

Hands-on Lab: Querying the Data Warehouse (Cubes, Rollups, Grouping Sets and Materialized Views)

Estimated time needed: **30** minutes

Objectives

In this lab you will learn how to create:

- Grouping sets
- Rollup
- Cube
- Materialized Query Tables (MQT)

Exercise 1 - Login to your Cloud IBM DB2

This lab requires that you complete the previous lab [Populate a Data Warehouse](#).

If you have not finished the Populate a Data Warehouse Lab yet, please finish it before you continue.

GROUPING SETS, CUBE, and ROLLUP allow us to easily create subtotals and grand totals in a variety of ways. All these operators are used along with the GROUP BY operator.

GROUPING SETS operator allows us to group data in a number of different ways in a single SELECT statement.

The ROLLUP operator is used to create subtotals and grand totals for a set of columns. The summarized totals are created based on the columns passed to the ROLLUP operator.

The CUBE operator produces subtotals and grand totals. In addition it produces subtotals and grand totals for every permutation of the columns provided to the CUBE operator.

Exercise 2 - Write a query using grouping sets

After you login to the cloud instance of IBM DB2, go to the sql tab and run the query below.

To create a grouping set for three columns labeled year, category, and sum of billedamount, run the sql statement below.

```

1  select year,category, sum(billedamount) as totalbilledamount
2  from factbilling
3  left join dimcustomer
4  on factbilling.customerid = dimcustomer.customerid
5  left join dimmonth
6  on factbilling.monthid=dimmonth.monthid
7  group by grouping sets(year,category)
8  order by year, category

```

The output of the above command will contain 13 rows. The partial output can be seen in the image below.

To see the full output click on the [open in the new tab](#) icon.

The screenshot shows a SQL IDE interface. On the left, a script editor contains the following SQL query:

```

1 select year,category, sum(billedamount) as totalbilledamount
2 from factbilling
3 left join dimcustomer
4 on factbilling.customerid = dimcustomer.customerid
5 left join dimmonth
6 on factbilling.monthid=dimmonth.monthid
7 group by grouping sets(year,category)
8 order by year, category

```

On the right, the 'Result' pane shows the output of the query. The title bar indicates 'Result - Oct 11, 2021 4:22:19 PM'. The query text is 'select year,category, sum(billedamou...' and the run time is '0.240 s'. The results are displayed in a table with the following columns: YEAR, CATEGORY, and TOTALBILLEDAMOUNT. The table shows data for the years 2009 through 2013.

YEAR	CATEGORY	TOTALBILLEDAMOUNT
2009		120263327
2010		119484658
2011		119427469
2012		120761543
2013		120859328

At the bottom of the result pane, a note states: 'Result set is truncated, only the first 13 rows have been loaded. Select "View all loaded data" on the right top of the result to view all loaded rows.' A 'More' link is provided.

Exercise 3 - Write a query using rollup

To create a rollup using the three columns year, category and sum of billedamount, run the sql statement below.

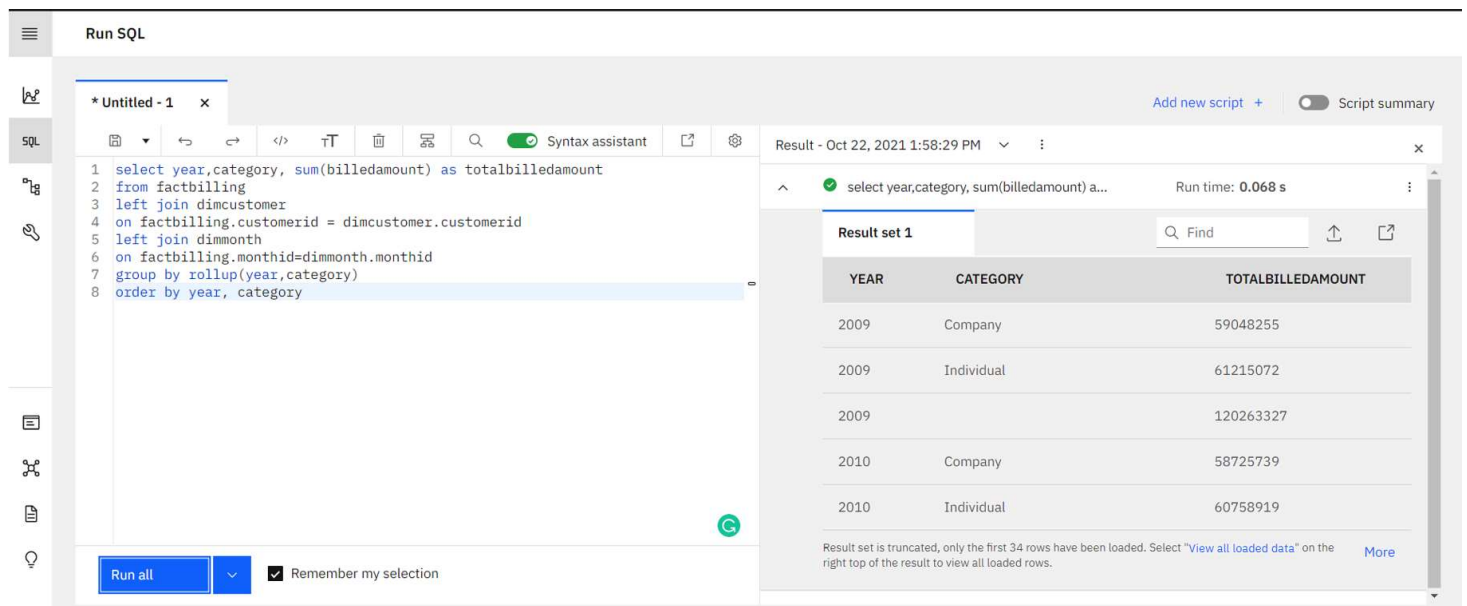
```

1  select year,category, sum(billedamount) as totalbilledamount
2  from factbilling
3  left join dimcustomer
4  on factbilling.customerid = dimcustomer.customerid
5  left join dimmonth
6  on factbilling.monthid=dimmonth.monthid
7  group by rollup(year,category)
8  order by year, category

```

The output of the above command will contain 408 rows. The partial output can be seen in the image below.

To see the full output click on the [open in the new tab](#) icon.



The screenshot shows a SQL IDE interface. On the left, a query editor titled '*Untitled - 1' contains the following SQL code:

```
1 select year,category, sum(billedamount) as totalbilledamount
2 from factbilling
3 left join dimcustomer
4 on factbilling.customerid = dimcustomer.customerid
5 left join dimmonth
6 on factbilling.monthid=dimmonth.monthid
7 group by rollup(year,category)
8 order by year, category
```

On the right, the 'Result' pane shows the output of the query. The title bar indicates 'Result - Oct 22, 2021 1:58:29 PM' and 'Run time: 0.068 s'. The results are displayed in a table with the following columns: YEAR, CATEGORY, and TOTALBILLEDAMOUNT. The table shows data for the years 2009 and 2010, categorized by Company and Individual. The results are truncated, showing only the first 34 rows.

YEAR	CATEGORY	TOTALBILLEDAMOUNT
2009	Company	59048255
2009	Individual	61215072
2009		120263327
2010	Company	58725739
2010	Individual	60758919

Exercise 4 - Write a query using cube

To create a cube using the three columns labeled year, category, and sum of billedamount, run the sql statement below.

```
1 select year,category, sum(billedamount) as totalbilledamount
2 from factbilling
3 left join dimcustomer
4 on factbilling.customerid = dimcustomer.customerid
5 left join dimmonth
6 on factbilling.monthid=dimmonth.monthid
7 group by cube(year,category)
8 order by year, category
```

The output of the above command will contain 468 rows. The partial output can be seen in the image below.

To see the full output click on the [open in the new tab](#) icon.

The screenshot shows a SQL IDE interface. On the left, a query editor window titled '*Untitled - 1' contains the following SQL code:

```

1 select year,category, sum(billedamount) as totalbilledamount
2 from factbilling
3 left join dimcustomer
4 on factbilling.customerid = dimcustomer.customerid
5 left join dimmonth
6 on factbilling.monthid=dimmonth.monthid
7 group by cube(year,category)
8 order by year, category

```

At the bottom of the editor, there is a 'Run all' button and a checkbox labeled 'Remember my selection' which is checked. On the right, a results pane titled 'Result - Oct 22, 2021 1:59:44 PM' shows the output of the query. It includes a search bar, a 'Find' button, and a table with the following data:

YEAR	CATEGORY	TOTALBILLEDAMOUNT
2009	Company	59048255
2009	Individual	61215072
2009		120263327
2010	Company	58725739
2010	Individual	60758919

Below the table, a message states: 'Result set is truncated, only the first 36 rows have been loaded. Select "View all loaded data" on the right top of the result to view all loaded rows.' A 'More' link is provided next to this message.

Exercise 5 - Create a Materialized Query Table(MQT)

In DB2 we can implement materialized views using Materialized Query Tables.

Step 1: Create the MQT.

Execute the sql statement below to create an MQT named countrystats.

```

1 CREATE TABLE countrystats (country, year, totalbilledamount) AS
2   (select country, year, sum(billedamount)
3    from factbilling
4    left join dimcustomer
5    on factbilling.customerid = dimcustomer.customerid
6    left join dimmonth
7    on factbilling.monthid=dimmonth.monthid
8    group by country,year)
9     DATA INITIALLY DEFERRED
10    REFRESH DEFERRED
11    MAINTAINED BY SYSTEM;

```

You may get a warning in the output as below.

The materialized query table may not be used to optimize the processing of queries.

You can safely ignore the warning and proceed to the next step.

The above command creates an MQT named `countrystats` that has 3 columns.

- country
- year
- totalbilledamount

The MQT is essentially the result of the below query, which gives you the country, year and the sum of billed amount grouped by country and year.

```
1  select country, year, sum(billedamount)
2  from factbilling
3  left join dimcustomer
4  on factbilling.customerid = dimcustomer.customerid
5  left join dimmonth
6  on factbilling.monthid=dimmonth.monthid
7  group by country,year
```

The settings

- DATA INITIALLY DEFERRED
- REFRESH DEFERRED
- MAINTAINED BY SYSTEM

Simple mean that data is not initially populated into this MQT. Whenever the underlying data changes, the MQT does NOT automatically refresh. The MQT is system maintained and not user maintained.

Step 2: Populate/refresh data into the MQT.

Execute the sql statement below to populate the MQT countrystats

```
1  refresh table countrystats;
```

The command above populates the MQT with relevant data.

Step 3: Query the MQT.

Once an MQT is refreshed, you can query it.

Execute the sql statement below to query the MQT countrystats.

```
1  select * from countrystats
```

Practice exercises

1. Problem:

Create a grouping set for the columns year, quartername, sum(billedamount).

▼ Click here for Hint

Select columns year, quartername, sum(billedamount), and use a group by query and join the dimcustomer and dimmonth tables to factbilling table.

▼ Click here for Solution

```

1  select year, quartername, sum(billedamount) as totalbilledamount
2  from factbilling
3  left join dimcustomer
4  on factbilling.customerid = dimcustomer.customerid
5  left join dimmonth
6  on factbilling.monthid=dimmonth.monthid
7  group by grouping sets(year, quartername)

```

2. Problem:

Create a rollup for the columns country, category, sum(billedamount).

▼ Click here for Hint

Select columns country, category, sum(billedamount), and use a group by query and join the dimcustomer and dimmonth tables to factbilling table.

▼ Click here for Solution

```

1  select country, category, sum(billedamount) as totalbilledamount
2  from factbilling
3  left join dimcustomer
4  on factbilling.customerid = dimcustomer.customerid
5  left join dimmonth
6  on factbilling.monthid=dimmonth.monthid
7  group by rollup(country,category)

```

3. Problem:

Create a cube for the columns year, country, category, sum(billedamount).

▼ Click here for Hint

Select columns year, country, category, sum(billedamount), and use a group by query and join the dimcustomer and dimmonth tables to factbilling table.

► Click here for Solution

4. Problem:

Create an MQT named average_billamount with columns year, quarter, category, country, average_bill_amount.

You can safely ignore the warning and proceed

▼ Click here for Hint

Select columns year, quarter, category, country, avg(billedamount), and use a group by query and join the dimcustomer and dimmonth tables to factbilling table.

▼ [Click here for Solution](#)

```
1 CREATE TABLE average_billamount (year,quarter,category,country, average_bill_amount
2 (select year,quarter,category,country, avg(billedamount) as average_bill_amou
3 from factbilling
4 left join dimcustomer
5 on factbilling.customerid = dimcustomer.customerid
6 left join dimmonth
7 on factbilling.monthid=dimmonth.monthid
8 group by year,quarter,category,country
9 )
10 DATA INITIALLY DEFERRED
11 REFRESH DEFERRED
12 MAINTAINED BY SYSTEM;
```

```
1 refresh table average_billamount;
```

Congratulations! You have successfully finished this lab.

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-09-28	0.1	Ramesh Sannareddy	Created initial version of the lab

IBM Corporation 2021. All rights reserved