SELECT
SELECT col1, col2
FROM table
WHERE condition
GROUP BY cols
HAVING condition
ORDER BY col;

Modifying Data

INSERT:
INSERT INTO tablename (col1, col2...) VALUES (val1, val2);

INSERT From Table:
INSERT INTO tablename (col1, col2...)
SELECT col1, col2...

UPDATE:
UPDATE tablename SET col1 = val1
WHERE condition;

DELETE:
DELETE FROM tablename WHERE condition;

TRUNCATE:
TRUNCATE TABLE tablename;

UPDATE with Join:
UPDATE t
SET col1 = val1
FROM tablename t
INNER JOIN table x ON t.id = x.tid
WHERE condition;

MERGE:
MERGE INTO table_name
USING table_name
ON (condition)
WHEN MATCHED THEN update_clause
DELETE where_clause
WHEN NOT MATCHED THEN insert_clause;

SELECT Keywords

DISTINCT: Removes duplicate results

BETWEEN: Matches a value between two other values (inclusive)

IN: Matches a value to one of many values

LIKE: Performs partial/wildcard matches

Order of Processing
1. FROM
2. JOIN
3. WHERE
4. GROUP BY
5. HAVING
6. SELECT
7. DISTINCT
8. ORDER BY
9. LIMIT

Joins

SELECT t1.*, t2.*
FROM t1
JOIN t2 ON t1.col = t2.col;

INNER JOIN: show all matching records in both tables.

LEFT JOIN: show all records from left table, and any matching records from right table.

RIGHT JOIN: show all records from right table, and any matching records from left table.

FULL JOIN: show all records from both tables, whether there is a match or not.

CROSS JOIN: show all combinations of records from both tables.

SELF JOIN: join a table to itself. Used for hierarchical data.

SELECT p.*, c.*
FROM yourtable p
INNER JOIN yourtable c ON p.id = c.parent_id;
### Create Table

**Create Table:**

CREATE TABLE tablename (column_name data_type);

**Create Table With Constraints:**

CREATE TABLE tablename (column_name data_type NOT NULL,
CONTRAINT pkname PRIMARY KEY (col),
CONTRAINT fkname FOREIGN KEY (col)
REFERENCES other_table(col_in_other_table),
CONTRAINT ucname UNIQUE (col),
CONTRAINT ckname CHECK (conditions));

**Drop Table:**

DROP TABLE tablename;

**Create Temporary Table:**

CREATE TEMPORARY TABLE tablename (colname datatype);

### Alter Table

**Add Column**

ALTER TABLE tablename ADD columnname datatype;

**Drop Column**

ALTER TABLE tablename DROP COLUMN columnname;

**Modify Column**

ALTER TABLE tablename CHANGE columnname newcolumnname newdatatype; [MySQL]

**Rename Column**

ALTER TABLE tablename CHANGE COLUMN currentname TO newname;

**Add Constraint**

ALTER TABLE tablename ADD CONSTRAINT constraintname constrainttype (columns);

**Drop Constraint**

ALTER TABLE tablename DROP constraint_type constraintname;

**Rename Table**

ALTER TABLE tablename RENAME TO newtablename;

### Indexes

**Create Index:**

CREATE INDEX indexname ON tablename (cols);

**Drop Index:**

DROP INDEX indexname;

### Set Operators

**UNION:** Shows unique rows from two result sets.

**UNION ALL:** Shows all rows from two result sets.

**INTERSECT:** Shows rows that exist in both result sets.

**MINUS:** Shows rows that exist in the first result set but not the second.

### Analytic Functions

function_name (arguments) OVER ([query_partition_clause]
[ORDER BY order_by_clause
[windowing_clause]]);

Example using RANK, showing the student details and their rank according to the fees_paid, grouped by gender:

```sql
SELECT student_id, first_name, last_name, gender, fees_paid,
RANK() OVER (PARTITION BY gender ORDER BY fees_paid) AS rank_val
FROM student;
```

### CASE Statement

**Simple Case:**

CASE name
WHEN 'John' THEN 'Name John'
WHEN 'Steve' THEN 'Name Steve'
ELSE 'Unknown'
END

**Searched Case:**


CASE
  WHEN name='John' THEN 'Name John'
  WHEN name='Steve' THEN 'Name Steve'
  ELSE 'Unknown'
END

With Clause/Common Table Expression
WITH queryname AS (  
  SELECT col1, col2  
  FROM firsttable)
SELECT col1, col2.
FROM queryname...;

Subqueries
Single Row:
SELECT id, last_name, salary  
FROM employee  
WHERE salary = (  
  SELECT MAX(salary)  
  FROM employee  
);

Multi Row
SELECT id, last_name, salary  
FROM employee  
WHERE salary IN (  
  SELECT salary  
  FROM employee  
    WHERE last_name LIKE 'C%'  
);

Aggregate Functions
SUM: Finds a total of the numbers provided
COUNT: Finds the number of records
AVG: Finds the average of the numbers provided
MIN: Finds the lowest of the numbers provided
MAX: Finds the highest of the numbers provided

Common Functions
LENGTH(string): Returns the length of the provided string
INSTR(string, substring): Returns the position of the substring within the specified string.
CAST(expression AS datatype): Converts an expression into the specified data type.
ADDDATE(input_date, days): Adds a number of days to a specified date.
NOW: Returns the current date, including time.
CEILING(input_val): Returns the smallest integer greater than the provided number.
FLOOR(input_val): Returns the largest integer less than the provided number.
ROUND(input_val, [round_to]): Rounds a number to a specified number of decimal places.
TRUNCATE(input_value, num_decimals): Truncates a number to a number of decimals.

REPLACE(whole_string, string_to_replace, replacement_string): Replaces one string inside the whole string with another string.
SUBSTRING(string, start_position): Returns part of a value, based on a position and length.