Querying the Data Warehouse (Cubes, Rollups, Grouping Sets and Materialized Views)

Exercise 1 - Launch a PostgreSQL server instance on Cloud IDE and open up the pgAdmin Graphical User Interface.

OPERATORS

- 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

```
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);
```

	year integer	category character varying (10)	totalbilledamount bigint
1	2015	[null]	119808719
2	2011	[null]	119427469
3	2014	[null]	119239283
4	2010	[null]	119484658
5	2017	[null]	119526654
6	2019	[null]	120820495
7	2016	[null]	120433289
8	2012	[null]	120761543
9	2018	[null]	119595980
10	2009	[null]	120263327
11	2013	[null]	120859328
12	[null]	Company	647445529
13	[null]	Individual	672775216

- Sets here just appears to give each individual column included in the query as it's own row
    - No combined aggregate output but simply each individual set separated (hence the null values for the year/category values for various rows above

- Maybe a LEFT JOIN is faster but inner join works just the same in terms of output

```
select year,category, sum(billedamount) as totalbilledamount
from "FactBilling"
inner join "DimCustomer"
on "FactBilling".customerid = "DimCustomer".customerid
inner join "DimMonth"
on "FactBilling".monthid="DimMonth".monthid
group by grouping sets(year,category);
```

Exercise 3 - Write a query using rollup

- Recall rollup will simply take the first column passed as a single attribute aggregate to return and dismiss the other column for single aggregate outputs

```
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;
```

- Snippet of Output

Data Output | Explain | Messages | Notifications

	year integer	category character varying (10)	totalbilledamount bigint
1	2009	Company	59048255
2	2009	Individual	61215072
3	2009	[null]	120263327
4	2010	Company	58725739
5	2010	Individual	60758919
6	2010	[null]	119484658
7	2011	Company	58559675
8	2011	Individual	60867794
9	2011	[null]	119427469
10	2012	Company	58981336
11	2012	Individual	61780207
12	2012	[null]	120761543
13	2013	Company	59450681
14	2013	Individual	61408647
15	2013	[null]	120859328
16	2014	Company	58375062

Exercise 4 - Write a query using cube

- Unlike rollup, we should have the combinations of the two fields and the respective individual summaries of the agg function for each field (column)
- Also the total is listed for the total sum w/o any grouping attribute

```
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;
```

Data Output | Explain | Messages | Notifications

	year integer	category character varying (10)	totalbilledamount bigint
1	2009	Company	59048255
2	2009	Individual	61215072
3	2009	[null]	120263327
4	2010	Company	58725739
5	2010	Individual	60758919
6	2010	[null]	119484658
7	2011	Company	58559675
8	2011	Individual	60867794
9	2011	[null]	119427469
10	2012	Company	58981336
11	2012	Individual	61780207
12	2012	[null]	120761543
13	2013	Company	59450681
14	2013	Individual	61408647
15	2013	[null]	120859328
16	2014	Company	58375062
17	2014	Individual	60864221
18	2014	[null]	119239283
19	2015	Company	58860727
20	2015	Individual	60947992
21	2015	[null]	119808719
22	2016	Company	58969379
23	2016	Individual	61463910

24	2016	[null]	120433289
25	2017	Company	58408457
26	2017	Individual	61118197
27	2017	[null]	119526654
28	2018	Company	58629460
29	2018	Individual	60966520
30	2018	[null]	119595980
31	2019	Company	59436758
32	2019	Individual	61383737
33	2019	[null]	120820495
34	[null]	Company	647445529
35	[null]	Individual	672775216
36	[null]	[null]	1320220745

Exercise 5 - Create a Materialized Query Table(MQT)

Step 1 Create

```
CREATE MATERIALIZED VIEW 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);
```

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

```
select year, quartername, 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, quartername);
```

Step 2: Populate/refresh data into the MQT.

Execute the sql statement below to populate the MQT countrystats.

```
REFRESH MATERIALIZED VIEW countrystats;
```

Step 3: Query the MQT.

```
select * from countrystats;
```

Data Output **Explain** Messages Notifications

	country character varying (40)	year integer	totalbilledamount bigint
1	Ukraine	2017	2200907
2	Antigua and Barbuda	2015	129228
3	Italy	2014	109511
4	Slovakia	2018	99512
5	Yemen	2012	706149
6	Austria	2009	128874
7	Venezuela	2019	370881
8	Uganda	2015	334817
9	Uzbekistan	2014	329642
10	Armenia	2009	588357
11	Portugal	2015	4555151
12	Costa Rica	2013	474298

Practice / Challenge Questions

1. Create a Grouping Set for the columns year, quartername, sum(billedamount)

```
-- Problem 1: Create a grouping set for the columns year, quartername,
sum(billedamount).
SELECT
      dm.year, dm.quartername, SUM(fb.billedamount)
FROM "FactBilling" AS fb
LEFT JOIN "DimMonth" AS dm
USING(monthid)
GROUP BY GROUPING SETS(dm.year, dm.quartername)
ORDER BY dm.year, dm.quartername;
```

Data Output Explain Messages Notifications

	year integer	quartername character varying (2)	sum bigint
1	2009	[null]	120263327
2	2010	[null]	119484658
3	2011	[null]	119427469
4	2012	[null]	120761543
5	2013	[null]	120859328
6	2014	[null]	119239283
7	2015	[null]	119808719
8	2016	[null]	120433289
9	2017	[null]	119526654
10	2018	[null]	119595980
11	2019	[null]	120820495
12	[null]	Q1	329926465
13	[null]	Q2	330321504
14	[null]	Q3	330991865
15	[null]	Q4	328980911

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

```sql
--  Create a rollup for the columns country, category, sum(billedamount).
SELECT
      dc.country, dc.category, SUM(fb.billedamount)
FROM "FactBilling" AS fb
LEFT JOIN "DimCustomer" AS dc
USING(customerid)
GROUP BY ROLLUP(dc.country, dc.category)
ORDER BY dc.country, dc.category;
```

Data Output Explain Messages Notifications

	country character varying (40)	category character varying (10)	sum bigint
1	Afghanistan	Individual	1322316
2	Afghanistan	[null]	1322316
3	Albania	Company	1146134
4	Albania	Individual	6364005
5	Albania	[null]	7510139
6	American Samoa	Company	1222436
7	American Samoa	Individual	1252762
8	American Samoa	[null]	2475198
9	Angola	Individual	1358358
10	Angola	[null]	1358358
11	Antigua and Barbuda	Individual	1337573
12	Antigua and Barbuda	[null]	1337573
13	Argentina	Company	5348972
14	Argentina	Individual	15984403
15	Argentina	[null]	21333375

328	Yemen	Company	2706099
329	Yemen	Individual	4003162
330	Yemen	[null]	6709261
331	[null]	[null]	1320220745

- Note here for the **ROLLUP** that the **country** has it's own individual sum file for the total regardless of the **category**

    - The second column in rollup is not aggregated on its' own unlike the country from the output which we can see is after each category listing
- Last point, at the end of the output (331th row) we can see that the total billed amount is included
    - This is the same as just the total sum of the field

```
SELECT SUM(billedamount) FROM "FactBilling";
```

Sum - bigint
1320220745

- Check for count of the customerID and billedamount having same total row (aka the join operation performed above would always have a customerID for any billed amount which can track to a country & category (which it does!)

```
SELECT COUNT(customerid) FROM "FactBilling"
UNION ALL
SELECT COUNT(billedamount) FROM "FactBilling"
```

count - bigint
132000
132000

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

```
--  Problem 3: Create a cube for the columns year,country, category,
sum(billedamount).
SELECT
      dm.year, dc.country, dc.category, SUM(fb.billedamount)
FROM "FactBilling" AS fb
LEFT JOIN "DimMonth" AS dm
      USING(monthid)
LEFT JOIN "DimCustomer" AS dc
```

```
      USING(customerid)
GROUP BY CUBE(dm.year, dc.country, dc.category)
ORDER BY dm.year, dc.country, dc.category;
```

	year integer	country character varying (40)	category character varying (10)	sum bigint
1	2009	Afghanistan	Individual	121329
2	2009	Afghanistan	[null]	121329
3	2009	Albania	Company	131691
4	2009	Albania	Individual	511600
5	2009	Albania	[null]	643291
6	2009	American Samoa	Company	118308
7	2009	American Samoa	Individual	116739
8	2009	American Samoa	[null]	235047
9	2009	Angola	Individual	130261
10	2009	Angola	[null]	130261
11	2009	Antigua and Barbuda	Individual	124201
12	2009	Antigua and Barbuda	[null]	124201
13	2009	Argentina	Company	574400
14	2009	Argentina	Individual	1443020
15	2009	Argentina	[null]	2017420

- Similar here to ROLLUP to start however at the end of the query for a year, we'll see the overall totals for the **Category & Year** by itself
    - Each country's individual total is output with the other aggregates with the ORDER BY declaration
    - See the snippet below for the changing of year 2018 to 2019 and how the aggregate outputs are returned to the data table (query output)

3325	2018	Yemen	Company	256090
3326	2018	Yemen	Individual	330350
3327	2018	Yemen	[null]	586440
3328	2018	[null]	Company	58629460
3329	2018	[null]	Individual	60966520
3330	2018	[null]	[null]	119595980
3331	2019	Afghanistan	Individual	155384
3332	2019	Afghanistan	[null]	155384
3333	2019	Albania	Company	84816
3334	2019	Albania	Individual	587178
3335	2019	Albania	[null]	671994