SQL Syntax Summary from Documents

Systematic Organization

October 10, 2025

SQL Syntax Summary

I. Data Definition Language (DDL)

DDL commands are used for defining, modifying, and dropping database objects like schemas and tables[cite: 265, 267, 452, 463].

Command	Purpose	Basic Structure/Keywords	Ref.
CREATE SCHEMA	Creates a new schema[cite: 272].	`CREATE SCHEMA <schema_name>` [cite: 272]	p. 4
CREATE TABLE	Creates a new table[cite: 327].	`CREATE TABLE <table_name>(<column_name><data_type>[co ...)` [cite: 328]	p. 9-13
DROP SCHEMA	Removes a schema and all objects in it (with `CASCADE`)[cite: 456].	`DROP SCHEMA <schema_name>[CASCADE | RESTRICT]` [cite: 455, 458]	p. 19
DROP TABLE	Removes a table and all objects referring to it (with `CASCADE`)[cite: 457].	`DROP TABLE <table_name>[CASCADE | RESTRICT]` [cite: 455, 459]	p. 19
ALTER TABLE	Modifies the attributes or constraints of a table[cite: 463, 466].	`ALTER TABLE <table_name><action>` (e.g., `ADD`, `DROP`, `ALTER`, `DROP CONSTRAINTS`, `ADD FOREIGN KEY`) [cite: 467, 468, 479]	p. 20-21

II. Data Types and Constraints

Concept	Purpose	Keywords/Examples	Ref.
Data Types	Specifies the type of data a column can hold[cite: 314].	**Numeric**: `INT`, `DECIMAL(n, m)`. **Character-string**: `CHAR(n)`, `VARCHAR(n)`. **Date/Time**: `DATE`, `TIME`, `TIMESTAMP`[cite: 316, 318, 320].	p. 8
NOT NULL	Ensures a column cannot have a `NULL` value[cite: 329].	`monthly_salary INT NOT NULL` [cite: 328]	p. 9
PRIMARY KEY	Specifies the column(s) that uniquely identify each row[cite: 329].	`PRIMARY KEY(id)` or `PRIMARY KEY (col1, col2)` [cite: 328, 355]	p. 9, 11
UNIQUE	Ensures all values in a column are distinct[cite: 336].	`UNIQUE (addr_str)` [cite: 335]	p. 10
DEFAULT	Specifies a default value for a column[cite: 405].	`store_id INT NOT NULL DEFAULT 1` [cite: 406]	p. 14
FOREIGN KEY	Defines a column(s) that references the primary key of another table[cite: 337].	`FOREIGN KEY(mgr_id) REFERENCES EMPLOYEE(id)` [cite: 335]	p. 10
Referential Triggers	Defines actions when a referred entity is deleted or updated[cite: 404].	`ON DELETE SET NULL`, `ON DELETE SET DEFAULT`, `ON DELETE CASCADE`, `ON UPDATE CASCADE`[cite: 411, 430, 438, 445].	p. 15-18

III. Data Manipulation Language (DML) - Retrieval Queries

Basic Select Statement Components

Component	Purpose	Keywords/Syntax	Ref.
SELECT	Specifies the columns to be retrieved[cite: 504].	`SELECT id, birthday`, `SELECT *` (all attributes) [cite: 517, 536]	p. 23, 24, 26
FROM	Specifies the tables involved in the query[cite: 505].	`FROM EMPLOYEE` [cite: 518]	p. 23
WHERE	Filters the rows based on a condition[cite: 506].	`WHERE name = 'Po-Lin Chen'` [cite: 519]	p. 23, 24
AS	Provides an alias for a table or column[cite: 687].	`SELECT s.id AS store_num, FROM EMPLOYEE AS e` [cite: 688, 689]	p. 38
DISTINCT	Eliminates duplicate tuples in the query outcome[cite: 608].	`SELECT DISTINCT monthly_salary FROM EMPLOYEE` [cite: 611]	p. 32
DISTINCT ON	Drops duplicates based on chosen columns[cite: 1214].	`SELECT DISTINCT ON (store_id) store_id, name, monthly_salary...` [cite: 1225, 1226]	p. 81, 82
ORDER BY	Sorts the final result[cite: 787].	`ORDER BY <attribute>[ASC | DESC]` [cite: 788]	p. 48
LIMIT	Restricts the number of rows returned[cite: 1175].	`LIMIT 3` [cite: 1183]	p. 78
OFFSET	Indicates how many rows to ignore from the beginning[cite: 1176].	`OFFSET 1` [cite: 1184]	p. 78

Filtering Conditions

Condition Type	Operators/Keywords	Examples	Ref.
Comparison	=, >, <, <>, ! =.	`WHERE monthly_salary > 80000` [cite: 976, 977]	p. 24
NULL check	`IS NULL`, `IS NOT NULL`[cite: 553, 559].	`WHERE supervisor_id IS NULL` [cite: 557]	p. 27

Condition Type	Operators/Keywords	Examples	Ref.
String Matching	`LIKE` with wildcards (`%`: any string, `_`: any single character)[cite: 565, 566, 567].	`WHERE name LIKE 'Chi%'` [cite: 571], `WHERE id LIKE 'A_____'` [cite: 580]	p. 28, 29
Range	`BETWEEN <value1> AND <value2>` (inclusive)[cite: 935].	`WHERE birthday BETWEEN '1995-01-01' AND '1999-12-31'` [cite: 935]	p. 61
Set Membership	`IN`, `NOT IN`[cite: 653].	`WHERE store_id IN (1, 2)` [cite: 658]	p. 35

Joins

Join Type	Purpose	Keywords/Syntax	Ref.
Traditional Join	Implicitly joins tables in `FROM`, filtered by `WHERE` for joining criterion[cite: 667, 676].	`FROM EMPLOYEE, STORE WHERE EMPLOYEE.store_id = STORE.id` [cite: 680, 681]	p. 36, 37
Inner Join	Returns only rows that satisfy the join condition[cite: 738].	`JOIN <table2> ON <condition>` [cite: 705]	p. 40
Cross Join	Computes the Cartesian product of the tables[cite: 728].	`CROSS JOIN <table2>` [cite: 731]	p. 42
Left Outer Join	Returns all rows from the left table[cite: 741].	`LEFT OUTER JOIN <table2> ON <condition>` [cite: 741]	p. 43
Right Outer Join	Returns all rows from the right table[cite: 743].	`RIGHT JOIN <table2> ON <condition>` [cite: 743, 762]	p. 43, 45
Full Outer Join	Returns rows matched in either table (applies rule to both sides)[cite: 745].	`FULL OUTER JOIN <table2> ON <condition>` [cite: 745]	p. 43

Join Type	Purpose	Keywords/Syntax	Ref.
USING	Used when join attribute names are the same[cite: 718].	`JOIN STORE_PHONE AS sp USING (store_id)` [cite: 722]	p. 41

Aggregation and Grouping

Concept	Purpose	Keywords/Functions	Ref.
Aggregate Functions	Calculates a single value over a set of rows[cite: 798].	`COUNT`, `SUM`, `MAX`, `MIN`, `AVG` [cite: 798]	p. 49
GROUP BY	Groups rows by a common value for aggregation[cite: 807].	`GROUP BY store_id` [cite: 810]	p. 50
HAVING	Filters the results of a `GROUP BY` clause based on an aggregate condition[cite: 816].	`HAVING COUNT(e.*) > 2` [cite: 821]	p. 51
FILTER	Applies an aggregate function only to the rows within a group that satisfy a condition[cite: 11].	`SUM(monthly_salary) FILTER (WHERE gender = 'W')` [cite: 72]	p. 6

Nested Queries (Subqueries)

Concept	Purpose	Keywords/Operators	Ref.
Subqueries	A query nested inside another query[cite: 958].	Parentheses `()` for the inner query [cite: 978]	p. 64
Comparison	Used with single-value results from subqueries (e.g., `MAX`)[cite: 972].	`WHERE monthly_salary > (<subquery>)` [cite: 977, 978]	p. 65
Set Comparison	Compares a value to a set of values returned by a subquery[cite: 959].	`ALL`, `ANY` (or `SOME`) [cite: 960, 993, 1006]	p. 66, 67

Concept	Purpose	Keywords/Operators	Ref.
Membership	Checks if a value is present in the set returned by a subquery[cite: 959].	`IN`, `NOT IN` [cite: 959, 1025]	p. 64, 68
Existence	Checks for the existence of any rows returned by the subquery[cite: 959].	`EXISTS`, `NOT EXISTS` [cite: 959, 1098, 1152]	p. 64, 73, 76

Functions

Function Type	Functions	Purpose/Example	Ref.
String Processing	`LEFT(<string>, n)`, `RIGHT(<string>, n)`, `SUBSTRING(<string>, m, n)`[cite: 588, 589, 590].	Selects a specified part of a string[cite: 588, 589, 590].	p. 30
Conditional Logic	`CASE WHEN <condition> THEN <value> [ELSE <value>] END`[cite: 1290].	Creates if-else selection for column values or updates[cite: 1289, 1292, 1293].	p. 83, 84
NULL Handling	`COALESCE(v1, v2, ...)`, `NULLIF(v1, v2)`[cite: 1191, 1206].	`COALESCE` returns the first non-NULL value[cite: 1189]. `NULLIF` returns NULL if $v1 = v2$[cite: 1205].	p. 79, 80
Window Functions	`RANK() OVER`, `ROW_NUMBER() OVER`, `AVG() OVER`[cite: 82, 95, 96].	Calculates values across a group of rows (a "window") retaining all rows[cite: 78, 80].	p. 7-15
Custom Function	`CREATE [OR REPLACE] FUNCTION...`[cite: 185].	Creates a user-defined function (UDF) to wrap transformation logic[cite: 181, 182].	p. 16, 18

IV. Data Manipulation Language (DML) - Modification

Command	Purpose	Basic Structure/Keywords	Ref.
INSERT INTO	Adds one or more rows into a table[cite: 846].	`INSERT INTO <table_name> [(<col_list>)] VALUES (<value_list>)` [cite: 847, 850]	p. 54
Bulk Insertion	Inserts data generated by a `SELECT` statement[cite: 862].	`INSERT INTO <table_name> (<col_list>) SELECT... FROM... GROUP BY...` [cite: 869, 870, 871]	p. 56
DELETE FROM	Removes rows from a table based on a condition[cite: 876].	`DELETE FROM <table_name> WHERE <condition>` (can use subqueries) [cite: 877, 880]	p. 57
UPDATE	Modifies existing values in rows[cite: 892].	`UPDATE <table_name> SET <col> = <value> [WHERE <condition>]` (can update multiple columns or use expressions/subqueries) [cite: 893, 897, 900]	p. 58

V. Views and CTE

Tools for structuring and simplifying complex queries[cite: 1338].

Concept	Purpose	Keywords/Structure	Ref.
CTE	Defines a temporary, named result set for use within a single query[cite: 1350].	`WITH CTE_name AS (SELECT...) [, another_CTE AS (SELECT...)] SELECT... FROM CTE_name...` [cite: 1353, 1357, 1362]	p. 90, 92
Materialized CTE	Forces the temporary result set to be stored (materialized)[cite: 1414, 1417].	`WITH CTE_name AS MATERIALIZED (...)` [cite: 1420]	p. 94
View	A virtual table derived from other tables, used to simplify queries[cite: 1433, 1435].	`CREATE VIEW <view_name> AS SELECT...` [cite: 1437]	p. 95
Materialized View	A view whose data is physically stored at creation time[cite: 1486].	`CREATE MATERIALIZED VIEW <view_name> AS SELECT... [WITH NO DATA];` [cite: 1481, 1499]	p. 99, 100

Concept	Purpose	Keywords/Structure	Ref.
Refresh View	Updates the data in a materialized view[cite: 1494].	`REFRESH MATERIALIZED VIEW <view_name>;` [cite: 1500]	p. 100