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

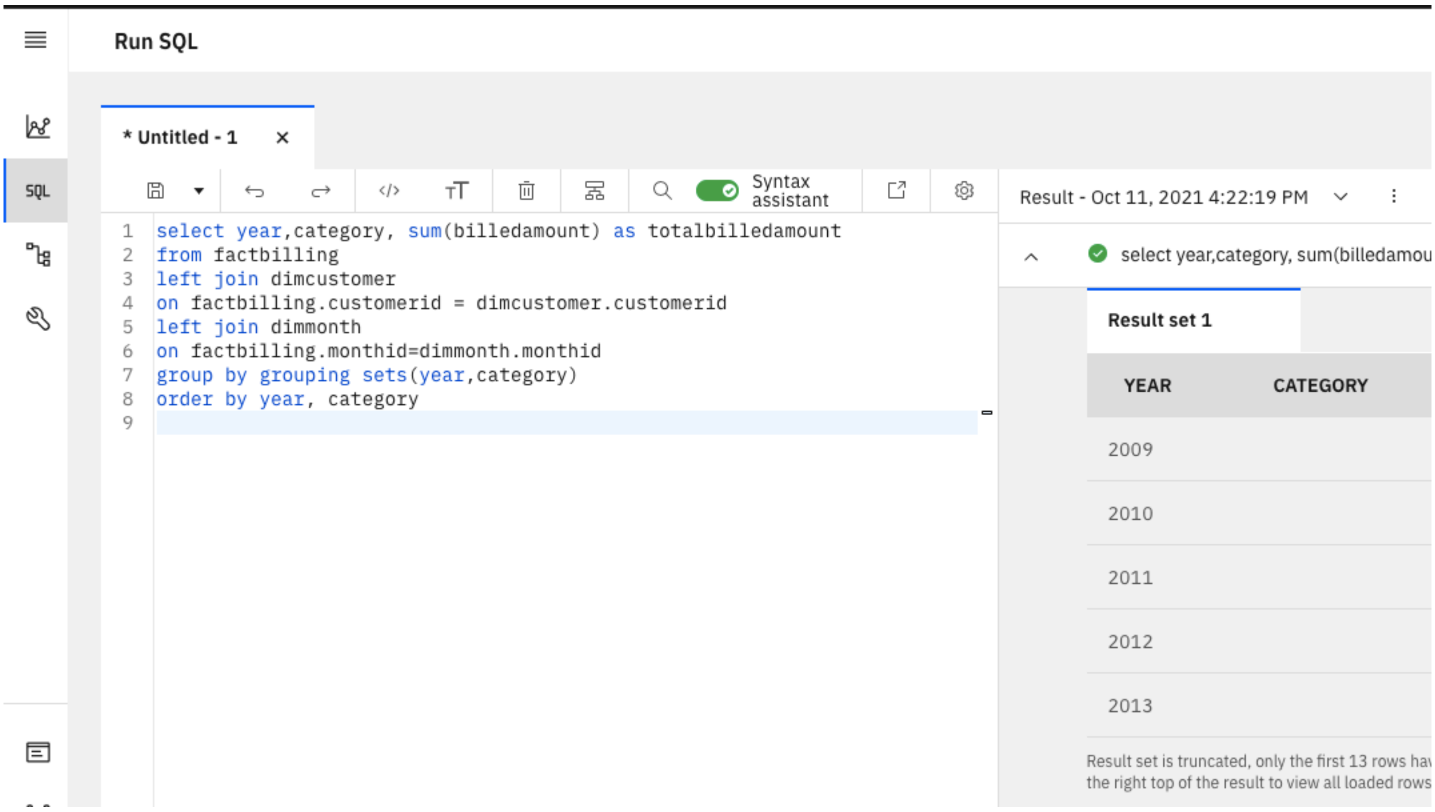
After you login to the cloud instance of IBM DB2, go to the sql tab and run the query below.

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

```
select year,category, sum(billedamount) as totalbilledamount
from factbilling
left join dimcustomer
on factbilling.customerid = dimcustomer.customerid
left join dimmonth
on factbilling.monthid=dimmonth.monthid
group by grouping sets(year,category)
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.


Run SQL
* Untitled - 1
Syntax assistant
Result - Oct 11, 2021 4:22:19 PM
select year,category, sum(billedamou
Result set 1

```
select year,category, sum(billedamount) as totalbilledamount
from factbilling
left join dimcustomer
on factbilling.customerid = dimcustomer.customerid
left join dimmonth
on factbilling.monthid=dimmonth.monthid
group by grouping sets(year,category)
order by year, category
```

YEAR	CATEGORY
2009	
2010	
2011	
2012	
2013	

Result set is truncated, only the first 13 rows hav
the right top of the result to view all loaded rows

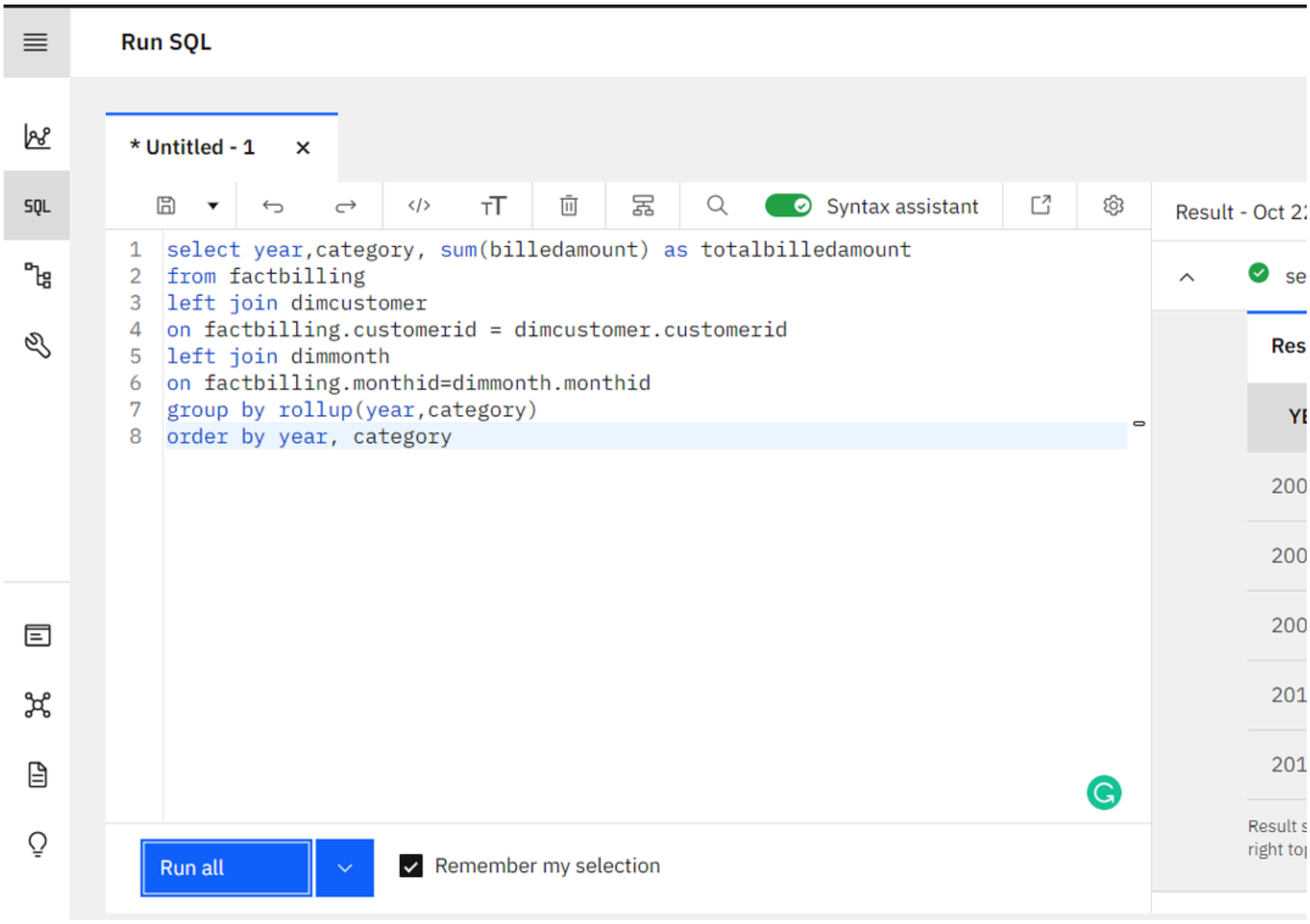
Exercise 3 - Write a query using rollup

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

```
select year,category, sum(billedamount) as totalbilledamount
from factbilling
left join dimcustomer
on factbilling.customerid = dimcustomer.customerid
left join dimmonth
on factbilling.monthid=dimmonth.monthid
group by rollup(year,category)
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.



```
select year,category, sum(billedamount) as totalbilledamount
from factbilling
left join dimcustomer
on factbilling.customerid = dimcustomer.customerid
left join dimmonth
on factbilling.monthid=dimmonth.monthid
group by rollup(year,category)
order by year, category
```

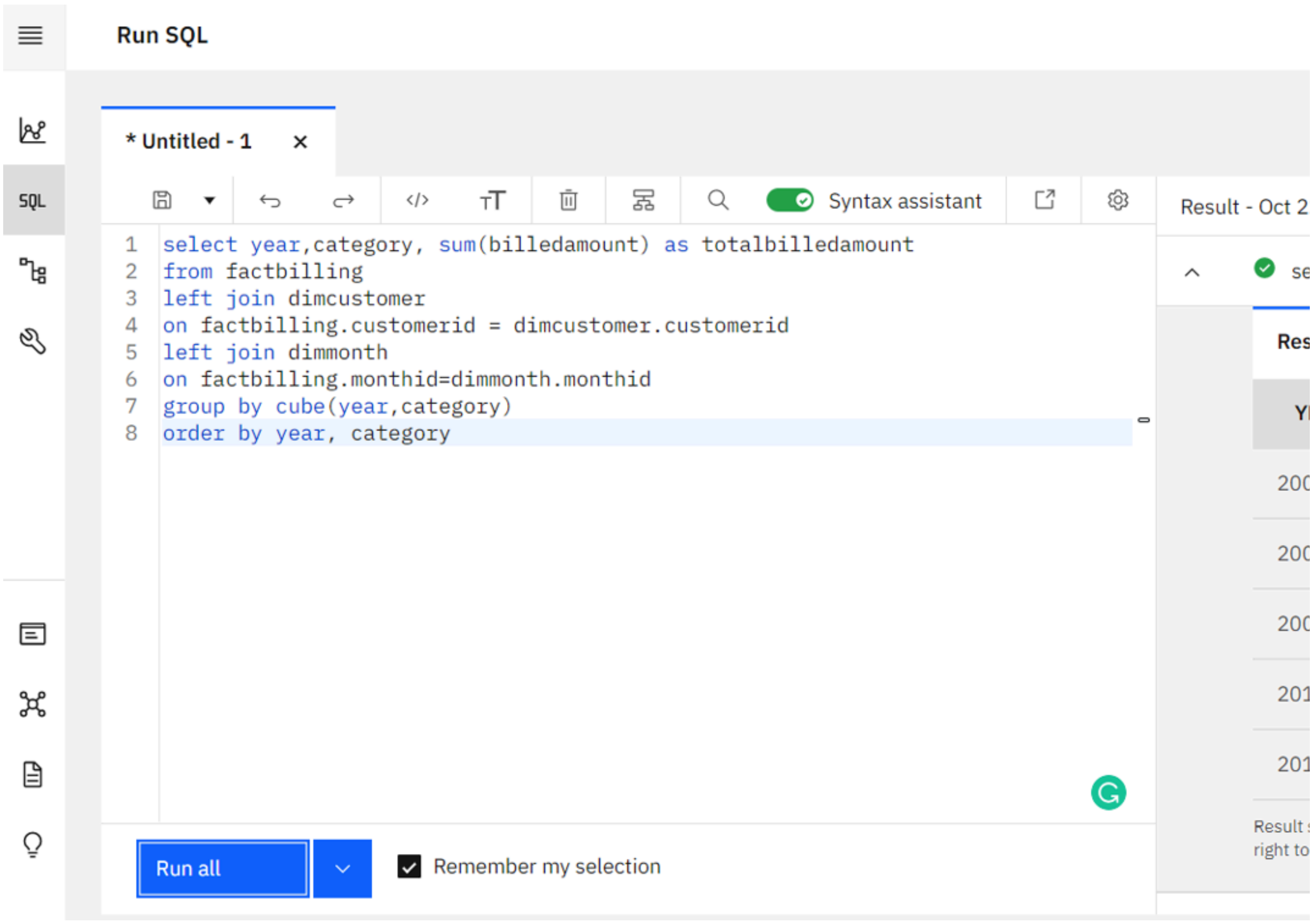
Exercise 4 - Write a query using cube

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

```
select year,category, sum(billedamount) as totalbilledamount
from factbilling
left join dimcustomer
on factbilling.customerid = dimcustomer.customerid
left join dimmonth
on factbilling.monthid=dimmonth.monthid
group by cube(year,category)
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.



```
select year,category, sum(billedamount) as totalbilledamount
from factbilling
left join dimcustomer
on factbilling.customerid = dimcustomer.customerid
left join dimmonth
on factbilling.monthid=dimmonth.monthid
group by cube(year,category)
order by year, category
```

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.

```
CREATE TABLE countrystats (country, year, totalbilledamount) AS
  (select country, year, sum(billedamount)
from factbilling
left join dimcustomer
on factbilling.customerid = dimcustomer.customerid
left join dimmonth
on factbilling.monthid=dimmonth.monthid
group by country,year)
     DATA INITIALLY DEFERRED
     REFRESH DEFERRED
     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.

```
select country, year, sum(billedamount)
from factbilling
left join dimcustomer
on factbilling.customerid = dimcustomer.customerid
left join dimmonth
```

```
on factbilling.monthid=dimmonth.monthid
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

```
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.

```
select * from countrystats
```

Practice exercises

1. Problem:

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

▶ Click here for Hint
▶ Click here for Solution

2. Problem:

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

▶ Click here for Hint
▶ Click here for Solution

3. Problem:

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

▶ Click here for Hint
▶ 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
▶ Click here for Solution

Congratulations! You have successfully finished this lab.

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja