4.3 GROUP BY, GROUP BY HAVING. ROLLUP, CUBE and DECODE

1. **GROUP BY:** An aggregate function takes multiple rows of data returned by a query and aggregates them into a single result row.

2. **GROUP BY HAVING:** The SQL HAVING Clause is used in combination with the GROUP BY Clause to restrict the groups of returned rows to only those whose the condition is TRUE.

   Example:

```
SELECT department, SUM(sales) AS "Total sales"
FROM order_details
GROUP BY department
HAVING SUM(sales) > 1000;
```

3. **ROLLUP:** ROLLUP enables a SELECT statement to *calculate multiple levels of subtotals across a specified group of dimensions.* It also calculates a grand total. ROLLUP is a simple extension to the GROUP BY clause, so its syntax is extremely easy to use. The ROLLUP extension is highly efficient, adding minimal overhead to a query.

   Syntax:

```
SELECT ... GROUP BY
  ROLLUP(grouping_column_reference_list)
```

   Details: ROLLUP will create subtotals at *n+1* levels, where n is the number of grouping columns. For instance, if a query specifies ROLLUP on grouping columns of Time, Region, and Department (n=3), the result set will include rows at four aggregation levels.

   Example:

```
select Dept,Desig,count(*)Total
from emp
group by rollup(Dept,Desig)
order by Dept,Desig;
```

   Sample Output:

```
   DEPT DESIG          TOTAL
---------- --------------------  ----------
10 Asst Manager  2
10 Manager 3
10 5
20 Asst Manager  3
20 Manager 2
20 5
30 Asst Manager  1
30 Manager 2
30 3
13
```

You should learn other possible options on ROLLUP.

When to Use ROLLUP?

(a) It is very helpful for subtotaling along a hierarchical dimension such as time or geography. For instance, a query could specify a ROLLUP of year/month/day or country/state/city.

(b) It simplifies and speeds the population and maintenance of summary tables. Data warehouse administrators may want to make extensive use of it. Note that population of summary tables is even faster if the ROLLUP query executes in parallel.

4. **CUBE:** Why CUBE? It has a strong relationship with Data Warehousing. Lets have a look on it.(Slides).

   CUBE enables a SELECT statement to calculate subtotals for all possible combinations of a group of dimensions. It also calculates a grand total. This is the set of information typically needed for all cross-tabular reports, so CUBE can calculate a cross-tabular report with a single SELECT statement. Like ROLLUP, CUBE is a simple extension to the GROUP BY clause.

   Syntax:

```
SELECT ...  GROUP BY
  CUBE (grouping_column_reference_list)
```

Example:

```
 select Dept,Desig,count(*)Total
from emp
```

```
group by cube(Dept,Desig)
order by Dept,Desig;
```

Sample Output:

```
 DEPT  DESIG      TOTAL
---------- -------------------- ----------
10 Asst Manager   2
10 Manager  3
10  5
20 Asst Manager   3
20 Manager  2
20  5
30 Asst Manager   1
30 Manager  2
30  3
   Asst Manager   6
   Manager  7

      DEPT DESIG      TOTAL
---------- -------------------- ----------
13
```

When to Use CUBE?

(a) Use CUBE in any situation requiring cross-tabular reports.

(b) CUBE is especially valuable in queries that use columns from multiple dimensions rather than columns representing different levels of a single dimension. For instance, a commonly requested cross-tabulation might need subtotals for all the combinations of month/state/product.

4.4 Date Operations

DATE is an Oracle datatype, just as VARCHAR2 and NUMBER are, and it has its own unique properties. The DATE datatype is stored in a special internal Oracle format that includes not just the month, day, and year, but also the hour, minute, and second. TIMESTAMP datatype stores fractional seconds.

```
 select SysDate from DUAL;

 SYSDATE
---------
```