L1 E2 - 3 - Grouping Sets

January 10, 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 [ ]:
```
# !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
```

1.0.1 Connect to the local database where Pagila is loaded

In [1]:
```
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[1]:
```
'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 [3]: %%sql
        SELECT SUM(sales_amount) AS "total revenue"
        FROM factsales

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


Out[3]: [(Decimal('67416.51'),)]
```

2.2 Revenue by Country

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

```
In [5]: %%sql
        SELECT country, SUM(sales_amount) as revenue
        FROM factsales f
        JOIN dimcustomer c
          ON f.customer_key = c.customer_key
        GROUP BY c.country

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


Out[5]: [('Cambodia', Decimal('191.47')),
         ('Turkey', Decimal('1662.12')),
         ('Germany', Decimal('831.04')),
         ('Chad', Decimal('135.68')),
         ('Madagascar', Decimal('93.78')),
         ('France', Decimal('374.04')),
         ('Iraq', Decimal('111.73')),
         ('Colombia', Decimal('709.41')),
         ('Japan', Decimal('3471.74')),
```

```
('American Samoa', Decimal('71.80')),
('Virgin Islands, U.S.', Decimal('122.68')),
('Vietnam', Decimal('746.28')),
('Kuwait', Decimal('111.74')),
('Ecuador', Decimal('390.13')),
('Nepal', Decimal('116.78')),
('Cameroon', Decimal('200.46')),
('Saudi Arabia', Decimal('523.79')),
('Bangladesh', Decimal('402.05')),
('Lithuania', Decimal('73.76')),
('Netherlands', Decimal('586.66')),
('Yugoslavia', Decimal('259.43')),
('Gambia', Decimal('129.70')),
('Nigeria', Decimal('1511.48')),
('Dominican Republic', Decimal('318.23')),
('Faroe Islands', Decimal('114.72')),
('Egypt', Decimal('694.39')),
('Zambia', Decimal('134.67')),
('United States', Decimal('4110.32')),
('Argentina', Decimal('1434.48')),
('Slovakia', Decimal('89.74')),
('Angola', Decimal('215.48')),
('Russian Federation', Decimal('3045.87')),
('Oman', Decimal('187.50')),
('Turkmenistan', Decimal('136.73')),
('Afghanistan', Decimal('67.82')),
('Czech Republic', Decimal('133.71')),
('Liechtenstein', Decimal('114.72')),
('Ethiopia', Decimal('91.77')),
('Azerbaijan', Decimal('245.43')),
('Brazil', Decimal('3200.52')),
('Israel', Decimal('422.01')),
('Senegal', Decimal('100.75')),
('Chile', Decimal('328.29')),
('New Zealand', Decimal('92.76')),
('Hungary', Decimal('111.71')),
('Mozambique', Decimal('344.20')),
('Mexico', Decimal('3307.04')),
('Finland', Decimal('101.74')),
('Taiwan', Decimal('1210.94')),
('Sri Lanka', Decimal('116.70')),
('Thailand', Decimal('419.04')),
('Estonia', Decimal('115.70')),
('Latvia', Decimal('262.40')),
('Venezuela', Decimal('683.30')),
('Peru', Decimal('468.88')),
('Holy See (Vatican City State)', Decimal('152.66')),
('Runion', Decimal('216.54')),
```

```
('Anguilla', Decimal('106.65')),
('Bahrain', Decimal('112.75')),
('Algeria', Decimal('383.10')),
('Malaysia', Decimal('369.15')),
('Iran', Decimal('950.75')),
('Spain', Decimal('606.58')),
('Ukraine', Decimal('730.42')),
('Tonga', Decimal('73.82')),
('South Korea', Decimal('574.65')),
('Belarus', Decimal('277.34')),
('French Guiana', Decimal('103.78')),
('Hong Kong', Decimal('142.70')),
('Malawi', Decimal('123.72')),
('Nauru', Decimal('148.69')),
('Romania', Decimal('235.38')),
('North Korea', Decimal('113.69')),
('Puerto Rico', Decimal('254.39')),
('Canada', Decimal('593.63')),
('China', Decimal('5802.73')),
('Austria', Decimal('307.22')),
('Moldova', Decimal('127.66')),
('South Africa', Decimal('1204.15')),
('Kenya', Decimal('253.46')),
('Poland', Decimal('877.97')),
('Kazakstan', Decimal('198.48')),
('United Kingdom', Decimal('924.80')),
('Brunei', Decimal('113.65')),
('Tunisia', Decimal('78.77')),
('Italy', Decimal('831.11')),
('Pakistan', Decimal('525.72')),
('Sweden', Decimal('144.66')),
('Armenia', Decimal('118.75')),
('Greece', Decimal('232.46')),
('India', Decimal('6630.27')),
('Yemen', Decimal('510.83')),
('Paraguay', Decimal('275.38')),
('Myanmar', Decimal('194.48')),
('Philippines', Decimal('2381.32')),
('Morocco', Decimal('307.29')),
('Greenland', Decimal('137.66')),
('Switzerland', Decimal('252.39')),
('Indonesia', Decimal('1510.33')),
('Tuvalu', Decimal('120.74')),
('Saint Vincent and the Grenadines', Decimal('96.75')),
('Bolivia', Decimal('183.53')),
('Congo, The Democratic Republic of the', Decimal('212.50')),
('Sudan', Decimal('227.46')),
('United Arab Emirates', Decimal('333.16')),
```

```
         ('Tanzania', Decimal('341.17')),
         ('French Polynesia', Decimal('235.46')),
         ('Bulgaria', Decimal('204.50'))]
```

2.3 Revenue by Month

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

```
In [6]: %%sql
        SELECT month, SUM(sales_amount) AS revenue
        FROM factsales f
        JOIN dimdate d
          ON f.date_key = d.date_key
        GROUP BY month

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


Out[6]: [(1, Decimal('4824.43')),
         (3, Decimal('23886.56')),
         (4, Decimal('28559.46')),
         (2, Decimal('9631.88')),
         (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 [9]: %%sql
        SELECT month, country, SUM(sales_amount) AS revenue
        FROM factsales f
        JOIN dimdate d
          ON f.date_key = d.date_key
        JOIN dimstore s
          ON f.store_key = s.store_key
        GROUP BY month, country
        ORDER BY month, country, revenue DESC

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


Out[9]: [(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 [16]: %%sql
          SELECT d.month, s.country, sum(sales_amount) AS revenue
          FROM factsales f
          JOIN dimstore s
```

```
          ON f.store_key = s.store_key
        JOIN dimdate d
          ON d.date_key = f.date_key
        GROUP BY GROUPING SETS (d.month, s.country, (), (d.month, s.country))
        ORDER BY d.month, s.country, revenue DESC
```

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

```
Out[16]: [(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, 'Australia', Decimal('33726.77')),
          (None, 'Canada', Decimal('33689.74')),
          (None, None, Decimal('67416.51'))]
```

```
<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>
```