ONLINE FASHION RETAIL DATABASE MANAGEMENT

Professor: Manuel Montrond

GROUP 6:

Vidhi Patel - 002641341

VenkataTadikonda - 002642314

Harshitha Sappidi - 002416707

Abhishek Kanakantty - 002246765

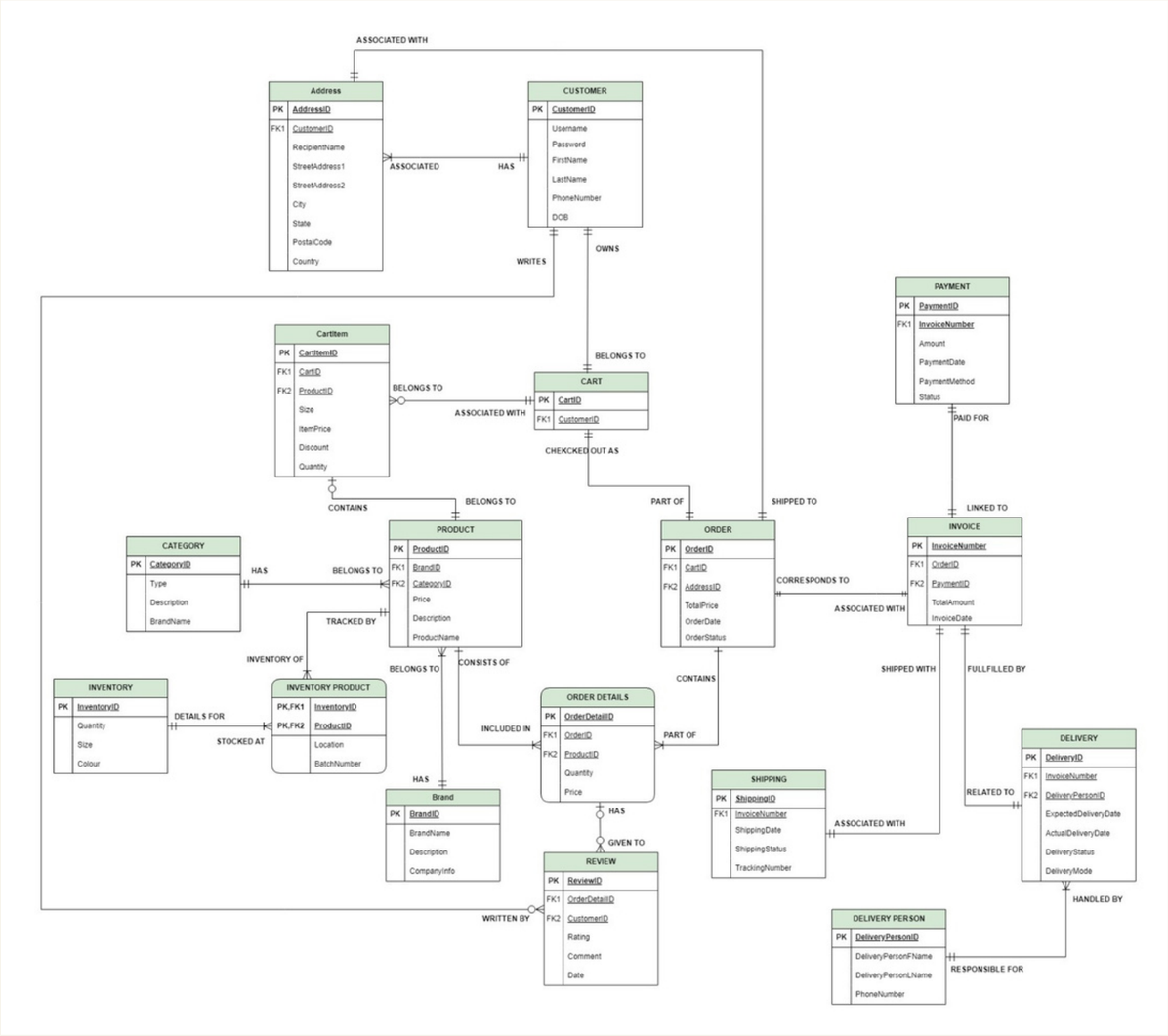
Nitesh More – 002697506

OBJECTIVE

The project aims to develop a comprehensive Database Management System for an e-commerce platform, addressing challenges associated with managing large volumes of online retail data. Key objectives include facilitating seamless navigation, product selection, and purchasing processes for customers, while efficiently managing user accounts, product listings, shopping carts, orders, payments, deliveries and reviews. The proliferation of online shopping has driven a significant shift in consumer behavior, necessitating sophisticated e-commerce databases to enhance user experience, streamline operations, and support informed decision-making. The primary purpose of the database is to empower businesses in the digital marketplace by optimizing inventory management, ensuring product availability, and delivering personalized experiences through data-driven insights. Automation is utilized to reduce costs and minimize errors, while scalability measures ensure adaptability to evolving demands. Robust security protocols safeguard customer information and transactions, fostering trust and confidence in the platform.

HIGH LEVEL DESIGN

ERD



DATABASE OBJECTS

STORED PROCEDURES

- Add Products into Database
- Get Sales Report of Orders
- Fetching orders details for a order placed

```sql
CREATE OR ALTER PROCEDURE AddProduct
    @BrandID INT,
    @CategoryID INT,
    @Price DECIMAL(10, 2),
    @Description TEXT,
    @ProductName VARCHAR(255),
    @Quantity INT,
    @Size VARCHAR(50),
    @Color VARCHAR(50),
    @Location VARCHAR(255),
    @BatchNumber VARCHAR(255),
    @NewProductID INT OUTPUT,
    @NewInventoryID INT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;
    -- Insert into Product table
    INSERT INTO Product (BrandID, CategoryID, Price, [Description], ProductName)
    VALUES (@BrandID, @CategoryID, @Price, @Description, @ProductName);
    -- Get the ID of the newly inserted product
    SET @NewProductID = SCOPE_IDENTITY();
    -- Insert into Inventory table
    INSERT INTO Inventory (Quantity, Size, Color)
    VALUES (@Quantity, @Size, @Color);
    SET @NewInventoryID = SCOPE_IDENTITY();
    -- Insert into InventoryProduct table
    INSERT INTO InventoryProduct (InventoryID, ProductID, [Location], BatchNumber)
    VALUES (@NewInventoryID, @NewProductID, @Location, @BatchNumber);
    SET NOCOUNT OFF;
END
GO
```

CHECK CONSTRAINTS

```sql
ALTER TABLE [Address]
ADD CONSTRAINT CHK_PostalCode CHECK (LEN(PostalCode) >= 5);

ALTER TABLE Product
ADD CONSTRAINT CHK_Price CHECK (Price >= 0);

ALTER TABLE Customer
ADD CONSTRAINT CHK_PhoneNumber_Format CHECK (
    PhoneNumber LIKE '+[0-9]%' OR PhoneNumber LIKE '[0-9]%'
);

ALTER TABLE [Order]
ADD CONSTRAINT CHK_OrderStatus_Valid CHECK (OrderStatus IN ('Pending', 'Processing', 'Shipped', 'Delivered', 'Cancelled','Ready to Ship','Failed','In Transit'))

ALTER TABLE CartItem
ADD CONSTRAINT CHK_Quantity_Positive CHECK (Quantity > 0)

ALTER TABLE CartItem
ADD CONSTRAINT CHK_Discount_Range CHECK (Discount BETWEEN 0 AND 100)

ALTER TABLE Payment
ADD CONSTRAINT CHK_Status CHECK (Status IN ('Success', 'Pending', 'Failed'))

ALTER TABLE Delivery
ADD CONSTRAINT CHK_DeliveryStatus CHECK (DeliveryStatus IN ('Pending', 'In Transit', 'Delivered', 'Failed'))

ALTER TABLE Review
ADD CONSTRAINT UQ_Customer_OrderDetails UNIQUE (CustomerID, OrderDetailsID);
```

DATABASE OBJECTS

TRIGGERS

- Update order and inventory status once payment is successful
- Trigger to clear the cart once the order is placed

```
CREATE OR ALTER TRIGGER UpdateInventoryOnPaymentSuccess
ON Payment
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

   IF(SELECT [Status] FROM inserted)=  'Success'
    BEGIN
        UPDATE inv
        SET inv.Quantity = inv.Quantity - od.Quantity
        FROM Inventory inv
        INNER JOIN InventoryProduct invp ON inv.InventoryID = invp.InventoryID
        INNER JOIN OrderDetails od ON invp.ProductID = od.ProductID
        INNER JOIN [Order] o ON od.OrderID = o.OrderID
        INNER JOIN Invoice invc ON o.OrderID = invc.OrderID
        INNER JOIN inserted i ON invc.InvoiceNumber = i.InvoiceNumber
        WHERE inv.[Size] = od.[Size]
        AND inv.Quantity >= od.Quantity
        AND i.[Status] = 'Success';
    END
END;
GO
```

VIEWS

- Top 5 costliest items sold
- No of successful devileries made
- Top 5 highest rated items

```
CREATE VIEW Top5CostliestItemsSold AS
SELECT TOP 5
  p.ProductName,
  od.Price,
  SUM(od.Quantity) AS TotalQuantitySold,
  SUM(od.Price * od.Quantity) AS TotalSalesValue
FROM
  OrderDetails od
INNER JOIN
  Product p ON od.ProductID = p.ProductID
GROUP BY
  p.ProductName,
  od.Price
ORDER BY
  TotalSalesValue DESC;

  SELECT * FROM Top5CostliestItemsSold;
```

DATABASE OBJECTS

5

UDF

```sql
CREATE FUNCTION dbo.CalculateOrderTotalPrice
(
    @CartID INT
)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @TotalPrice DECIMAL(10, 2);

    SELECT @TotalPrice = ISNULL(SUM((ItemPrice - (ItemPrice * Discount / 100)) * Quantity), 0)
    FROM CartItem
    WHERE CartID = @CartID;

    RETURN @TotalPrice;
END;
GO

CREATE FUNCTION dbo.GetDiscountedPrice
(
    @ProductID INT,
    @Discount DECIMAL(5, 2)
)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @Price DECIMAL(10, 2);

    SELECT @Price = Price FROM Product WHERE ProductID = @ProductID;

    RETURN @Price - (@Price * @Discount / 100);
END;
GO
```

ENCRYPTION

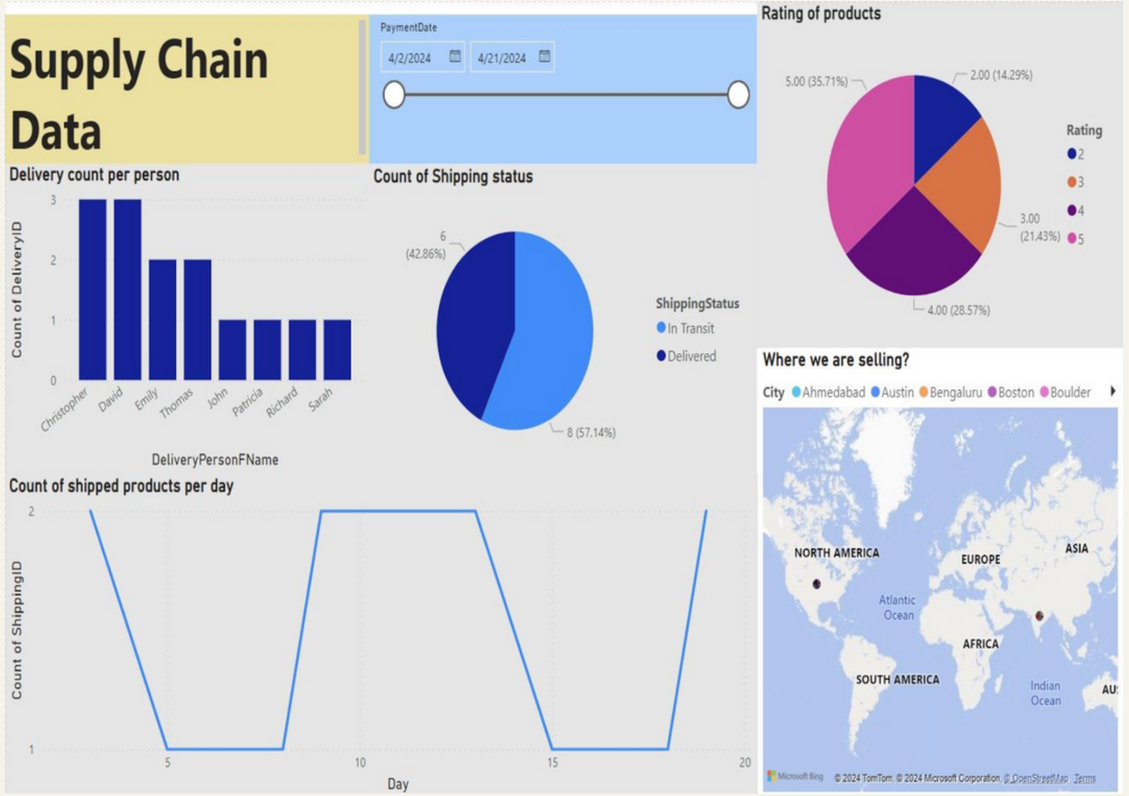
```sql
-----------------------------------------Encryption of the Password Column of Customer Table---
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'DataManagement@6201';

CREATE CERTIFICATE passwordEncryptCertificate WITH SUBJECT = 'Customer Password Encryption';

CREATE SYMMETRIC KEY passwordEncryptionKey WITH ALGORITHM = AES_256

ENCRYPTION BY CERTIFICATE passwordEncryptCertificate;

-- Encrypt the Password columns

OPEN SYMMETRIC KEY passwordEncryptionKey
DECRYPTION BY CERTIFICATE passwordEncryptCertificate;
UPDATE Customer
SET [Password] = EncryptByKey(Key_GUID('passwordEncryptionKey'), [Password]);
CLOSE SYMMETRIC KEY passwordEncryptionKey;


-- Decrypt and View the Password data
OPEN SYMMETRIC KEY passwordEncryptionKey
DECRYPTION BY CERTIFICATE passwordEncryptCertificate;
SELECT
    [Password],
    CONVERT(VARCHAR, DecryptByKey([Password])) AS DecryptedPassword
FROM
    Customer;
CLOSE SYMMETRIC KEY passwordEncryptionKey;
```

NON-CLUSTERED INDEX

```sql
CREATE NONCLUSTERED INDEX IX_Customer_Username ON Customer(Username);
CREATE NONCLUSTERED INDEX IX_Invoices_InvoiceNumber ON Invoice(InvoiceNumber);
CREATE NONCLUSTERED INDEX IX_Orders_OrderID ON [Order](OrderID);
```

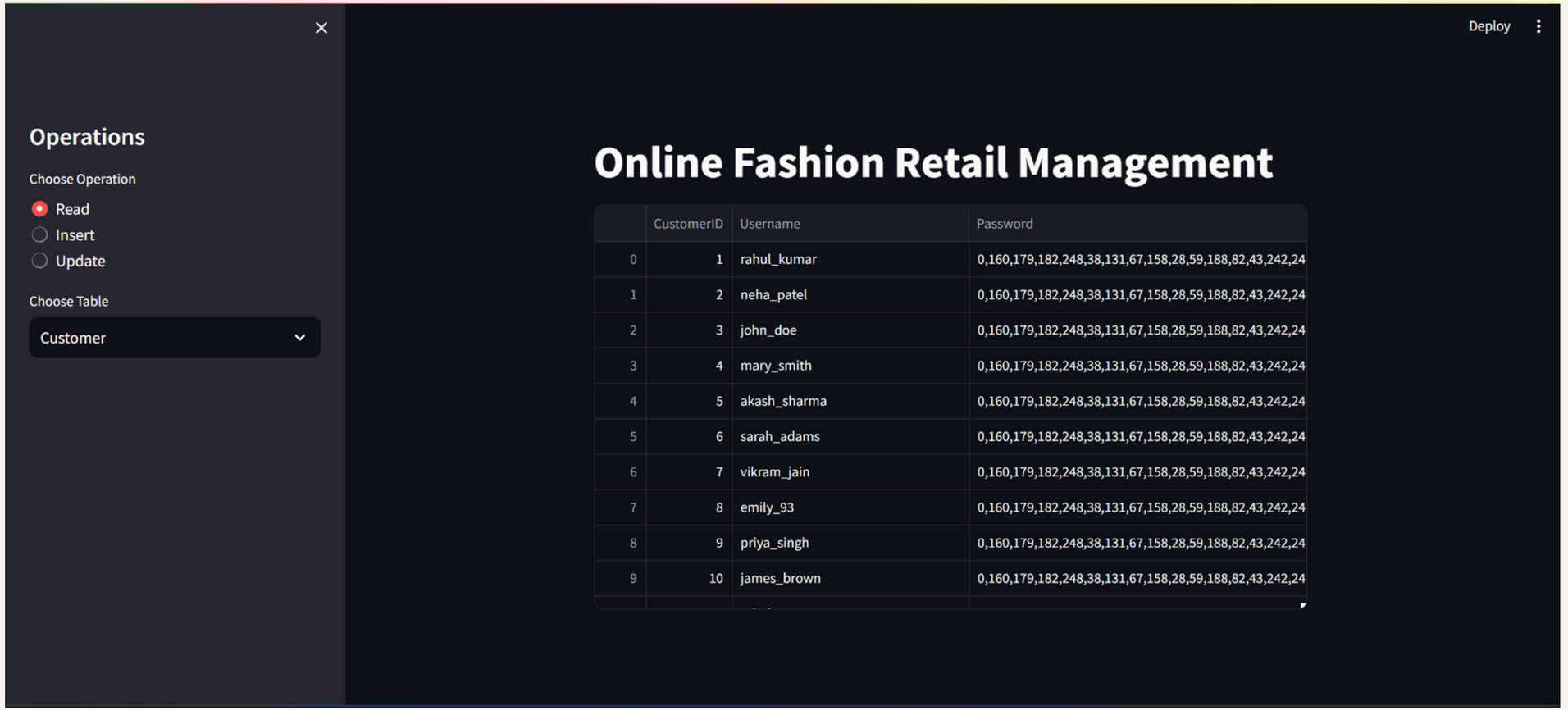
PRESENTATION LAYER

POWER BI



PRESENTATION LAYER

GUI



	CustomerID	Username	Password
0	1	rahul_kumar	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24
1	2	neha_patel	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24
2	3	john_doe	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24
3	4	mary_smith	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24
4	5	akash_sharma	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24
5	6	sarah_adams	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24
6	7	vikram_jain	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24
7	8	emily_93	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24
8	9	priya_singh	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24
9	10	james_brown	0,160,179,182,248,38,131,67,158,28,59,188,82,43,242,24

THANK YOU