Database Systems

WholeSale Managment System

Yusuf Karamuk 2210206053

PROJECT SCENARIO:

The Wholesale Management System is crucial for effective handling of stocks, buyers, customers, and financial transactions. This system ensures smooth and efficient operations of the wholesale business by detailed managing all critical aspects.

KEY FEATURES:

1. Inventory Management
    - Stock Records
        - Maintains detailed records of stock items including IDs, names, and quantities.
        - Ensures precise inventory management.
    - Supply Chain Management
        - Tracks buyers by recording their IDs, names, addresses, and the specific items they provide.
        - Simplifies supply chain management.
2. Customer Management
    - Customer Information
        - Records customer details such as names, addresses, and unique IDs.
        - Facilitates smooth order processing and delivery management.
    - Defaulters Management
        - Keeps a record of defaulters to identify customers with outstanding payments.
        - Aids in receivables management and cash flow maintenance.
3. Financial Transactions
    - Payment Records
        - Tracks payment details including ID, customer details, amount, and status.
        - Ensures accurate tracking of all transactions.
        - Addresses pending payments promptly.
    - Monthly Profit Calculation
        - Calculates monthly profits.
        - Offers insights into financial performance.
        - Supports decision-making.
4. Stock Level Monitoring
    - Inventory Alerts
        - Monitors stock levels.
        - Issues alerts when quantities fall below a certain threshold.
        - Ensures optimal inventory levels and timely reordering.

Entitiy and Attributes

Entities

Entitiy1	Entitiy2	Entitiy3	Entitiy4	Entitiy5	Entitiy6	Entitiy7	Entitiy8	Entitiy9
STOCKS	BUYERS	CUSTOMERS	DEFAULTERS	PAYMENTS	STOCKREORDER	MONTHLYPROFIT	DELIVERYDATES	SALES

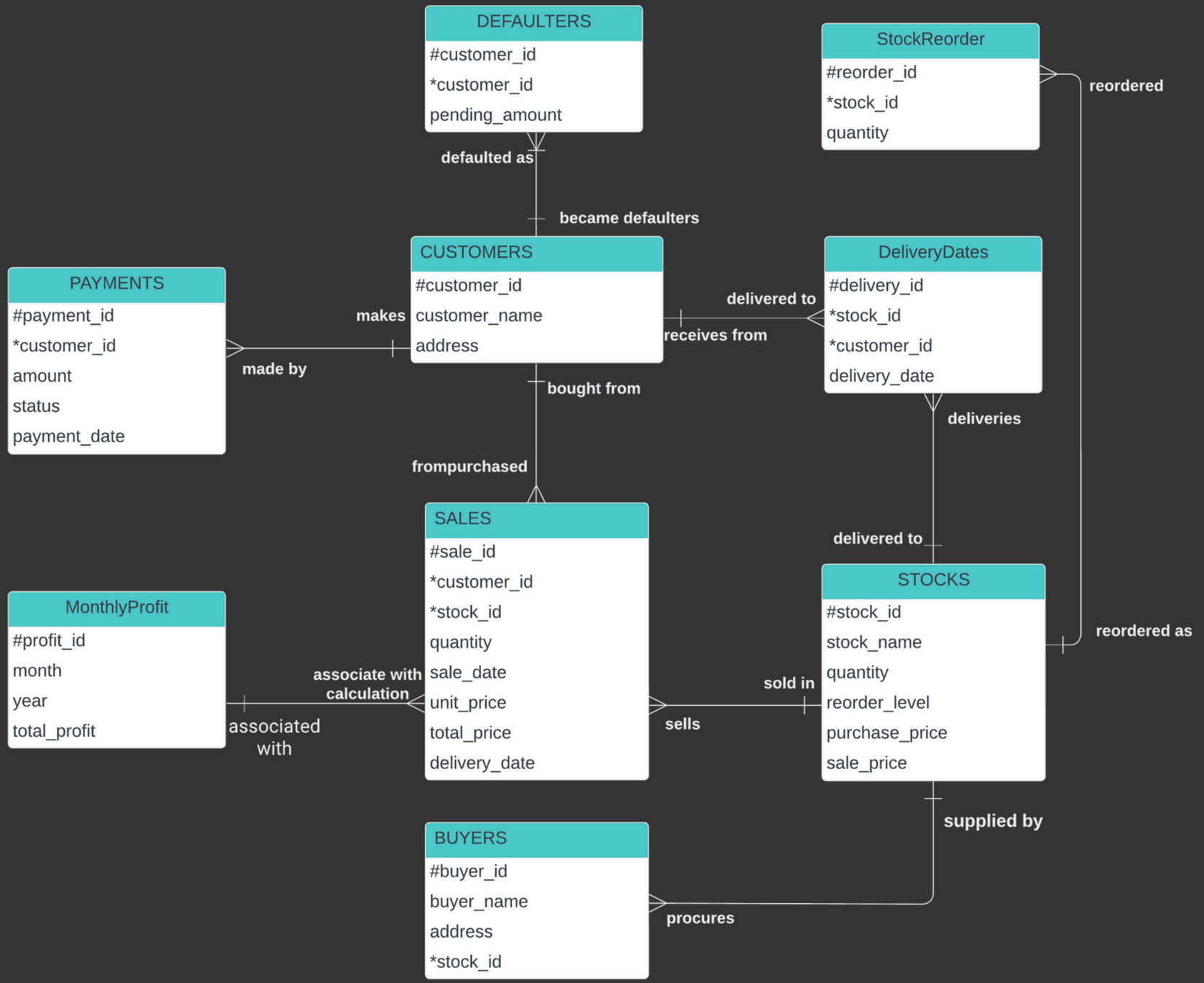
Attributes

STOCKS	BUYERS	CUSTOMERS	DEFAULTERS	PAYMENTS	STOCKREORDER	MONTHLYPROFIT	DELIVERY DATES	SALES
#STOCKID	#BUYERID	#CUSTOMERID	#CUSTOMERID	#PAYMENTID	#REORDERID	#PROFITID	#DELIVERID	#SALEID
STOCKNAME	BUYERNAME	CUSTOMERNAME	*CUSTOMERID	*CUSTOMERID	*STOCKID	MONTH	*STOCKID	*CUSTOMERID
QUANTITY	ADDRESS	ADDRESS	PENDINGAMOUNT	AMOUNT	QUANTITY	YEAR	*CUSTOMERID	*STOCKID
REORDERLEVEL	*STOCKID			STATUS		TOTALPROFIT	DELIVERYDATE	QUANTITY
PURCHASEPRICE				PAYMENTDATE				SALEDATE
SALEPRICE								UNITPRICE
								TOTALPRICE
								DELIVERYDATE

Matrix Diagram

	STOCKS	BUYERS	CUSTOMERS	DEFAULTERS	PAYMENTS	STOCKREORDER	MONTHLYPROFIT	DELIVERY DATES	SALES
STOCKS	*********	supplied by				reordered as		delivered to	sold in
BUYERS	procures	*********							
CUSTOMERS			*********	can become defaulter	makes payments			receives deliveries from	buys from
DEFAULTERS			defaults as	*********					
PAYMENTS			made by		*********				
STOCKREORDER	reordered					*********			
MONTHYLYPROFIT							*********		associated with
DELIVERYDATES	delivered		delivered to					*********	
SALES	sells		buys from				associated with		*********

Entitiy Relationship Diagram (ERD)



SQL DDL Codes

(Create and Insert)

STOCKS

```sql
CREATE TABLE Stocks (
    StockID INT PRIMARY KEY,
    Name VARCHAR(100),
    Quantity INT,
    ReorderLevel INT,
    PurchasePrice DECIMAL(10, 2),
    SalePrice DECIMAL(10, 2)
);

```

```sql
insert into STOCKS (STOCKID , STOCKNAME , QUANTITY, REORDERLEVEL ,PURCHASEPRICE ,SALEPRICE) VALUES (1,'RAM', 1000, 100 ,20, 95) ;
insert into STOCKS (STOCKID , STOCKNAME , QUANTITY, REORDERLEVEL ,PURCHASEPRICE ,SALEPRICE) VALUES (2,'MOTHER BOARD', 4562, 80 ,35, 195)
insert into STOCKS (STOCKID , STOCKNAME , QUANTITY, REORDERLEVEL ,PURCHASEPRICE ,SALEPRICE) VALUES (3,'CPU', 842, 50 ,41, 78) ;
insert into STOCKS (STOCKID , STOCKNAME , QUANTITY, REORDERLEVEL ,PURCHASEPRICE ,SALEPRICE) VALUES (4,'PSU', 80, 65 ,30, 85) ;
insert into STOCKS (STOCKID , STOCKNAME , QUANTITY, REORDERLEVEL ,PURCHASEPRICE ,SALEPRICE) VALUES (5,'ssd', 482, 75 ,50, 81) ;
```

BUYERS

```sql
CREATE TABLE Buyers (
    BuyerID INT PRIMARY KEY,
    Name VARCHAR(100),
    Address VARCHAR(255),
    StockID INT,
    FOREIGN KEY (StockID) REFERENCES Stocks(StockID)
);
```

```sql
INSERT INTO BUYERS (BUYERID, BUYERNAME, ADDRESS, STOCKID)
VALUES (1, 'KINGSTON', 'ENGLAND', 1);
INSERT INTO BUYERS (BUYERID, BUYERNAME, ADDRESS, STOCKID)
VALUES (2, 'MSI', 'ENGLAND', 2);
INSERT INTO BUYERS (BUYERID, BUYERNAME, ADDRESS, STOCKID)
VALUES (3, 'AMD', 'GERMANY', 3);
INSERT INTO BUYERS (BUYERID, BUYERNAME, ADDRESS, STOCKID)
VALUES (4, 'ASUS', 'USA', 4);
INSERT INTO BUYERS (BUYERID, BUYERNAME, ADDRESS, STOCKID)
VALUES (5, 'GIGABYTE', 'HOLLAND', 5);
```

CUSTOMERS

```sql
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    Address VARCHAR(255)
);
```

```sql
insert  into CUSTOMERS (CUSTOMERID, CUSTOMERNAME ,  ADDRESS ) VALUES (1,'MERT BULUT', '100.SOKAK') ;
insert  into CUSTOMERS (CUSTOMERID, CUSTOMERNAME ,  ADDRESS ) VALUES (2,'YUSUF KARAMUK', '101.SOKAK') ;
insert  into CUSTOMERS (CUSTOMERID, CUSTOMERNAME ,  ADDRESS ) VALUES (3,'YUSUF EREN ERIKCI', 'KRAL.SOKAK') ;
insert  into CUSTOMERS (CUSTOMERID, CUSTOMERNAME ,  ADDRESS ) VALUES (4,'BERKER SIMSEK', 'KRAL.SOKAK') ;
insert  into CUSTOMERS (CUSTOMERID, CUSTOMERNAME ,  ADDRESS ) VALUES (5,'KEMAL GUNES', 'GENC.SOKAK') ;
```

DEFAULTERS

```sql
CREATE TABLE Defaulters (
    CustomerID INT PRIMARY KEY,
    PendingAmount DECIMAL(10, 2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

```sql
INSERT INTO DEFAULTERS (CUSTOMERID ,PENDINGAMOUNT)
VALUES (1, 7000);
INSERT INTO DEFAULTERS (CUSTOMERID ,PENDINGAMOUNT)
VALUES (2, 20000);
```

PAYMENTS

```sql
CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY,
    CustomerID INT,
    Amount DECIMAL(10, 2),
    Status VARCHAR(50),
    PaymentDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

```sql
INSERT INTO Payments (PaymentID, CustomerID, Amount, Status, PaymentDate)
VALUES (1, 1, 5000, 'PAID', TO_DATE('2024-04-04', 'YYYY-MM-DD'));
INSERT INTO Payments (PaymentID, CustomerID, Amount, Status, PaymentDate)
VALUES (2, 2, 6000, 'PAID', TO_DATE('2024-05-01', 'YYYY-MM-DD'));
```

STOCKREORDER

```sql
CREATE TABLE StockReorder (
    ReorderID INT PRIMARY KEY,
    StockID INT,
    Quantity INT,
    FOREIGN KEY (StockID) REFERENCES Stocks(StockID)
);

```

```sql
INSERT INTO STOCKREORDER (REORDERID ,STOCKID,QUANTITY)
VALUES (1, 1,1000);
INSERT INTO STOCKREORDER (REORDERID ,STOCKID,QUANTITY)
VALUES (2, 2,1000);
INSERT INTO STOCKREORDER (REORDERID ,STOCKID,QUANTITY)
VALUES (3, 3,1000);
INSERT INTO STOCKREORDER (REORDERID ,STOCKID,QUANTITY)
VALUES (4, 4,1000);
INSERT INTO STOCKREORDER (REORDERID ,STOCKID,QUANTITY)
VALUES (5, 5,1000);
```

MONTHLYPROFIT

```sql
CREATE TABLE MonthlyProfit (
    ProfitID INT PRIMARY KEY,
    Month INT,
    Year INT,
    TotalProfit DECIMAL(10, 2)
);

```

```sql
CREATE OR REPLACE PROCEDURE calculate_monthly_profit(p_month INT, p_year INT) IS
    v_total_sales DECIMAL(10, 2);
    v_total_cost DECIMAL(10, 2);
    v_profit DECIMAL(10, 2);
BEGIN
    -- Toplam satış gelirini hesapla
    SELECT SUM(TotalPrice) INTO v_total_sales
    FROM Sales
    WHERE EXTRACT(MONTH FROM SaleDate) = p_month
      AND EXTRACT(YEAR FROM SaleDate) = p_year;

    -- Toplam satın alma maliyetini hesapla
    SELECT SUM(s.Quantity * st.PurchasePrice) INTO v_total_cost
    FROM Sales s
    JOIN Stocks st ON s.StockID = st.StockID
    WHERE EXTRACT(MONTH FROM s.SaleDate) = p_month
      AND EXTRACT(YEAR FROM s.SaleDate) = p_year;

    -- Karı hesapla
    v_profit := v_total_sales - v_total_cost;

    -- MonthlyProfit tablosuna ekle
    INSERT INTO MonthlyProfit (ProfitID, Month, Year, TotalProfit)
    VALUES (seq_profit_id.NEXTVAL, p_month, p_year, v_profit);
END;
/
```

--There is no need to provide a sample INSERT command to execute this procedure, as the procedure is directly executable.

DELIVERYDATES

```
CREATE TABLE DeliveryDates (
    DeliveryID INT PRIMARY KEY,
    StockID INT,
    CustomerID INT,
    DeliveryDate DATE,
    FOREIGN KEY (StockID) REFERENCES Stocks(StockID),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
```

```
CREATE OR REPLACE TRIGGER record_delivery_date
AFTER INSERT ON Sales
FOR EACH ROW
BEGIN
    INSERT INTO DeliveryDates (DeliveryID, StockID, CustomerID, DeliveryDate)
    VALUES (seq_delivery_id.NEXTVAL, :NEW.StockID, :NEW.CustomerID, :NEW.DeliveryDate);
END;
/
```

SALES

```sql
CREATE TABLE Sales (
    SaleID INT PRIMARY KEY,
    CustomerID INT,
    StockID INT,
    Quantity INT,
    SaleDate DATE,
    UnitPrice DECIMAL(10, 2),
    TotalPrice DECIMAL(10, 2) GENERATED ALWAYS AS (Quantity * UnitPrice) STORED,
    DeliveryDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (StockID) REFERENCES Stocks(StockID)
);

```

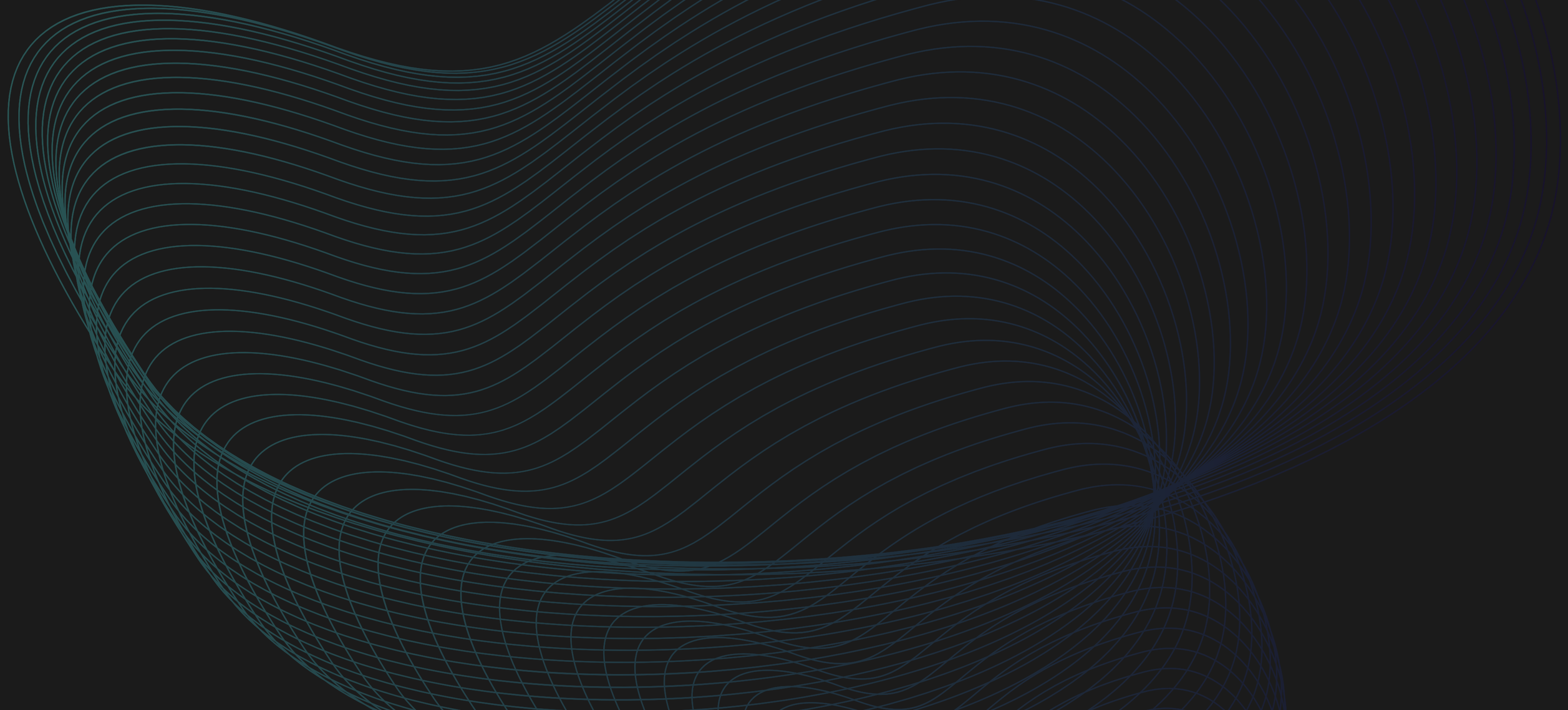
```sql
INSERT INTO SALES (SALEID, CUSTOMERID, STOCKID, QUANTITY, SALEDATE, UNITPRICE, DELIVERYDATE)
SELECT 1, 1, 1, 10, TO_DATE('26-05-2024', 'DD-MM-YYYY'), SALEPRICE, TO_DATE('05-06-2024', 'DD-MM-YYYY')
FROM stocks
WHERE STOCKID = 1;

INSERT INTO SALES (SALEID, CUSTOMERID, STOCKID, QUANTITY, SALEDATE, UNITPRICE, DELIVERYDATE)
SELECT 2, 2, 3, 44, TO_DATE('15-04-2024', 'DD-MM-YYYY'), SALEPRICE, TO_DATE('22-04-2024', 'DD-MM-YYYY')
FROM stocks
WHERE STOCKID = 3;

INSERT INTO SALES (SALEID, CUSTOMERID, STOCKID, QUANTITY, SALEDATE, UNITPRICE, DELIVERYDATE)
SELECT 3, 3, 2, 44, TO_DATE('13-05-2024', 'DD-MM-YYYY'), SALEPRICE, TO_DATE('20-05-2024', 'DD-MM-YYYY')
FROM stocks
WHERE STOCKID = 2;


INSERT INTO SALES (SALEID, CUSTOMERID, STOCKID, QUANTITY, SALEDATE, UNITPRICE, DELIVERYDATE)
SELECT 4, 4, 5, 22, TO_DATE('13-05-2024', 'DD-MM-YYYY'), SALEPRICE, TO_DATE('20-05-2024', 'DD-MM-YYYY')
FROM stocks
WHERE STOCKID = 5;
```

SQL DML Codes



Subquery

```sql
SELECT StockID,STOCKNAME, Quantity, ReorderLevel, PurchasePrice, SalePrice
FROM Stocks
WHERE StockID = (
    SELECT StockID
    FROM Sales
    ORDER BY UnitPrice DESC
    FETCH FIRST 1 ROW ONLY);
```

Results | Explain | Describe | Saved SQL | History

STOCKID	STOCKNAME	QUANTITY	REORDERLEVEL	PURCHASEPRICE	SALEPRICE
2	MOTHER BOARD	4430	80	35	195

-Retrieves the information of the product with the highest sale price using STOCK table:

Join

```sql
SELECT s.SaleID, s.CustomerID, s.StockID, s.Quantity, s.SaleDate, s.UnitPrice, st.STOCKNAME, st.PurchasePrice
FROM Sales s
JOIN Stocks st ON s.StockID = st.StockID;
```

Results | Explain | Describe | Saved SQL | History

SALEID	CUSTOMERID	STOCKID	QUANTITY	SALEDATE	UNITPRICE	STOCKNAME	PURCHASEPRICE
2	2	3	44	15-Apr-2024	78	CPU	41
4	4	5	22	13-May-2024	81	ssd	50
1	1	1	10	26-May-2024	100	RAM	20
3	3	2	44	13-May-2024	195	MOTHER BOARD	35

-Retrieves the stock information and prices along with the details of each sale using a JOIN statement:

Group by

```sql
SELECT StockID, SUM(Quantity) AS TotalQuantity
FROM Sales
GROUP BY StockID;
```

Results | Explain | Describe | Saved SQL | History

STOCKID	TOTALQUANTITY
1	10
2	44
5	22
3	44

-Calculates the total sales quantity for each stock

Date Function

```sql
SELECT EXTRACT(YEAR FROM SaleDate) AS SaleYear,
       EXTRACT(MONTH FROM SaleDate) AS SaleMonth,
       EXTRACT(DAY FROM SaleDate) AS SaleDay
FROM Sales;
```

Results | Explain | Describe | Saved SQL | History

SALEYEAR	SALEMONTH	SALEDAY
2024	4	15
2024	5	13
2024	5	26
2024	5	13

-This query extracts month, and day information from the SaleDate column. The EXTRACT() function separates specified units (MONTH, DAY) from a date value.

Character Function

```sql
SELECT UPPER(CUSTOMERNAME) AS UpperName,
       LENGTH(CUSTOMERNAME) AS NameLength
FROM Customers;
```

Results | Explain | Describe | Saved SQL | History

UPPERNAME	NAMELENGTH
YUSUF KARAMUK	13
YUSUF EREN ERIKCI	17
BERKER SIMSEK	13
KEMAL GUNES	11
MERT BULUT	10

-This query transforms the values in the "Name" column of the Customers table to uppercase using the UPPER function, and simultaneously calculates the length of each name using the LENGTH function.

Update

```sql
UPDATE Customers
SET Address = 'Tuncer Sokak'
WHERE CustomerID = 2;
```

Results | Explain | Describe | Saved SQL | History

1 row(s) updated.

0.00 seconds

EDIT	CUSTOMERID	CUSTOMERNAME	ADDRESS
	2	Yusuf Karamuk	Tuncer Sokak

-This query updates the address of the customer with the customer id “2” in the Customers table to 'Tuncer Sokak'.

Alter Table

```
1  ALTER TABLE CUSTOMERS ADD TELNO NUMBER;
```

Results | Explain | Describe | Saved SQL | History

```
Table altered.

0.03 seconds
```

-This ALTER TABLE statement adds a new column named TELNO to the CUSTOMERS table.

EDIT	CUSTOMERID	CUSTOMERNAME	ADDRESS	TELNO
	2	Yusuf Karamuk	Tuncer Sokak	-
	6	ismail kurt	-	-
	3	YUSUF EREN ERİKCİ	KRAL.SOKAK	-
	4	BERKER SIMSEK	KRAL.SOKAK	-
	5	KEMAL GUNES	GENC.SOKAK	-
	1	MERT BULUT	100.SOKAK	-

```
1  ALTER TABLE CUSTOMERS RENAME COLUMN TELNO TO ILETISIM;
```

Results | Explain | Describe | Saved SQL | History

```
Table altered.

0.01 seconds
```

-This ALTER TABLE statement renames the column TELNO in the CUSTOMERS table to ILETISIM.

EDIT	CUSTOMERID	CUSTOMERNAME	ADDRESS	ILETISIM
	2	Yusuf Karamuk	Tuncer Sokak	-
	6	ismail kurt	-	-
	3	YUSUF EREN ERİKCİ	KRAL.SOKAK	-
	4	BERKER SIMSEK	KRAL.SOKAK	-
	5	KEMAL GUNES	GENC.SOKAK	-
	1	MERT BULUT	100.SOKAK	-

```
1  ALTER TABLE CUSTOMERS MODIFY ILETISIM VARCHAR(150);
```

Results | Explain | Describe | Saved SQL | History

```
Table altered.
```

0.13 seconds

-This ALTER TABLE statement modifies the data type of the ILETISIM column in the CUSTOMERS table to VARCHAR(150).

Column Name	Data Type	Nullable	Default	Primary Key
CUSTOMERID	NUMBER	No	-	1
CUSTOMERNAME	VARCHAR2(100)	Yes	-	-
ADDRESS	VARCHAR2(255)	Yes	-	-
ILETISIM	VARCHAR2(150)	Yes	-	-

```
1  ALTER TABLE CUSTOMERS DROP COLUMN ILETISIM;
```

Results | Explain | Describe | Saved SQL | History

```
Table altered.
```

0.02 seconds

-Deletes column from the CUSTOMERS table:

EDIT	CUSTOMERID	CUSTOMERNAME	ADDRESS
	2	Yusuf Karamuk	Tuncer Sokak
	6	ismail kurt	-
	3	YUSUF EREN ERİKCİ	KRAL.SOKAK
	4	BERKER SIMSEK	KRAL.SOKAK
	5	KEMAL GUNES	GENC.SOKAK
	1	MERT BULUT	100.SOKAK

End of the presentation.