Content

[**FOR XAMP:** 1](#_Toc167136800)

[**For SQL SERVER DATABASE:** 5](#_Toc167136801)

[**DROP QUERRIES:** 8](#_Toc167136802)

**FOR XAMP:**-- Creating Tables

CREATE TABLE brands (

bid INT AUTO\_INCREMENT PRIMARY KEY,

bname VARCHAR(20)

);

CREATE TABLE inv\_user (

user\_id VARCHAR(20) PRIMARY KEY,

name VARCHAR(20),

password VARCHAR(20),

last\_login TIMESTAMP,

user\_type VARCHAR(10)

);

CREATE TABLE categories (

cid INT AUTO\_INCREMENT PRIMARY KEY,

category\_name VARCHAR(20)

);

CREATE TABLE stores (

sid INT AUTO\_INCREMENT PRIMARY KEY,

sname VARCHAR(20),

address VARCHAR(50),

mobno VARCHAR(10)

);

CREATE TABLE product (

pid INT AUTO\_INCREMENT PRIMARY KEY,

cid INT,

bid INT,

sid INT,

pname VARCHAR(20),

p\_stock INT,

price DECIMAL(10, 2),

added\_date DATE,

image BLOB,

FOREIGN KEY (cid) REFERENCES categories(cid),

FOREIGN KEY (bid) REFERENCES brands(bid),

FOREIGN KEY (sid) REFERENCES stores(sid)

);

CREATE TABLE provides (

bid INT,

sid INT,

discount DECIMAL(5, 2),

FOREIGN KEY (bid) REFERENCES brands(bid),

FOREIGN KEY (sid) REFERENCES stores(sid)

);

CREATE TABLE customer\_cart (

cust\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(20),

mobno VARCHAR(10)

);

CREATE TABLE select\_product (

cust\_id INT,

pid INT,

quantity INT,

FOREIGN KEY (cust\_id) REFERENCES customer\_cart(cust\_id),

FOREIGN KEY (pid) REFERENCES product(pid)

);

CREATE TABLE transaction (

id INT AUTO\_INCREMENT PRIMARY KEY,

total\_amount DECIMAL(10, 2),

paid DECIMAL(10, 2),

due DECIMAL(10, 2),

gst DECIMAL(5, 2),

discount DECIMAL(5, 2),

payment\_method VARCHAR(10),

cart\_id INT,

FOREIGN KEY (cart\_id) REFERENCES customer\_cart(cust\_id)

);

CREATE TABLE invoice (

item\_no INT,

product\_name VARCHAR(20),

quantity INT,

net\_price DECIMAL(10, 2),

transaction\_id INT,

FOREIGN KEY (transaction\_id) REFERENCES transaction(id)

);

-- Procedures

CREATE PROCEDURE get\_due\_amount(IN c\_id INT)

BEGIN

DECLARE due1 DECIMAL(10, 2);

SELECT due INTO due1 FROM transaction WHERE cart\_id = c\_id;

SELECT due1 AS due\_amount;

END;

CREATE PROCEDURE show\_products()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE p\_id INT;

DECLARE p\_name VARCHAR(20);

DECLARE p\_stock INT;

DECLARE cur CURSOR FOR SELECT pid, pname, p\_stock FROM product;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO p\_id, p\_name, p\_stock;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT CONCAT(p\_id, ' ', p\_name, ' ', p\_stock) AS product\_details;

END LOOP;

CLOSE cur;

END;

CREATE PROCEDURE check\_stock(IN b INT)

BEGIN

DECLARE a INT;

SELECT p\_stock INTO a FROM product WHERE pid = b;

IF a < 2 THEN

SELECT 'Stock is Less' AS stock\_status;

ELSE

SELECT 'Enough Stock' AS stock\_status;

END IF;

END;

-- Triggers

CREATE TRIGGER after\_select\_product\_insert

AFTER INSERT ON select\_product

FOR EACH ROW

BEGIN

UPDATE product

SET p\_stock = p\_stock - NEW.quantity

WHERE pid = NEW.pid;

END;

CREATE TRIGGER after\_transaction\_update

AFTER UPDATE ON transaction

FOR EACH ROW

BEGIN

IF NEW.paid != OLD.paid THEN

UPDATE transaction

SET due = total\_amount - paid

WHERE id = NEW.id;

END IF;

END;

-- Insert demo data into stores

INSERT INTO stores (sname, address, mobno) VALUES

('Store One', '123 Main St, Cityville', '1234567890'),

('Store Two', '456 High St, Townsville', '0987654321'),

('Store Three', '789 Market St, Villagetown', '1122334455');

-- Insert demo data into categories

INSERT INTO categories (category\_name) VALUES

('Electronics'),

('Clothing'),

('Groceries');

-- Insert demo data into brands

INSERT INTO brands (bname) VALUES

('Brand A'),

('Brand B'),

('Brand C');

-- Insert demo data into products

INSERT INTO product (cid, bid, sid, pname, p\_stock, price, added\_date, image) VALUES

(1, 1, 1, 'Smartphone', 50, 299.99, '2023-05-20', NULL),

(2, 2, 2, 'Jeans', 100, 49.99, '2023-05-21', NULL),

(3, 3, 3, 'Apple', 200, 0.99, '2023-05-22', NULL);

# **For SQL SERVER DATABASE:**

-- Creating Tables

CREATE TABLE brands (

bid INT IDENTITY(1,1) PRIMARY KEY,

bname VARCHAR(20)

);

CREATE TABLE inv\_user (

user\_id VARCHAR(20) PRIMARY KEY,

name VARCHAR(20),

password VARCHAR(20),

last\_login DATETIME,

user\_type VARCHAR(10)

);

CREATE TABLE categories (

cid INT IDENTITY(1,1) PRIMARY KEY,

category\_name VARCHAR(20)

);

CREATE TABLE stores (

sid INT IDENTITY(1,1) PRIMARY KEY,

sname VARCHAR(20),

address VARCHAR(50),

mobno VARCHAR(10)

);

CREATE TABLE product (

pid INT IDENTITY(1,1) PRIMARY KEY,

cid INT,

bid INT,

sid INT,

pname VARCHAR(20),

p\_stock INT,

price DECIMAL(10, 2),

added\_date DATE,

image VARCHAR(255),

FOREIGN KEY (cid) REFERENCES categories(cid),

FOREIGN KEY (bid) REFERENCES brands(bid),

FOREIGN KEY (sid) REFERENCES stores(sid)

);

CREATE TABLE provides (

bid INT,

sid INT,

discount DECIMAL(5, 2),

FOREIGN KEY (bid) REFERENCES brands(bid),

FOREIGN KEY (sid) REFERENCES stores(sid)

);

CREATE TABLE customer\_cart (

cust\_id INT IDENTITY(1,1) PRIMARY KEY,

name VARCHAR(20),

mobno VARCHAR(10)

);

CREATE TABLE select\_product (

cust\_id INT,

pid INT,

quantity INT,

FOREIGN KEY (cust\_id) REFERENCES customer\_cart(cust\_id),

FOREIGN KEY (pid) REFERENCES product(pid)

);

CREATE TABLE transaction\_data (

id INT IDENTITY(1,1) PRIMARY KEY,

total\_amount DECIMAL(10, 2),

paid DECIMAL(10, 2),

due DECIMAL(10, 2),

gst DECIMAL(5, 2),

discount DECIMAL(5, 2),

payment\_method VARCHAR(10),

cart\_id INT,

FOREIGN KEY (cart\_id) REFERENCES customer\_cart(cust\_id)

);

CREATE TABLE invoice (

item\_no INT,

product\_name VARCHAR(20),

quantity INT,

net\_price DECIMAL(10, 2),

transaction\_id INT,

FOREIGN KEY (transaction\_id) REFERENCES transaction\_data(id)

);

-- Procedures

CREATE PROCEDURE get\_due\_amount(@c\_id INT)

AS

BEGIN

DECLARE @due1 DECIMAL(10, 2);

SELECT @due1 = due FROM transaction\_data WHERE cart\_id = @c\_id;

SELECT @due1 AS due\_amount;

END;

CREATE PROCEDURE show\_products

AS

BEGIN

DECLARE @done INT = 0;

DECLARE @p\_id INT;

DECLARE @p\_name NVARCHAR(20);

DECLARE @p\_stock INT;

DECLARE @cur CURSOR;

SET @cur = CURSOR FOR SELECT pid, pname, p\_stock FROM product;

OPEN @cur;

FETCH NEXT FROM @cur INTO @p\_id, @p\_name, @p\_stock;

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT CONCAT(@p\_id, ' ', @p\_name, ' ', @p\_stock) AS product\_details;

FETCH NEXT FROM @cur INTO @p\_id, @p\_name, @p\_stock;

END;

CLOSE @cur;

DEALLOCATE @cur;

END;

CREATE PROCEDURE check\_stock(@b INT)

AS

BEGIN

DECLARE @a INT;

SELECT @a = p\_stock FROM product WHERE pid = @b;

IF @a < 2

BEGIN

SELECT 'Stock is Less' AS stock\_status;

END

ELSE

BEGIN

SELECT 'Enough Stock' AS stock\_status;

END

END;

-- Triggers

CREATE TRIGGER after\_select\_product\_insert

ON select\_product

AFTER INSERT

AS

BEGIN

UPDATE product

SET p\_stock = p\_stock - i.quantity

FROM product

INNER JOIN inserted i ON product.pid = i.pid;

END;

CREATE TRIGGER after\_transaction\_update

ON transaction\_data

AFTER UPDATE

AS

BEGIN

IF UPDATE(paid)

BEGIN

UPDATE td

SET due = td.total\_amount - td.paid

FROM transaction\_data td

INNER JOIN inserted i ON td.id = i.id;

END;

END;

-- Insert demo data into stores

INSERT INTO stores (sname, address, mobno) VALUES

('Store One', '123 Main St, Cityville', '1234567890'),

('Store Two', '456 High St, Townsville', '0987654321'),

('Store Three', '789 Market St, Villagetown', '1122334455');

-- Insert demo data into categories

INSERT INTO categories (category\_name) VALUES

('Electronics'),

('Clothing'),

('Groceries');

-- Insert demo data into brands

INSERT INTO brands (bname) VALUES

('Brand A'),

('Brand B'),

('Brand C');

-- Insert demo data into products

INSERT INTO product (cid, bid, sid, pname, p\_stock, price, added\_date, image) VALUES

(1, 1, 1, 'Smartphone', 50, 299.99, '2023-05-20', NULL),

(2, 2, 2, 'Jeans', 100, 49.99, '2023-05-21', NULL),

(3, 3, 3, 'Apple', 200, 0.99, '2023-05-22', NULL);

# **DROP QUERRIES:**

DECLARE @sql NVARCHAR(MAX) = N'';

SELECT @sql += N'DROP TABLE ' + QUOTENAME(TABLE\_SCHEMA) + '.' + QUOTENAME(TABLE\_NAME) + ';

'

FROM demo.INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE = 'BASE TABLE';

EXEC sp\_executesql @sql;

DECLARE @sql NVARCHAR(MAX) = N'';

SELECT @sql += N'DROP PROCEDURE ' + QUOTENAME(SPECIFIC\_SCHEMA) + '.' + QUOTENAME(SPECIFIC\_NAME) + ';

'

FROM INFORMATION\_SCHEMA.ROUTINES

WHERE ROUTINE\_TYPE = 'PROCEDURE' AND SPECIFIC\_SCHEMA = 'dbo'; -- Specify the schema if necessary

EXEC sp\_executesql @sql;

DECLARE @sql NVARCHAR(MAX) = N'';

SELECT @sql += N'DROP TRIGGER ' + QUOTENAME(OBJECT\_SCHEMA\_NAME(parent\_id)) + '.' + QUOTENAME(name) + ';

'

FROM sys.triggers

WHERE parent\_class = 0; -- 0 indicates it's a user table trigger

EXEC sp\_executesql @sql;

# Config

const config = {

user: "admin",

password: "123",

server: "DESKTOP-0O8U1BU",

database: "demo",

options: {

trustServerCertificate: true,

trustConnection: false,

enableArithAbort: true,

instancename: "UMAIRKHAN",

},

port: 1433,

};