Essential SQL Commands

Command Name	Description	[1]Example

Query Commands

Command Name	Description	Example
SELECT	Basic query building block to retrieve data.	SELECT 1 FROM table_name;
SELECT *	Using * with SELECT returns all columns.	SELECT * FROM table_name;
SELECT column	Specify exact columns with their name.	SELECT column_name FROM table_name;
SELECT table.column	Reference a column from a specific table.	SELECT table_name.column_name FROM table_name, table_2_name;
FROM	Specify where to find data.	SELECT column_name FROM table_name;
AS	Temporarily alias a table name or column to a new name.	SELECT new_table_name.*, column_name AS new_column FROM table_name AS new_table_name;
WHERE	Filter results with a condition.	SELECT * FROM table_name WHERE column_name = 'value';
AND	Use multiple conditions with a WHERE clause. Results must match all conditions.	SELECT * FROM table_name WHERE column_name < 10 AND column_name > 1;
OR	Use multiple conditions with a WHERE clause. Results only need to match one condition.	SELECT * FROM table_name WHERE column_name < 10 OR column_name = 15;
ORDER BY	Order the results by a column. The database chooses how to order.	SELECT * FROM table_name ORDER BY column_name;
ORDER BY column ASC	Order the results by a column in ascending order.	SELECT * FROM table_name ORDER BY column_name ASC;
ORDER BY column DESC	Order the results by a column in descending order.	SELECT * FROM table_name ORDER BY column_name DESC;
LIMIT	Restrict the number of results returned.	SELECT * FROM table_name LIMIT 5;
OFFSET	Skip the first OFFSET number of rows. Often used with LIMIT.	SELECT * FROM table_name LIMIT 5 OFFSET 10;
SUBQUERY	Run a query to retrieve data for another query.	SELECT column FROM table_name where column_name IN (SELECT column_2_name FROM table_2_name);

Aggregate Functions[2]

Command Name	Description	Example
COUNT	Count the number of rows that match the query.	SELECT COUNT(column_name) FROM table_name;
MAX	Return the highest value in a numeric column.	SELECT MAX(column_name) FROM table_name;
MIN	Return the lowest value in a numeric column.	SELECT MIN(column_name) FROM table_name;
SUM	Sum the values of a numeric column.	SELECT SUM(column_name) FROM table_name;
AVG	Calculate the average value for a numeric column.	SELECT AVG(column_name) FROM table_name;
HAVING	Used with aggregate functions instead of the WHERE clause.	SELECT COUNT(column_name) FROM table_name HAVING column_name > 10;
GROUP BY	Used to refine an aggregate result.	SELECT COUNT(column_name) FROM table_name GROUP BY column_2_name;

Operators

Command Name	Description	Example
LIKE	Case-sensitive search for a pattern with a wildcard operator (%).	SELECT column_name FROM table_name WHERE column_name LIKE '%VALUE%';
ILIKE	Case-insensitive search for a pattern with a wildcard operator (%).	SELECT column_name FROM table_name WHERE column_name ILIKE '%value%';
BETWEEN	Search for a value between two values. Works with dates or numbers.	SELECT column_name FROM table_name WHERE column_name BETWEEN 1 AND 10;
>	Search for values greater than a condition.	SELECT column_name FROM table_name WHERE column_name > 10;
>=	Search for values greater or equal to a condition.	SELECT column_name FROM table_name WHERE column_name >= 10;
<	Search for values less than a condition.	SELECT column_name FROM table_name WHERE column_name < 10;
<=	Search for values less than or equal to a condition.	SELECT column_name FROM table_name WHERE column_name <= 10;
=	Search for values matching a condition exactly.	SELECT column_name FROM table_name where column_name = 10;
<>	Search for values not equal to a condition.	SELECT column_name FROM table_name WHERE column_name <> 10;
UNION	Combine two unique queries (with the same columns) into one result.	SELECT column_name FROM table_name UNION SELECT column_2_name FROM table_2_name;
UNION ALL	Combine two queries (with the same columns) into one result. Duplicates allowed.	SELECT column_name FROM table_name UNION ALL SELECT column_2_name FROM table_2_name;
IN	Shorthand for WHERE. Specifies multiple OR conditions.	SELECT column_name FROM table_name where column_name IN ('A', 'B', 'C');
NOT IN	Shorthand for WHERE. Specifies multiple OR conditions (inverted) or not equal to.	SELECT column_name FROM table_name where column_name NOT IN ('A', 'B', 'C');
IS NULL	Check for empty values.	SELECT column_name FROM table_name WHERE column_name IS NULL;
IS NOT NULL	Check for no empty values.	SELECT column_name FROM table_name WHERE column_name IS NOT NULL;
INTERSECT	Return results which match two queries.	SELECT column_name FROM table_name INTERSECT SELECT column_2_name FROM table_2_name;
MINUS	[2]Return results in one query which are not in another query.	SELECT column_name FROM table_name MINUS SELECT column_2_name FROM table_2_name;

Command Name	Description	Example

Joins

Command Name	Description	Example
ON	Used to specify the column to compare and match results.	SELECT * FROM table_name LEFT OUTER JOIN table_2_name ON table_name.column_name = table_2_name.column_name;
USING	Shorthand for ON, used when the column name is the same in both tables.	SELECT * FROM table_name LEFT OUTER JOIN table_2_name ON table_name.column_name = table_2_name.column_2_name;
LEFT OUTER JOIN	All the results from the left table, with only the matching results from the right table.	SELECT * FROM table_name LEFT OUTER JOIN table_2_name ON table_name.column_name = table_2_name.column_2_name;
LEFT OUTER JOIN (WITH NULL)	(With null) All the results from the left table but not in the right table.	SELECT * FROM table_name LEFT OUTER JOIN table_2_name ON table_name.column_name = table_2_name.column_2_name WHERE table_2_name.column_2_name IS NULL;
INNER JOIN	All the results that match in both the left and right tables.	SELECT * FROM table_name INNER JOIN table_2_name ON table_name.column_name = table_2_name.column_2_name;
FULL OUTER JOIN	All the results from both the left and right tables.	SELECT * FROM table_name FULL OUTER JOIN table_2_name ON table_name.column_name = table_2_name.column_2_name;
FULL OUTER JOIN (WITH NULL)	(With null) All the results from both the left and right tables excluding results in both tables.	SELECT * FROM table_name FULL OUTER JOIN table_2_name ON table_name.column_name = table_2_name.column_2_name WHERE table_name.column_name IS NULL OR table_2_name.column_2_name IS NULL;
RIGHT OUTER JOIN	All the results from the right table, with only the matching results from the left table.	SELECT * FROM table_2_name RIGHT OUTER JOIN table_name ON table_2_name.column_2_name = table_name.column_name;
RIGHT OUTER JOIN (WITH NULL)	(With null) All the results from the right table but not in the left table.	SELECT * FROM table_2_name RIGHT OUTER JOIN table_name ON table_2_name.column_2_name = table1.column_name WHERE table_name.column_name IS NULL;

Creating and Editing Tables

Command Name	Description	Example
CREATE TABLE	Create a new table.	CREATE TABLE table_name (column_name datatype column_2_name datatype);
NULL	Allow empty values for this field.	CREATE TABLE table_name (column_name column_name datatype NULL);
NOT NULL	Don't allow empty values for this field.	CREATE TABLE table_name (column_name column_name datatype NOT NULL);
DEFAULT	A value to populate the field with if one is not supplied.	CREATE TABLE table_name (column_name datatype DEFAULT 'makeuseof');
AS	Create a new table based on the structure of an existing table. The new table will contain the data from the old table.	CREATE TABLE table_2_name AS SELECT * FROM table_name;
ALTER TABLE (ADD COLUMN)	Add a new column to an existing table.	ALTER TABLE table_name ADD COLUMN column_2_name datatype;
ALTER TABLE (DROP COLUMN)	Remove a column from an existing table.	ALTER TABLE table_name DROP COLUMN column_2_name;
ALTER TABLE (ALTER COLUMN)	Change the datatype of an existing column.	ALTER TABLE table_2_name ALTER COLUMN column_name datatype;
ALTER TABLE (RENAME COLUMN)	Rename an existing column.	ALTER TABLE table_name RENAME COLUMN column_name TO new_column_name datatype;
ALTER TABLE (RENAME TABLE)	Rename an existing table.	RENAME TABLE table_name TO new_table_name;
ALTER TABLE (MODIFY NULL)	Allow null values for a column.	ALTER TABLE table_name MODIFY column_name datatype NULL;
ALTER TABLE (MODIFY NOT NULL)	Prevent null values for a column.	ALTER TABLE table_name MODIFY column_name datatype NOT NULL;
DROP TABLE	Delete a table and all its data.	DROP TABLE table_name;
TRUNCATE TABLE	Delete all the data in a table, but not the table itself.	TRUNCATE TABLE table_name;

Constraints

Command Name	Description	Example
PRIMARY KEY	A value that uniquely identifies a record in a table. A combination of NOT NULL and UNIQUE.	CREATE TABLE table_name (column_name datatype column_2_name datatype, PRIMARY KEY (column_name, column_2_name));
FOREIGN KEY	References a unique value in another table. Often a primary key in the other table.	CREATE TABLE table_name (column_name datatype column_2_name datatype, FOREIGN KEY (column_name) REFERENCES table_2_name (column_2_name));
UNIQUE	Enforce unique values for this column per table.	CREATE TABLE table_name (column_name datatype column_2_name datatype, UNIQUE(column_name, column_2_name));
CHECK	Ensure values meet a specific condition.	CREATE TABLE table_name (column_name datatype column_2_name datatype, CHECK(column_name > 10));
INDEX (CREATE)	Optimize tables and greatly speed up queries by adding an index to a column.	CREATE INDEX index_name ON table_name(column_name);
INDEX (CREATE UNIQUE)	Create an index that does not allow duplicate values.	CREATE UNIQUE INDEX index_name ON table_name(column_name);
INDEX (DROP)	Remove an index.	DROP INDEX index_name;

Command Name	**Description**	**Example**

Creating and Editing Data

INSERT (SINGLE VALUE)	Add a new record to a table.	INSERT INTO table_name(column_name) VALUES(value_1);
INSERT (MULTIPLE VALUES)	Add several new records to a table.	INSERT INTO table_name(column_name) VALUES(value_1),(value_2);
INSERT (SELECT)	Add records to a table, but get the values from an existing table.	INSERT INTO table_name(column_name) SELECT * FROM table_2_name;
UPDATE (ALL)	Modify all existing records in a table.	UPDATE table_name SET column_name = 10;
UPDATE (WHERE)	Modify existing records in a table which match a condition.	UPDATE table_name SET column_name = 10 WHERE column_2_name = 5;
DELETE (ALL)	Remove all records from a table.	DELETE FROM table_name;
DELETE (WHERE)	Remove records from a table which match a condition.	DELETE FROM table_name WHERE column_name = 5;

[2]Creating and Editing Triggers

CREATE TRIGGER	Create a trigger.	CREATE TRIGGER trigger_name BEFORE INSERT ON table_name FOR EACH ROW EXECUTE stored_procedure;
CREATE TRIGGER (OR MODIFY)	Create a trigger, or update an existing trigger if one is found with the same name.	CREATE OR MODIFY TRIGGER trigger_name BEFORE INSERT ON table_name FOR EACH ROW EXECUTE stored_procedure;
WHEN (BEFORE)	Run the trigger before the event happens.	CREATE TRIGGER trigger_name BEFORE INSERT ON table_name FOR EACH ROW EXECUTE stored_procedure;
WHEN (AFTER)	Run the trigger after the event happens.	CREATE TRIGGER trigger_name AFTER INSERT ON table_name FOR EACH ROW EXECUTE stored_procedure;
EVENT (INSERT)	Run the trigger before or after an insert happens.	CREATE TRIGGER trigger_name BEFORE INSERT ON table_name FOR EACH ROW EXECUTE stored_procedure;
EVENT (UPDATE)	Run the trigger before or after an update happens.	CREATE TRIGGER trigger_name BEFORE UPDATE ON table_name FOR EACH ROW EXECUTE stored_procedure;
EVENT (DELETE)	Run the trigger before or after a delete happens.	CREATE TRIGGER trigger_name BEFORE DELETE ON table_name FOR EACH ROW EXECUTE stored_procedure;
ON	Specify which table to target with this trigger.	CREATE TRIGGER trigger_name BEFORE INSERT ON table_name FOR EACH ROW EXECUTE stored_procedure;
TRIGGER_TYPE (FOR EACH ROW)	Execute the trigger for every row changed.	CREATE TRIGGER trigger_name BEFORE INSERT ON table_name FOR EACH ROW EXECUTE stored_procedure;
TRIGGER_TYPE (FOR EACH STATEMENT)	Execute the trigger once per SQL statement, regardless of how many rows are altered.	CREATE TRIGGER trigger_name BEFORE INSERT ON table_name FOR EACH ROW STATEMENT stored_procedure;
EXECUTE	Keyword to indicate the end of the main trigger definition.	CREATE TRIGGER trigger_name BEFORE INSERT ON table_name FOR EACH ROW EXECUTE stored_procedure;
DROP TRIGGER	Delete a trigger.	DROP TRIGGER trigger_name;

Creating and Editing Views

CREATE VIEW	Create a new view.	CREATE VIEW view_name(column_name) AS SELECT * FROM table_name;
AS	Define where to retrieve the data for a view.	CREATE VIEW view_name(column_name) AS SELECT * FROM table_name;
WITH CASCADED CHECK OPTION	Ensure any data modified through a view meets the rules defined by the rule. Apply this to any other views.	CREATE VIEW view_name(column_name) AS SELECT * FROM table_name WITH CASCADED CHECK OPTION;
WITH LOCAL CHECK OPTION	Ensure any data modified through a view meets the rules defined by the rule. Ignore this for any other views.	CREATE VIEW view_name(column_name) AS SELECT * FROM table_name WITH LOCAL CHECK OPTION;
CREATE RECURSIVE VIEW	Create a recursive view (one that refers to a recursive common table expression).	CREATE RECURSIVE VIEW view_name(column_name) AS SELECT * FROM table_name;
CREATE TEMPORARY VIEW	Create a view that exists for the current session only.	CREATE TEMPORARY VIEW view_name(column_name) AS SELECT * FROM table_name;
DROP VIEW	Delete a view.	DROP VIEW view_name;

[2]Common Table Expressions (CTEs)

WITH	Create a new common table expression.	WITH cte_name (column_name) AS (SELECT * FROM table_name) SELECT * FROM cte_name;
AS	Specify the data to use in the CTE.	WITH cte_name (column_name) AS (SELECT * FROM table_name) SELECT * FROM cte_name;
, (COMMA)	Chain multiple CTEs.	WITH cte_name (column_name) AS (SELECT * FROM table_name), cte_2_name (column_2_name) AS (SELECT * FROM table_2_name) SELECT * FROM cte_name;

[1]Examples given in MySQL syntax.
[2]Database engine implementations and support often vary.