SQL (Structured Query Lan.)

CHEET SHEET BY @CHARLSTOWN

CREATOR: DONALD D. CHAMBERLIN
YEAR: 1974
LATEST VERSION (2019): SQL2012
TOP DE SQL: SQLITE, MYSQL, POSTGRESQL, ORACLEDB

QUERYING DATA FROM A TABLE

>> Selecting columns

```
SELECT c1, c2 FROM table;                    > Selecting columns

SELECT * FROM table                          > Selects n columns
LIMIT n OFFSET n;

SELECT DISTINCT c1 FROM table;               > Selecting uniques
```

>> Applying conditions: (after select * from t)

```
WHERE c1 LIKE ['ST_ING' / 'S%' / '%s'];      > LIKE condition

WHERE c1 BETWEEN a AND b;                    > BETWEEN cond.

ORDER BY C1 [ASC/DESC];                      > Sorting columns

SELECT c1 AS 'col1' FROM table;              > Renaming column

WHERE c1 IS NOT NULL;                        > No nulls

WHERE C1 IN ('01', '02');                    > Contain cond.

LIMIT 5 OFFSET 10;                           > Show limit rows

SELECT c1, aggregate(c2)                     > Grooping options
FROM table
GROUP BY C1;

SELECT c1, count(c2) FROM table              > Filter groups
GROUP BY 2
HAVING count(c2) > 5;
```

****HIERARCHY:** **WHERE > GROUP BY > ORDER BY > LIMIT;**

>> Aggregate functions:

```
COUNT(c)        > Number of rows
SUM(c)          > Total sum
MAX(c)          > Largest value
MIN(c)          > Smallest value
AVG(c)          > Average value
FIRST(c)        > First value
LAST(c)         > Last value
```

>> Scalar functions:

```
UCASE(c)        > Converts a field to upper case
LCASE(c)        > Converts a field to lower case
MID(c)          > Extract characters from a text
LEN(c)          > Returns the length of a text
ROUND(c)        > Rounds a numbers to decimals
NOW()           > Returns the current time
FORMAT('0.5', 'percent')   > How field is displayed
```

>> String and other functions:

```
'str01' || 'add' || 'str02'          > Concatenate strings
CONCAT_WS ( '_' , 'str1', 'str2')    > Concatenate with symbol
SUBSTR ('string', 1, 3)              > Slice string by index
FIND_IN_SET ('x', c1)                > Find letters or symbols
INSTR ('string', 's')                > Count ocurrences of s
LEFT ('words', 2)                    > First two letters
TRIM ('  str  ')                     > Strip the spaces
LOCATE ('dog', 'my dog', 1)          > Locates a word in a str.
LPAD ('dog', 5, 'X')                 > Fill size with symbols
REPEAT ('str', 2)                    > Repeats the string n
REPLACE ('xrayx', 'x')               > Erase the string x
REVERSE ('asdf')                     > Returns reversed
SPACE(2)                             > Insert spaces
STRCMP ('str1', 'str2')              > Compare lenght strings
SUBSTRING_INDEX('s.tr', '.', 1)      > String before '.'
strftime('%Y', col)                  >  Extracts year from col.
```

QUERYING DATA FROM MULTIPLE TABLES

>> Selecting columns

```
SELECT c1, c2 FROM table                     > INNER join tables
JOIN table2;

SELECT c1, c2 FROM t1                        > Joinning columns
JOIN t2 ON t1.col_id = t2.col_id;

SELECT * FROM t1                             > Join all combin. I
CROSS JOIN t2;

SELECT c1, c2 FROM t1, t2                    > Join all combin. II

SELECT * FROM t1                             > Union 2 tables
UNION
SELECT * FROM t2;
```

>> Join Types

LEFT / INNER / RIGHT / FULL OUTER / CROSS JOIN

SHORTCUTS AND SUBQUERIES

```
SELECT x.c1, y.c2                            > Alias tables
FROM t1 as x, t2 as y;

SELECT c1 as x, c2 y FROM t;                 > As no needed

SELECT c1 / (SELECT * FROM t2)               > Subquery
FROM t1
WHERE c2 > (SELECT c3 FROM t2)

WITH previous AS(                            > Temp. subquery
      SELECT * FROM t1)
SELECT * FROM previous
JOIN t2 ON previous.c1 = t2.c1;

CREATE VIEW db.view_name AS                  > Permanent subq.
      SELECT * FROM db.t1;

SELECT C1,                                   > Conditional cases
      CASE
            WHEN a > b THEN C
            ELSE d
      END as c_name
FROM T;
```

QUERYING DATA FROM MULTIPLE TABLES

```
SELECT c1, c2 FROM table                    > INNER join tables
JOIN table2;

SELECT c1, c2 FROM t1                       > Joinning columns
JOIN t2 ON t1.col_id = t2.col_id;

SELECT c1, c2 FROM t1
INNER JOIN t2 ON condition;

SELECT c1, c2 FROM t1
LEFT JOIN t2 ON condition;

SELECT c1, c2 FROM t1
RIGHT JOIN t2 ON condition;

SELECT c1, c2 FROM t1
FULL OUTER JOIN t2 ON condition;

SELECT * FROM t1                            > Join all combin. I
CROSS JOIN t2;

SELECT * FROM t1                            > Union 2 tables
UNION
SELECT * FROM t2;

SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2 ON condition;

SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;

SELECT c1, c2 FROM t1
WHERE
SELECT c1, c2 FROM t2;

SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```

CREATE TABLES

```
CREATE TABLE table (                        > Create table
      c1 TYPE
      , c2 TYPE);

CREATE TABLE table (                        >Primary key
  c1 TYPE PRIMARY KEY,
  c2 TYPE);

CREATE TABLE table (                        > Foreign Key
      c1 TYPE,
      c2 TYPE,
      PRIMARY KEY (C1, C2)
      FOREIGN KEY (c1) REFERENCES t(c1));

DROP TABLE table name;                      > Delete table

TRUNCATE TABLE t;                           > Remove table data
```

MODIFY TABLES

```
INSERT INTO t (c1, c2)                      > Insert rows
      VALUES (1, 2)
      VALUES (2, 3);

DELETE FROM t                               > Delete rows
      WHERE condition;

ALTER TABLE t                               > Add column
      ADD COLUMN c1 TYPE;

ALTER TABLE t                               > Add column
      DROP COLUMN c1 TYPE;

ALTER TABLE t1                              > Rename table
      RENAME TO t2;

ALTER TABLE t RENAME c1 TO C2;              > Rename column

ALTER TABLE t                               > Add constraint
      ADD CONTRAINT name
      PRIMARY KEY (c1);

ALTER TABLE t DROP constraint;              > Remove constraint

UPDATE t                                    > Update column
      SET c1 =  (SELECT * FROM table);
```

INDEXES

```
CREATE INDEX name
ON t(c1, c2);

CREATE UNIQUE INDEX name
ON t(c1, c2);

DROP INDEX name;
```

SQL SHELL — SQLITE3

```
.tables                 > Show all tables in the file.db
.mode column            > Visualize by columns
.help                   > Get documentation
.shell clear            > Shell command clear console
.shell ls               > Shell command listing console
.schema table           > Show data types
sqlite3 new_db          > New database
.quit                   > Exit sqlite3
.headers on             > Visualize column names
.mode csv               > Ready to import CSV files
.import file.csv table  > Import CSV files as SQL
```

MANAGE TRIGGERS

```
CREATE TRIGGER name                 > Create trigger
      WHEN EVENT
      ON t1 TRIGGER_TYPE
      SET c1 = 0;

** WHEN                              > when the trigger occurs
      BEFORE
      AFTER

** EVENT                             > Table events
      INSERT
      UPDATE
      DELETE

** TRIGGER TYPE                      > Trigger type
      FOR EACH ROW
      FOR EACH STATEMENT

DROP TRIGGER trigger_name;          > Erase trigger
```