Create, Delete and Update Queries

```
-- INSERT QUERIES
-- Case insensitive

USE sql_store;

SELECT *
FROM customers;

-- A single insert
INSERT INTO customers
VALUES (11, 'Sumeet', 'Malik', '1988-08-11', '999', 'XYZ', 'Atlanta', 'GA', 800);

-- Multiple rows insertion
INSERT INTO customers
VALUES
(12, 'ABC', 'DEF', '1988-08-11', '999', 'XYZ', 'Atlanta', 'GA', 800),
(13, 'GHI', 'JKL', '1988-08-11', '999', 'XYZ', 'Atlanta', 'GA', 800);

-- Defaults = not providing values for columns who can make their own value
INSERT INTO customers
VALUES
(DEFAULT, 'ABC', 'DEF', '1988-08-11', '999', 'XYZ', 'Atlanta', 'GA', 800),
(DEFAULT, 'GHI', 'JKL', '1988-08-11', '999', 'XYZ', 'Atlanta', 'GA', 800);

-- Named column
INSERT INTO customers (first_name, last_name, birth_date, phone, address, city, state, points)
VALUES
('ABC', 'DEF', '1988-08-11', '999', 'XYZ', 'Atlanta', 'GA', 800),
('GHI', 'JKL', '1988-08-11', '999', 'XYZ', 'Atlanta', 'GA', 800);

 -- Skipping nullable columns and auto-increment
 INSERT INTO customers (first_name, last_name, address, city, state, points)
VALUES
('ABC', 'DEF', 'XYZ', 'Atlanta', 'GA', 800),
('GHI', 'JKL', 'XYZ', 'Atlanta', 'GA', 800);

-- Reordering columns
INSERT INTO customers (address, first_name, last_name, city, state, points)
VALUES
('XYZ', 'ABC', 'DEF', 'Atlanta', 'GA', 800),
```

```
('XYZ', 'GHI', 'JKL', 'Atlanta', 'GA', 800);


-- Update Queries
UPDATE customers
SET first_name = 'ABCD', last_name = 'EF'
WHERE customer_id = 12;

-- increase points of customers with id <= 10 by 1000
UPDATE customers
SET points = points + 1000
WHERE customer_id <= 10;

-- Always add the where clause for update and delete queries

-- Delete Queries
DELETE FROM customers
WHERE customer_id >= 18;
```

Select Queries

```
USE sql_store;

SELECT 34, "Sumeet", "Malik";

 -- selecting all columns
SELECT *
FROM customers;

-- operations on columns and alias
SELECT first_name, last_name, city, 2 * points AS TwiceOfPoints, points
FROM customers;

-- DISTINCT in action
SELECT DISTINCT first_name
FROM customers;

SELECT DISTINCT first_name, last_name
FROM customers;

-- by default string operations are case insensitive
```

```
SELECT *
FROM customers
WHERE first_name = 'abc';

-- find all customers with more than 500 points and living in GA state
-- both the conditions must be true
SELECT *
FROM customers
WHERE state = 'GA' AND points >= 500;

-- OR = either of the conditions should be true
-- find all customers who live in Chicago or have more than 2000 points
SELECT *
FROM customers
WHERE city = 'Chicago' OR points >= 2000;

-- NOT reverses the entire meaning (will select everything except 1, 3, 5, 6, 7, 9)
SELECT *
FROM customers
WHERE NOT (city = 'Chicago' OR points >= 2000);

-- city != 'Chicago' AND points < 2000

-- Guideline : Always specify the parenthesis when combining conditions via AND, OR, NOT

-- GIVE ME CUSTOMERS WHOSE CUSTOMER ID IS EITHER 2 OR 4 OR 8 OR 10 OR 11
-- WRONG WAY TO WRITE
SELECT *
FROM customers
WHERE customer_id = 2 OR
    customer_id = 4 OR
    customer_id = 8 OR
    customer_id = 10 OR
    customer_id = 11;

-- Better way = IN keyword
SELECT *
FROM customers
WHERE customer_id IN (2, 4, 8, 10, 11);

-- Get customers born between 1990 and 1995
SELECT *
FROM customers
WHERE birth_date >= '1990-01-01' AND birth_date <= '1995-12-31';
```

```sql
SELECT *
FROM customers
WHERE birth_date BETWEEN '1990-01-01' AND '1995-12-31';

SELECT *
FROM customers
WHERE points BETWEEN 1200 AND 1457;

SELECT *
FROM customers
WHERE city LIKE '%c_g%';


-- %Aug% = Aug, 2022Aug, Aug2022

-- customers wtih name starting with a or b or c
SELECT *
FROM customers
WHERE first_name LIKE 'a%' OR first_name LIKE 'b%' OR first_name LIKE 'c%';

-- IS NULL (customers with no phone number)
SELECT *
FROM customers
WHERE phone IS NULL;

SELECT (NULL = NULL), (NULL IS NULL);

-- ORDER BY
SELECT *
FROM customers
ORDER BY points DESC, first_name, last_name DESC;

-- LIMIT
SELECT *
FROM customers
ORDER BY points DESC
LIMIT 5;

-- sharma, sharmaa, sharmb

SELECT *
FROM customers
WHERE LEFT(first_name, 1) BETWEEN 'a' AND 'c';
```

```
SELECT LEFT(first_name, 10)
FROM customers;
```

JOINS

```
USE sql_store;

SELECT *
FROM customers;

SELECT *
FROM orders;

-- Get customer firstname, phone, order_id, and order_date for orders between a begin date
and an end date

SELECT c.first_name, c.phone, o.order_id, o.order_date
FROM customers c
JOIN orders o
        ON o.customer_id = c.customer_id
WHERE o.order_date BETWEEN '2018-01-01' AND '2018-12-31';

USE sql_hr;
SELECT *
FROM employees;

-- Get employee, manager for all the employees who have managers
SELECT e.first_name, m.first_name
FROM employees e
JOIN employees m
        ON e.reports_to = m.employee_id;


USE sql_store;

SELECT *
FROM customers;

SELECT *
FROM orders;

SELECT *
```

```
FROM order_items;

SELECT *
FROM products;

-- get (customer first_name, product name) for all the products ordered by customers living in
MA and VA state
SELECT c.first_name, p.name, o.order_id, o.order_date
FROM customers c
JOIN orders o
        ON c.customer_id = o.customer_id
JOIN order_items oi
        ON o.order_id = oi.order_id
JOIN products p
        ON oi.product_id = p.product_id
WHERE c.state = 'MA' OR c.state = 'VA';


-- using keyword
SELECT c.customer_id, o.order_id
FROM customers c
JOIN orders o
        USING (customer_id);

-- natural join
SELECT c.customer_id, o.order_id
FROM customers c
NATURAL JOIN orders o;

-- all customers with all products
SELECT c.customer_id, p.product_id
FROM customers c
CROSS JOIN products p


-- a bad way of writing cross join
SELECT c.customer_id, p.product_id
FROM customers c, products p;


-- bad way to write an inner join
SELECT c.first_name, c.phone, o.order_id, o.order_date
FROM customers c, orders o
```

```
WHERE o.customer_id = c.customer_id AND o.order_date BETWEEN '2018-01-01' AND
'2018-12-31';
```