## Sql From Programming with Mosh :

### Select Clause :

To select data from tables

```
SELECT
    last_name,
    first_name,
    points,
    (points + 10) * 100 AS 'discount factor'
FROM customers
```

### Where Clause :

To select data from tables with some conditions

```
>
>=
<
<=
=
!=
<>
```

```sql
SELECT *
FROM Customers
WHERE state = 'VA'
```

```sql
SELECT *
FROM orders
WHERE order_date >= '2019-01-01'
```

## And, Or, Not :

Just like if else in programming :

```sql
SELECT *
FROM Customers
WHERE birth_date > '1990-01-01' OR
      (points > 1000 AND state = 'VA')
```

## IN / Between operator :

IN - Just like in python to check whether a char present or not
Between - range (includes st and end)

## LIKE Operator:

It is like string slicing using % for multiple characters
And _ for single character

```sql
SELECT *
FROM customers
WHERE last_name LIKE 'b____y'
-- % any number of characters
-- _ single character
```

```sql
SELECT *
FROM customers
WHERE address LIKE '%trail%' OR
      address LIKE '%avenue%'
```

**REGEXP Operator:**

Just like (LIKE) operator but advanced

^char if it should be in first
char$ if it should be in last
Char if it is present

```sql
SELECT *
FROM customers
WHERE last_name LIKE '%field%'
WHERE last_name REGEXP 'field'
```

Both are same

```sql
SELECT *
FROM customers
WHERE last_name REGEXP 'field|mac|rose'
```

Like using OR for multiple values

```sql
SELECT *
FROM customers
WHERE last_name REGEXP '[gim]e'
ge
ie
me
```

Same can be done after char e[....]

```sql
SELECT *
FROM customers
WHERE last_name REGEXP '[a-h]e'
-- ^ beginning
-- $ end
-- | logical or
-- [abcd]
-- [a-f]
```

## IS NULL :

Selects values only if NULL (not present)

```
SELECT *
FROM orders
WHERE shipper_id IS NULL
```

## ORDER BY :

Default by primary key column, so we can change order by using this clause

```
SELECT *
FROM customers
ORDER BY state, first_name
```

First looks at State, then if more than 1 present, looks at first_name

```sql
Use sql_store;

SELECT *, (quantity * unit_price) AS total_price
FROM order_items
WHERE order_id = 2
ORDER BY (quantity * unit_price)
```

| order_id | product_id | quantity | unit_price | total_price |
|---|---|---|---|---|
| 2 | 6 | 2 | 2.94 | 5.88 |
| 2 | 4 | 4 | 1.66 | 6.64 |
| 2 | 1 | 2 | 9.10 | 18.20 |

Created a new column for clarity

## LIMIT OPERATOR :

When using select clause, we can limit our needs

```sql
SELECT *
FROM customers
LIMIT 6, 3
-- page 1: 1 - 3
-- page 2: 4 - 6
-- page 3: 7 - 9
```

6 is offset which skips

```sql
SELECT
    *
FROM customers
ORDER BY points DESC
LIMIT 3
```

Top 3

## Order of Clauses :

```sql
SELECT *
FROM customers
WHERE
ORDER BY points DESC
LIMIT 3
```

## INNER JOIN :

Inner is optional, it's inner by default

```
SELECT
    order_id, customers.customer_id, first_name, last_name, phone
FROM orders
JOIN customers
    ON orders.customer_id = customers.customer_id;
```

## Syntax :

JOIN **target_table**
ON **source_table.column_name = target_table.column_name**

We can also use alias for table names
O for orders
C for customers

```
SELECT
    order_id, c.customer_id, first_name, last_name, phone
FROM orders o
JOIN customers c
    ON o.customer_id = c.customer_id;
```

Remember like variable names

```sql
SELECT
    order_id, p.product_id, name,
    (quantity * o.unit_price) AS total_price
FROM order_items o
JOIN products p
ON o.product_id = p.product_id
```

If both tables have same column_name, must specify variable name at beginning

**Joining across different DB :**

```sql
SELECT
    order_id, p.product_id, name,
    (quantity * o.unit_price) AS total_price
FROM order_items o
JOIN sql_inventory.products p
ON o.product_id = p.product_id
```

## SELF JOIN :

We can join info within same table

```
SELECT
    e.first_name,
    e.job_title,
    e.employee_id,
    m.first_name AS manager_name
FROM employees e
JOIN employees m
    ON e.reports_to = m.employee_id
```

**Only employee table**

Manager is also an employee, so we got infor about manager within this table using JOIN

## JOINING multiple tables :

```
SELECT
    order_id, c.first_name, o.order_date, os.name
FROM orders o
JOIN order_statuses os
    ON o.status = os.order_status_id
JOIN customers c
    ON o.customer_id = c.customer_id;
```

Same join syntax,
But two times

```sql
SELECT
    p.payment_id,
    p.date,
    pm.name AS payment_method,
    c.name AS client_name,
    c.address AS client_address
FROM payments p
JOIN payment_methods pm
    ON p.payment_method = pm.payment_method_id
JOIN clients c
    ON p.client_id = c.client_id
```

3 tables joined between payment, payment_methods, clients

## OUTER JOIN :

When using INNER join, if any value in table 1 has no value in table 2, it just skips

For example, 6 out of 10 customers have ordered,
So when joining customers and orders tables,
The final table will skip 4 customers who didn't order anything.

```sql
JOIN orders o
    ON c.customer_id = o.customer_id
```

If there is no customer id in orders table, it skips, that's it.

## LEFT JOIN :

Gets all values from left table even when missing in right table
Puts null in right table values when missing

```sql
SELECT
    c.customer_id,
    c.first_name,
    o.order_id
FROM customers c
LEFT JOIN orders o
    ON c.customer_id = o.customer_id
ORDER BY c.customer_id
```

Gets all customers values and puts null in orders if not present

```sql
SELECT
    p.product_id,
    p.name AS product_name,
    oi.quantity
FROM products p
LEFT JOIN order_items oi
    ON p.product_id = oi.product_id;
```

Gets all products values and puts null in order items if not present

## RIGHT JOIN :

Same as LEFT JOIN, gets all values from righttable even when missing in left table
Puts null in left table values when missing

```sql
SELECT
    p.product_id,
    p.name AS product_name,
    oi.quantity
FROM order_items oi
RIGHT JOIN products p
    ON p.product_id = oi.product_id;
```

Order items and products joins
Gets all values from right side (products)
And puts null in orders if not available

Multiple tables OUTER JOIN :

Same like INNER JOIN, we can JOIN multiple tables using LEFT or RIGHT
Even when some values missing, it puts null

```sql
SELECT
    c.customer_id,
    c.first_name,
    o.order_id,
    sh.name AS shipper
FROM customers c
LEFT JOIN orders o
    ON c.customer_id = o.customer_id
LEFT JOIN shippers sh
    ON o.shipper_id = sh.shipper_id;
```

Here,
**Main table is customers, shows all customers**
**And shows orders, when customer didnt order shows null**
**And there is no order means, shipper also null**

## JOINS FINAL QUESTION :

```sql
SELECT
    o.order_date,
    o.order_id,
    c.first_name,
    sh.name AS shipper,
    os.name AS status
FROM orders o
LEFT JOIN customers c
    ON o.customer_id = c.customer_id
LEFT JOIN shippers sh
    ON o.shipper_id = sh.shipper_id
LEFT JOIN order_statuses os
    ON o.status = os.order_status_id
ORDER BY status
```

| order_date | order_id | first_name | shipper | status |
|---|---|---|---|---|
| 2019-01-30 | 1 | Elka | NULL | Processed |
| 2017-12-01 | 3 | Thacher | NULL | Processed |
| 2017-01-22 | 4 | Ines | NULL | Processed |
| 2018-11-18 | 6 | Levy | NULL | Processed |
| 2018-06-08 | 8 | Clemmie | NULL | Processed |
| 2018-08-02 | 2 | Ilene | Mraz, Renner and Nolan | Shipped |
| 2017-08-25 | 5 | Clemmie | Satterfield LLC | Shipped |
| 2018-09-22 | 7 | Ines | Mraz, Renner and Nolan | Shipped |
| 2017-07-05 | 9 | Levy | Hettinger LLC | Shipped |
| 2018-04-22 | 10 | Elka | Schinner-Predovic | Shipped |

Combined 4 tables
Main table = orders

Shows everything in orders
Shows customer who ordered

Shows shipper who shipped, if it's still processing shows null in shipper
And ORDERED BY shipping status

## USING keyword :

If column names of two tables are same, we can use this keyword

```
SELECT
    o.order_id,
    c.first_name
FROM orders o
JOIN customers c
    -- ON o.customer_id = c.customer_id
    USING (customer_id)
```

## NATURAL JOIN :

If two columns from tables have same name,
We can use this without USING or ON

```sql
SELECT
    o.order_id,
    c.first_name
FROM orders o
NATURAL JOIN customers c
```

## UNION :

Combine multiple queries

```sql
SELECT
    order_id,
    customer_id,
    'Active' AS status
FROM orders
WHERE order_date >= '2018-01-01'
UNION
SELECT
    order_id,
    customer_id,
    'Archived' AS status
FROM orders
WHERE order_date < '2018-01-01';
```

```sql
SELECT first_name
FROM customers
UNION
SELECT name
FROM shippers
```

Can use it to combine values from multiple tables too.