SQL Fundamentals

- **SELECT statement**
- ORDER BY, LIMIT clauses
- WHERE and Logical Operators

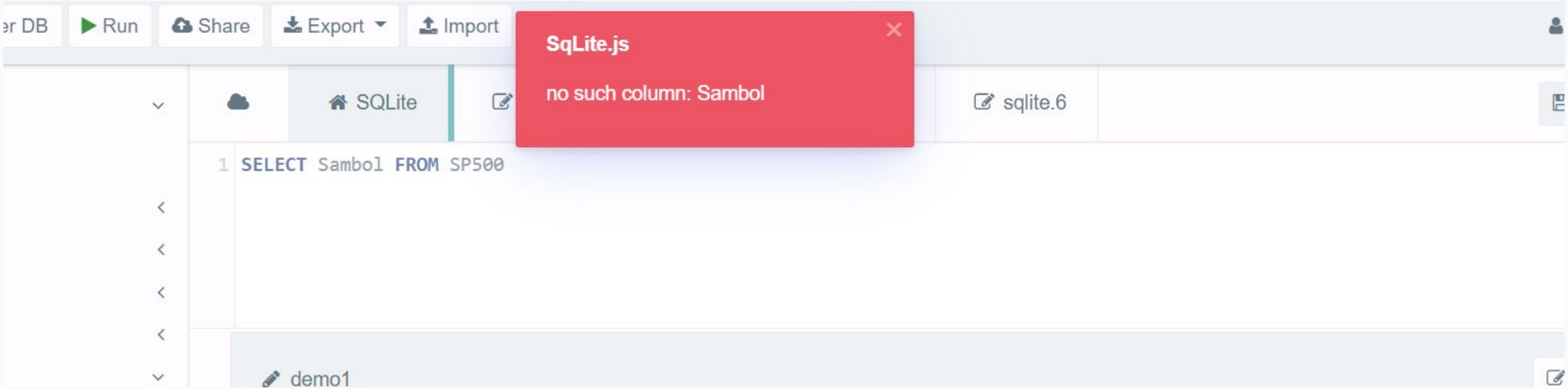
SQL FUNDAMENTALS: SELECT statement

SELECT …. FROM ….

- *SELECT * FROM table*
- *SELECT column name FROM table*
- *SELECT Column1, Column2, Column3 FROM table*
- Comments (-- , /* ….. */)
- Code formatting:
  - **Different columns should be separated by comma!**
  - The SQL Keywords are case-insensitive (SELECT , FROM , WHERE , etc), but are often written in all caps as a good practice
  - Usually the column and table name are also case-insensitive

SQL FUNDAMENTALS - Errors

- **Getting errors is absolutely OK**
- Usually IDE have suggestions of what is the type of error



SQL Fundamentals

- SELECT statement
- **ORDER BY, LIMIT clauses**
- WHERE and Logical Operators

SQL FUNDAMENTALS: ORDER BY

- ORDER BY clause is sorting the result based on one or more columns in different order.
- Column name by which you want to sort after the ORDER BY clause followed by the ASC or DESC keyword.
  - The ASC keyword means ascending.
  - And the DESC keyword means descending.
- If not specified ASC order is by default
- You can sort the result set using a column that does not appear in the select list of the SELECT clause.
- Use a comma (,) to separate multiple order columns

```
SELECT
    select_list
FROM
    table
ORDER BY
     column_1 ASC,
     column_2 DESC;
```

SQL FUNDAMENTALS: LIMIT

- The LIMIT clause is an optional part of the SELECT statement. You use the LIMIT clause to constrain the number of rows returned by the query.
- We can retrieve 10 rows instead of 1 MLN rows
- The **row_count** is a positive integer that specifies the number of rows returned (5, 10, 20, 1000, etc.)

```
SELECT
        column_list
FROM
        table
LIMIT row_count;
```

SQL Fundamentals

- SELECT statement
- ORDER BY, LIMIT clauses
- **WHERE and Logical Operators**

SQL FUNDAMENTALS: WHERE

- The WHERE clause is an optional clause of the SELECT statement. It appears after the FROM clause as the following statement
- When evaluating a SELECT statement with a WHERE clause, SQLite uses the following steps:
    - First, check the table in the FROM clause.
    - Second, evaluate the conditions in the WHERE clause to get the rows that met these conditions.
    - Third, make the final result set based on the rows in the previous step with columns in the SELECT clause.

```
SELECT
        column_list
FROM
        table
WHERE
        search_condition;
```

SQL FUNDAMENTALS: WHERE

For example, you can form a search condition as follows:

- WHERE column_1 **=** 100;
- WHERE column_2 **IN** (1,2,3); **IN** (‘Canada’, ‘UK’); **NOT IN** (‘Canada’, ‘Bulgaria’)
- WHERE column_3 **LIKE** 'An%';
- ‘%text%’ (will look for ‘text’ anywhere in the text); **NOT LIKE**
- WHERE column_4 **BETWEEN** 10 AND 20;
- WHERE column_5 **IS NULL**; **NOT NULL**
- WHERE column_6 <> “”

- Use **AND** or **OR** operator for multiple conditions

SQL FUNDAMENTALS: Data Types

Storage Class & Description	
NULL The value is a NULL value.	IS NULL / IS NOT NULL
INTEGER The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.	=, <, >, <>, etc.
REAL The value is a floating point value, stored as an 8-byte IEEE floating point number.	=, <, >, <>, etc.
TEXT The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE)	=, <> (use 'text' or "text") IS CASE-SENSITIVE
BLOB The value is a blob of data, stored exactly as it was input.	

SQL FUNDAMENTALS: SQL Comparison Operators

- A comparison operator tests if two expressions are the same. The following table illustrates the comparison operators that you can use to construct expressions:

Operator	Meaning
=	Equal to
<> or !=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to