MSDS 597

Lecture 2

Renaming

- AS operator
  - Change column names
  - Change table names
  - Example:
    - SELECT old_column AS new_column
- You can also skip the AS operator, but this is not recommended as it makes the query more difficult to read

Creating new columns and basic math operations

- Instead of **selecting** a column from a table, you can put down a single value like a number or a string
- Note that no new rows are being generated
- Data can also be manipulated on a row-by-row basis using mathematical functions in the SELECT
  - Standard mathematical functions are all supported (+, -, *, /)

Queries without a FROM clause

- SQL doesn’t require a FROM clause
  - Useful for testing
- You can also create “singletons” in SQL, which are queries that return a single value only
  - Singletons can be used as if it were that value

String Functions

LEFT(string, x) => Take x characters from the left of the string

RIGHT(string, x) => Take x characters from the right of the string

LOWER(string) => Convert all characters of the string to lowercase

UPPER(string) => Convert all characters of the string to uppercase

LENGTH(string) => Return the length of a string

TRIM(string) => Remove whitespace from the start and end of a string

CONCAT(string1, string2, string3...) => Concatenate 2 or more strings into 1 string

More Math functions

ABS(x) => Return the absolute value of x

ROUND(x, n) => Return x rounded to n decimal places

LEAST(x, y, z, ...) => Return the smallest value of the input numbers

GREATEST(x, y, z ...) => Return the greatest value of the input numbers

Integer division and casting

- In most variants of SQL, dividing one integer by another will result in an integer
    - This must be paid attention to when calculating fractions, as you will not get a float
    - In the cases where you want a float, you should either multiply by a float (e.g. 1.0) or cast the numerator or denominator to a float or a decimal before dividing
        - column_name::FLOAT
        - column_name::DECIMAL
        - CAST(column_name AS FLOAT)
        - CAST(column_name AS DECIMAL)

Query Evaluation Order

WHERE happens before SELECT

- For efficiency, the WHERE clause filters the source data first
- Then, the SELECT clause acts on the filtered data
- Thus, any columns defined in the SELECT are not available in the WHERE clause

BETWEEN, LIKE, and ILIKE

col_name BETWEEN x AND y

- Returns TRUE for any values between x and y inclusive
- Equivalent to
  - col_name >= x AND col_name <= y

col_name LIKE ‘%string%’

- Allows the use of wildcards to match strings
  - % matches 0 or more of any character
  - _ matches exactly one of any character

col_name ILIKE ‘%string%’

- Similar to LIKE but is **case insensitive**

CASE statements

Conditional transformations

- Use to create a new column based on conditional logic

```
CASE WHEN <insert condition> THEN <value>
     WHEN <insert condition> THEN <value>
     .......
     ELSE <value> END AS new_column_name
```

CASE statements continued

Secondary syntax

```
CASE <column_name>
    WHEN <equality_value> THEN <value>
    WHEN <equality_value> THEN <value>
    ....
    END AS new_column_name
```

DISTINCT

DISTINCT operator is used to remove duplicates in result data

Syntax

```
SELECT DISTINCT
      col1,
      col2,
      ...
```

Returns a distinct set of the combination of the columns in the SELECT statement.
DISTINCT can operate on one or more columns.

More filtering

Compare column_name against multiple values

```
WHERE
    column_name IN (x, y, z, ...)

WHERE
    column_name NOT IN (x, y, z...)

WHERE
    column_name IN (SELECT some_column FROM some_table)
```