Oracle ORDER BY Clause

In Oracle, a table stores its rows in unspecified order regardless of the order which rows were inserted into the database. To query rows in either ascending or descending order by a column, you must explicitly instruct Oracle Database that you want to do so.

To sort data, you add the `ORDER BY` clause to the `SELECT` statement as follows:

```
SELECT
    column_1,
    column_2,
    column_3,
    ...
FROM
    table_name
ORDER BY
    column_1 [ASC | DESC] [NULLS FIRST | NULLS LAST],
    column_1 [ASC | DESC] [NULLS FIRST | NULLS LAST],
    ...Code language: CSS (css)
```

To sort the result set by a column, you list that column after the `ORDER BY` clause.

Following the column name is a sort order that can be:

- `ASC` for sorting in ascending order
- `DESC` for sorting in descending order

By default, the `ORDER BY` clause sorts rows in ascending order whether you specify `ASC` or not. If you want to sort rows in descending order, you use `DESC` explicitly.

`NULLS FIRST` places NULL values before non-NULL values and `NULLS LAST` puts the NULL values after non-NULL values.

The `ORDER BY` clause allows you to sort data by multiple columns where each column may have different sort orders.

Note that the `ORDER BY` clause is always the last clause in a `SELECT` statement.

retrieves customer name, address, and credit limit from the `customers` table:

```
SELECT
    name,
    address,
    credit_limit
FROM
    customers;
```

A) Sorting rows by a column example

To sort the customer data by names alphabetically in ascending order,

```
SELECT
    name,
    address,
    credit_limit
FROM
    customers
ORDER BY
    name ASC;
```

To sort customer by name alphabetically in descending order, we explicitly use `DESC` after the column name in the `ORDER BY` clause as follows:

```
SELECT
    name,
    address,
    credit_limit
FROM
    customers
ORDER BY
    name DESC;
```

B) Sorting rows by multiple columns example

To sort multiple columns, you separate each column in the `ORDER BY` clause by a comma.

to sort contacts by their first names in ascending order and their last names in descending order,

```
SELECT
  first_name,
  last_name
FROM
  contacts
ORDER BY
  first_name,
  last_name DESC;
```

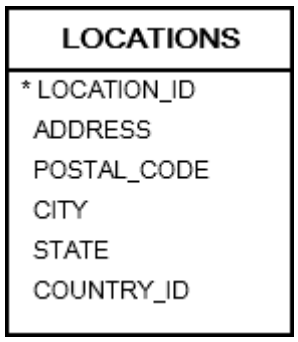
C) Sort rows by column’s positions example

```
SELECT
    name,
    credit_limit
FROM
    customers
ORDER BY
    2 DESC,
    1;
```

In this example, the position of `name` column is 1 and `credit_limit` column is 2.

In the `ORDER BY` clause, we used these column positions to instruct the Oracle to sort the rows.

C) Sorting rows with NULL values examples




```
SELECT
    country_id,
    city,
    state
FROM
    locations
ORDER BY
    city,
    state;
```

```
SELECT
    country_id,
    city,
    state
FROM
    locations
ORDER BY
    state ASC NULLS FIRST;
```

```
SELECT
   country_id,
   city,
   state
FROM
   locations
ORDER BY
   state
ASC NULLS LAST;
```

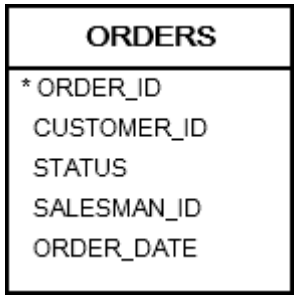
D) Sorting rows by the result of a function or expression

The `ORDER BY` clause allows you to apply a function e.g., string function and math function on a column and sorts the data by the result of the function.

```
SELECT
  customer_id,
```

```
  name
FROM
  customers
ORDER BY
  UPPER( name );
```

E) Sorting by date example




```
SELECT
   order_id,
   customer_id,
   status,
   order_date
FROM
   orders
ORDER BY
   order_date DESC;
```