NAME: Heet Dhanuka
ROLL NO.: B -34
BATCH: B2

DWDM Practical 2

Aim: Write and Execute SQL aggregation queries for data warehouse.

Details: To run queries for CUBE, PARTIAL CUBE, ROLLUP, PARTIAL ROLLUP, GROUPING, GROUPING SETS

Theory:
Aggregation is a fundamental part of data warehousing. To improve aggregation performance in your warehouse, Oracle Database provides the following functionality:
CUBE and ROLLUP extensions to the GROUP BY clause

- Three GROUPING functions
- GROUPING SETS expression
- Pivoting operations

The CUBE, ROLLUP, and GROUPING SETS extensions to SQL make querying and reporting easier and faster. CUBE, ROLLUP, and grouping sets produce a single result set that is equivalent to a UNION ALL of differently grouped rows. ROLLUP calculates aggregations such as SUM, COUNT, MAX, MIN, and AVG at increasing levels of aggregation, from the most detailed up to a grand total. CUBE is an extension similar to ROLLUP, enabling a single statement to calculate all possible combinations of aggregations. The CUBE, ROLLUP, and the GROUPING SETS extension lets you specify just the groupings needed in the GROUP BY clause. This allows efficient analysis across multiple dimensions without performing a CUBE operation. Computing a CUBE creates a heavy processing load, so replacing cubes with grouping sets can significantly increase performance.

To enhance performance, CUBE, ROLLUP, and GROUPING SETS can be parallelized: multiple processes can simultaneously execute all of these statements. These capabilities make aggregate calculations more efficient, thereby enhancing database performance, and scalability.

The three GROUPING functions help you identify the group each row belongs to and enable sorting subtotal rows and filtering results.

ROLLUP Extension to GROUP BY

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.

The action of ROLLUP is straightforward: it creates subtotals that roll up from the most detailed level to a grand total, following a grouping list specified in the ROLLUP clause. ROLLUP takes as its argument an ordered list of grouping columns. First, it calculates the standard aggregate values specified in the GROUP BY clause. Then, it creates progressively higher-level subtotals, moving from right to left through the list of grouping columns. Finally, it creates a grand total.

ROLLUP creates subtotals at n+1 levels, where n is the number of grouping columns. For instance, if a query specifies ROLLUP on grouping columns oftime , region, and department (n=3), the result set will include rows at four aggregation levels.

You might want to compress your data when using ROLLUP. This is particularly useful when there are few updates to older partitions.

When to Use ROLLUP

Use the ROLLUP extension in tasks involving subtotals.

- It is very helpful for subtotaling along a hierarchical dimension such as time or geography. For instance, a query could specify a ROLLUP (y, m, day) or ROLLUP (country, state, city).
- For data warehouse administrators using summary tables, ROLLUP can simplify and speed up the maintenance of summary tables.

ROLLUP Syntax

ROLLUP appears in the GROUP BY clause in a SELECT statement. Its form is:

SELECT ... GROUP BY ROLLUP (grouping_column_reference_list)

PARTIAL ROLLUP

You can also roll up so that only some of the sub-totals will be included. This partial rollup uses the following syntax:

GROUP BY expr1, ROLLUP (expr2, expr3);

In this case, the GROUP BY clause creates subtotals at (2+1=3) aggregation levels. That is, at level (expr1, expr2, expr3), (expr1, expr2), and (expr1).

CUBE Extension to GROUP BY

CUBE takes a specified set of grouping columns and creates subtotals for all of their possible

combinations. In terms of multidimensional analysis, CUBE generates all the subtotals that could be calculated for a data cube with the specified dimensions. If you have specified CUBE (time, region,department), the result set will include all the values that would be included in an equivalent ROLLUP statement plus additional combinations. For instance, in Figure 21-1, the departmental totals across regions (279,000 and 319,000) would not be calculated by a ROLLUP (time, region,department) clause, but they would be calculated by a CUBE (time, region, department) clause. If n columns are specified for a CUBE, there will be 2 to the n combinations of subtotals returned. Example 21-4 gives an example of a three-dimension cube.

When to Use CUBE

Consider Using CUBE in any situation requiring cross-tabular reports. The data needed for cross-tabular reports can be generated with a single SELECT using CUBE.
Like ROLLUP, CUBE can be helpful in generating summary tables. Note that population of summary tables is even faster if the CUBE query executes in parallel.

CUBE is typically most suitable 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, and product. These are three independent dimensions, and analysis of all possible subtotal combinations is commonplace. In contrast, a cross-tabulation showing all possible combinations of year, month, and day would have several values of limited interest, because there is a natural hierarchy in the time dimension. Subtotals such as profit by day of month summed across year would be unnecessary in most analyses. Relatively few users need to ask "What were the total sales for the 16th of each month across the year?" See "Hierarchy Handling in ROLLUP and CUBE" for an example of handling rollup calculations efficiently.

CUBE Syntax

CUBE appears in the GROUP BY clause in a SELECT statement. Its form is:

SELECT ... GROUP BY CUBE (grouping_column_reference_list)

Partial CUBE

Partial CUBE resembles partial ROLLUP in that you can limit it to certain dimensions and precede it with columns outside the CUBE operator. In this case, subtotals of all possible combinations are limited to the dimensions within the cube list (in parentheses), and they are combined with the preceding items in the GROUP BY list.

The syntax for partial CUBE is as follows:

GROUP BY expr1, CUBE(expr2, expr3)

This syntax example calculates 2*2, or 4, subtotals. That is:

- (expr1, expr2, expr3)
- (expr1, expr2)
- (expr1, expr3)
- (expr1)

GROUPING Functions

Two challenges arise with the use of ROLLUP and CUBE. First, how can you programmatically determine which result set rows are subtotals, and how do you find the exact level of aggregation for a given subtotal? You often need to use subtotals in calculations such as percent-of-totals, so you need an easy way to determine which rows are the subtotals. Second, what happens if query results contain both stored NULL values and "NULL" values created by a ROLLUP or CUBE?
How can you differentiate between the two? This section discusses some of these situations.

GROUPING Function

GROUPING handles these problems. Using a single column as its argument, GROUPING returns 1 when it encounters a NULL value created by a ROLLUP or CUBE operation. That is, if the NULL indicates the row is a subtotal, GROUPING returns a 1. Any other type of value, including a stored NULL, returns a 0.

GROUPING appears in the selection list portion of a SELECT statement. Its form is:

SELECT ... [GROUPING (dimension_column)...] ...
GROUP BY ... {CUBE | ROLLUP| GROUPING SETS} (dimension_column)

GROUPING SETS Expression

You can selectively specify the set of groups that you want to create using a GROUPING SETS expression within a GROUP BY clause. This allows precise specification across multiple dimensions without computing the whole CUBE.

DATA WAREHOUSE SCHEMA USED: SH
The hypothetical company has sales across the world and tracks sales by both dollars and quantities information. SH is sales history.

Write the queries for the following:

Q1. Find the total sales by country_id and channel_desc for the US and GB through the Internet and direct sales in September 2000 and October 2000 using ROLL-UP Extension. The query should return the following:

- The aggregation rows that would be produced by GROUP BY
- The First-level subtotals aggregating across country_iso_code for each combination of channel_desc and calendar_month.
- Second-level subtotals aggregating across calendar_month_desc and country_iso_code for each channel_desc value.
- A grand total row.

Query:

```
SQL> SELECT channels.channel_desc, calendar_month_desc,
  2  countries.country_iso_code,
  3  TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
  4  FROM sales, customers, times, channels, countries
  5  WHERE sales.time_id=times.time_id
  6  AND sales.cust_id=customers.cust_id
  7  AND customers.country_id = countries.country_id
  8  AND sales.channel_id = channels.channel_id
  9  AND channels.channel_desc IN ('Direct Sales', 'Internet')
 10  AND times.calendar_month_desc IN ('2000-09', '2000-10')
 11  AND countries.country_iso_code IN ('GB', 'US')
 12  GROUP BY ROLLUP(channels.channel_desc, calendar_month_desc,
 13  countries.country_iso_code);

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Internet             2000-09  GB          16,569
Internet             2000-09  US         124,224
Internet             2000-09             140,793
Internet             2000-10  GB          14,539
Internet             2000-10  US         137,054
Internet             2000-10             151,593
Internet                                 292,387
Direct Sales         2000-09  GB          85,223
Direct Sales         2000-09  US         638,201
Direct Sales         2000-09             723,424
Direct Sales         2000-10  GB          91,925

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Direct Sales         2000-10  US         682,297
Direct Sales         2000-10             774,222
Direct Sales                           1,497,646
                                       1,790,032

15 rows selected.
```

Q2. Find the total sales by country_iso_code and channel_desc for the US and GB through the Internet and direct sales in September 2000 and October 2000 using CUBE aggregation across three dimensions- channel_desc, calendar_month_desc, countries. Country_iso_code.

Query:

```
SQL> SELECT channel_desc, calendar_month_desc, country_iso_code,
  2         TO_CHAR(SUM(amount_sold), '9,999,999,999') AS SALES$
  3  FROM sales, customers, times, channels, countries
  4  WHERE sales.time_id = times.time_id
  5    AND sales.cust_id = customers.cust_id
  6    AND customers.country_id = countries.country_id
  7    AND sales.channel_id = channels.channel_id
  8    AND channels.channel_desc IN ('Direct Sales', 'Internet')
  9    AND times.calendar_month_desc IN ('2000-09', '2000-10')
 10    AND countries.country_iso_code IN ('US', 'GB')
 11  GROUP BY CUBE(channel_desc, calendar_month_desc, countries.country_iso_code);

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
                                  1,790,032
                              GB    208,257
                              US  1,581,775
                     2000-09        864,217
                     2000-09  GB    101,792
                     2000-09  US    762,425
                     2000-10        925,815
                     2000-10  GB    106,465
                     2000-10  US    819,351
Internet                            292,387
Internet                     GB      31,109

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Internet                     US     261,278
Internet             2000-09        140,793
Internet             2000-09  GB     16,569
Internet             2000-09  US    124,224
Internet             2000-10        151,593
Internet             2000-10  GB     14,539
Internet             2000-10  US    137,054
Direct Sales                      1,497,646
Direct Sales                 GB     177,148
Direct Sales                 US  1,320,497
Direct Sales         2000-09        723,424

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Direct Sales         2000-09  GB     85,223
Direct Sales         2000-09  US    638,201
Direct Sales         2000-10        774,222
Direct Sales         2000-10  GB     91,925
Direct Sales         2000-10  US    682,297

27 rows selected.
```

Q3. Find the total sales by country_iso and channel_desc for the US and France through the Internet and direct sales in September 2000

Query:

```
SQL> SELECT channel_desc, country_iso_code,
  2         TO_CHAR(SUM(amount_sold), '9,999,999,999') AS SALES$
  3  FROM sales, customers, times, channels, countries
  4  WHERE sales.time_id = times.time_id
  5    AND sales.cust_id = customers.cust_id
  6    AND customers.country_id = countries.country_id
  7    AND sales.channel_id = channels.channel_id
  8    AND channels.channel_desc IN ('Direct Sales', 'Internet')
  9    AND times.calendar_month_desc = '2000-09'
 10    AND countries.country_iso_code IN ('US', 'FR')
 11  GROUP BY channel_desc, countries.country_iso_code;

CHANNEL_DESC         CO SALES$
-------------------- -- --------------
Direct Sales         FR          61,202
Direct Sales         US         638,201
Internet             US         124,224
Internet             FR           9,597
```

Q4. Find the total sales by country_iso and channel_desc for the US and GB through the Internet and direct sales in September 2000 and October 2009 using PARTIAL ROLL-UP. The query should return the following:

- Regular aggregation rows that would be produced by GROUP BY without using ROLLUP.
- First-level subtotals aggregating across country_iso for each combination of channel_desc and calendar_month_desc.
- Second-level subtotals aggregating across calendar_month_desc and country_iso for each channel_desc value.
- It does not produce a grand total row.

Query:

```
SQL> SELECT channel_desc, calendar_month_desc, country_iso_code,
  2         TO_CHAR(SUM(amount_sold), '9,999,999,999') AS SALES$
  3  FROM sales, customers, times, channels, countries
  4  WHERE sales.time_id = times.time_id
  5    AND sales.cust_id = customers.cust_id
  6    AND customers.country_id = countries.country_id
  7    AND sales.channel_id = channels.channel_id
  8    AND channels.channel_desc IN ('Direct Sales', 'Internet')
  9    AND times.calendar_month_desc IN ('2000-09', '2000-10')
 10    AND countries.country_iso_code IN ('US', 'GB')
 11  GROUP BY ROLLUP(channel_desc, calendar_month_desc, countries.country_iso_code);

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Internet             2000-09  GB          16,569
Internet             2000-09  US         124,224
Internet             2000-09             140,793
Internet             2000-10  GB          14,539
Internet             2000-10  US         137,054
Internet             2000-10             151,593
Internet                                 292,387
Direct Sales         2000-09  GB          85,223
Direct Sales         2000-09  US         638,201
Direct Sales         2000-09             723,424
Direct Sales         2000-10  GB          91,925

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Direct Sales         2000-10  US         682,297
Direct Sales         2000-10             774,222
Direct Sales                           1,497,646
                                       1,790,032

15 rows selected.
```

Q5. Find the total sales by country_id and channel_desc for the US and GB through the Internet and direct sales in September 2000 and October 2009 using PARTIAL CUBE aggregation on month and country code and GROUP BY on channel_desc.

Query:

```
SQL> SELECT channel_desc, calendar_month_desc, country_iso_code,
  2         TO_CHAR(SUM(amount_sold), '9,999,999,999') AS SALES$
  3  FROM sales, customers, times, channels, countries
  4  WHERE sales.time_id = times.time_id
  5    AND sales.cust_id = customers.cust_id
  6    AND customers.country_id = countries.country_id
  7    AND sales.channel_id = channels.channel_id
  8    AND channels.channel_desc IN ('Direct Sales', 'Internet')
  9    AND times.calendar_month_desc IN ('2000-09', '2000-10')
 10    AND countries.country_iso_code IN ('US', 'GB')
 11  GROUP BY channel_desc, ROLLUP(calendar_month_desc, countries.country_iso_code);

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Internet             2000-09  GB          16,569
Internet             2000-09  US         124,224
Internet             2000-09             140,793
Internet             2000-10  GB          14,539
Internet             2000-10  US         137,054
Internet             2000-10             151,593
Internet                                 292,387
Direct Sales         2000-09  GB          85,223
Direct Sales         2000-09  US         638,201
Direct Sales         2000-09             723,424
Direct Sales         2000-10  GB          91,925

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Direct Sales         2000-10  US         682,297
Direct Sales         2000-10             774,222
Direct Sales                           1,497,646

14 rows selected.
```

Q6. Use GROUPING to create a set of mask columns for the result set of Q1.

- Create grouping on channel_desc and name it as CH
- Create grouping calendar_month_desc and name it as MO
- Create grouping on country_iso_code and name it as CO

Query:

```
SQL> SELECT channel_desc, calendar_month_desc, country_iso_code,
  2         TO_CHAR(SUM(amount_sold), '9,999,999,999') AS SALES$,
  3         GROUPING(channel_desc) AS CH,
  4         GROUPING(calendar_month_desc) AS MO,
  5         GROUPING(country_iso_code) AS CO
  6  FROM sales, customers, times, channels, countries
  7  WHERE sales.time_id = times.time_id
  8    AND sales.cust_id = customers.cust_id
  9    AND customers.country_id = countries.country_id
 10    AND sales.channel_id = channels.channel_id
 11    AND channels.channel_desc IN ('Direct Sales', 'Internet')
 12    AND times.calendar_month_desc IN ('2000-09', '2000-10')
 13    AND countries.country_iso_code IN ('GB', 'US')
 14  GROUP BY ROLLUP(channel_desc, calendar_month_desc, countries.country_iso_code);

CHANNEL_DESC         CALENDAR CO SALES$                CH         MO         CO
-------------------- -------- -- -------------- ---------- ---------- ----------
Internet             2000-09  GB          16,569          0          0          0
Internet             2000-09  US         124,224          0          0          0
Internet             2000-09             140,793          0          0          1
Internet             2000-10  GB          14,539          0          0          0
Internet             2000-10  US         137,054          0          0          0
Internet             2000-10             151,593          0          0          1
Internet                                 292,387          0          1          1
Direct Sales         2000-09  GB          85,223          0          0          0
Direct Sales         2000-09  US         638,201          0          0          0
Direct Sales         2000-09             723,424          0          0          1
Direct Sales         2000-10  GB          91,925          0          0          0

CHANNEL_DESC         CALENDAR CO SALES$                CH         MO         CO
-------------------- -------- -- -------------- ---------- ---------- ----------
Direct Sales         2000-10  US         682,297          0          0          0
Direct Sales         2000-10             774,222          0          0          1
Direct Sales                           1,497,646          0          1          1
                                       1,790,032          1          1          1

15 rows selected.
```

Q7. Find the total sales by country_id and channel_desc for the US and GB through the Internet and direct sales in September 2000 and October 2009 using GROUPING SETS. Calculate aggregates over three groupings:
(channel_desc, calendar_month_desc, country_iso_code)

Query:

```
SQL> SELECT channel_desc, calendar_month_desc, country_iso_code,
  2         TO_CHAR(SUM(amount_sold), '9,999,999,999') AS SALES$
  3  FROM sales, customers, times, channels, countries
  4  WHERE sales.time_id = times.time_id
  5    AND sales.cust_id = customers.cust_id
  6    AND customers.country_id = countries.country_id
  7    AND sales.channel_id = channels.channel_id
  8    AND channels.channel_desc IN ('Direct Sales', 'Internet')
  9    AND times.calendar_month_desc IN ('2000-09', '2000-10')
 10    AND countries.country_iso_code IN ('US', 'GB')
 11  GROUP BY
 12    GROUPING SETS (
 13      (channel_desc, calendar_month_desc, countries.country_iso_code),
 14      (channel_desc, countries.country_iso_code),
 15      (calendar_month_desc, countries.country_iso_code)
 16    );

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Internet             2000-09  GB          16,569
Direct Sales         2000-09  GB          85,223
Internet             2000-09  US         124,224
Direct Sales         2000-09  US         638,201
Internet             2000-10  GB          14,539
Direct Sales         2000-10  GB          91,925
Internet             2000-10  US         137,054
Direct Sales         2000-10  US         682,297
                     2000-09  GB         101,792
                     2000-09  US         762,425
                     2000-10  GB         106,465

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
                     2000-10  US         819,351
Direct Sales                  GB         177,148
Internet                      GB          31,109
Direct Sales                  US       1,320,497
Internet                      US         261,278

16 rows selected.
```

Q8. Consider the following Query and make conclusion from the result obtained.

Query: (scott Schema)

SELECT deptno, job, SUM(sal)

FROM emp

GROUP BY CUBE(deptno, job)

Query:

```
SQL> SELECT deptno, job, SUM(sal)
  2  FROM emp
  3  GROUP BY CUBE(deptno, job);

    DEPTNO JOB          SUM(SAL)
---------- --------- ----------
                           29025
           CLERK            4150
           ANALYST          6000
           MANAGER          8275
           SALESMAN         5600
           PRESIDENT        5000
        10                  8750
        10 CLERK            1300
        10 MANAGER          2450
        10 PRESIDENT        5000
        20                 10875

    DEPTNO JOB          SUM(SAL)
---------- --------- ----------
        20 CLERK            1900
        20 ANALYST          6000
        20 MANAGER          2975
        30                  9400
        30 CLERK             950
        30 MANAGER          2850
        30 SALESMAN         5600

18 rows selected.
```

Q9. Calculate the salary for each department present in different cities of hr schema using rollup.

Query:

```
SQL> SELECT
  2      e.department_id,
  3      l.city,
  4      SUM(e.salary) AS total_salary
  5  FROM
  6      employees e
  7  JOIN
  8      departments d ON e.department_id = d.department_id
  9  JOIN
 10      locations l ON d.location_id = l.location_id
 11  GROUP BY
 12      ROLLUP(e.department_id, l.city);

DEPARTMENT_ID CITY                           TOTAL_SALARY
------------- ------------------------------ ------------
           10 Seattle                              4400
           10                                      4400
           20 Toronto                             19000
           20                                     19000
           30 Seattle                             24900
           30                                     24900
           40 London                               6500
           40                                      6500
           50 South San Francisco                156400
           50                                    156400
           60 Southlake                           28800

DEPARTMENT_ID CITY                           TOTAL_SALARY
------------- ------------------------------ ------------
           60                                      28800
           70 Munich                              10000
           70                                     10000
           80 Oxford                             304500
           80                                    304500
           90 Seattle                             58000
           90                                     58000
          100 Seattle                             51608
          100                                     51608
          110 Seattle                             20308
          110                                     20308

DEPARTMENT_ID CITY                           TOTAL_SALARY
------------- ------------------------------ ------------
                                                684416

23 rows selected.
```

Q10. Calculate the salary for each department present in different cities of hr schema using cube.

Query:

```
SQL> SELECT
  2      e.department_id,
  3      l.city,
  4      SUM(e.salary) AS total_salary
  5  FROM
  6      employees e
  7  JOIN
  8      departments d ON e.department_id = d.department_id
  9  JOIN
 10      locations l ON d.location_id = l.location_id
 11  GROUP BY
 12      CUBE(e.department_id, l.city);

DEPARTMENT_ID CITY                           TOTAL_SALARY
------------- ------------------------------ ------------
                                                   684416
              London                                 6500
              Munich                                10000
              Oxford                               304500
              Seattle                              159216
              Toronto                               19000
              Southlake                             28800
              South San Francisco                  156400
           10                                        4400
           10 Seattle                                4400
           20                                       19000

DEPARTMENT_ID CITY                           TOTAL_SALARY
------------- ------------------------------ ------------
           20 Toronto                               19000
           30                                       24900
           30 Seattle                               24900
           40                                        6500
           40 London                                 6500
           50                                      156400
           50 South San Francisco                  156400
           60                                       28800
           60 Southlake                             28800
           70                                       10000
           70 Munich                                10000

DEPARTMENT_ID CITY                           TOTAL_SALARY
------------- ------------------------------ ------------
           80                                      304500
           80 Oxford                               304500
           90                                       58000
           90 Seattle                               58000
          100                                       51608
          100 Seattle                               51608
          110                                       20308
          110 Seattle                               20308

30 rows selected.
```