Riphah International University Lahore

Project Title: Active POS

Submitted to: **Mr. Talha Tariq**

Group Members:

Shakeel Ahmad (48237)

Abdul Samad (48865)

DEPARTMENT OF COMPUTER SCIENCE RIPHAH INTERNATIONAL UNIVERSITY LAHORE.

Introduction

Active POS is an advanced online platform that simplifies purchasing and inventory management for retailers and wholesalers. It offers features like user authentication, product management, and invoice generation for smooth transactions. Active POS tackles common sales and inventory challenges with solutions for cart management, vendor integration, and detailed reporting. This helps businesses manage stock and sales efficiently, boosting their effectiveness and profitability.

Objective

Create an online POS system that boosts sales, manages inventory, and handles invoicing to improve business operations and customer satisfaction.

Final Outcome

A user-friendly online POS platform that connects businesses with customers, enhances inventory tracking, and simplifies sales processes while offering detailed financial records.

Goals

- Improve sales operations with efficient product and inventory management.
- Enhance customer experience with a smooth and reliable purchasing process.
- Optimize resources by accurately tracking stock and sales data.
- Offer detailed reporting tools for better financial and inventory decisions.

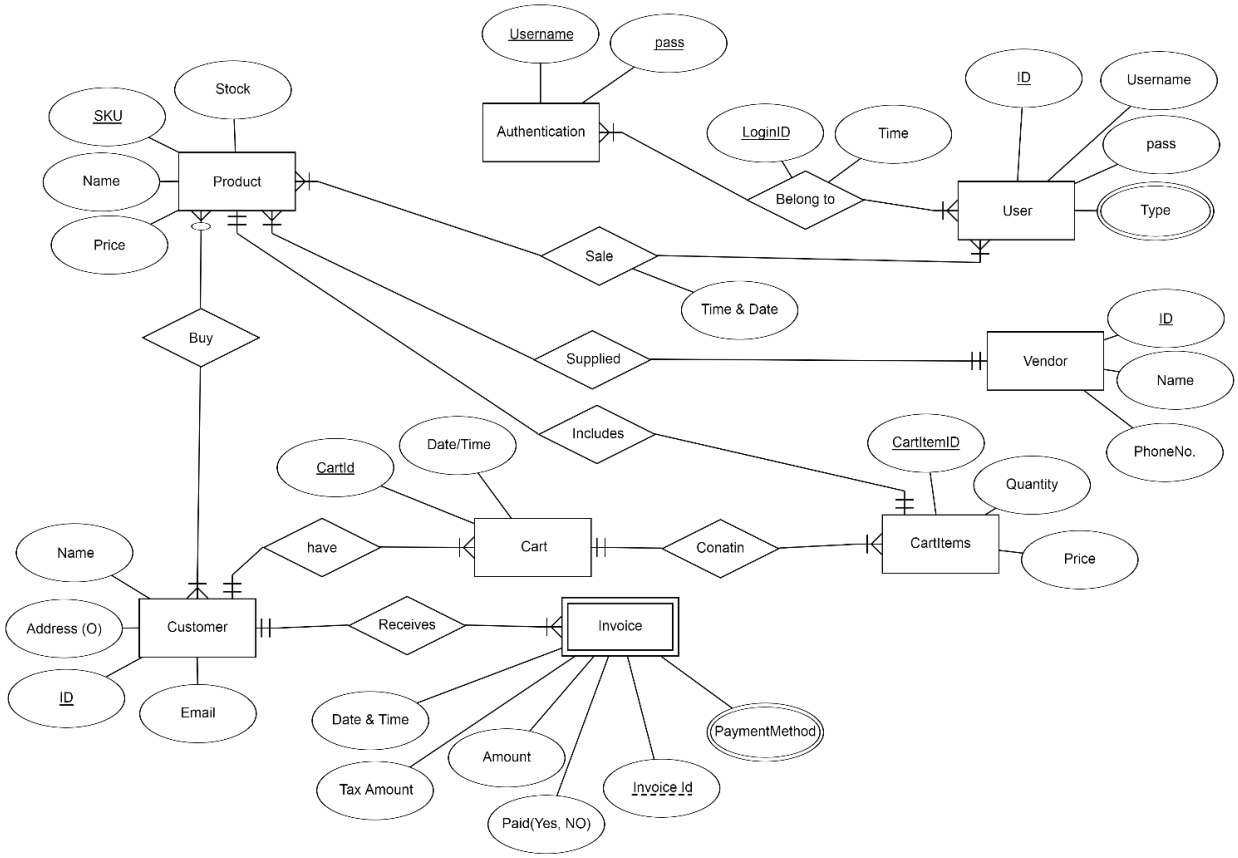
Functions

- User authentication and profile management.
- Product listing and inventory management.
- Sales processing and cart management.
- Invoice generation and payment processing.
- Vendor management and product supply tracking.
- Detailed reporting and analytics for sales and inventory.

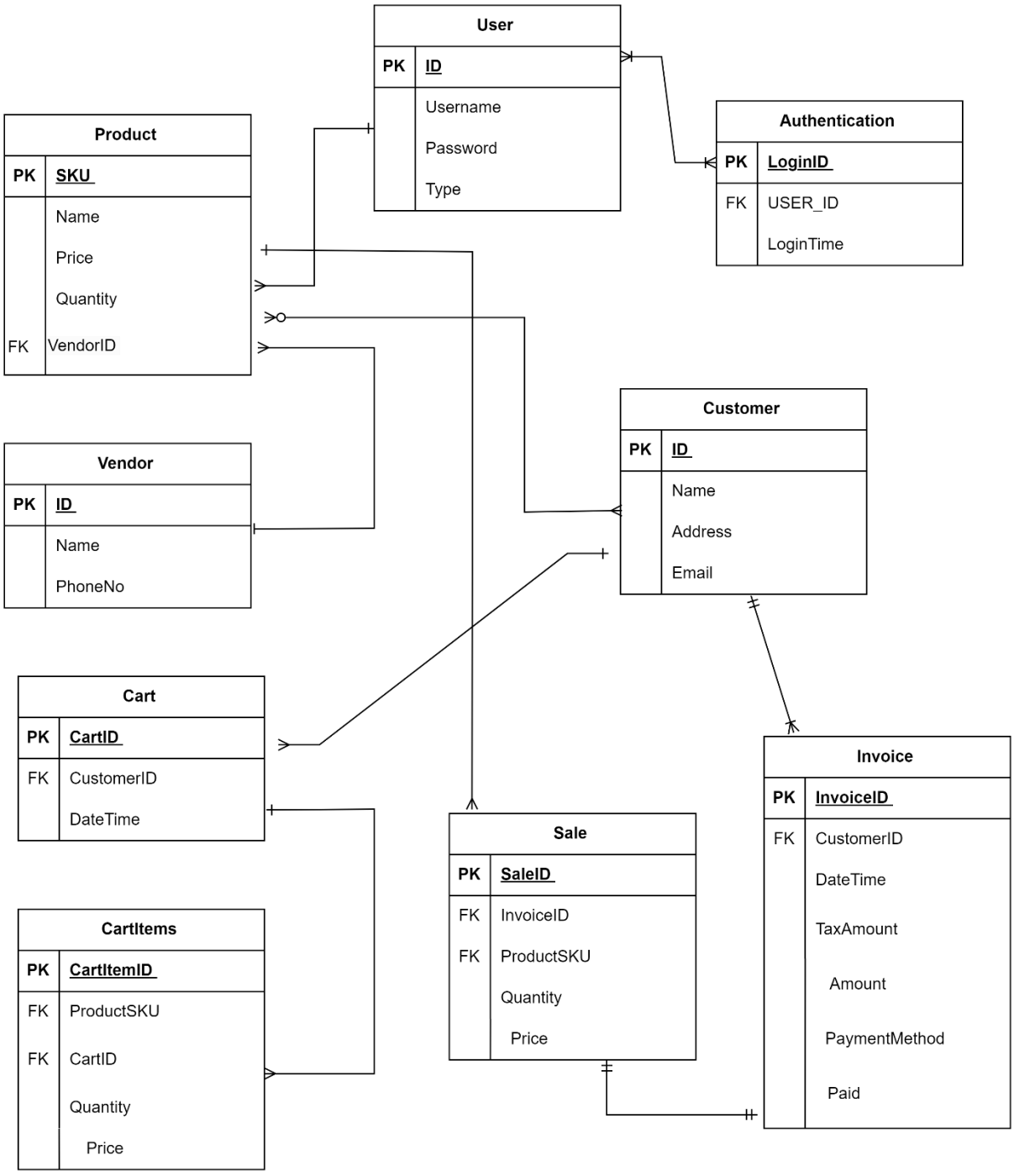
Methodology

- Develop a user-friendly interface for both administrators and customers.
- Implement robust product and inventory management functionality.
- Integrate secure payment gateways and invoicing features.

ERD OF POS:

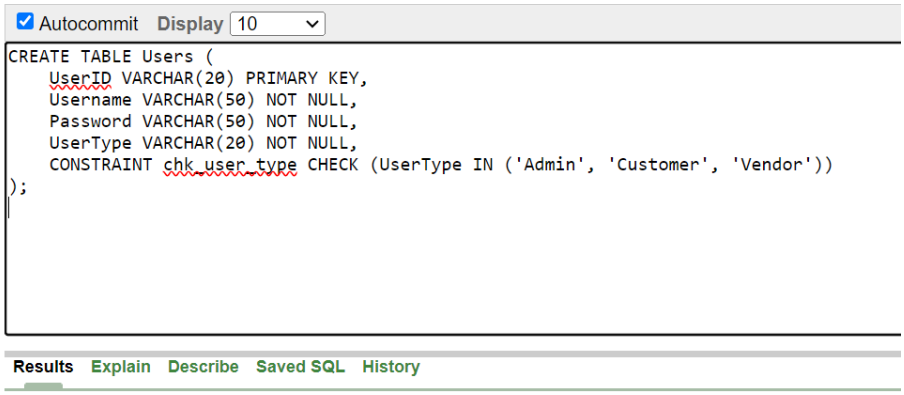

Username
pass
Authentication
Stock
SKU
Name
Price
Product
LoginID
Time
Belong to
ID
Username
pass
User
Type
Sale
Time & Date
Buy
Supplied
Vendor
ID
Name
PhoneNo.
Includes
CartItemID
Quantity
CartItems
Price
Date/Time
CartId
Cart
have
Conatin
Name
Address (O)
ID
Customer
Email
Receives
Invoice
Date & Time
Tax Amount
Amount
Paid(Yes, NO)
Invoice Id
PaymentMethod

SCHEMA OF POS:



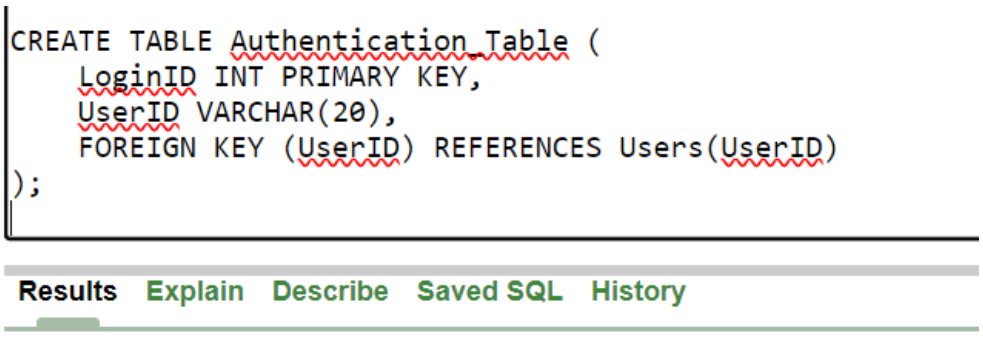
CREATION OF TABLES

- **Users**




```
Table created.
```

- **Authentication_Table**




```
Table created.
```

- **Vendor**

```
CREATE TABLE Vendor (
    ID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    PhoneNo VARCHAR(15)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

- **Product**

```
CREATE TABLE Product (
    SKU INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Price DECIMAL(10, 2) NOT NULL,
    Quantity INT NOT NULL,
    ID INT,
    FOREIGN KEY (ID ) REFERENCES Vendor(ID)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

- **Customer_**

```
CREATE TABLE Customer_ (
    ID INT PRIMARY KEY ,
    Name VARCHAR(100) NOT NULL,
    Address VARCHAR(255),
    Email VARCHAR(100) NOT NULL
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

- **Cart**

```
CREATE TABLE Cart (
    CartID INT PRIMARY KEY ,
    ID INT,
    DateTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (ID ) REFERENCES Customer_(ID)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

- **CartItems**

```
CREATE TABLE CartItems (
    CartItemID INT PRIMARY KEY,
    CartID INT,
    SKU INT,
    Quantity INT NOT NULL,
    Price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (CartID) REFERENCES Cart(CartID),
    FOREIGN KEY (SKU ) REFERENCES Product(SKU)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

- **Invoice**

```
CREATE TABLE Invoice (
    InvoiceID INT PRIMARY KEY,
    ID INT,
    DateTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    TaxAmount DECIMAL(10, 2),
    Amount DECIMAL(10, 2) NOT NULL,
    PaymentMethod VARCHAR2(10) CHECK (PaymentMethod IN ('Cash', 'Card')) NOT NULL,
    Paid VARCHAR(3) DEFAULT 'No' CHECK (Paid IN ('Yes', 'No')),
    CONSTRAINT fk_customer_id FOREIGN KEY (ID) REFERENCES Customer_(ID)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

- **Sale**

```
CREATE TABLE Sale (
    SaleID INT PRIMARY KEY ,
    InvoiceID INT,
    SKU INT,
    Quantity INT NOT NULL,
    Price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (InvoiceID) REFERENCES Invoice(InvoiceID),
    FOREIGN KEY (SKU ) REFERENCES Product(SKU)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

Inserted rows in Users:

```
-- Inserting an Admin user
INSERT INTO Users (UserID, Username, Password, UserType)
VALUES ('123', 'shakeel', 'abc', 'Admin');

-- Inserting a Customer user
INSERT INTO Users (UserID, Username, Password, UserType)
VALUES ('234', 'samad', 'xyz', 'Customer');

-- Inserting a Vendor user
INSERT INTO Users (UserID, Username, Password, UserType)
VALUES ('999', 'Ali', 'tyz', 'Vendor');
```

Result

```
select *from Users ;
```

Results | Explain | Describe | Saved SQL | History

USERID	USERNAME	PASSWORD	USERTYPE
U001	admin1	password123	Admin
U002	customer1	securepass	Customer
U003	vendor1	vendorpass	Vendor
123	shakeel	abc	Admin
234	samad	xyz	Customer
999	Ali	tyz	Vendor

6 rows returned in 0.00 seconds CSV Export

Inserted rows in Authentication_Table:

```
INSERT INTO Authentication_Table (LoginID, UserID)
VALUES (9, '123');

INSERT INTO Authentication_Table (LoginID, UserID)
VALUES (8, '234');

INSERT INTO Authentication_Table (LoginID, UserID)
VALUES (7, '999');
```

Result

```
select *from Authentication_Table ;
```

Results | Explain | Describe | Saved SQL | History

LOGINID	USERID
1	U001
2	U002
3	U003
9	123
8	234
7	999

6 rows returned in 0.00 seconds CSV Export

Inserted rows in Vendor:

```
INSERT INTO Vendor (ID, Name, PhoneNo)
VALUES
(1, 'Vendor A', '123-456-7890'),
(2, 'Vendor B', '003-456-7890'),
(3, 'Vendor C', '123-456-7340'),
(4, 'Vendor D', '123-423-7890'),
(5, 'Vendor E', '123-456-7120');
```

Result:

```
select *from Vendor;
```

Results Explain Describe Save

ID	NAME	PHONENO
1	Vendor A	123-456-7890
2	Vendor B	003-456-7890
3	Vendor C	123-456-7340
4	Vendor D	123-423-7890
5	Vendor E	123-456-7120

5 rows returned in 0.00 seconds

Inserted rows in Product:

```
INSERT INTO Product (SKU, Name, Price, Quantity, ID)
VALUES
(1, 'Product A', 19.99, 100, 1),
(2, 'Product B', 29.99, 150, 2),
(3, 'Product C', 9.99, 200, 3),
(4, 'Product D', 49.99, 50, 4),
(5, 'Product E', 39.99, 75, 5);
```

Result:

```
select *from Product ;
```

Results Explain Describe Saved SQL History

SKU	NAME	PRICE	QUANTITY	ID
1	Product A	19.99	100	1
2	Product B	29.99	150	2
3	Product C	9.99	200	3
4	Product D	49.99	50	4
5	Product E	39.99	75	5

5 rows returned in 0.04 seconds CSV Export

Inserted rows in Customer_:

```
INSERT INTO Customer_ (ID, Name, Address, Email)
VALUES
(1, 'Adeel', 'Gajumata Lahore', 'adeel.doe@13.com');
(2, 'ALI', 'Cubrji Lahore', 'ali@12e.com'),
(3, 'Eman', 'Anar Kali Lahore', 'em.davisKali e.com'),
(4, 'Michael', 'Jhang', 'michaeln@exui.com'),
(5, 'Jessica', 'TX', 'jessica.wilson@exdf.com');
```

Result:

```
select *from Customer_;
```

Results Explain Describe Saved SQL History

ID	NAME	ADDRESS	EMAIL
3	Eman	Anar Kali Lahore	em.davisKali e.com
4	Michael	Jhang	michaeln@exui.com
5	Jessica	TX	jessica.wilson@exdf.com
1	Adeel	Gajumata Lahore	adeel.doe@13.com
2	ALI	Cubrji Lahore	ali@12e.com

5 rows returned in 0.03 seconds CSV Export

Inserted rows in Cart

```
INSERT INTO Cart (CartID, ID)
VALUES
(1, 1);
(2, 2),
(3, 3),
(4, 4),
(5, 5);
```

Result

```
select *from Cart ;
```

Results Explain Describe Saved SQL History

CARTID	ID	DATETIME
1	1	22-JUN-24 05.26.14.605000 PM
2	2	22-JUN-24 05.29.26.240000 PM
4	4	22-JUN-24 05.29.54.356000 PM
5	5	22-JUN-24 05.30.04.209000 PM
3	3	22-JUN-24 05.30.28.794000 PM

5 rows returned in 0.00 seconds CSV Export

Inserted rows in CartItems

```
INSERT INTO CartItems (CartItemID, CartID, SKU, Quantity, Price)
VALUES
(10, 1, 1, 20, 19.99);
(20, 2, 2, 10, 29.99),
(39, 3, 3, 3, 9.99),
(41, 4, 4, 1, 49.99),
(55, 5, 5, 2, 39.99),
```

Result

```
select *from CartItems ;
```

Results Explain Describe Saved SQL History

CARTITEMID	CARTID	SKU	QUANTITY	PRICE
10	1	1	20	19.99
20	2	2	10	29.99
39	3	3	3	9.99
41	4	4	1	49.99
55	5	5	2	39.99

5 rows returned in 0.00 seconds CSV Export

Inserted rows in Invoice

```
INSERT INTO Invoice (InvoiceID, ID, TaxAmount, Amount, PaymentMethod, Paid)
VALUES
(1, 1, 1.50, 30.00, 'Cash', 'Yes'),
(2, 2, 2.50, 50.00, 'Card', 'No'),
(3, 3, 0.75, 15.00, 'Card', 'Yes'),
(4, 4, 3.00, 60.00, 'Cash', 'No'),
(5, 5, 1.25, 25.00, 'Card', 'Yes');
```

Result

Results Explain Describe Saved SQL History

INVOICEID	ID	DATETIME	TAXAMOUNT	AMOUNT	PAYMENTMETHOD	PAID
4	4	22-JUN-24 05.56.53.235000 PM	3	60	Cash	No
5	5	22-JUN-24 05.57.01.366000 PM	1.25	25	Card	Yes
1	1	22-JUN-24 05.55.18.999000 PM	1.5	30	Cash	Yes
2	2	22-JUN-24 05.56.31.811000 PM	2.5	50	Card	No
3	3	22-JUN-24 05.56.43.856000 PM	.75	15	Card	Yes

5 rows returned in 0.00 seconds CSV Export

Inserted rows in Sale

```
INSERT INTO Sale (SaleID, InvoiceID, SKU, Quantity, Price)
VALUES
(1, 1, 1, 2, 19.99),
(2, 1, 2, 1, 29.99),
(3, 2, 3, 3, 9.99),
(4, 3, 4, 1, 49.99),
(5, 4, 5, 2, 39.99);
```

Result

```
select *from Sale  ;
```

Results Explain Describe Saved SQL History

SALEID	INVOICEID	SKU	QUANTITY	PRICE
1	1	1	2	19.99
2	1	2	1	29.99
3	2	3	3	9.99
4	3	4	1	49.99
5	4	5	2	39.99

5 rows returned in 0.02 seconds CSV Export

Update sale table

```
update sale
set price = 90.99
where saleid= 3;
```

Results Explain Describe Saved SQL History

SALEID	INVOICEID	SKU	QUANTITY	PRICE
1	1	1	2	19.99
2	1	2	1	29.99
3	2	3	3	90.99
4	3	4	1	49.99
5	4	5	2	39.99

5 rows returned in 0.00 seconds CSV Export

Add Column

```
Alter table Customer_
Add Age int;

select *from Customer_ ;
```

Results Explain Describe Saved SQL History

ID	NAME	ADDRESS	EMAIL	AGE
3	Eman	Anar Kali Lahore	em.davisKali e.com	-
4	Michael	Jhang	michaeln@exui.com	-
5	Jessica	TX	jessica.wilson@exdf.com	-
1	Adeel	Gajumata Lahore	adeel.doe@13.com	-
2	ALI	Cubrji Lahore	ali@12e.com	-

5 rows returned in 0.03 seconds CSV Export

Rename:

```
ALTER TABLE Sale
RENAME COLUMN Quantity  to qunata;

select *from Sale ;
```

Results Explain Describe Saved SQL History

SALEID	INVOICEID	SKU	QUNATA	PRICE
1	1	1	2	19.99
2	1	2	1	29.99
3	2	3	3	90.99
4	3	4	1	49.99
5	4	5	2	39.99

5 rows returned in 0.00 seconds CSV Export

AGGREGATE FUNCTION

- **MIN**

```
select MIN(Price)
from Sale;

select *from Sale ;
```

Results Explain Describe Saved

MIN(PRICE)
19.99

1 rows returned in 0.00 seconds

- **MAX**

```
select Max(Quantity )
from CartItems ;
```

Results Explain Describe Sav

MAX(QUANTITY)
20

1 rows returned in 0.00 seconds

- **COUNT**

```
select count(Email )
from Customer_ ;
```

Results Explain Describe Save

COUNT(EMAIL)
5

1 rows returned in 0.01 seconds

- **SUM**

```
select SUM(Price )
from product;
```

Results Explain Describe Save

SUM(PRICE)
149.95

1 rows returned in 0.00 seconds

- **AVG**

```
select AVG(TaxAmount )
from Invoice ;
```

Results Explain Describe Save

AVG(TAXAMOUNT)
1.8

1 rows returned in 0.00 seconds

JOINS

- **Left Join**

```
SELECT Users.UserType, Authentication_Table.UserID
FROM Users
LEFT JOIN Authentication_Table
ON Users.UserID = Authentication_Table.UserID;
```

Results Explain Describe Saved SQL History

USERTYPE	USERID
Admin	U001
Customer	U002
Vendor	U003
Admin	123
Customer	234
Vendor	999

6 rows returned in 0.01 seconds CSV Export

- **Right Join**

```
SELECT Users.UserType, Authentication_Table.UserID
FROM Users
Right JOIN Authentication_Table
ON Users.UserID = Authentication_Table.UserID
ORDER BY Users.UserType;
```

Results Explain Describe Saved SQL History

USERTYPE	USERID
Admin	U001
Admin	123
Customer	234
Customer	U002
Vendor	999
Vendor	U003

6 rows returned in 0.01 seconds CSV Export

- **Inner Join**

```
SELECT Vendor.PhoneNo, Product.Price
FROM Vendor
INNER JOIN Product
ON Vendor.ID = Product.ID
ORDER BY Product.Price;
```

Results Explain Describe Saved SQL History

PHONENO	PRICE
123-456-7340	9.99
123-456-7890	19.99
003-456-7890	29.99
123-456-7120	39.99
123-423-7890	49.99

5 rows returned in 0.00 seconds CSV Export

- **Full outer Join**

```
SELECT Sale .Quantity , Invoice .Amount
FROM Sale
Full outer JOIN Invoice
ON Sale.Quantity =  Invoice .Amount
ORDER BY Invoice .Amount ;
```

Results Explain Describe Saved SQL History

QUANTITY	AMOUNT
-	15
-	25
-	30
-	50
-	60
2	-
1	-
1	-
3	-
2	-

10 rows returned in 0.00 seconds CSV Export

Nested (Subqueries)

1.

```
SELECT
    v.Name AS VendorName,
    (
        SELECT SUM(p.Quantity)
        FROM Product p
        WHERE p.ID = v.ID
    ) AS TotalQuantity
FROM Vendor v;
```

Results Explain Describe Saved SQL History

VENDORNAME	TOTALQUANTITY
Vendor A	100
Vendor B	150
Vendor C	200
Vendor D	50
Vendor E	75

5 rows returned in 0.00 seconds CSV Export

2.

```
SELECT
    i.InvoiceID,
    i.Amount AS InvoiceAmount,
    i.TaxAmount,
    (
        SELECT COUNT(*)
        FROM Sale s
        WHERE s.InvoiceID = i.InvoiceID
    ) AS SaleCount
FROM Invoice i;
```

Results Explain Describe Saved SQL History

INVOICEID	INVOICEAMOUNT	TAXAMOUNT	SALECOUNT
4	60	3	1
5	25	1.25	0
1	30	1.5	2
2	50	2.5	1
3	15	.75	1

5 rows returned in 0.02 seconds CSV Export

3.

```
SELECT
    c.CartID,
    c.DateTime,
    (
        SELECT Name
        FROM Customer_
        WHERE ID = c.ID
    ) AS CustomerName
FROM Cart c;
```

Results Explain Describe Saved SQL History

CARTID	DATETIME	CUSTOMERNAME
1	22-JUN-24 05.26.14.605000 PM	Adeel
2	22-JUN-24 05.29.26.240000 PM	ALI
4	22-JUN-24 05.29.54.356000 PM	Michael
5	22-JUN-24 05.30.04.209000 PM	Jessica
3	22-JUN-24 05.30.28.794000 PM	Eman

5 rows returned in 0.02 seconds CSV Export

Normalization

Now apply normalization on Active POS system through the

- First Normal Form (1NF),
- Second Normal Form (2NF),
- Third Normal Form (3NF),
- Boyce-Codd Normal Form (BCNF).

Product table

ProductID	Name	VendorID	Vendor_Name
1	ProductA	4	Vendor_A
2	ProductB	5	Vendor_B

- **First Normal Form (1NF)**

ProductID	Name	VendorID	SupplierName
1	ProductA	2	Vendor_E
2	ProductB	1	Vendor_C

The given table already satisfies 1NF as all values are atomic and the records are unique.

- **Second Normal Form (2NF)**

  The primary key of the table is ProductID. Vendor_Name is dependent on VendorID, not directly on ProductID. This indicates a partial dependency.

  To resolve this, we separate the table into two tables:

  Product Table (with primary key ProductID)
  Vendor Table (with primary key VendorID).

  Product Table

ProductID	Name	VendorID
1	ProductA	1
2	ProductB	3

  Vendor Table

VendorID	Vendor_Name
1	Vendor_C
3	Vendor_B

- **Third Normal Form (3NF)**

  3NF requires the table to be in 2NF and all the attributes to be functionally dependent only on the primary key (no transitive dependency).

  Product Table: In 2NF and no transitive dependencies are present.

  Vendor Table: In 2NF and no transitive dependencies are present.

- **Boyce-Codd Normal Form (BCNF)**

  Product Table:
  In 3NF and the only functional dependency is ProductID -> {Name, VendorID}, where ProductID is a super key.

  Vendor Table:
  In 3NF and the only functional dependency is VendorID -> Vendor_Name, where VendorID is a super key.

- **Final Normalized Tables**

Product Table

ProductID	Name	VendorID
1	ProductA	4
2	ProductB	5

Vendor Table

SupplierID	Vendor_Name
100	Vendor_A
101	Vendor_D