Oracle GROUP BY

The `GROUP BY` clause is used in a `SELECT` statement to group rows into a set of summary rows by values of columns or expressions. The `GROUP BY` clause returns one row per group.

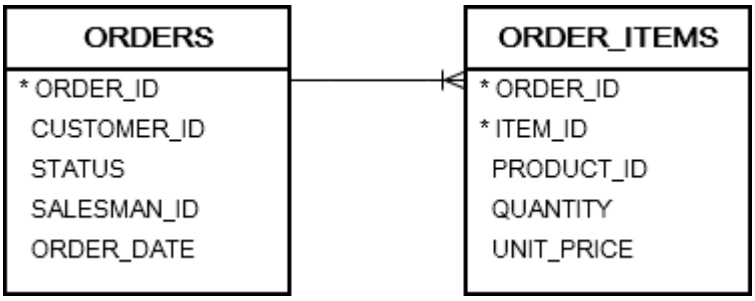
The `GROUP BY` clause is often used with aggregate functions such as `AVG()`, `COUNT()`, `MAX()`, `MIN()` and `SUM()`.
In this case, the aggregate function returns the summary information per group. For example, given groups of products in several categories, the `AVG()` function returns the average price of products in each category.

```
SELECT
    column_list
FROM
    T
GROUP BY c1,c2,c3;
```

The `GROUP BY` clause appears after the `FROM` clause. In case `WHERE` clause is presented, the `GROUP BY` clause must be placed after the `WHERE` clause

```
SELECT
   column_list
FROM
  T
WHERE
   condition
GROUP BY c1, c2, c3;
```

The `GROUP BY` clause groups rows by values in the grouping columns such as `c1`, `c2` and `c3`. The `GROUP BY` clause must contain only aggregates or grouping columns.



A) Oracle `GROUP BY`

find unique order statuses from the `orders` table:

```
SELECT
    status
FROM
    orders
GROUP BY
    status;
```

B) Oracle `GROUP BY` with an aggregate function

returns the number of orders by customers

```
SELECT
    customer_id,
    COUNT( order_id )
FROM
    orders
GROUP BY
    customer_id
ORDER BY
    customer_id;
```

To get more meaningful data, you can join the `orders` table with the `customers` table

```
SELECT
    name,
    COUNT( order_id )
FROM
    orders
INNER JOIN customers
        USING(customer_id)
GROUP BY
    name
ORDER BY
    name;
```

C) Oracle `GROUP BY` with an expression

groups the orders by year and returns the number of orders per year

```
SELECT
    EXTRACT(YEAR FROM order_date) YEAR,
    COUNT( order_id )
FROM
    orders
GROUP BY
    EXTRACT(YEAR FROM order_date)
```

```
ORDER BY
    YEAR;
```

D) Oracle `GROUP BY` with `WHERE` clause

The `GROUP BY` clause with a `WHERE` clause to return the number of shipped orders for every customer:

```
SELECT
   name,
   COUNT( order_id )
FROM orders
   INNER JOIN customers USING(customer_id)
WHERE
   status = 'Shipped'
GROUP BY
   name
ORDER BY
   name;
```

E) Oracle `GROUP BY` with `ROLLUP`

computes the sales amount and groups them by `customer_id`, `status`, and (`customer_id`, `status`):

```
SELECT
    customer_id,
    status,
    SUM( quantity * unit_price ) sales
FROM
    orders
INNER JOIN order_items
        USING(order_id)
GROUP BY
    ROLLUP(
        customer_id,
        status
    );
```