# Logical Operator Keywords

Here are the most important Logical Operators summarized in a table.

Logical Operators can be used for conditions as they show a result in form of a `boolean` (True/False) or Unknown. So, e.g. if an exact value is `True` for a value, a Logical Operator can proof that it's True.

| Logical Operator | Explanation |
|---|---|
| ALL | If all comparisons are True: return True |
| ANY | If any comparison is True: return True |
| AND | If both expressions are True: return True |
| EXISTS | If a subquery contains rows: return True |
| IN | If compared value is equal to at least one value: return True |
| BETWEEN | If there are values in given range: return True |
| NOT | Reverses the value of any boolean |
| OR | If either expression is True: return True |

## `HAVING` Clause

Unlike where clause which imposes conditions on columns `Having` clause enables you to specify conditions that filter which group results appear in the results.

## Syntax

```sql
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

## Description

- Used with `aggregate functions`
- Must follow `GROUP BY` clause in the query

## Aggregate Functions

- SQL aggregation is the task of collecting a set of values to return a single value.
- An aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

## Aggregate Functions Examples

Suppose this are the table given to us

| **Students** | **table** | |
|---|---|---|
| rollno | name | class |
| 1 | Sanskriti | TE |
| 1 | Shree | BE |
| 2 | Harry | TE |
| 3 | John | TE |
| 3 | Shivani | TE |

| **purchase** | **table** | |
|---|---|---|
| item | price | customer_name |
| Pen | 10 | Sanskriti |
| Bag | 1000 | Sanskriti |
| Vegetables | 500 | Sanskriti |
| Shoes | 5000 | Sanskriti |
| Water Bottle | 800 | XYZ |
| Mouse | 120 | ABC |
| Sun Glasses | 1350 | ABC |

### AVG function

Calculates `average` of the given column of values

```
SELECT AVG(price) AS Avg_Purchase, customer_name
FROM purchase
GROUP BY customer_name;
```

| **Avg_Purchase** | **customer_name** |
|---|---|
| 1627.5000 | Sanskriti |

## SUM function

Calculates `sum` of values of given column.

```
SELECT SUM(price) AS Total_Bill, customer_name
FROM purchase
GROUP BY customer_name;
```

| Total_Bill | customer_name |
|---|---|
| 6510 | Sanskriti |

## COUNT function

Gives `count` of entries/ values in given column.

```
SELECT COUNT(item) AS Total_Items, customer_name
FROM purchase
GROUP BY customer_name;
```

| Total_Items | customer_name |
|---|---|
| 4 | Sanskriti |

## MAX function

Return `maximum` value from the number of values in the column.

```
SELECT MAX(price) AS Highest_Purchase, customer_name
FROM purchase
GROUP BY customer_name;
```

| Highest_Purchase | customer_name |
|---|---|
| 5000 | Sanskriti |

## MIN function

Return `minimum` value from the number of values in the column.

```
SELECT MIN(price) AS Lowest_Purchase, customer_name
FROM purchase
GROUP BY customer_name;
```

| Lowest_Purchase | customer_name |
|---|---|
| 10 | Sanskriti |

## Having clause Examples

### Example 1

```
SELECT COUNT(class) AS strength, class
FROM Students
GROUP BY class
HAVING COUNT(class) > 2;
```

Above query gives number of students in a class `having` number of students > 2

| strength | class |
|---|---|
| 4 | TE |

### Example 2

```
SELECT customer_name, MIN(price) AS MIN_PURCHASE
FROM purchase
GROUP BY customer_name
HAVING MIN(price) > 10;
```

Above query finds `minimum` price which is > 10

| customer_name | MIN_PURCHASE |
|---|---|
| XYZ | 800 |
| ABC | 120 |

### Example 3

```
SELECT customer_name, AVG(price) AS Average_Purchase
FROM purchase
GROUP BY customer_name
HAVING AVG(price) > 550
ORDER BY customer_name DESC;
```

Above query calculates `average` of price and prints customer name and average price which is greater than 550 with descending `order` of customer names.

| customer_name | Average_Purchase |
|---|---|
| XYZ | 800.0000 |
| Sanskriti | 1627.5000 |
| ABC | 735.0000 |

## Example 4

```
SELECT customer_name, SUM(price) AS Total_Purchase
FROM purchase
WHERE customer_name
LIKE "S%"
GROUP BY customer_name
HAVING SUM(price) > 1000;
```

Calculates `SUM` of price and returns customer name and sum > 1000.

| customer_name | Total_Purchase |
|---|---|
| Sanskriti | 6510 |