**Table of Contents**

1. [Finding Data Queries.](https://github.com/enochtangg/quick-SQL-cheatsheet?fbclid=PAQ0xDSwLfXFtleHRuA2FlbQIxMAABp7XWP4nX4WPfEQ2EvU0tkS6PWeZF3lOxFRtoeREYsfTkO4cs9Cyki1r1TTO5_aem_LXzwS3CaYXQjchY0Hpv_4A#find)
2. [Data Modification Queries.](https://github.com/enochtangg/quick-SQL-cheatsheet?fbclid=PAQ0xDSwLfXFtleHRuA2FlbQIxMAABp7XWP4nX4WPfEQ2EvU0tkS6PWeZF3lOxFRtoeREYsfTkO4cs9Cyki1r1TTO5_aem_LXzwS3CaYXQjchY0Hpv_4A#modify)
3. [Reporting Queries.](https://github.com/enochtangg/quick-SQL-cheatsheet?fbclid=PAQ0xDSwLfXFtleHRuA2FlbQIxMAABp7XWP4nX4WPfEQ2EvU0tkS6PWeZF3lOxFRtoeREYsfTkO4cs9Cyki1r1TTO5_aem_LXzwS3CaYXQjchY0Hpv_4A#report)
4. [Join Queries.](https://github.com/enochtangg/quick-SQL-cheatsheet?fbclid=PAQ0xDSwLfXFtleHRuA2FlbQIxMAABp7XWP4nX4WPfEQ2EvU0tkS6PWeZF3lOxFRtoeREYsfTkO4cs9Cyki1r1TTO5_aem_LXzwS3CaYXQjchY0Hpv_4A#joins)
5. [View Queries.](https://github.com/enochtangg/quick-SQL-cheatsheet?fbclid=PAQ0xDSwLfXFtleHRuA2FlbQIxMAABp7XWP4nX4WPfEQ2EvU0tkS6PWeZF3lOxFRtoeREYsfTkO4cs9Cyki1r1TTO5_aem_LXzwS3CaYXQjchY0Hpv_4A#view)
6. [Altering Table Queries.](https://github.com/enochtangg/quick-SQL-cheatsheet?fbclid=PAQ0xDSwLfXFtleHRuA2FlbQIxMAABp7XWP4nX4WPfEQ2EvU0tkS6PWeZF3lOxFRtoeREYsfTkO4cs9Cyki1r1TTO5_aem_LXzwS3CaYXQjchY0Hpv_4A#alter)
7. [Creating Table Query.](https://github.com/enochtangg/quick-SQL-cheatsheet?fbclid=PAQ0xDSwLfXFtleHRuA2FlbQIxMAABp7XWP4nX4WPfEQ2EvU0tkS6PWeZF3lOxFRtoeREYsfTkO4cs9Cyki1r1TTO5_aem_LXzwS3CaYXQjchY0Hpv_4A#create)

**1. Finding Data Queries**

**SELECT: used to select data from a database**

* SELECT \* FROM table\_name;

**DISTINCT: filters away duplicate values and returns rows of specified column**

* SELECT DISTINCT column\_name;

**WHERE: used to filter records/rows**

* SELECT column1, column2 FROM table\_name WHERE condition;
* SELECT \* FROM table\_name WHERE condition1 AND condition2;
* SELECT \* FROM table\_name WHERE condition1 OR condition2;
* SELECT \* FROM table\_name WHERE NOT condition;
* SELECT \* FROM table\_name WHERE condition1 AND (condition2 OR condition3);
* SELECT \* FROM table\_name WHERE EXISTS (SELECT column\_name FROM table\_name WHERE condition);

**ORDER BY: used to sort the result-set in ascending or descending order**

* SELECT \* FROM table\_name ORDER BY column;
* SELECT \* FROM table\_name ORDER BY column DESC;
* SELECT \* FROM table\_name ORDER BY column1 ASC, column2 DESC;

**SELECT TOP: used to specify the number of records to return from top of table**

* SELECT TOP number columns\_names FROM table\_name WHERE condition;
* SELECT TOP percent columns\_names FROM table\_name WHERE condition;
* Not all database systems support SELECT TOP. The MySQL equivalent is the LIMIT clause
* SELECT column\_names FROM table\_name LIMIT offset, count;

**LIKE: operator used in a WHERE clause to search for a specific pattern in a column**

* % (percent sign) is a wildcard character that represents zero, one, or multiple characters
* \_ (underscore) is a wildcard character that represents a single character
* SELECT column\_names FROM table\_name WHERE column\_name LIKE pattern;
* LIKE ‘a%’ (find any values that start with “a”)
* LIKE ‘%a’ (find any values that end with “a”)
* LIKE ‘%or%’ (find any values that have “or” in any position)
* LIKE ‘\_r%’ (find any values that have “r” in the second position)
* LIKE ‘a\_%\_%’ (find any values that start with “a” and are at least 3 characters in length)
* LIKE ‘[a-c]%’ (find any values starting with “a”, “b”, or “c”

**IN: operator that allows you to specify multiple values in a WHERE clause**

* essentially the IN operator is shorthand for multiple OR conditions
* SELECT column\_names FROM table\_name WHERE column\_name IN (value1, value2, …);
* SELECT column\_names FROM table\_name WHERE column\_name IN (SELECT STATEMENT);

**BETWEEN: operator selects values within a given range inclusive**

* SELECT column\_names FROM table\_name WHERE column\_name BETWEEN value1 AND value2;
* SELECT \* FROM Products WHERE (column\_name BETWEEN value1 AND value2) AND NOT column\_name2 IN (value3, value4);
* SELECT \* FROM Products WHERE column\_name BETWEEN #01/07/1999# AND #03/12/1999#;

**NULL: values in a field with no value**

* SELECT \* FROM table\_name WHERE column\_name IS NULL;
* SELECT \* FROM table\_name WHERE column\_name IS NOT NULL;

**AS: aliases are used to assign a temporary name to a table or column**

* SELECT column\_name AS alias\_name FROM table\_name;
* SELECT column\_name FROM table\_name AS alias\_name;
* SELECT column\_name AS alias\_name1, column\_name2 AS alias\_name2;
* SELECT column\_name1, column\_name2 + ‘, ‘ + column\_name3 AS alias\_name;

**UNION: set operator used to combine the result-set of two or more SELECT statements**

* Each SELECT statement within UNION must have the same number of columns
* The columns must have similar data types
* The columns in each SELECT statement must also be in the same order
* SELECT columns\_names FROM table1 UNION SELECT column\_name FROM table2;
* UNION operator only selects distinct values, UNION ALL will allow duplicates

**INTERSECT: set operator which is used to return the records that two SELECT statements have in common**

* Generally used the same way as **UNION** above
* SELECT columns\_names FROM table1 INTERSECT SELECT column\_name FROM table2;

**EXCEPT: set operator used to return all the records in the first SELECT statement that are not found in the second SELECT statement**

* Generally used the same way as **UNION** above
* SELECT columns\_names FROM table1 EXCEPT SELECT column\_name FROM table2;

**ANY|ALL: operator used to check subquery conditions used within a WHERE or HAVING clauses**

* The ANY operator returns true if any subquery values meet the condition
* The ALL operator returns true if all subquery values meet the condition
* SELECT columns\_names FROM table1 WHERE column\_name operator (ANY|ALL) (SELECT column\_name FROM table\_name WHERE condition);

**GROUP BY: statement often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns**

* SELECT column\_name1, COUNT(column\_name2) FROM table\_name WHERE condition GROUP BY column\_name1 ORDER BY COUNT(column\_name2) DESC;

**HAVING: this clause was added to SQL because the WHERE keyword could not be used with aggregate functions**

* SELECT COUNT(column\_name1), column\_name2 FROM table GROUP BY column\_name2 HAVING COUNT(column\_name1) > 5;

**WITH: often used for retrieving hierarchical data or re-using temp result set several times in a query. Also referred to as "Common Table Expression"**

* WITH RECURSIVE cte AS (  
    SELECT c0.\* FROM categories AS c0 WHERE id = 1 # Starting point  
    UNION ALL  
    SELECT c1.\* FROM categories AS c1 JOIN cte ON c1.parent\_category\_id = cte.id  
  )  
  SELECT \*  
  FROM cte

**2. Data Modification Queries**

**INSERT INTO: used to insert new records/rows in a table**

* INSERT INTO table\_name (column1, column2) VALUES (value1, value2);
* INSERT INTO table\_name VALUES (value1, value2 …);

**UPDATE: used to modify the existing records in a table**

* UPDATE table\_name SET column1 = value1, column2 = value2 WHERE condition;
* UPDATE table\_name SET column\_name = value;

**DELETE: used to delete existing records/rows in a table**

* DELETE FROM table\_name WHERE condition;
* DELETE \* FROM table\_name;

**3. Reporting Queries**

**COUNT: returns the # of occurrences**

* SELECT COUNT (DISTINCT column\_name);

**MIN() and MAX(): returns the smallest/largest value of the selected column**

* SELECT MIN (column\_names) FROM table\_name WHERE condition;
* SELECT MAX (column\_names) FROM table\_name WHERE condition;

**AVG(): returns the average value of a numeric column**

* SELECT AVG (column\_name) FROM table\_name WHERE condition;

**SUM(): returns the total sum of a numeric column**

* SELECT SUM (column\_name) FROM table\_name WHERE condition;

**4. Join Queries**

**INNER JOIN: returns records that have matching value in both tables**

* SELECT column\_names FROM table1 INNER JOIN table2 ON table1.column\_name=table2.column\_name;
* SELECT table1.column\_name1, table2.column\_name2, table3.column\_name3 FROM ((table1 INNER JOIN table2 ON relationship) INNER JOIN table3 ON relationship);

**LEFT (OUTER) JOIN: returns all records from the left table (table1), and the matched records from the right table (table2)**

* SELECT column\_names FROM table1 LEFT JOIN table2 ON table1.column\_name=table2.column\_name;

**RIGHT (OUTER) JOIN: returns all records from the right table (table2), and the matched records from the left table (table1)**

* SELECT column\_names FROM table1 RIGHT JOIN table2 ON table1.column\_name=table2.column\_name;

**FULL (OUTER) JOIN: returns all records when there is a match in either left or right table**

* SELECT column\_names FROM table1 FULL OUTER JOIN table2 ON table1.column\_name=table2.column\_name;

**Self JOIN: a regular join, but the table is joined with itself**

* SELECT column\_names FROM table1 T1, table1 T2 WHERE condition;

**5. View Queries**

**CREATE: create a view**

* CREATE VIEW view\_name AS SELECT column1, column2 FROM table\_name WHERE condition;

**SELECT: retrieve a view**

* SELECT \* FROM view\_name;

**DROP: drop a view**

* DROP VIEW view\_name;

**6. Altering Table Queries**

**ADD: add a column**

* ALTER TABLE table\_name ADD column\_name column\_definition;

**MODIFY: change data type of column**

* ALTER TABLE table\_name MODIFY column\_name column\_type;

**DROP: delete a column**

* ALTER TABLE table\_name DROP COLUMN column\_name;

**7. Creating Table Query**

**CREATE: create a table**

* CREATE TABLE table\_name (  
  column1 datatype,  
  column2 datatype,  
  column3 datatype,  
  column4 datatype,  
  );