use ECOM

-- Create Customers table

CREATE TABLE customers (

cust\_id INT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(100) UNIQUE,

phone\_number VARCHAR(20),

address VARCHAR(255)

);

-- Create Orders table

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

cust\_id INT,

order\_date DATE NOT NULL,

total\_amount DECIMAL(10, 2) NOT NULL,

status VARCHAR(50) DEFAULT 'Pending',

FOREIGN KEY (cust\_id) REFERENCES customers(cust\_id)

);

-- Insert data into the customers table

INSERT INTO customers (cust\_id, first\_name, last\_name, email, phone\_number, address) VALUES

(1, 'Alice', 'Johnson', 'alice.j@email.com', '555-1234', '123 Main St, Anytown'),

(2, 'Bob', 'Williams', 'bob.w@email.com', '555-5678', '456 Oak Ave, Somewhere'),

(3, 'Charlie', 'Brown', 'charlie.b@email.com', NULL, '789 Pine Rd, Otherplace');

-- Insert data into the orders table

INSERT INTO orders (order\_id, cust\_id, order\_date, total\_amount) VALUES

(101, 1, '2025-08-01', 75.50),

(102, 1, '2025-08-05', 150.00),

(103, 2, '2025-08-02', 25.75),

(104, 3, '2025-08-10', 200.25),

(105, 1, '2025-08-14', 50.00);

select \* from customers

select \* from orders

--Create View for showing all order details

CREATE VIEW customer\_order\_details AS

SELECT

c.first\_name,

c.last\_name,

o.order\_id,

o.order\_date,

o.total\_amount,

o.status

FROM

customers c

JOIN

orders o ON c.cust\_id = o.cust\_id;

--Show the result of View which we created

select \* from customer\_order\_details

--Create view with complex select query to shows the total spending for each customer.

CREATE VIEW customer\_spending AS

SELECT

c.first\_name,

c.last\_name,

COUNT(o.order\_id) AS number\_of\_orders,

SUM(o.total\_amount) AS total\_spent

FROM

customers c

JOIN

orders o ON c.cust\_id = o.cust\_id

GROUP BY

c.first\_name, c.last\_name;

--Show the result of View which we created

SELECT first\_name, last\_name, total\_spent FROM customer\_spending;

--create a view that hides a customer's address and email for public users.

CREATE VIEW public\_customer\_info AS

SELECT

cust\_id,

first\_name,

last\_name,

phone\_number

FROM

customers;

--Show the result of View which we created

select \* from public\_customer\_info

--Reusing the view without mentioning joins

SELECT first\_name, last\_name FROM customer\_spending

where total\_spent > 250;

--creating view for Updating Data

CREATE VIEW updatable\_customers AS

SELECT cust\_id, first\_name, last\_name, email FROM customers;

-- Updating data

UPDATE updatable\_customers SET email = 'johnson.a@email.com' WHERE cust\_id = 1;

--Show the result of View which we created

select \* from updatable\_customers

--creating view for non-Updating Data

CREATE VIEW non\_updatable\_view AS

SELECT cust\_id, COUNT(order\_id) AS total\_orders FROM orders GROUP BY cust\_id;

-- Updating data

UPDATE non\_updatable\_view SET cust\_id = 0 WHERE total\_orders = 3;

--Drop a view

DROP VIEW customer\_order\_details;

select \* from customer\_order\_details

--create a view for orders with a total\_amount greater than 100.

CREATE VIEW large\_orders AS

SELECT order\_id, cust\_id, total\_amount

FROM orders

WHERE total\_amount > 100

WITH CHECK OPTION;

--Show the result of View which we created

select \* from large\_orders

--Indexed view using Schemabinding

CREATE VIEW dbo.CustomerOrderSummary0

WITH SCHEMABINDING

AS

SELECT

C.cust\_id,

C.first\_name,

C.last\_name,

SUM(O.total\_amount) AS TotalSpent,

COUNT\_BIG(\*) AS NumberOfOrders -- Corrected to use COUNT\_BIG(\*)

FROM

dbo.customers AS C

JOIN

dbo.orders AS O ON C.cust\_id = O.cust\_id

GROUP BY

C.cust\_id,

C.first\_name,

C.last\_name;

select \* from dbo.CustomerOrderSummary0

--Indexed view using Unique cluster

CREATE UNIQUE CLUSTERED INDEX IX\_CustomerOrderSummary0\_CustomerID

ON dbo.CustomerOrderSummary0 (cust\_id);

select \* from dbo.CustomerOrderSummary1

-- Create a materialized view

CREATE MATERIALIZED VIEW customer\_spending\_mv AS

SELECT

c.first\_name,

c.last\_name,

SUM(o.total\_amount) AS total\_spent

FROM

customers c

JOIN

orders o ON c.customer\_id = o.customer\_id

GROUP BY

c.first\_name, c.last\_name;

-- To refresh the data

REFRESH MATERIALIZED VIEW customer\_spending\_mv;