L1 E2 - 1 - Slicing and Dicing

January 10, 2023

1 Exercise 02 - OLAP Cubes - Slicing and Dicing

All the databases table in this demo are based on public database samples and transformations - `Sakila` is a sample database created by `MySql` Link - The postgresql version of it is called `Pagila` Link - The facts and dimension tables design is based on O'Reilly's public dimensional modelling tutorial schema Link

Start by creating and connecting to the database by running the cells below.

```
In [2]: !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.s
```

```
 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 [3]: 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
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
postgresql://student:student@127.0.0.1:5432/pagila_star
```

```
Out[3]: 'Connected: student@pagila_star'
```

1.0.2 Star Schema

2 Start with a simple cube

TODO: Write a query that calculates the revenue (sales_amount) by day, rating, and city. Remember to join with the appropriate dimension tables to replace the keys with the dimension labels. Sort by revenue in descending order and limit to the first 20 rows. The first few rows of your output should match the table below.

```
In [4]: %%time
        %%sql

        SELECT d.day, m.rating, c.city, SUM(f.sales_amount) AS revenue
        FROM factsales f
        JOIN dimmovie m
          ON f.movie_key = m.movie_key
        JOIN dimdate d
          ON f.date_key = d.date_key
        JOIN dimcustomer c
          ON f.customer_key = c.customer_key
        GROUP BY d.day, m.rating, c.city
        ORDER BY revenue DESC
        LIMIT 20

 * postgresql://student:***@127.0.0.1:5432/pagila_star
20 rows affected.
CPU times: user 5.59 ms, sys: 257 µs, total: 5.85 ms
Wall time: 42.3 ms


Out[4]: [(30, 'G', 'San Bernardino', Decimal('24.97')),
         (30, 'NC-17', 'Apeldoorn', Decimal('23.95')),
         (21, 'NC-17', 'Belm', Decimal('22.97')),
         (28, 'R', 'Mwanza', Decimal('21.97')),
         (30, 'PG-13', 'Zanzibar', Decimal('21.97')),
         (21, 'G', 'Citt del Vaticano', Decimal('21.97')),
         (22, 'R', 'Yangor', Decimal('19.97')),
         (1, 'R', 'Qomsheh', Decimal('19.97')),
         (17, 'G', 'Rajkot', Decimal('19.97')),
         (28, 'PG-13', 'Dhaka', Decimal('19.97')),
         (19, 'PG', 'Najafabad', Decimal('19.96')),
         (30, 'R', 'Fengshan', Decimal('19.95')),
         (28, 'PG', 'So Leopoldo', Decimal('18.98')),
         (21, 'G', 'Wroclaw', Decimal('18.98')),
         (1, 'NC-17', 'Memphis', Decimal('18.97')),
         (30, 'G', 'Omdurman', Decimal('18.97')),
         (29, 'PG-13', 'Shimoga', Decimal('18.97')),
         (30, 'PG-13', 'Osmaniye', Decimal('18.97')),
```

```
        (19, 'PG', 'Sokoto', Decimal('18.97')),
        (21, 'PG-13', 'Asuncin', Decimal('18.95'))]
```

```
<tbody><tr>
    <th>day</th>
    <th>rating</th>
    <th>city</th>
    <th>revenue</th>
</tr>
<tr>
    <td>30</td>
    <td>G</td>
    <td>San Bernardino</td>
    <td>24.97</td>
</tr>
<tr>
    <td>30</td>
    <td>NC-17</td>
    <td>Apeldoorn</td>
    <td>23.95</td>
</tr>
<tr>
    <td>21</td>
    <td>NC-17</td>
    <td>Belm</td>
    <td>22.97</td>
</tr>
<tr>
    <td>30</td>
    <td>PG-13</td>
    <td>Zanzibar</td>
    <td>21.97</td>
</tr>
<tr>
    <td>28</td>
    <td>R</td>
    <td>Mwanza</td>
    <td>21.97</td>
</tr>
```

2.1 Slicing

Slicing is the reduction of the dimensionality of a cube by 1 e.g. 3 dimensions to 2, fixing one of the dimensions to a single value. In the example above, we have a 3-dimensional cube on day, rating, and country.

TODO: Write a query that reduces the dimensionality of the above example by limiting the results to only include movies with a `rating` of "PG-13". Again, sort by revenue in descending order and limit to the first 20 rows. The first few rows of your output should match the table

below.

```
In [5]: %%time
        %%sql

        SELECT d.day, m.rating, c.city, SUM(f.sales_amount) AS revenue
        FROM factsales f
        JOIN dimmovie m
          ON f.movie_key = m.movie_key
        JOIN dimdate d
          ON f.date_key = d.date_key
        JOIN dimcustomer c
          ON f.customer_key = c.customer_key
        WHERE m.rating = 'PG-13'
        GROUP BY d.day, m.rating, c.city
        ORDER BY revenue DESC
        LIMIT 20

 * postgresql://student:***@127.0.0.1:5432/pagila_star
20 rows affected.
CPU times: user 5.61 ms, sys: 255 µs, total: 5.86 ms
Wall time: 17.8 ms


Out[5]: [(30, 'PG-13', 'Zanzibar', Decimal('21.97')),
         (28, 'PG-13', 'Dhaka', Decimal('19.97')),
         (30, 'PG-13', 'Osmaniye', Decimal('18.97')),
         (29, 'PG-13', 'Shimoga', Decimal('18.97')),
         (21, 'PG-13', 'Asuncin', Decimal('18.95')),
         (21, 'PG-13', 'Parbhani', Decimal('17.98')),
         (20, 'PG-13', 'Baha Blanca', Decimal('17.98')),
         (30, 'PG-13', 'Nagareyama', Decimal('17.98')),
         (30, 'PG-13', 'Tanauan', Decimal('17.96')),
         (17, 'PG-13', 'Ikerre', Decimal('17.95')),
         (30, 'PG-13', 'Zhoushan', Decimal('16.98')),
         (30, 'PG-13', 'Yerevan', Decimal('16.97')),
         (30, 'PG-13', 'Newcastle', Decimal('16.97')),
         (20, 'PG-13', 'Santa Rosa', Decimal('15.98')),
         (17, 'PG-13', 'Yantai', Decimal('15.98')),
         (30, 'PG-13', 'Santa Rosa', Decimal('15.98')),
         (8, 'PG-13', 'Pontianak', Decimal('15.98')),
         (30, 'PG-13', 'Toulouse', Decimal('15.97')),
         (23, 'PG-13', 'Tiefa', Decimal('15.97')),
         (1, 'PG-13', 'Moscow', Decimal('15.97'))]

<tbody><tr>
    <th>day</th>
    <th>rating</th>
    <th>city</th>
```

```
    <th>revenue</th>
</tr>
<tr>
    <td>30</td>
    <td>PG-13</td>
    <td>Zanzibar</td>
    <td>21.97</td>
</tr>
<tr>
    <td>28</td>
    <td>PG-13</td>
    <td>Dhaka</td>
    <td>19.97</td>
</tr>
<tr>
    <td>29</td>
    <td>PG-13</td>
    <td>Shimoga</td>
    <td>18.97</td>
</tr>
<tr>
    <td>30</td>
    <td>PG-13</td>
    <td>Osmaniye</td>
    <td>18.97</td>
</tr>
<tr>
    <td>21</td>
    <td>PG-13</td>
    <td>Asuncin</td>
    <td>18.95</td>
</tr>
```

2.2 Dicing

Dicing is creating a subcube with the same dimensionality but fewer values for two or more dimensions.

TODO: Write a query to create a subcube of the initial cube that includes moves with: * ratings of PG or PG-13 * in the city of Bellevue or Lancaster * day equal to 1, 15, or 30

The first few rows of your output should match the table below.

```
In [6]: %%time
        %%sql

        SELECT d.day, m.rating, c.city, SUM(f.sales_amount) AS revenue
        FROM factsales f
        JOIN dimmovie m
          ON f.movie_key = m.movie_key
```

```
        JOIN dimdate d
          ON f.date_key = d.date_key
        JOIN dimcustomer c
          ON f.customer_key = c.customer_key
        WHERE m.rating IN ('PG', 'PG-13')
              AND c.city IN ('Bellevue', 'Lancaster')
              AND d.day IN(1, 5, 30)
        GROUP BY d.day, m.rating, c.city
        ORDER BY revenue DESC
        LIMIT 20
```

```
 * postgresql://student:***@127.0.0.1:5432/pagila_star
5 rows affected.
CPU times: user 6.8 ms, sys: 0 ns, total: 6.8 ms
Wall time: 12.2 ms
```

```
Out[6]: [(30, 'PG', 'Lancaster', Decimal('12.98')),
         (1, 'PG-13', 'Lancaster', Decimal('5.99')),
         (30, 'PG-13', 'Bellevue', Decimal('3.99')),
         (30, 'PG-13', 'Lancaster', Decimal('2.99')),
         (1, 'PG', 'Bellevue', Decimal('0.99'))]
```

```
<tbody><tr>
    <th>day</th>
    <th>rating</th>
    <th>city</th>
    <th>revenue</th>
</tr>
<tr>
    <td>30</td>
    <td>PG</td>
    <td>Lancaster</td>
    <td>12.98</td>
</tr>
<tr>
    <td>1</td>
    <td>PG-13</td>
    <td>Lancaster</td>
    <td>5.99</td>
</tr>
<tr>
    <td>30</td>
    <td>PG-13</td>
    <td>Bellevue</td>
    <td>3.99</td>
</tr>
<tr>
```

```
    <td>30</td>
    <td>PG-13</td>
    <td>Lancaster</td>
    <td>2.99</td>
</tr>
<tr>
    <td>15</td>
    <td>PG-13</td>
    <td>Bellevue</td>
    <td>1.98</td>
</tr>
```