L1 E2 - 4 - CUBE

July 1, 2021

1 Exercise 02 - OLAP Cubes - CUBE

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 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
```

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 CUBE

- Group by CUBE (dim1, dim2, ..) , produces all combinations of different lenghts in one go.
- This view could be materialized in a view and queried which would save lots repetitive aggregations

TODO: Write a query that calculates the various levels of aggregation done in the grouping sets exercise (total, by month, by country, by month & country) using the CUBE function. Your output should match the table below.

```
In [3]: %%time
        %%sql
        select dimDate.month, dimStore.country, sum(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 cube (dimDate.month, dimStore.country)
        order by dimDate.month, dimStore.country, revenue desc

 * postgresql://student:***@127.0.0.1:5432/pagila_star
18 rows affected.
CPU times: user 4.06 ms, sys: 497 ţs, total: 4.55 ms
Wall time: 33.4 ms


Out[3]: [(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>
```

2.1 Revenue Total, by Month, by Country, by Month & Country All in one shot, NAIVE way

The naive way to create the same table as above is to write several queries and UNION them together. Grouping sets and cubes produce queries that are shorter to write, easier to read, and more performant. Run the naive query below and compare the time it takes to run to the time it takes the cube query to run.

```
In [4]: %%time
        %%sql
        SELECT  NULL as month, NULL as country, sum(sales_amount) as revenue
        FROM factSales
            UNION all
        SELECT NULL, dimStore.country,sum(sales_amount) as revenue
        FROM factSales
        JOIN dimStore on (dimStore.store_key = factSales.store_key)
        GROUP by  dimStore.country
            UNION all
        SELECT cast(dimDate.month as text) , NULL, sum(sales_amount) as revenue
        FROM factSales
        JOIN dimDate on (dimDate.date_key = factSales.date_key)
        GROUP by dimDate.month
            UNION all
        SELECT cast(dimDate.month as text),dimStore.country,sum(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 (dimDate.month, dimStore.country)

 * postgresql://student:***@127.0.0.1:5432/pagila_star
18 rows affected.
CPU times: user 7.58 ms, sys: 0 ns, total: 7.58 ms
Wall time: 43.2 ms


Out[4]: [(None, None, Decimal('67416.51')),
         (None, 'Australia', Decimal('33726.77')),
         (None, 'Canada', Decimal('33689.74')),
         ('1', None, Decimal('4824.43')),
         ('3', None, Decimal('23886.56')),
         ('4', None, Decimal('28559.46')),
         ('2', None, Decimal('9631.88')),
         ('5', None, Decimal('514.18')),
         ('4', 'Canada', Decimal('14423.39')),
         ('3', 'Australia', Decimal('12060.33')),
         ('5', 'Canada', Decimal('243.10')),
         ('1', 'Australia', Decimal('2364.19')),
         ('2', 'Canada', Decimal('4736.78')),
         ('2', 'Australia', Decimal('4895.10')),
```

```
        ('1', 'Canada', Decimal('2460.24')),
        ('5', 'Australia', Decimal('271.08')),
        ('3', 'Canada', Decimal('11826.23')),
        ('4', 'Australia', Decimal('14136.07'))]
```

In [ ]: