Reference Notes of Oracle 12c SQL Part 3

Oracle Trainer :- Sekhar

Oracle uses ROWNUM to fetch limited number of records.

To execute in oracle the command is

```
sql> select * from salespeople
    where rownum <=4;
```

The above query will execute the top 4 records.

==

SQL - Distinct Keyword

****

The SQL DISTINCT keyword is used in conjunction with SELECT statement to eliminate all the duplicate records and fetching only unique records.

There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only unique records instead of fetching duplicate records.

Syntax:

```
sql>select distinct city from salespeople;

sql> select count(distinct city) from salespeople;
```

==

Between Operator

****

The BETWEEN operator is used to select values within a range.

The SQL BETWEEN Operator

The BETWEEN operator selects values within a range. The values can be numbers, text, or dates.

==

<u>NOT BETWEEN Operator Example</u>

BETWEEN Operator with Text Value Example

****

The following SQL statement selects all salespeople whose name begins with any of the letter BETWEEN 'C' and 'M':

```
sql> SELECT * FROM salespeople;
    WHERE Name  BETWEEN 'C' AND 'M';
```

BETWEEN Operator with IN Example

****

The following SQL statement selects all employees with a salary

BETWEEN 50000 AND 150000

but employee number of 99, 2 or 390 should not be displayed:

```
sql> SELECT * FROM employee
    WHERE (basic BETWEEN 50000 AND 150000)
            AND NOT empno IN (99,2,390,66);
```

In the above query we are using the a sub query and also the in operator with not operator so that only those employees are print who draw salary in the given range but particular employees no should not be printed due to as per client requirement.

NOT BETWEEN Operator with Text Value Example

****

The following SQL statement selects all employees with empno

beginning with any of the letter NOT BETWEEN 'C' and 'M':

```
sql>SELECT * FROM employee
   WHERE Name not   BETWEEN 'C' AND 'M';
```

==

Assume for the following examples we have an employee table.

****

```
SQL> SELECT * FROM employee
        ORDER BY NAME;
```

The above query will sort the record based on name in ascending order.

```
sql> select empname, basic from employee
     order by basic desc;
```

the abovery query will sort in descending order.

****

<u>Group By</u>

****

The SQL GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.

The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

If you want to know the total amount of salary on each designation,

then GROUP BY query would be as follows:

```
SQL> SELECT desig, SUM(basic) FROM employee
     GROUP BY desig;
```

In the above query group by is used so that the query in the memory of the computer will group the records based on designation and then each group will have only 1 output in the screen along with the salary total of each group.

----

write a query where you will print the maximum salary drawn in each group using aggregate functions.

```
sql> SELECT desig, Max(basic) FROM employee
     GROUP BY desig;
```

```
sql> SELECT city SUM(basic) FROM employee
     GROUP BY city;
```

in the above query we are showing city wise salary using the group by clause.

```
sql> SELECT city, min(basic) FROM employee
     GROUP BY city;
```

in the above query we are showing city wise minimum salary using the group by clause.

```
sql> SELECT city, avg(basic) FROM employee
     GROUP BY city;
```

In the above query we will be showing city wise average salary.

```
sql>SELECT city, avg(basic)
    max(basic), min(basic), sum(basic)
    FROM employee
    GROUP BY city;
```

The above query will print city wise sum, max, min & average salary.

```
sql> select city, count(basic)
    from employee
    group by city;
```

the above query will print the number of employees in each city.using count function.

The HAVING Clause

****

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

The HAVING clause enables you to specify conditions that filter which group results appear in the final results.

The WHERE clause places conditions on the selected columns, where as the HAVING clause places conditions on groups created by the GROUP BY clause.

```
sql> SELECT Desig, count(desig)
FROM employee
GROUP BY Desig
HAVING COUNT(Desig) >= 2;
```

The above query will display those designation that appears more than twice in the table and for that we are using the group by and having clause and the agregate function.

==

waq where you will print all those city where their are 2 or less than 2 salespeople;

```
sql> select city, count(city)
       from salespeople
        group by city
          having count(city) <=2
          order by city desc;
```

<u>Views</u>

*****

Views are logical table based on real table which are called base tables.

In views our query is stored which can be executed from time to time.
You can add records, modify records through views.

Write a view called london to see salespeople who residing in london.

```
sql> create view london
         as select * from salespeople where city = 'London';
```

```
sql> select * from london;
```

The above query will display those salespeople who stay in london.

To add records through view london

```
sql> insert into london values
        (1900, 'James Bond', 'New York', null);


sql>  insert into london values
        (1450, 'Jack Patel', 'London', 654.34);
```

To see all the london records through view

```
sql> select * from london;
```

To drop a view london

```
sql> drop view london
```
