DATA 210 Notes: Chapter 3 Supplement
SQL SELECT Statements
Conditions in WHERE and in HAVING clauses

When formulating queries with conditions (either in the WHERE clause or the HAVING clause), there are certain operators that can be useful.

1. Logical Operators – AND, OR, NOT

There are three logical operators -- AND, OR, and NOT – that can be used to compare two conditions at a time to determine whether a tuple or a group can be selected for the output.

Examples:

(a) `WHERE quantity > 50 AND price < 25`

Evaluates as **true** only if both quantity > 50 AND price < 25 evaluate as **true** for a tuple or group

(b) `WHERE quantity > 50 OR price < 25`

Evaluates as **true** if one, or both, of quantity > 50 and price < 25 evaluate as **true** for a tuple or group

(c) `WHERE NOT (quantity > 50)`

Evaluates as **false** if quantity > 50 is **true**, and evaluates as **true** if quantity ≤ 50. In this case we could also have expressed the consition as simply

```
WHERE quantity <= 50
```

Shortly, however, we will see better use for NOT.

2. IS NULL; IS NOT NULL

When checking to see whether an attribute has a NULL value, one does not use the compasrison operator =, instead uses the special operator **IS NULL**.

Example: `WHERE DeptId IS NULL`

We can turn this around and can check whether an attribute in a given tuple does not have a NULL value is via a condition such as the following, using the NOT operator above

```
WHERE NOT DeptId IS NULL
```

However, SQL also provides an IS NOT NULL operator to allow such a query to read more naturally, so the preferred form would be

```
WHERE DeptId IS NOT NULL
```

3. More Comparison Keywords

IS NULL and IS NOT NULL are example of ***comparison keywords***. There are a few others that areavailable in SQL that can be used to expand the comparison capabilities of SQL. These are "LIKE", "IN", and "BETWEEN...AND"

a. **LIKE** is used to check whether a column with a character-types value matches a prescribed pattern. We illustrate the use of this comparison keywork via some examples:

- **`LIKE 'ABC%'`**

  Here the percent character % means "any sequence of characters, regardless of values, of length 0 or greater." A CHAR or VARCHAR value would evaluate as true against this condition if it began with the letters ABC (or was only ABC)

- **`LIKE 'A%BC'`**

  A CHAR or VARCHAR value would evaluate as true against this condition if it began with the letter A and ended with the letters BC. It does not matter how many letters are between the A and the BC.

- `LIKE 'S_ _ _ _ _'` (this is an S followed by five underscores)

  Here the underscore character represents a single character, regardless of value. A CHAR or VARCHAR value would evaluate as true against this condition if it consisted of 5 characters only, the first of which was an S.

- `LIKE '%9'`

  Matches any sequence of characters ending with a 9 (so the length has to be at least 1)

- `LIKE '%101%'`

  Matches a sequence of characters of any length containing '101' (so the length has to be at least 3.

- `NOT LIKE '%101%'`

  The sequence of characters cannot contain '101' as a subsequence anywhere

b. **IN** checks whaether column value is equal to any one of a specified set of values. In its simplest form the sepcified are given as a comma-separated list of values, with the list surrpounded by parentheses.

Example:

```
Grade IN ('A', 'B', 'C')
```

Evaluates as true if the value of Grade is either 'A', 'B', or 'C'

c. **BETWEEN...AND**: Evaluates as true if an attribute value is between two values, including the end values specified in the range.

Example: `Price BETWEEN 21.25 AND 75.50`

This could be also done using the relational operator >=, <= and the logical operator AND.

The BETWEEN condition can also be combined with the SQL NOT operator, as as with the case of IS NULL, wi9th a more euphemistic syntax

Example: Instead of

```
NOT (Price BETWEEN 21.25 AND 75.50)
```

we can write

```
Price NOT BETWEEN 21.25 AND 75.50
```