SQL Basics

Zeina Elguindi

2024

Contents

1 Select Clause

1.1 Preparatory Information

Starter Data Types

Data Type	Format
Numeric	Positive/Negative Integers and Floats
Non-numeric	Must be Enclosed In Quotations
Date	'yyyy-mm-dd'

Comparison Operators

Operator	Description
=	Equal to
<> | Not equal to |
< | Less than |
<= | Less than or equal to |
> | Greater than |
>= | Greater than or equal to |

1.2 Select Clause

The **SELECT** clause selects and returns data from one or more tables. It must be accompanied by a **FROM** statement, indicating which table(s) to query from. All SQL queries must have these 2 statements.

1.2.1 Selecting Columns

Select All Columns: Use keyword "ALL" or "*" in the clause.

Select Specific Columns: Specify comma-separated list of columns to pull in the clause.

Simultaneous Calculations: You can perform operations on columns as you query them

- You can assign an alias to this column, this will change the column name to the alias name in the output. Otherwise, the new column will have its original name.

```
SELECT employee_name, employee_ID, employee_salary * 1.05 AS new_salary
FROM employees;
```

1.2.2 Selecting Rows

Select Without Duplicates: Use the "DISTINCT" keyword after your **SELECT** statement to avoid outputting any duplicate rows.

- If you select multiple columns, **DISTINCT** will evaluate a combination of the column values to determine duplicates. If you want to select multiple columns, but only remove duplicates from one column, use the **GROUP BY** clause (3.2).

2 Filtering Rows/Conditions

2.1 Where Clause

The **WHERE** clause filters the returned results. It uses logic to include/exclude certain columns, based on a comparison statement(s) a.k.a predicates.

- The comparison statements must correctly reference the SQL data types (chapter: 1.1).
- This clause must immediately follow the **FROM** clause.

2.2 Limit Clause

The **LIMIT** clause limits the number of rows returned. You can include an optional clause, **OFFSET**, that will skip the offset number of rows before returning.

2.3 Having Clause

The **HAVING** clause serves the same purpose as the **WHERE** clause, although the latter cannot perform conditions on rows after they are grouped.

3 Ordering Grouping Rows

3.1 Ordering Rows

"ORDER BY" keyword sorts the rows returned by **SELECT** clause in either ascending or descending order.
You can also sort rows by giving sortation priorities to specific columns. The following will sort column_1 in an ascending order, and then column_1 in descending order.

```
ORDER BY
    column_1 ASC,
    column_2 DESC;
```

3.2 Grouping Rows

The "GROUP BY" keyword groups rows based on the values of 1 or more columns.

- This is useful for aggregating data by grouping rows that share common values.

The following query counts the number of records for each Pet and Breed in vet_table. The **GROUP BY** clause aggregates the count in the 'Total' column.

Pet	Breed	VisitReason
Cat	Siamese	Check-up
Cat	Siamese	Vaccination
Dog	Poodle	Check-up

Table 1: vet_table

Pet	Breed	Total
Dog	Poodle	1
Cat	Siamese	2

Table 2: Query Result

```
    SELECT Pet, Breed, COUNT(*) AS Total
    FROM vet_table
    GROUP BY Pet, Breed
    ORDER BY Pet DESC;
```

You can now use **HAVING** to place conditions on the aggregate function 'COUNT(*)' (2.3).

4 Aggregate Functions

The parameters of statistical functions must be a single column or expression.
The output of these functions is a single column per function.
You can rename the outputted column by using an alias.
Note: To filter for columns with an aggregate function applied, you must use the **HAVING** clause (2.3).

4.1 Count

Parameter	Description
COUNT (∗)	Count all rows in the table
COUNT (*column_name*)	Count all non-null rows in the column
COUNT (*DISTINCT column_name*)	Count unique values in the column, excluding NULL rows.

Using the **GROUP BY** clause is especially useful when paired with the COUNT function:

From (3.2), we can see that a 'Total' column returned (using the count function), containing the count of each Pet and Breed combination.

Note: It is required to have any non-aggregate function stated within the **SELECT** statement to be stated within the **GROUP BY** statement

4.2 Avg & Sum

The **AVG()** function calculates and returns the average value or a set/column.

AVG ([ALL — DiSTINCT] expression)

The **SUM()** function calculates and returns the average value or a set/column.

SUM ([ALL — DiSTINCT] expression)

By default, AVG() and SUM() calculate using all values in the column, although, using the DISTINCT keyword would result in the calculation of unique values only.
The values within the column **must** be numeric.

4.3 Max & Min

The **MAX()** function returns the maximum value of a set/column.

MAX (expression)

The **MIN()** function returns the minimum value of a set/column.

MIN (expression)

The **DISTINCT** option is not available for MAX and MIN functions.

5 Advanced Operators

5.1 Is Null

5.2 Like

5.3 Between

5.4 Exists

5.5 In