Sri Lanka Institute of Information Technology

Database Design & Development |IT2140

2025 | Year 2 Semester 1

Group ID: 2025-Y2-S1-MLB-B6G2-07

Web-based Musical Instrument Selling System

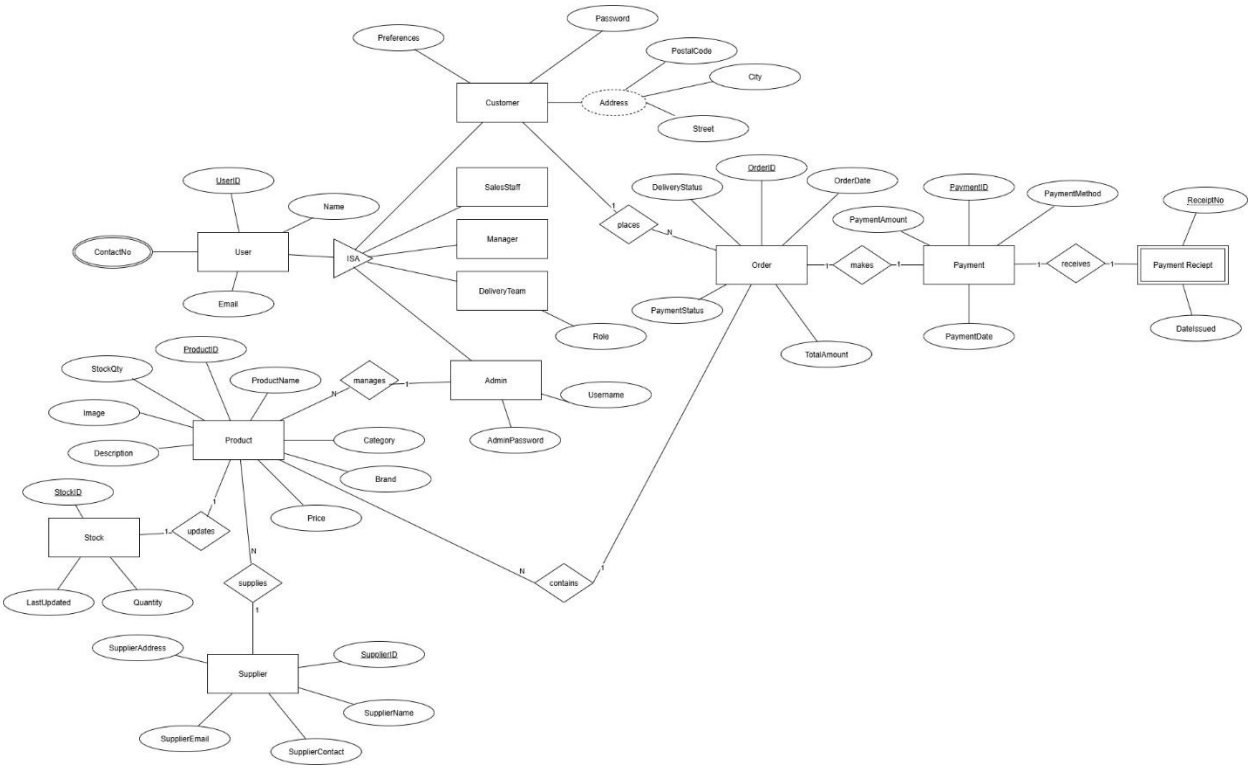
Group Assignment - Part 02

Student ID	**Name**
IT24102374	Wickramaarachchi M. H
IT24101551	Kuyilini. T
IT24101536	Alwis P.G.R.M
IT24101325	Gammanpila J.P
IT24101520	Athukorala A. P. H. B
IT24101376	Nishshanka N.P.S.M.

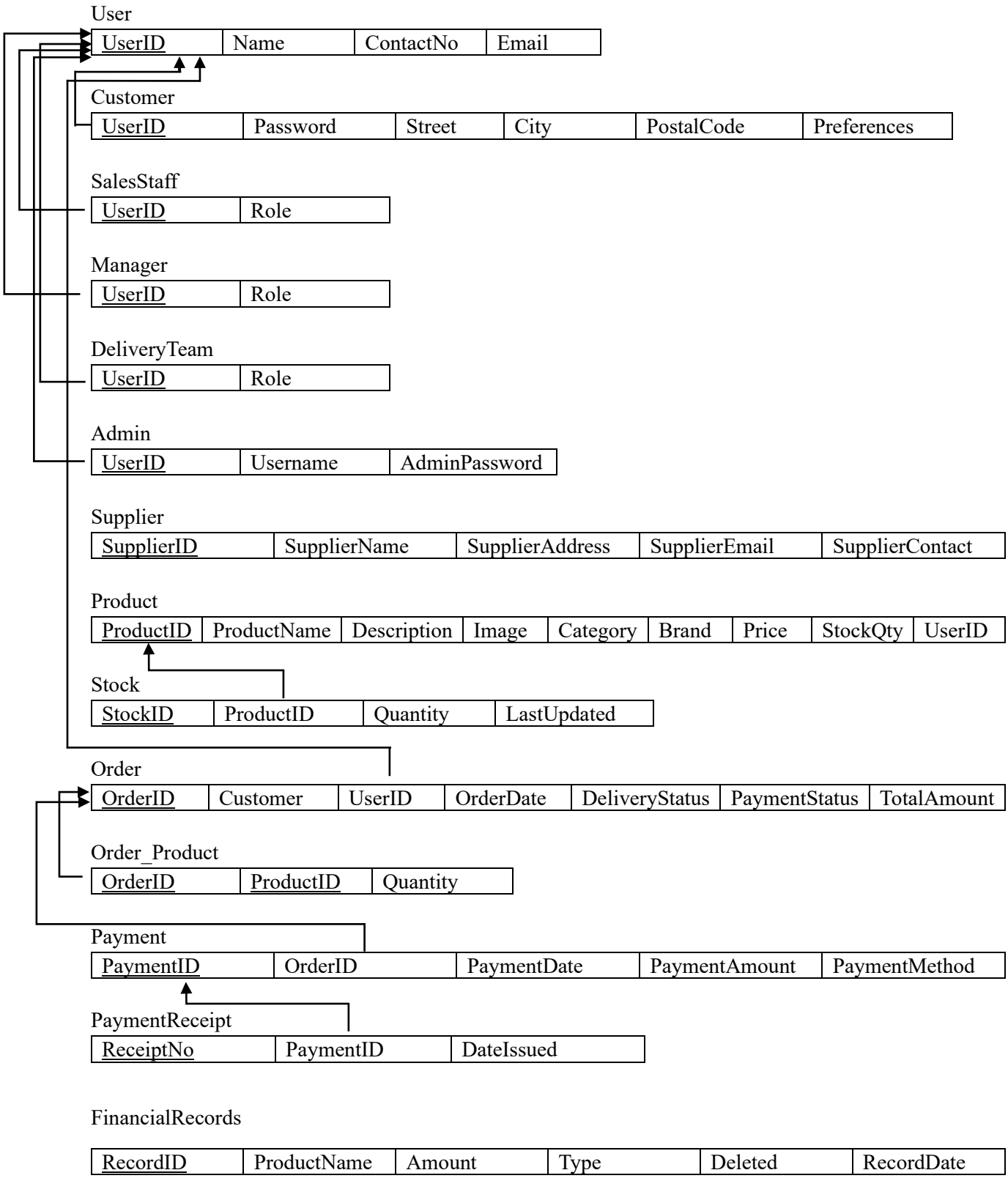
B.Sc. (Hons) in Information Technology

Contents

EER Diagram Design



Mapping to relational schema



ISA Mapping choices

The Mapping Used: Multiple Tables for All (Method 3)

The database uses a "multiple tables for all" approach for user roles, where each role (Customer, Staff, Admin) gets its own table linked to a central Users table. This design provides clear separation between different user types and ensures that all users must first exist in the base Users table, maintaining data integrity through foreign key constraints. Each role table contains only relevant attributes, like customer addresses or admin credentials, making the structure conceptually clean and aligned with business roles.

However, this mapping creates practical problems, particularly with the staff hierarchy where SalesStaff, Manager, and DeliveryTeam tables have identical attributes. This redundancy makes a flawed design that leads to inefficient querying to get all staff and poor scalability when adding new role types. The approach also introduces inconsistent security with separate password systems, making authentication more complicated than necessary.

Refined Schema

User (UserID, Name, ContactNo, Email, UserType)
Customer (UserID ,Password, Street, City, PostalCode, Preferences)
Staff (UserID, Role, EmploymentType, HireDate)
Admin (UserID , Username, Password)

Supplier (SupplierID, SupplierName, Address, Email, ContactNo)

Product (ProductID, ProductName, Description, ImageURL, Category, Brand,
Price, SupplierID, IsActive)
ProductInventory (ProductID, StockQuantity, LastRestocked, ReorderLevel)

CustomerOrder (OrderID, CustomerID,
OrderDate, ShippingStreet, ShippingCity, ShippingPostalCode, TotalAmount, OrderStatus)
OrderItem (OrderID, ProductID, Quantity, UnitPriceAtTime)

Payment (PaymentID, OrderID, PaymentDate, Amount, PaymentMethod, TransactionID, PaymentStatus)
PaymentReceipt (ReceiptID, PaymentID, DateIssued, ReceiptURL)

FinancialRecords(RecordID, ProductName, Amount, Type, Deleted, RecordDate)

SQL DDL Implementation

```
--CREATE DATABASE
CREATE DATABASE musicalDB;

--USERS TABLE
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    ContactNo VARCHAR(15),
    Email VARCHAR(100) UNIQUE NOT NULL,
    UserType VARCHAR(20) CHECK (UserType IN ('Customer', 'Staff', 'Admin'))
);

--CUSTOMER TABLE
CREATE TABLE Customer (
    UserID INT PRIMARY KEY,
    Password VARCHAR(255) NOT NULL,
    Street VARCHAR(100),
    City VARCHAR(50),
    PostalCode VARCHAR(10),
    Preferences TEXT,
    FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE
);
```

```
--STAFF TABLE
CREATE TABLE Staff (
    UserID INT PRIMARY KEY,
    Role VARCHAR(50) NOT NULL,
    EmploymentType VARCHAR(20) CHECK (EmploymentType IN ('Full-Time', 'Part-Time',
'Contract')),
    HireDate DATE NOT NULL,
    FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE
);

--ADMIN TABLE
CREATE TABLE Admin (
    UserID INT PRIMARY KEY,
    Username VARCHAR(50) UNIQUE NOT NULL,
    Password VARCHAR(255) NOT NULL,
    FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE
);

--SUPPLIER TABLE
CREATE TABLE Supplier (
    SupplierID INT PRIMARY KEY,
    SupplierName VARCHAR(100) NOT NULL,
    Address VARCHAR(150),
    Email VARCHAR(100) UNIQUE,
    ContactNo VARCHAR(15)
);
```

```
--PRODUCT TABLE
CREATE TABLE Product (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL,
    Description TEXT,
    ImageURL VARCHAR(255),
    Category VARCHAR(50),
    Brand VARCHAR(50),
    Price DECIMAL(10,2) CHECK (Price >= 0),
    SupplierID INT,
    IsActive BIT DEFAULT 1,
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID) ON DELETE SET NULL
);


--PRODUCT INVENTORY TABLE
CREATE TABLE ProductInventory (
    ProductID INT PRIMARY KEY,
    StockQuantity INT DEFAULT 0 CHECK (StockQuantity >= 0),
    LastRestocked DATE,
    ReorderLevel INT DEFAULT 10 CHECK (ReorderLevel >= 0),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID) ON DELETE CASCADE
);
```

```
--CUSTOMER ORDER TABLE
CREATE TABLE CustomerOrder (
    OrderID INT PRIMARY KEY,
    CustomerID INT NOT NULL,
    OrderDate DATE NOT NULL,
    ShippingStreet VARCHAR(100),
    ShippingCity VARCHAR(50),
    ShippingPostalCode VARCHAR(10),
    TotalAmount DECIMAL(10,2) CHECK (TotalAmount >= 0),
    OrderStatus VARCHAR(20) CHECK (OrderStatus IN ('Pending', 'Shipped', 'Delivered', 'Cancelled')),
    FOREIGN KEY (CustomerID) REFERENCES Customer(UserID) ON DELETE CASCADE
);

--ORDER ITEM TABLE
CREATE TABLE OrderItem (
    OrderID INT,
    ProductID INT,
    Quantity INT CHECK (Quantity > 0),
    UnitPriceAtTime DECIMAL(10,2) CHECK (UnitPriceAtTime >= 0),
    PRIMARY KEY (OrderID, ProductID),
    FOREIGN KEY (OrderID) REFERENCES CustomerOrder(OrderID) ON DELETE CASCADE,
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID) ON DELETE CASCADE
);
```

```
--PAYMENT TABLE
CREATE TABLE Payment (
    PaymentID INT PRIMARY KEY,
    OrderID INT NOT NULL,
    PaymentDate DATE NOT NULL,
    Amount DECIMAL(10,2) CHECK (Amount >= 0),
    PaymentMethod VARCHAR(20) CHECK (PaymentMethod IN ('Credit Card', 'Debit Card', 'Cash', 'Online')),
    TransactionID VARCHAR(100) UNIQUE,
    PaymentStatus VARCHAR(20) CHECK (PaymentStatus IN ('Pending', 'Completed', 'Failed')),
    FOREIGN KEY (OrderID) REFERENCES CustomerOrder(OrderID) ON DELETE CASCADE
);
--PAYMENT RECEIPT TABLE
CREATE TABLE PaymentReceipt (
    ReceiptID INT PRIMARY KEY,
    PaymentID INT NOT NULL,
    DateIssued DATE NOT NULL,
    ReceiptURL VARCHAR(255),
    FOREIGN KEY (PaymentID) REFERENCES Payment(PaymentID) ON DELETE CASCADE
 );
--FINANCIAL RECORDS TABLE
CREATE TABLE FinancialRecords (
    RecordID INT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL,
    Amount DECIMAL(10,2) CHECK (Amount >= 0),
    Type VARCHAR(20) CHECK (Type IN ('Income', 'Expense')),
    Deleted BIT DEFAULT 0,
    RecordDate DATE DEFAULT GETDATE());
```

Sample Data

sampleData.sql - L...musicalDB (sa (73))* | tables.sql - LAPTOP...musicalDB (sa (52))

```sql
USE musicalDB;

--USERS table
INSERT INTO Users (UserID, Name, ContactNo, Email, UserType) VALUES
(1, 'Alice Perera', '0771234567', 'alice@gmail.com', 'Customer'),
(2, 'Brian Silva', '0719876543', 'brian@gmail.com', 'Customer'),
(3, 'Chathura Fernando', '0752223344', 'chathura@gmail.com', 'Staff'),
(4, 'Dinusha Jayasena', '0769988776', 'dinusha@gmail.com', 'Admin'),
(5, 'Eshan Weerasinghe', '0704455667', 'eshan@gmail.com', 'Staff'),
(6, 'Janith Gunarathne', '0775566449', 'janith@gmail.com', 'Customer'),
(7, 'Nimal Perera', '0723344556', 'nimal@gmail.com', 'Customer'),
(8, 'Saman Fernando', '0756677889', 'saman@gmail.com', 'Customer');

SELECT * FROM Users;
```

100 %

Results | Messages

	UserID	Name	ContactNo	Email	UserType
1	1	Alice Perera	0771234567	alice@gmail.com	Customer
2	2	Brian Silva	0719876543	brian@gmail.com	Customer
3	3	Chathura Fernando	0752223344	chathura@gmail.com	Staff
4	4	Dinusha Jayasena	0769988776	dinusha@gmail.com	Admin
5	5	Eshan Weerasinghe	0704455667	eshan@gmail.com	Staff
6	6	Janith Gunarathne	0775566449	janith@gmail.com	Customer
7	7	Nimal Perera	0723344556	nimal@gmail.com	Customer
8	8	Saman Fernando	0756677889	saman@gmail.com	Customer

```sql
--STAFF table
INSERT INTO Staff (UserID, Role, EmploymentType, HireDate) VALUES
--(3, 'Sales Assistant', 'Full-Time', '2023-01-15'),
--(5, 'Inventory Clerk', 'Part-Time', '2024-03-01'),
(9, 'Cashier', 'Full-Time', '2022-05-10'),
(10, 'Delivery Staff', 'Contract', '2023-07-01'),
(11, 'Technician', 'Full-Time', '2024-01-20');

SELECT * FROM Staff;
```

100 %

Results | Messages

	UserID	Role	EmploymentType	HireDate
1	3	Sales Assistant	Full-Time	2023-01-15
2	5	Inventory Clerk	Part-Time	2024-03-01
3	9	Cashier	Full-Time	2022-05-10
4	10	Delivery Staff	Contract	2023-07-01
5	11	Technician	Full-Time	2024-01-20

```sql
--CUSTOMER table
INSERT INTO Customer (UserID, Password, Street, City, PostalCode, Preferences) VALUES
(1, 'pass123', 'No.12, Flower Rd', 'Colombo', '00100', 'Guitars'),
(2, 'pass456', 'No.33, Main St', 'Kandy', '20000', 'Keyboards'),
(6, 'pass789', 'No.15, Lake Rd', 'Galle', '80000', 'Drums'),
(7, 'pass321', 'No.28, Hill St', 'Negombo', '11500', 'Violins'),
(8, 'pass654', 'No.40, Garden Ave', 'Matara', '81000', 'Microphones');

SELECT * FROM Customer;
```

100 %

Results | Messages

	UserID	Password	Street	City	PostalCode	Preferences
1	1	pass123	No.12, Flower Rd	Colombo	00100	Guitars
2	2	pass456	No.33, Main St	Kandy	20000	Keyboards
3	6	pass789	No.15, Lake Rd	Galle	80000	Drums
4	7	pass321	No.28, Hill St	Negombo	11500	Violins
5	8	pass654	No.40, Garden Ave	Matara	81000	Microphones

sampleData.sql - L...musicalDB (sa (73)) | tables.sql - LAPTOP...musicalDB (sa (52))

```sql
--ADMIN table
INSERT INTO Admin (UserID, Username, Password) VALUES
(4, 'adminD', 'admin@123'),
(12, 'superadmin1', 'admin1pass'),
(13, 'superadmin2', 'admin2pass'),
(14, 'superadmin3', 'admin3pass'),
(15, 'superadmin4', 'admin4pass');

SELECT * FROM Admin;
```

100 %

Results | Messages

	UserID	Username	Password
1	4	adminD	admin@123
2	12	superadmin1	admin1pass
3	13	superadmin2	admin2pass
4	14	superadmin3	admin3pass
5	15	superadmin4	admin4pass

```sql
--SUPPLIER table
INSERT INTO Supplier (SupplierID, SupplierName, Address, Email, ContactNo) VALUES
(1, 'Melody Instruments', '45 Queen Street, Colombo', 'melody@instruments.com', '0112456789'),
(2, 'SoundWave Ltd', '23 Kings Road, Kandy', 'soundwave@music.com', '0812233445'),
(3, 'ToneWorld', '78 Harmony Lane, Galle', 'toneworld@music.com', '0915674321'),
(4, 'BeatBox Traders', '10 Lake View, Kurunegala', 'beatbox@music.com', '0373344556'),
(5, 'Classic Strings', '22 Garden Rd, Negombo', 'classic@strings.com', '0315566778');

SELECT * FROM Supplier;
```

100 %

Results | Messages

	SupplierID	SupplierName	Address	Email	ContactNo
1	1	Melody Instruments	45 Queen Street, Colombo	melody@instruments.com	0112456789
2	2	SoundWave Ltd	23 Kings Road, Kandy	soundwave@music.com	0812233445
3	3	ToneWorld	78 Harmony Lane, Galle	toneworld@music.com	0915674321
4	4	BeatBox Traders	10 Lake View, Kurunegala	beatbox@music.com	0373344556
5	5	Classic Strings	22 Garden Rd, Negombo	classic@strings.com	0315566778

```sql
--PRODUCT table
INSERT INTO Product (ProductID, ProductName, Description, ImageURL, Category, Brand, Price, SupplierID, IsActive) VALUES
(1, 'Acoustic Guitar', 'Wooden 6-string acoustic guitar', 'img1.jpg', 'Guitar', 'Yamaha', 45000, 1, 1),
(2, 'Electric Guitar', 'Classic Fender Stratocaster', 'img2.jpg', 'Guitar', 'Fender', 85000, 1, 1),
(3, 'Digital Keyboard', '61-key Yamaha keyboard', 'img3.jpg', 'Keyboard', 'Yamaha', 60000, 2, 1),
(4, 'Drum Set', '5-piece beginner drum set', 'img4.jpg', 'Drums', 'Pearl', 120000, 3, 1),
(5, 'Violin', '4/4 full size beginner violin', 'img5.jpg', 'Strings', 'Stentor', 35000, 5, 1);

SELECT * FROM Product;
```

100 %

Results | Messages

	ProductID	ProductName	Description	ImageURL	Category	Brand	Price	SupplierID	IsActive
1	1	Acoustic Guitar	Wooden 6-string acoustic guitar	img1.jpg	Guitar	Yamaha	45000.00	1	1
2	2	Electric Guitar	Classic Fender Stratocaster	img2.jpg	Guitar	Fender	85000.00	1	1
3	3	Digital Keyboard	61-key Yamaha keyboard	img3.jpg	Keyboard	Yamaha	60000.00	2	1
4	4	Drum Set	5-piece beginner drum set	img4.jpg	Drums	Pearl	120000.00	3	1
5	5	Violin	4/4 full size beginner violin	img5.jpg	Strings	Stentor	35000.00	5	1

```sql
--PRODUCT INVENTORY table
INSERT INTO ProductInventory (ProductID, StockQuantity, LastRestocked, ReorderLevel) VALUES
(1, 10, '2024-12-20', 3),
(2, 5, '2025-02-10', 2),
(3, 8, '2025-03-05', 2),
(4, 4, '2025-04-15', 1),
(5, 12, '2025-01-25', 4);

SELECT * FROM ProductInventory;
```

100 %

Results | Messages

	ProductID	StockQuantity	LastRestocked	ReorderLevel
1	1	10	2024-12-20	3
2	2	5	2025-02-10	2
3	3	8	2025-03-05	2
4	4	4	2025-04-15	1
5	5	12	2025-01-25	4

```sql
--CUSTOMER ORDER table
INSERT INTO CustomerOrder (OrderID, CustomerID, OrderDate, ShippingStreet, ShippingCity, ShippingPostalCode, TotalAmount, OrderStatus) VALUES
(1, 1, '2025-03-01', 'No.12, Flower Rd', 'Colombo', '00100', 45000, 'Delivered'),
(2, 1, '2025-04-10', 'No.12, Flower Rd', 'Colombo', '00100', 85000, 'Pending'),
(3, 2, '2025-05-12', 'No.33, Main St', 'Kandy', '20000', 95000, 'Shipped'),
(4, 6, '2025-06-05', 'No.15, Lake Rd', 'Galle', '80000', 35000, 'Cancelled'),
(5, 7, '2025-07-02', 'No.28, Hill St', 'Negombo', '11500', 60000, 'Pending');

SELECT * FROM CustomerOrder;
```

100 %

Results | Messages

	OrderID	CustomerID	OrderDate	ShippingStreet	ShippingCity	ShippingPostalCode	TotalAmount	OrderStatus
1	1	1	2025-03-01	No.12, Flower Rd	Colombo	00100	45000.00	Delivered
2	2	1	2025-04-10	No.12, Flower Rd	Colombo	00100	85000.00	Pending
3	3	2	2025-05-12	No.33, Main St	Kandy	20000	95000.00	Shipped
4	4	2	2025-06-05	No.33, Main St	Kandy	20000	35000.00	Cancelled
5	5	1	2025-07-02	No.12, Flower Rd	Colombo	00100	60000.00	Pending

```sql
--ORDER ITEM table
INSERT INTO OrderItem (OrderID, ProductID, Quantity, UnitPriceAtTime) VALUES
(1, 1, 1, 45000),
(2, 2, 1, 85000),
(3, 3, 1, 60000),
(3, 5, 1, 35000),
(5, 3, 1, 60000);

SELECT * FROM OrderItem;
```

100 %

Results | Messages

	OrderID	ProductID	Quantity	UnitPriceAtTime
1	1	1	1	45000.00
2	2	2	1	85000.00
3	3	3	1	60000.00
4	3	5	1	35000.00
5	5	3	1	60000.00

```sql
--PAYMENT table
INSERT INTO Payment (PaymentID, OrderID, PaymentDate, Amount, PaymentMethod, TransactionID, PaymentStatus) VALUES
(1, 1, '2025-03-01', 45000, 'Cash', 'TXN001', 'Completed'),
(2, 2, '2025-04-11', 85000, 'Credit Card', 'TXN002', 'Pending'),
(3, 3, '2025-05-13', 95000, 'Online', 'TXN003', 'Completed'),
(4, 4, '2025-06-06', 35000, 'Debit Card', 'TXN004', 'Failed'),
(5, 5, '2025-07-03', 60000, 'Online', 'TXN005', 'Pending');

SELECT * FROM Payment;
```

100 %

Results | Messages

	PaymentID	OrderID	PaymentDate	Amount	PaymentMethod	TransactionID	PaymentStatus
1	1	1	2025-03-01	45000.00	Cash	TXN001	Completed
2	2	2	2025-04-11	85000.00	Credit Card	TXN002	Pending
3	3	3	2025-05-13	95000.00	Online	TXN003	Completed
4	4	4	2025-06-06	35000.00	Debit Card	TXN004	Failed
5	5	5	2025-07-03	60000.00	Online	TXN005	Pending

```sql
--PAYMENT RECEIPT table
INSERT INTO PaymentReceipt (ReceiptID, PaymentID, DateIssued, ReceiptURL) VALUES
(1, 1, '2025-03-01', 'receipt1.pdf'),
(2, 2, '2025-04-11', 'receipt2.pdf'),
(3, 3, '2025-05-13', 'receipt3.pdf'),
(4, 4, '2025-06-06', 'receipt4.pdf'),
(5, 5, '2025-07-03', 'receipt5.pdf');

SELECT * FROM PaymentReceipt;
```

100 %

Results | Messages

	ReceiptID	PaymentID	DateIssued	ReceiptURL
1	1	1	2025-03-01	receipt1.pdf
2	2	3	2025-05-13	receipt2.pdf
3	3	5	2025-07-03	receipt3.pdf
4	4	2	2025-04-11	receipt4.pdf
5	5	4	2025-06-06	receipt5.pdf

```sql
--Financial records table
INSERT INTO FinancialRecords (RecordID, ProductName, Amount, Type, Deleted, RecordDate)
VALUES
(1, 'Acoustic Guitar', 1200.00, 'Income', 0, '2025-10-01'),
(2, 'Electric Guitar', 850.00, 'Income', 0, '2025-10-02'),
(3, 'Drum Set', 450.00, 'Expense', 0, '2025-10-03'),
(4, 'Keyboard', 700.00, 'Income', 0, '2025-10-05'),
(5, 'Violin', 300.00, 'Expense', 0, '2025-10-06');

SELECT * FROM FinancialRecords;
```

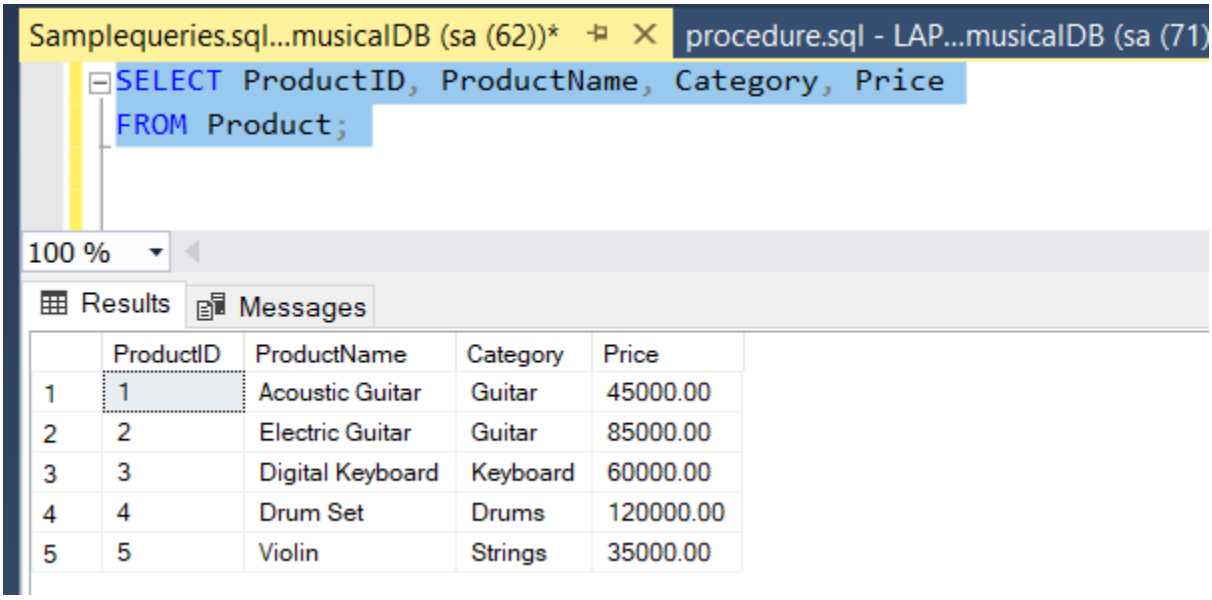
100 %

Results | Messages

	RecordID	ProductName	Amount	Type	Deleted	RecordDate
1	1	Acoustic Guitar	1200.00	Income	0	2025-10-01
2	2	Electric Guitar	850.00	Income	0	2025-10-02
3	3	Drum Set	450.00	Expense	0	2025-10-03
4	4	Keyboard	700.00	Income	0	2025-10-05
5	5	Violin	300.00	Expense	0	2025-10-06

SQL Queries & Outputs

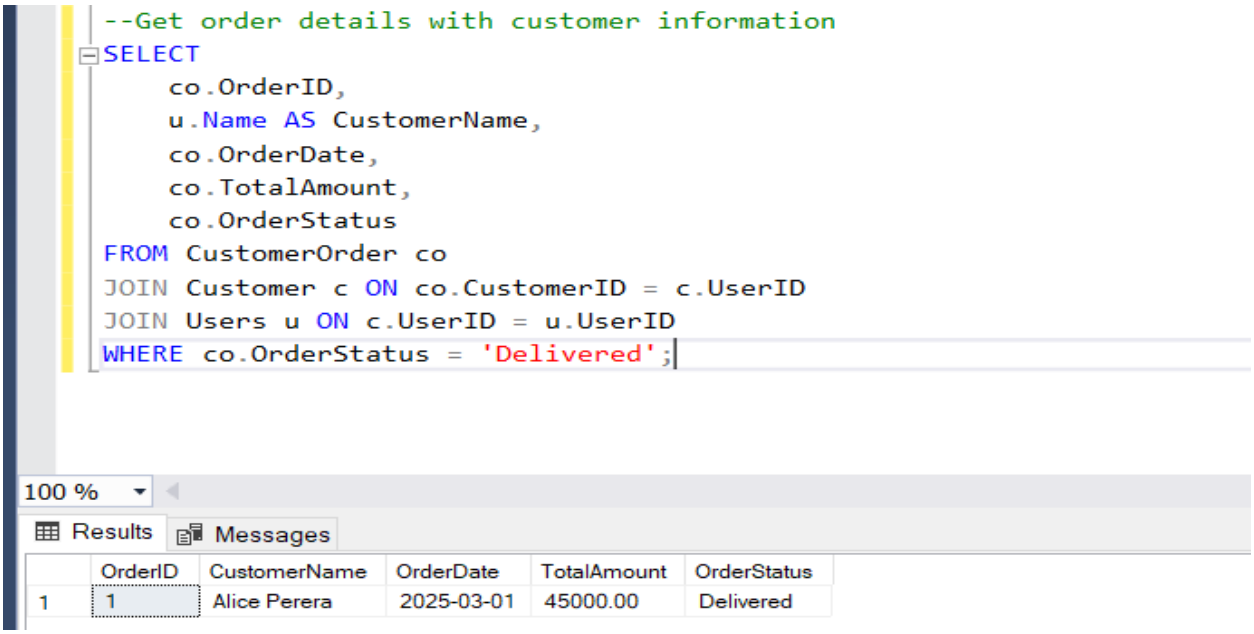
SELECT



	ProductID	ProductName	Category	Price
1	1	Acoustic Guitar	Guitar	45000.00
2	2	Electric Guitar	Guitar	85000.00
3	3	Digital Keyboard	Keyboard	60000.00
4	4	Drum Set	Drums	120000.00
5	5	Violin	Strings	35000.00

Showing a list of all products with only their ID, name, category, and price.

JOIN



	OrderID	CustomerName	OrderDate	TotalAmount	OrderStatus
1	1	Alice Perera	2025-03-01	45000.00	Delivered

Shows all delivered orders with the customer's name by connecting order table, customer table , and user table information together.

AGGREGATION

```sql
-- Calculate total sales by payment method
SELECT
    PaymentMethod,
    COUNT(*) AS TotalTransactions,
    SUM(Amount) AS TotalAmount,
    AVG(Amount) AS AverageAmount
FROM Payment
WHERE PaymentStatus = 'Completed'
GROUP BY PaymentMethod;
```

100 %

Results | Messages

	PaymentMethod	TotalTransactions	TotalAmount	AverageAmount
1	Cash	1	45000.00	45000.000000
2	Online	1	95000.00	95000.000000

For each payment method, count how many successful payments were made, and calculate the total and average amount.

GROUP BY / HAVING

```sql
--Find product categories with average price over 20000
SELECT
    Category,
    AVG(Price) AS AveragePrice
FROM Product
GROUP BY Category
HAVING AVG(Price) > 20000;
```

100 %

Results | Messages

	Category	AveragePrice
1	Drums	120000.000000
2	Guitar	65000.000000
3	Keyboard	60000.000000
4	Strings	35000.000000

Shows product categories where the average price of products is more than 20,000.

SubQuery

```sql
--Find customers who have placed orders above average order value
SELECT
    u.Name,
    u.Email,
    co.OrderID,
    co.TotalAmount
FROM CustomerOrder co
JOIN Customer c ON co.CustomerID = c.UserID
JOIN Users u ON c.UserID = u.UserID
WHERE co.TotalAmount > (
    SELECT AVG(TotalAmount)
    FROM CustomerOrder
    WHERE OrderStatus != 'Cancelled'
)
ORDER BY co.TotalAmount DESC;
```

100 %

Results | Messages

	Name	Email	OrderID	TotalAmount
1	Brian Silva	brian@gmail.com	3	95000.00
2	Alice Perera	alice@gmail.com	2	85000.00

Finds customers who spent more than the average order value and show their big orders from highest to lowest.

Stored Function/Procedure

```sql
--- Procedure: Calculate total amount spent by a customer (Input: Customer ID, Output: Total Amount)

CREATE PROCEDURE GetTotalSpent
    @custID INT,
    @totalSpent MONEY OUTPUT
AS
BEGIN
    SELECT @totalSpent = SUM(TotalAmount)
    FROM CustomerOrder
    WHERE CustomerID = @custID
END;

--- Call the procedure

DECLARE @total MONEY;

EXEC GetTotalSpent 1, @total OUTPUT;

PRINT @total;
```

100 % 3 0

Messages

```
130000.00

Completion time: 2025-10-11T11:26:10.6432510+05:30
```

Calculates total amount spent by a customer.

Trigger

```sql
--- Trigger: Update inventory after new order item is inserted

CREATE TRIGGER reduce_stock_trigger
ON OrderItem
FOR INSERT
AS
BEGIN
    DECLARE @pid INT
    DECLARE @qty INT

    SELECT @pid = ProductID, @qty = Quantity FROM inserted

    UPDATE ProductInventory
    SET StockQuantity = StockQuantity - @qty
    WHERE ProductID = @pid
END
```

```sql
--- Test trigger

--- Check initial stock
SELECT * FROM ProductInventory WHERE ProductID = 3;

--- Insert order item
INSERT INTO CustomerOrder (OrderID, CustomerID, OrderDate, ShippingStreet, ShippingCity, ShippingPostalCode, TotalAmount, OrderStatus)
VALUES (7, 1, '2025-10-11', '123 Main Rd', 'Colombo', '00100', 60000, 'Pending');

INSERT INTO OrderItem (OrderID, ProductID, Quantity, UnitPriceAtTime)
VALUES (7, 3, 1, 60000);

--- Check stock again to confirm update
SELECT * FROM ProductInventory WHERE ProductID = 3;
```

100 % 25 0 Ln: 34 Ch: 1 SPC CRL

Results Messages

	ProductID	StockQuantity	LastRestocked	ReorderLevel
1	3	8	2025-03-05	2

	ProductID	StockQuantity	LastRestocked	ReorderLevel
1	3	7	2025-03-05	2

Reduces stock quantity after a new order is placed.