

MS SQL Sample data

1. store_db

customers

```
CREATE TABLE Customers (  
  
    customer_id INT IDENTITY(100,1) PRIMARY KEY,  
  
    customer_name VARCHAR(100) NOT NULL,  
  
    email VARCHAR(100) UNIQUE  
  
);
```

```
INSERT INTO Customers (customer_name, email)
```

```
VALUES
```

```
('Raju', 'raju@example.com'),
```

```
('Sham', 'sham@example.com'),
```

```
('Baburao', 'baburao@example.com');
```

Orders

```
CREATE TABLE Orders (  
  
    order_id INT IDENTITY(500,1) PRIMARY KEY,  
  
    order_date DATE NOT NULL,  
  
    total_amount DECIMAL(10, 2),  
  
    customer_id INT,  
  
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)  
  
);
```

```
INSERT INTO Orders (order_date, total_amount, customer_id)

VALUES

('2025-09-15', 1500.00, 100), -- This links to Raju (customer_id 100)

('2025-09-28', 800.00, 101), -- This links to Sham (customer_id 101)

('2025-10-05', 2200.00, 100), -- This links to Raju (customer_id 100)

('2025-10-12', 500.00, 102), -- This links to Baburao (customer_id 102)

('2025-10-17', 1200.00, 101); -- New order for Sham (customer_id 101)
```

=====

Institute

Table Creation

- **courses**

- **Create Table**

```
CREATE TABLE courses (
  course_id INT IDENTITY(1,1) PRIMARY KEY,
  course_name VARCHAR(100) NOT NULL,
  course_fee NUMERIC(10, 2) NOT NULL
);
```

- **Data**

```
INSERT INTO courses (course_name, course_fee)
VALUES
('Mathematics', 500.00),
('Physics', 600.00),
('Chemistry', 700.00);
```

- **students**

- **Create Table**

```
CREATE TABLE students (
  student_id INT IDENTITY(1,1) PRIMARY KEY,
  student_name VARCHAR(100) NOT NULL
);
```

- **Data**

```
INSERT INTO Students (student_name) VALUES  
('Raju'),  
('Sham'),  
('Baburao'),  
('Alex');
```

- **enrollment**

- **Create Table**

```
CREATE TABLE enrollment (  
    enrollment_id INT IDENTITY(1,1) PRIMARY KEY,  
    student_id INT NOT NULL,  
    course_id INT NOT NULL,  
    enrollment_date DATE NOT NULL,  
  
    FOREIGN KEY (student_id) REFERENCES students(student_id),  
    FOREIGN KEY (course_id) REFERENCES courses(course_id)  
);
```

- **Data**

```
INSERT INTO enrollment (student_id, course_id, enrollment_date)  
VALUES  
(1, 1, '2025-01-01'), -- Raju enrolled in Mathematics  
(1, 2, '2025-01-15'), -- Raju enrolled in Physics  
(2, 1, '2025-02-01'), -- Sham enrolled in Mathematics  
(2, 3, '2025-02-15'), -- Sham enrolled in Chemistry  
(3, 3, '2025-03-25'); -- Alex enrolled in Chemistry
```

SHOW DATA

```
SELECT s.student_name, c.course_name, e.enrollment_date, c.course_fee  
  
FROM enrollment e  
  
INNER JOIN students s ON e.student_id = s.student_id  
  
INNER JOIN courses c ON e.course_id = c.course_id
```

```
SELECT c.course_name, COUNT(s.student_id), SUM(c.course_fee)

FROM enrollment e

INNER JOIN students s ON e.student_id = s.student_id

INNER JOIN courses c ON e.course_id = c.course_id

GROUP BY c.course_name
```

=====

TASK E-StoreDB

- **customers**

```
CREATE TABLE customers (
    cust_id INT IDENTITY(1,1) PRIMARY KEY,
    cust_name VARCHAR(100) NOT NULL
);
```

```
INSERT INTO customers (cust_name)
VALUES
    ('Raju'), ('Sham'), ('Paul'), ('Alex'),('Baburao');
```

- **orders**

```
CREATE TABLE orders (
    ord_id INT IDENTITY(1,1) PRIMARY KEY,
    ord_date DATE NOT NULL,
    cust_id INT NOT NULL,
    FOREIGN KEY (cust_id) REFERENCES customers(cust_id) ON DELETE CASCADE
);
```

```
INSERT INTO orders (ord_date, cust_id)
VALUES
    ('2025-01-01', 1), -- Raju first order
    ('2025-02-01', 2), -- Sham first order
    ('2025-03-01', 3), -- Paul first order
```

('2025-04-04', 2); -- Sham second order

- **products**

```
CREATE TABLE products (  
  p_id INT IDENTITY(1,1) PRIMARY KEY,  
  p_name VARCHAR(100) NOT NULL,  
  price NUMERIC NOT NULL  
);
```

```
INSERT INTO products (p_name, price)  
VALUES  
  ('Laptop', 55000.00),  
  ('Mouse', 500),  
  ('Keyboard', 800.00),  
  ('Cable', 250.00),  
  ('Monitor', 12000.00);
```

- **order_items**

```
CREATE TABLE order_items (  
  item_id INT IDENTITY(1,1) PRIMARY KEY,  
  ord_id INT NOT NULL,  
  p_id INT NOT NULL,  
  quantity INT NOT NULL,  
  FOREIGN KEY (ord_id) REFERENCES orders(ord_id),  
  FOREIGN KEY (p_id) REFERENCES products(p_id)  
);
```

```
INSERT INTO order_items (ord_id, p_id, quantity)  
VALUES  
  (1, 1, 1), -- Raju ordered 1 Laptop  
  (1, 4, 2), -- Raju ordered 2 Cables  
  (2, 1, 1), -- Sham ordered 1 Laptop  
  (3, 2, 1), -- Paul ordered 1 Mouse  
  (3, 4, 5), -- Paul ordered 5 Cables  
  (4, 3, 1); -- Sham ordered 1 Keyboard
```

=====

To see overall report

	cust_name character varying (100) 🔒	ord_date date 🔒	p_name character varying (100) 🔒	price numeric 🔒	quantity integer 🔒	total_price numeric 🔒
1	Raju	2024-01-01	Laptop	55000.00	1	55000.00
2	Raju	2024-01-01	Cable	250.00	2	500.00
3	Sham	2024-02-01	Laptop	55000.00	1	55000.00
4	Paul	2024-03-01	Mouse	500	1	500
5	Paul	2024-03-01	Cable	250.00	5	1250.00
6	Sham	2024-04-04	Keyboard	800.00	1	800.00

SELECT

c.cust_name,
o.ord_date,
p.p_name,
p.price,
oi.quantity,
(oi.quantity*p.price) AS total_price

FROM order_items oi

JOIN

products p ON oi.p_id=p.p_id

JOIN

orders o ON o.ord_id=oi.ord_id

JOIN

customers c ON o.cust_id=c.cust_id;

All Queries

-- Database

CREATE DATABASE test;

-- Select a Database

USE test;

-- To check the selected/current Database

SELECT db_name();

-- To Delete/Drop Database

DROP DATABASE test;

-- Creating Table

CREATE TABLE users (

id INT,

name VARCHAR(100),

city VARCHAR(100)

);

-- Checking Table Created

EXEC sp_help 'users';

-- Inserting sample data

INSERT INTO users(id, name, city)

VALUES (101, 'Raju', 'Delh');

-- Inserting multiple data

INSERT INTO users

VALUES (102, 'Sham', 'Bhopal'), (103, 'Baburao', 'Mumbai');

-- READ Data

SELECT * FROM users; -- All Data

SELECT name FROM users; -- Only name column

SELECT * FROM users WHERE name='Raju' -- Only show data for Raju

-- UPDATE Data

UPDATE users SET city='London' WHERE name='Sham'

-- DELETE Data

DELETE FROM users WHERE name='Baburao'

-- Employee Table using IDENTITY, Primary Key, Unique key, Default

CREATE TABLE employees (

emp_id INT IDENTITY(1,1) PRIMARY KEY,

fname VARCHAR(50) NOT NULL,

lname VARCHAR(50) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

dept VARCHAR(50),

salary DECIMAL(10,2) DEFAULT 30000.00,

hire_date DATE NOT NULL DEFAULT GETDATE())

);

-- Inserting employee data

INSERT INTO employees (fname, lname, email, dept, salary, hire_date)

VALUES

('Raj', 'Sharma', 'raj.sharma@example.com', 'IT', 50000.00, '2020-01-15'),

('Priya', 'Singh', 'priya.singh@example.com', 'HR', 45000.00, '2019-03-22'),

('Arjun', 'Verma', 'arjun.verma@example.com', 'IT', 55000.00, '2021-06-01'),

('Suman', 'Patel', 'suman.patel@example.com', 'Finance', 60000.00, '2018-07-30'),

('Kavita', 'Rao', 'kavita.rao@example.com', 'HR', 47000.00, '2020-11-10'),

('Amit', 'Gupta', 'amit.gupta@example.com', 'Marketing', 52000.00, '2020-09-25'),

('Neha', 'Desai', 'neha.desai@example.com', 'IT', 48000.00, '2019-05-18'),

('Rahul', 'Kumar', 'rahul.kumar@example.com', 'IT', 53000.00, '2021-02-14'),

('Anjali', 'Mehta', 'anjali.mehta@example.com', 'Finance', 61000.00, '2018-12-03'),

('Vijay', 'Nair', 'vijay.nair@example.com', 'Marketing', 50000.00, '2020-04-19');

-- CLAUSES DISTINCT | TOP | LIKE | ORDER BY

SELECT DISTINCT dept FROM employees; -- To only show dept

SELECT TOP 3 * from employees; -- Top 3 rows

SELECT * FROM employees ORDER BY fname; -- Sorting data based on first name

-- LIKE --

SELECT * FROM employees WHERE hire_date LIKE '%2020%' -- 2020 present in column

SELECT * FROM employees WHERE fname LIKE 'A%' -- Name start with A

SELECT * FROM employees WHERE fname LIKE '%a' -- Name end with a

SELECT * FROM employees WHERE fname LIKE '____' -- Name with 4 letters

-- Relational Operators --

SELECT * FROM employees WHERE salary > 50000;

SELECT * FROM employees WHERE salary != 50000;

-- Logical Operators --

SELECT * FROM employees WHERE dept='IT' AND salary>50000;

SELECT * FROM employees WHERE dept='IT' OR salary=50000;

-- IN, NOT IN, BETWEEN ----

SELECT * FROM employees WHERE dept IN ('HR', 'Marketing');

SELECT * FROM employees WHERE dept NOT IN ('HR', 'Marketing');

SELECT * FROM employees WHERE salary BETWEEN 50000 AND 60000;

-- Aggregate Functions COUNT, MIN, MAX, AVG, SUM --

SELECT COUNT(salary) FROM employees;

SELECT SUM(salary) FROM employees;

SELECT MIN(salary) FROM employees;

SELECT MAX(salary) FROM employees;

SELECT AVG(salary) FROM employees;

-- GROUP BY -----

SELECT dept FROM employees GROUP BY dept;

SELECT dept, COUNT(emp_id) FROM employees GROUP BY dept;

-- String Functions

```
SELECT CONCAT('Hello', 'BUDDY');

SELECT CONCAT_WS('-', 'One', 'Two', 'Three')

SELECT SUBSTRING('Hey Buddy', 1, 4);

SELECT REPLACE('Hey Buddy', 'Hey', 'Hello')

SELECT REVERSE('Hello World');

SELECT LEN('Hello World');

SELECT UPPER('Hello World');

SELECT LOWER('Hello World');

SELECT CHARINDEX('OM','ThOMAS');

SELECT TRIM(' Alright! ');
```

-- Exercise

```
SELECT CONCAT_WS(':', emp_id, fname, lname, dept) FROM employees

SELECT CONCAT_WS(':', emp_id, CONCAT_WS(' ', fname, lname), dept) FROM employees

SELECT CONCAT_WS(':', emp_id, fname, UPPER(dept)) FROM employees WHERE emp_id=4

SELECT CONCAT(LEFT(dept,1), emp_id), fname FROM employees
```

```
SELECT LEFT(dept, 1) FROM employees

SELECT RIGHT(dept, 1) FROM employees

SELECT * FROM employees WHERE fname LIKE 'A%'
```

```
SELECT CONCAT(fname, lname) FROM employees;

SELECT CONCAT_WS('-', fname, lname) FROM employees;
```

----- ALTER Table -----

-- Adding new Column to a table

ALTER TABLE employees ADD city VARCHAR(50);

SELECT * FROM employees;

-- Dropping Column from a table

ALTER TABLE employees DROP COLUMN city;

-- Renaming Column name

EXEC sp_rename 'employees.first_name', 'fname', 'COLUMN';

-- Modify Column datatype

Alter table employees

Alter column fname VARCHAR(60);

-- Modify Column to add a Default Constraint

ALTER TABLE employees

ADD CONSTRAINT default_dept DEFAULT 'Trainee' FOR dept;

----- CHECK CONSTRAINT -----

create table contacts(

name varchar(50),

mobile varchar(20) UNIQUE CHECK (Len(mobile)>=10)

)

insert into contacts values ('raju', '123456780')

-- Check our constraint

EXEC sp_help 'contacts';

----- CASE -----

SELECT fname, salary,

CASE

 WHEN salary >=50000 Then 'High'

 ELSE 'Low'

END as sal_stat

FROM employees

----- Relationship -----

CREATE TABLE customers (

 cust_id INT IDENTITY(1,1) PRIMARY KEY,

 cust_name VARCHAR(100) NOT NULL

);

CREATE TABLE orders (

 ord_id INT IDENTITY(1,1) PRIMARY KEY,

 ord_date DATE NOT NULL,

 price DECIMAL(10, 2) NOT NULL,

 cust_id INT NOT NULL,

 FOREIGN KEY (cust_id) REFERENCES customers(cust_id)

);

INSERT INTO customers (cust_name)

VALUES

 ('Raju'), ('Sham'), ('Paul'), ('Alex');

INSERT INTO orders (ord_date, cust_id, price)

VALUES

('2024-01-01', 1, 250.00),

('2024-01-15', 1, 300.00),

('2024-02-01', 2, 150.00),

('2024-03-01', 3, 450.00),

('2024-04-04', 2, 550.00);

select * from customers;

select * from orders;

-- JOINS -----

---- CROSS JOIN -----

SELECT * FROM customers

CROSS JOIN orders;

---- INNER JOIN -----

SELECT * FROM customers

INNER JOIN orders

ON customers.cust_id=orders.cust_id;

SELECT cust_name FROM customers

INNER JOIN orders

**ON customers.cust_id=orders.cust_id GROUP BY cust_name; -- Customer names who made
atleast one order**

---- LEFT JOIN -----

SELECT * FROM customers

LEFT JOIN orders

ON customers.cust_id=orders.cust_id;

---- RIGHT JOIN -----

SELECT * FROM customers

RIGHT JOIN orders

ON customers.cust_id=orders.cust_id;

SELECT

c.cust_id,

c.cust_name,

o.ord_id,

o.ord_date,

o.price

FROM

customers c

FULL JOIN

orders o ON c.cust_id = o.cust_id;