IN

- Used to compare a value of the column for equality to a list of literal values that have been specified.
- Syntax:

```
select ….from…. where
 <column-name> in
(value1, value2, …);
```

BETWEEN

- Used to check whether a value of a column exists within a given range or not.
- Syntax:

```
select ….from…. where
 <column-name> between
<minvalue> AND <maxvalue>
```

IS NULL

- Used to compare a value with NULL .

NOT

- A predicate used for negation.
- It can be used in conjunction with other predicates e.g. `LIKE`, `IN`, `BETWEEN`, `EXISTS`, `IS NULL` etc.

ORDER BY

ORDER BY

- Used to fetch records from the table in sorted order.
- Syntax:

```
SELECT….FROM TABLE ORDER BY
<column-name> [order-type]
[,<column-name> [order-type]]
```

- Order types can be:
  - ASC (DEFAULT)
  - DESC

SQL Row Limiting Clause

By Rahul Barve

SQL Row Limiting Clause

- It is possible to limit the no of rows returned by a query.
- Syntax:

```
SELECT…
    FROM…
    [WHERE…]
    [ORDER BY…]
    [OFFSET offset {row | ROWS}]
    [FETCH {FIRST|NEXT}
    [row_count |     percent PERCENT}]
    {row | ROWS}{ONLY | WITH TIES}]
```

SQL Row Limiting Clause

- Example

```
SELECT ENAME FROM EMP ORDER BY SAL
FETCH FIRST 5 ROWS ONLY

SELECT ENAME FROM EMP ORDER BY SAL
OFFSET 5 ROWS
FETCH NEXT 5 ROWS ONLY
```

Substitution Variables

By Rahul Barve

Substitution Variables

- Substitution variables are used to run the same SQL query multiple times with different values.
- They are denoted by ‘&’.
- E.g.

```
SELECT ENAME FROM EMP WHERE SAL
BETWEEN &MIN AND &MAX;
```

Substitution Variables

- Substitution variables can also be used in place of column names, conditions and so on.
- E.g.

```
SELECT EMPNO, &COL FROM EMP WHERE
&CONDITION ORDER BY &ORDER_COL;
```

Substitution Variables

- To reuse the variable value without prompting the user each time, '&&' is used.
- E.g.

```
SELECT ENAME, &&COL FROM EMP ORDER
BY &COL;
```

DEFINE Command

By Rahul Barve

DEFINE Command

- DEFINE command is used to create a variable and assign a value to it.
- E.g.

```
DEFINE ENO = 7369
SELECT ENAME FROM EMP WHERE EMPNO=&ENO;
```

Dual

By Rahul Barve

Dual

- The DUAL table is a special one-row, one-column table present by default in Oracle and other database installations.
- In Oracle, the table has a single `VARCHAR2(1)` column called `DUMMY` that has a value of '`X`' .

Dual

- Suitable for use in selecting a pseudo column such as `SYSDATE` or `USER`.
- E.g.

```
select sysdate from dual;
```

SQL Functions

By Rahul Barve

SQL Functions

- SQL provides various types of built-in functions which are used to perform data manipulation.

SQL Functions

- SQL functions are categorized as:
  - Utility Function (Scalar Functions)
  - Aggregate Functions
  - String Functions
  - Date Functions

Utility Functions

By Rahul Barve

Utility Functions

- Used to perform generic mathematical operations.
- Accept a single value and return the result.

Utility Functions

- Utility functions are:
  - `abs()`
  - `floor()`
  - `ceil()`
  - `round()`
  - `sqrt()`
  - `sin()`
  - `cos()`
  - `tan()`

Aggregate Functions

By Rahul Barve

Aggregate Functions

- SQL aggregate functions return a single value, calculated from values in a column.

Aggregate Functions

- Aggregate functions are:
  - `sum()`
  - `avg()`
  - `max()`
  - `min()`
  - `count()`

String Functions

By Rahul Barve

String Functions

- Used to perform manipulation on the string data.

String Functions

- String functions are:
  - `length()`
  - `reverse()`
  - `lower()`
  - `upper()`
  - `substr()`
  - `ltrim()`
  - `rtrim()`
  - `trim()`

Date Functions

By Rahul Barve

Date Functions

- Used to perform manipulation on the columns of type `date`.

Date Functions

- Date functions are:
  - `add_months(date, no_of_months)`
  - `last_day(date)`
  - `next_day(date,day_of_week)`
  - `months_between(date1, date2)`