MySql

➔ For creating table

```
CREATE TABLE IF NOT EXISTS tasks (
    task_id INT AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    start_date DATE,
    due_date DATE,
    priority TINYINT NOT NULL DEFAULT 3,
    description TEXT,
    PRIMARY KEY (task_id)
);
```

➔ Insert data into a table

```
INSERT INTO tasks(title,start_date,due_date,description)
VALUES('Assignment 4',CURRENT_DATE(),CURRENT_DATE(),"This assignment is related to ai")
```

➔ Update data into the table

```
UPDATE tasks
SET
    priority = 4
WHERE
    task_id = 3;
```

Filtering data:-

➔ `Distinct:-`

When querying data from a table, you may get duplicate rows. In order to remove these duplicate rows, you use the `Distict` clause in the select statement.

```
SELECT DISTINCT name FROM Student;
```

➔ **Like:-**

The `LIKE` operator is a logical operator that tests whether a string contains a specified pattern or not.

```
SELECT * FROM Student where name LIKE 'a%'
```

```
SELECT * FROM Student where name LIKE '%a'
```

```
SELECT * FROM Student where name LIKE '%a%'
```

➔ **Limit:-**

The `LIMIT` clause is used in the `SELECT` statement to constrain the number of rows to return.

```
SELECT * FROM `Student` LIMIT 2,4
```

➔ Here 2 is offset and 4 is row_count.
➔ The `offset` specifies the offset of the first row to return. The `offset` of the first row is 0, not 1.
➔ The `row_count` specifies the maximum number of rows to return.

➔ Join

Create table Student with following fields

Id	Int & Auto-increment
Name	varchar
Age	Int
City	Int

Create a city table with the following fields

cid	Int
City	varchar

- ➔ Inner join Query

  SELECT * from Student Inner JOIN City on Student.city=City.cid;
  Or
  SELECT * from Student JOIN City on Student.city=City.cid;
  Or
  SELECT * from Student s Inner JOIN City c on s.city=c.cid;

- ➔ Left join

  SELECT * from Student s LEFT JOIN City c on s.city=c.cid

  It will display all the common records from the left table and right table and all records from the left table.

- ➔ Right join

  SELECT * from Student s RIGHT JOIN City c on s.city=c.cid

  It will display all the common records from the right table and left table and all records from the Right table.

➔ Alias for columns

SELECT CONCAT_WS(', ', id, name) AS `Full name` FROM Student

Grouping data:-

- ➔ Group By:-
  The `GROUP BY` clause groups a set of rows into a set of summary rows by values of columns or expressions.

  The aggregate function allow you to perform the calculation of a set of rows and return a single value. The `GROUP BY` clause is often used with an aggregate function to perform calculation and return a single value for each subgroup.

  SELECT status,COUNT(*) FROM orders GROUP BY status DESC;

- ➔ Having :- filter the groups by a specific condition.

  SELECT status,COUNT(*) FROM orders GROUP BY status HAVING COUNT(*)>4;

Sub query:-

➔ A MySQL subquery is called an inner query while the query that contains the subquery is called an outer query. A subquery can be used anywhere that expression is used and must be closed in parentheses.

SELECT * from Student WHERE city in (SELECT cid from City)

Derived table:-

➔ A derived table is a virtual table returned from a `select` statement. A derived table is similar to a temporary table, but using a derived table in the `SELECT` statement is much simpler than a temporary table because it does not require steps of creating the temporary table.

Exists:-

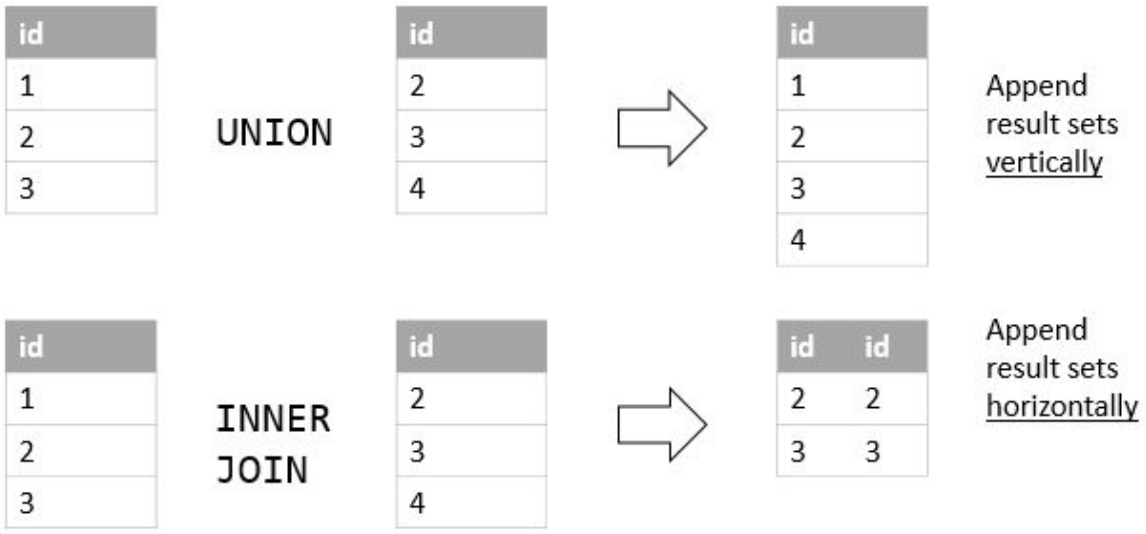
➔ The `EXISTS` operator is a Boolean operator that returns either true or false. The `EXISTS` operator is often used to test for the existence of rows returned by the subquery.

```
SELECT customerNumber, customerName FROM customers WHERE
EXISTS( SELECT 1 FROM orders WHERE orders.customernumber =
customers.customernumber);
```

Set operators:-

➔ Union:-

A `JOIN` combines result sets horizontally, a `UNION` appends result set vertically.



MySQL `UNION` operator allows you to combine two or more result sets of queries into a single result set.

```
SELECT id FROM Student UNION SELECT cid FROM City
```

➔ **Intersect:-**

The `INTERSECT` operator is a set operator that returns only distinct rows of two queries or more queries.

```
SELECT DISTINCT id FROM Student WHERE id IN (SELECT cid FROM City)
```

➔ **Minus:-**

SELECT id FROM t1 MINUS SELECT id FROM t2;