L1 E2 - 3 - Grouping Sets

February 6, 2023

1 Exercise 02 - OLAP Cubes - Grouping Sets

Start by connecting to the database by running the cells below. If you are coming back to this exercise, then uncomment and run the first cell to recreate the database. If you recently completed the slicing and dicing exercise, then skip to the second cell.

```
In [1]:  !PGPASSWORD=student createdb -h 127.0.0.1 -U student pagila_star
         !PGPASSWORD=student psql -q -h 127.0.0.1 -U student -d pagila_star -f Data/pagila-star.
```

```
 set_config
------------

(1 row)

 setval
--------
    200
(1 row)

 setval
--------
    605
(1 row)

 setval
--------
     16
(1 row)

 setval
--------
    600
(1 row)

 setval
--------
    109
```

```
(1 row)

 setval
--------
    599
(1 row)

 setval
--------
      1
(1 row)

 setval
--------
      1
(1 row)

 setval
--------
      1
(1 row)

 setval
--------
      1
(1 row)

 setval
--------
  16049
(1 row)

 setval
--------
   1000
(1 row)

 setval
--------
   4581
(1 row)

 setval
--------
      6
(1 row)

 setval
```

```
--------
  32098
(1 row)

 setval
--------
  16049
(1 row)

 setval
--------
      2
(1 row)

 setval
--------
      2
(1 row)
```

1.0.1 Connect to the local database where Pagila is loaded

```
In [2]: import sql
        %load_ext sql

        DB_ENDPOINT = "127.0.0.1"
        DB = 'pagila_star'
        DB_USER = 'student'
        DB_PASSWORD = 'student'
        DB_PORT = '5432'

        # postgresql://username:password@host:port/database
        conn_string = "postgresql://{}:{}@{}:{}/{}" \
                                .format(DB_USER, DB_PASSWORD, DB_ENDPOINT, DB_PORT, DB)

        print(conn_string)
        %sql $conn_string
```

```
postgresql://student:student@127.0.0.1:5432/pagila_star
```

```
Out[2]: 'Connected: student@pagila_star'
```

1.0.2 Star Schema

2 Grouping Sets

- It happens often that for 3 dimensions, you want to aggregate a fact:

  - by nothing (total)
  - then by the 1st dimension
  - then by the 2nd
  - then by the 3rd
  - then by the 1st and 2nd
  - then by the 2nd and 3rd
  - then by the 1st and 3rd
  - then by the 1st and 2nd and 3rd
- Since this is very common, and in all cases, we are iterating through all the fact table anyhow, there is a more clever way to do that using the SQL grouping statement "GROUPING SETS"

2.1 Total Revenue

TODO: Write a query that calculates total revenue (sales_amount)

```
In [4]: %%sql
        SELECT sum(sales_amount) revenue
        FROM factsales;
```

```
 * postgresql://student:***@127.0.0.1:5432/pagila_star
1 rows affected.
```

```
Out[4]: [(Decimal('67416.51'),)]
```

2.2 Revenue by Country

TODO: Write a query that calculates total revenue (sales_amount) by country

```
In [9]: %%sql
        SELECT dimstore.country, sum(factsales.sales_amount) as total_revenue
        FROM factsales
        JOIN dimstore ON (factsales.store_key = dimstore.store_key)
        GROUP By dimstore.country;
```

```
 * postgresql://student:***@127.0.0.1:5432/pagila_star
2 rows affected.
```

```
Out[9]: [('Australia', Decimal('33726.77')), ('Canada', Decimal('33689.74'))]
```

2.3 Revenue by Month

TODO: Write a query that calculates total revenue (sales_amount) by month

```
In [13]: %%sql
         SELECT dimdate.month, sum(factsales.sales_amount) as revenue
         FROM factsales
         JOIN dimdate ON (dimdate.date_key=factsales.date_key)
         GROUP BY month
         ORDER BY month asc;
```

```
 * postgresql://student:***@127.0.0.1:5432/pagila_star
5 rows affected.
```

```
Out[13]: [(1, Decimal('4824.43')),
          (2, Decimal('9631.88')),
          (3, Decimal('23886.56')),
          (4, Decimal('28559.46')),
          (5, Decimal('514.18'))]
```

2.4 Revenue by Month & Country

TODO: Write a query that calculates total revenue (sales_amount) by month and country. Sort the data by month, country, and revenue in descending order. The first few rows of your output should match the table below.

```
In [21]: %%sql
         SELECT dimdate.month, dimstore.country, sum(factsales.sales_amount) as revenue
         FROM factsales
         JOIN dimdate ON (dimdate.date_key=factsales.date_key)
         JOIN dimstore ON (dimstore.store_key=factsales.store_key)
         GROUP BY (month, country)
         ORDER BY (month, country) ASC
```

```
 * postgresql://student:***@127.0.0.1:5432/pagila_star
10 rows affected.
```

```
Out[21]: [(1, 'Australia', Decimal('2364.19')),
          (1, 'Canada', Decimal('2460.24')),
          (2, 'Australia', Decimal('4895.10')),
          (2, 'Canada', Decimal('4736.78')),
          (3, 'Australia', Decimal('12060.33')),
          (3, 'Canada', Decimal('11826.23')),
          (4, 'Australia', Decimal('14136.07')),
          (4, 'Canada', Decimal('14423.39')),
          (5, 'Australia', Decimal('271.08')),
          (5, 'Canada', Decimal('243.10'))]
```

```
<tbody><tr>
    <th>month</th>
    <th>country</th>
    <th>revenue</th>
</tr>
<tr>
    <td>1</td>
    <td>Australia</td>
    <td>2364.19</td>
</tr>
```

```
<tr>
    <td>1</td>
    <td>Canada</td>
    <td>2460.24</td>
</tr>
<tr>
    <td>2</td>
    <td>Australia</td>
    <td>4895.10</td>
</tr>
<tr>
    <td>2</td>
    <td>Canada</td>
    <td>4736.78</td>
</tr>
<tr>
    <td>3</td>
    <td>Australia</td>
    <td>12060.33</td>
</tr>
```

2.5 Revenue Total, by Month, by Country, by Month & Country All in one shot

TODO: Write a query that calculates total revenue at the various grouping levels done above (total, by month, by country, by month & country) all at once using the grouping sets function. Your output should match the table below.

```
In [23]: %%sql
         SELECT dimdate.month, dimstore.country, sum(factsales.sales_amount) as revenue
         FROM factsales
         JOIN dimdate ON (dimdate.date_key=factsales.date_key)
         JOIN dimstore ON (dimstore.store_key=factsales.store_key)
         GROUP BY grouping sets ((), (dimdate.month), (dimstore.country),  (dimdate.month, dimst

 * postgresql://student:***@127.0.0.1:5432/pagila_star
18 rows affected.


Out[23]: [(1, 'Australia', Decimal('2364.19')),
          (1, 'Canada', Decimal('2460.24')),
          (1, None, Decimal('4824.43')),
          (2, 'Australia', Decimal('4895.10')),
          (2, 'Canada', Decimal('4736.78')),
          (2, None, Decimal('9631.88')),
          (3, 'Australia', Decimal('12060.33')),
          (3, 'Canada', Decimal('11826.23')),
          (3, None, Decimal('23886.56')),
          (4, 'Australia', Decimal('14136.07')),
          (4, 'Canada', Decimal('14423.39')),
```

```
          (4, None, Decimal('28559.46')),
          (5, 'Australia', Decimal('271.08')),
          (5, 'Canada', Decimal('243.10')),
          (5, None, Decimal('514.18')),
          (None, None, Decimal('67416.51')),
          (None, 'Australia', Decimal('33726.77')),
          (None, 'Canada', Decimal('33689.74'))]

<tbody><tr>
    <th>month</th>
    <th>country</th>
    <th>revenue</th>
</tr>
<tr>
    <td>1</td>
    <td>Australia</td>
    <td>2364.19</td>
</tr>
<tr>
    <td>1</td>
    <td>Canada</td>
    <td>2460.24</td>
</tr>
<tr>
    <td>1</td>
    <td>None</td>
    <td>4824.43</td>
</tr>
<tr>
    <td>2</td>
    <td>Australia</td>
    <td>4895.10</td>
</tr>
<tr>
    <td>2</td>
    <td>Canada</td>
    <td>4736.78</td>
</tr>
<tr>
    <td>2</td>
    <td>None</td>
    <td>9631.88</td>
</tr>
<tr>
    <td>3</td>
    <td>Australia</td>
    <td>12060.33</td>
</tr>
```

```
<tr>
    <td>3</td>
    <td>Canada</td>
    <td>11826.23</td>
</tr>
<tr>
    <td>3</td>
    <td>None</td>
    <td>23886.56</td>
</tr>
<tr>
    <td>4</td>
    <td>Australia</td>
    <td>14136.07</td>
</tr>
<tr>
    <td>4</td>
    <td>Canada</td>
    <td>14423.39</td>
</tr>
<tr>
    <td>4</td>
    <td>None</td>
    <td>28559.46</td>
</tr>
<tr>
    <td>5</td>
    <td>Australia</td>
    <td>271.08</td>
</tr>
<tr>
    <td>5</td>
    <td>Canada</td>
    <td>243.10</td>
</tr>
<tr>
    <td>5</td>
    <td>None</td>
    <td>514.18</td>
</tr>
<tr>
    <td>None</td>
    <td>None</td>
    <td>67416.51</td>
</tr>
<tr>
    <td>None</td>
    <td>Australia</td>
```

```
    <td>33726.77</td>
</tr>
<tr>
    <td>None</td>
    <td>Canada</td>
    <td>33689.74</td>
</tr>
```