '10 - 500 g pkgs.', 21.50),

(72, 'Mozzarella di Giovanni', 14, 4, '24 - 200 g pkgs.', 34.80),

(73, 'Rød Kaviar', 17, 8, '24 - 150 g jars', 15.00),

(74, 'Longlife Tofu', 4, 7, '5 kg pkg.', 10.00),

(75, 'Rhönbräu Klosterbier', 12, 1, '24 - 0.5 l bottles', 7.75),

(76, 'Lakkalikööri', 23, 1, '500 ml', 18.00),

(77, 'Original Frankfurter grüne Soße', 12, 2, '12 boxes', 13.00);

**1. Basic Cursor to Select All Products**

DELIMITER $$

CREATE PROCEDURE SelectAllProducts()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE ProductName VARCHAR(255);

DECLARE cur CURSOR FOR SELECT ProductID, ProductName FROM Products;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, ProductName;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT ProductID, ProductName;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**2. Cursor to Update Product Prices**

DELIMITER $$

CREATE PROCEDURE UpdateProductPricesWithCursor(IN Percentage FLOAT)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE CurrentPrice DECIMAL(10, 2);

DECLARE cur CURSOR FOR SELECT ProductID, Price FROM Products;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, CurrentPrice;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE Products

SET Price = CurrentPrice + (CurrentPrice \* Percentage / 100)

WHERE ProductID = ProductID;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**3. Cursor to Select Products by Category**

DELIMITER $$

CREATE PROCEDURE SelectProductsByCategory(IN CatID INT)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE ProductName VARCHAR(255);

DECLARE cur CURSOR FOR SELECT ProductID, ProductName FROM Products WHERE CategoryID = CatID;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, ProductName;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT ProductID, ProductName;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**4. Cursor to Sum Prices by Supplier**

DELIMITER $$

CREATE PROCEDURE SumPricesBySupplier(IN SupID INT, OUT TotalPrice DECIMAL(10, 2))

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE CurrentPrice DECIMAL(10, 2);

DECLARE cur CURSOR FOR SELECT Price FROM Products WHERE SupplierID = SupID;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

SET TotalPrice = 0;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO CurrentPrice;

IF done THEN

LEAVE read\_loop;

END IF;

SET TotalPrice = TotalPrice + CurrentPrice;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**5. Cursor to Delete Products by Category**

DELIMITER $$

CREATE PROCEDURE DeleteProductsByCategory(IN CatID INT)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE cur CURSOR FOR SELECT ProductID FROM Products WHERE CategoryID = CatID;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID;

IF done THEN

LEAVE read\_loop;

END IF;

DELETE FROM Products WHERE ProductID = ProductID;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**6. Cursor to Find Products Below a Price**

DELIMITER $$

CREATE PROCEDURE SelectProductsBelowPrice(IN MaxPrice DECIMAL(10, 2))

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE ProductName VARCHAR(255);

DECLARE cur CURSOR FOR SELECT ProductID, ProductName FROM Products WHERE Price < MaxPrice;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, ProductName;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT ProductID, ProductName;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**7. Cursor to List Products from a Specific Supplier**

DELIMITER $$

CREATE PROCEDURE SelectProductsBySupplier(IN SupID INT)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE ProductName VARCHAR(255);

DECLARE cur CURSOR FOR SELECT ProductID, ProductName FROM Products WHERE SupplierID = SupID;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, ProductName;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT ProductID, ProductName;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**8. Cursor to Count Products in Each Category**

DELIMITER $$

CREATE PROCEDURE CountProductsInCategories()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE CatID INT;

DECLARE ProductCount INT;

DECLARE cur CURSOR FOR SELECT CategoryID, COUNT(\*) FROM Products GROUP BY CategoryID;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO CatID, ProductCount;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT CatID, ProductCount;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**9. Cursor to Update Stock Levels**

DELIMITER $$

CREATE PROCEDURE UpdateStockLevels(IN IncreaseBy INT)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE CurrentStock INT;

DECLARE cur CURSOR FOR SELECT ProductID, Quantity FROM Products;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, CurrentStock;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE Products

SET Quantity = CurrentStock + IncreaseBy

WHERE ProductID = ProductID;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**10. Cursor to Find Out of Stock Products**

DELIMITER $$

CREATE PROCEDURE SelectOutOfStockProducts()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE ProductName VARCHAR(255);

DECLARE cur CURSOR FOR SELECT ProductID, ProductName FROM Products WHERE Quantity = 0;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, ProductName;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT ProductID, ProductName;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**11. Cursor to Select Products with Low Stock**

DELIMITER $$

CREATE PROCEDURE SelectLowStockProducts(IN Threshold INT)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE ProductName VARCHAR(255);

DECLARE cur CURSOR FOR SELECT ProductID, ProductName FROM Products WHERE Quantity < Threshold;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, ProductName;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT ProductID, ProductName;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**12. Cursor to Update Units of a Specific Category**

DELIMITER $$

CREATE PROCEDURE UpdateUnitsByCategory(IN CatID INT, IN NewUnit VARCHAR(255))

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE cur CURSOR FOR SELECT ProductID FROM Products WHERE CategoryID = CatID;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE Products

SET Unit = NewUnit

WHERE ProductID = ProductID;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**13. Cursor to List Product Names in Alphabetical Order**

DELIMITER $$

CREATE PROCEDURE SelectProductsAlphabetically()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE ProductName VARCHAR(255);

DECLARE cur CURSOR FOR SELECT ProductID, ProductName FROM Products ORDER BY ProductName;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, ProductName;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT ProductID, ProductName;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

**14. Cursor to Calculate Average Price by Category**

DELIMITER $$

CREATE PROCEDURE CalculateAveragePriceByCategory(IN CatID INT, OUT AvgPrice DECIMAL(10, 2))

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE CurrentPrice DECIMAL(10, 2);

DECLARE cur CURSOR FOR SELECT Price FROM Products WHERE CategoryID = CatID;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

SET AvgPrice = 0;

DECLARE TotalPrice DECIMAL(10, 2) DEFAULT 0;

DECLARE Count INT DEFAULT 0;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO CurrentPrice;

IF done THEN

LEAVE read\_loop;

END IF;

SET TotalPrice = TotalPrice + CurrentPrice;

SET Count = Count + 1;

END LOOP;

SET AvgPrice = TotalPrice / Count;

CLOSE cur;

END $$

DELIMITER ;

**15. Cursor to List Products and Their Prices**

DELIMITER $$

CREATE PROCEDURE ListProductsAndPrices()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ProductID INT;

DECLARE ProductName VARCHAR(255);

DECLARE ProductPrice DECIMAL(10, 2);

DECLARE cur CURSOR FOR SELECT ProductID, ProductName, Price FROM Products;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO ProductID, ProductName, ProductPrice;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT ProductID, ProductName, ProductPrice;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;