L1 E1 - Solution

March 10, 2022

1 Demo 01 - Sakila Star Schema & ETL

All the database tables 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

2 STEP0: Using ipython-sql

- load ipython-sql: `%load_ext sql`
- To execute SQL queries you write one of the following atop of your cell:
  - `%sql`
    - For a one-liner SQL query
    - You can access a python var using `$`
  - `%%sql`
    - For a multi-line SQL query
    - You can **NOT** access a python var using `$`
- Running a connection string like: `postgresql://postgres:postgres@db:5432/pagila` connects to the database

3 STEP1 : Connect to the local database where Pagila is loaded

3.1 1.1 Create the pagila db and fill it with data

- Adding `"!"` at the beginning of a jupyter cell runs a command in a shell, i.e. we are not running python code but we are running the `createdb` and `psql` postgresql commmand-line utilities

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

```
createdb: database creation failed: ERROR:  database "pagila" already exists
psql:Data/pagila-schema.sql:43: ERROR:  type "mpaa_rating" already exists
psql:Data/pagila-schema.sql:53: ERROR:  type "year" already exists
psql:Data/pagila-schema.sql:70: ERROR:  function "_group_concat" already exists with same argume
psql:Data/pagila-schema.sql:87: ERROR:  function "film_in_stock" already exists with same argume
psql:Data/pagila-schema.sql:104: ERROR:  function "film_not_in_stock" already exists with same a
psql:Data/pagila-schema.sql:149: ERROR:  function "get_customer_balance" already exists with sam
psql:Data/pagila-schema.sql:171: ERROR:  function "inventory_held_by_customer" already exists wi
psql:Data/pagila-schema.sql:208: ERROR:  function "inventory_in_stock" already exists with same
psql:Data/pagila-schema.sql:226: ERROR:  function "last_day" already exists with same argument t
psql:Data/pagila-schema.sql:241: ERROR:  function "last_updated" already exists with same argume
psql:Data/pagila-schema.sql:255: ERROR:  relation "customer_customer_id_seq" already exists
psql:Data/pagila-schema.sql:279: ERROR:  relation "customer" already exists
psql:Data/pagila-schema.sql:343: ERROR:  function "rewards_report" already exists with same argu
psql:Data/pagila-schema.sql:355: ERROR:  function "group_concat" already exists with same argume
psql:Data/pagila-schema.sql:369: ERROR:  relation "actor_actor_id_seq" already exists
psql:Data/pagila-schema.sql:383: ERROR:  relation "actor" already exists
psql:Data/pagila-schema.sql:397: ERROR:  relation "category_category_id_seq" already exists
psql:Data/pagila-schema.sql:410: ERROR:  relation "category" already exists
psql:Data/pagila-schema.sql:424: ERROR:  relation "film_film_id_seq" already exists
psql:Data/pagila-schema.sql:448: ERROR:  relation "film" already exists
psql:Data/pagila-schema.sql:461: ERROR:  relation "film_actor" already exists
psql:Data/pagila-schema.sql:474: ERROR:  relation "film_category" already exists
psql:Data/pagila-schema.sql:497: ERROR:  relation "actor_info" already exists
psql:Data/pagila-schema.sql:511: ERROR:  relation "address_address_id_seq" already exists
psql:Data/pagila-schema.sql:529: ERROR:  relation "address" already exists
psql:Data/pagila-schema.sql:543: ERROR:  relation "city_city_id_seq" already exists
psql:Data/pagila-schema.sql:557: ERROR:  relation "city" already exists
psql:Data/pagila-schema.sql:571: ERROR:  relation "country_country_id_seq" already exists
psql:Data/pagila-schema.sql:584: ERROR:  relation "country" already exists
psql:Data/pagila-schema.sql:609: ERROR:  relation "customer_list" already exists
psql:Data/pagila-schema.sql:632: ERROR:  relation "film_list" already exists
psql:Data/pagila-schema.sql:646: ERROR:  relation "inventory_inventory_id_seq" already exists
psql:Data/pagila-schema.sql:660: ERROR:  relation "inventory" already exists
psql:Data/pagila-schema.sql:674: ERROR:  relation "language_language_id_seq" already exists
psql:Data/pagila-schema.sql:687: ERROR:  relation "language" already exists
psql:Data/pagila-schema.sql:710: ERROR:  relation "nicer_but_slower_film_list" already exists
psql:Data/pagila-schema.sql:724: ERROR:  relation "payment_payment_id_seq" already exists
psql:Data/pagila-schema.sql:740: ERROR:  relation "payment" already exists
psql:Data/pagila-schema.sql:751: ERROR:  relation "rental_rental_id_seq" already exists
psql:Data/pagila-schema.sql:768: ERROR:  relation "rental" already exists
psql:Data/pagila-schema.sql:787: ERROR:  relation "sales_by_film_category" already exists
psql:Data/pagila-schema.sql:801: ERROR:  relation "staff_staff_id_seq" already exists
psql:Data/pagila-schema.sql:822: ERROR:  relation "staff" already exists
psql:Data/pagila-schema.sql:836: ERROR:  relation "store_store_id_seq" already exists
psql:Data/pagila-schema.sql:850: ERROR:  relation "store" already exists
psql:Data/pagila-schema.sql:872: ERROR:  relation "sales_by_store" already exists
psql:Data/pagila-schema.sql:893: ERROR:  relation "staff_list" already exists
```

```
psql:Data/pagila-schema.sql:903: ERROR:  multiple primary keys for table "actor" are not allowed
psql:Data/pagila-schema.sql:911: ERROR:  multiple primary keys for table "address" are not allow
psql:Data/pagila-schema.sql:919: ERROR:  multiple primary keys for table "category" are not allo
psql:Data/pagila-schema.sql:927: ERROR:  multiple primary keys for table "city" are not allowed
psql:Data/pagila-schema.sql:935: ERROR:  multiple primary keys for table "country" are not allow
psql:Data/pagila-schema.sql:944: ERROR:  multiple primary keys for table "film_actor" are not al
psql:Data/pagila-schema.sql:952: ERROR:  multiple primary keys for table "film_category" are not
psql:Data/pagila-schema.sql:960: ERROR:  multiple primary keys for table "film" are not allowed
psql:Data/pagila-schema.sql:968: ERROR:  multiple primary keys for table "inventory" are not all
psql:Data/pagila-schema.sql:976: ERROR:  multiple primary keys for table "language" are not allo
psql:Data/pagila-schema.sql:984: ERROR:  multiple primary keys for table "rental" are not allowe
psql:Data/pagila-schema.sql:992: ERROR:  multiple primary keys for table "staff" are not allowed
psql:Data/pagila-schema.sql:1000: ERROR:  multiple primary keys for table "store" are not allowe
psql:Data/pagila-schema.sql:1007: ERROR:  relation "film_fulltext_idx" already exists
psql:Data/pagila-schema.sql:1014: ERROR:  relation "idx_actor_last_name" already exists
psql:Data/pagila-schema.sql:1021: ERROR:  relation "idx_fk_address_id" already exists
psql:Data/pagila-schema.sql:1028: ERROR:  relation "idx_fk_city_id" already exists
psql:Data/pagila-schema.sql:1035: ERROR:  relation "idx_fk_country_id" already exists
psql:Data/pagila-schema.sql:1042: ERROR:  relation "idx_fk_customer_id" already exists
psql:Data/pagila-schema.sql:1049: ERROR:  relation "idx_fk_film_id" already exists
psql:Data/pagila-schema.sql:1056: ERROR:  relation "idx_fk_inventory_id" already exists
psql:Data/pagila-schema.sql:1063: ERROR:  relation "idx_fk_language_id" already exists
psql:Data/pagila-schema.sql:1070: ERROR:  relation "idx_fk_original_language_id" already exists
psql:Data/pagila-schema.sql:1077: ERROR:  relation "idx_fk_payment_customer_id" already exists
psql:Data/pagila-schema.sql:1083: ERROR:  relation "idx_fk_payment_staff_id" already exists
psql:Data/pagila-schema.sql:1092: ERROR:  relation "idx_fk_store_id" already exists
psql:Data/pagila-schema.sql:1099: ERROR:  relation "idx_last_name" already exists
psql:Data/pagila-schema.sql:1106: ERROR:  relation "idx_store_id_film_id" already exists
psql:Data/pagila-schema.sql:1113: ERROR:  relation "idx_title" already exists
psql:Data/pagila-schema.sql:1120: ERROR:  relation "idx_unq_manager_staff_id" already exists
psql:Data/pagila-schema.sql:1127: ERROR:  relation "idx_unq_rental_rental_date_inventory_id_cust
psql:Data/pagila-schema.sql:1133: ERROR:  trigger "film_fulltext_trigger" for relation "film" al
psql:Data/pagila-schema.sql:1140: ERROR:  trigger "last_updated" for relation "actor" already ex
psql:Data/pagila-schema.sql:1147: ERROR:  trigger "last_updated" for relation "address" already
psql:Data/pagila-schema.sql:1154: ERROR:  trigger "last_updated" for relation "category" already
psql:Data/pagila-schema.sql:1161: ERROR:  trigger "last_updated" for relation "city" already exi
psql:Data/pagila-schema.sql:1168: ERROR:  trigger "last_updated" for relation "country" already
psql:Data/pagila-schema.sql:1175: ERROR:  trigger "last_updated" for relation "customer" already
psql:Data/pagila-schema.sql:1182: ERROR:  trigger "last_updated" for relation "film" already exi
psql:Data/pagila-schema.sql:1189: ERROR:  trigger "last_updated" for relation "film_actor" alrea
psql:Data/pagila-schema.sql:1196: ERROR:  trigger "last_updated" for relation "film_category" al
psql:Data/pagila-schema.sql:1203: ERROR:  trigger "last_updated" for relation "inventory" alread
psql:Data/pagila-schema.sql:1210: ERROR:  trigger "last_updated" for relation "language" already
psql:Data/pagila-schema.sql:1217: ERROR:  trigger "last_updated" for relation "rental" already e
psql:Data/pagila-schema.sql:1224: ERROR:  trigger "last_updated" for relation "staff" already ex
psql:Data/pagila-schema.sql:1231: ERROR:  trigger "last_updated" for relation "store" already ex
psql:Data/pagila-schema.sql:1239: ERROR:  constraint "address_city_id_fkey" for relation "addres
psql:Data/pagila-schema.sql:1247: ERROR:  constraint "city_country_id_fkey" for relation "city"
```

```
psql:Data/pagila-schema.sql:1255: ERROR:  constraint "customer_address_id_fkey" for relation "cu
psql:Data/pagila-schema.sql:1263: ERROR:  constraint "customer_store_id_fkey" for relation "cust
psql:Data/pagila-schema.sql:1271: ERROR:  constraint "film_actor_actor_id_fkey" for relation "fi
psql:Data/pagila-schema.sql:1279: ERROR:  constraint "film_actor_film_id_fkey" for relation "fil
psql:Data/pagila-schema.sql:1287: ERROR:  constraint "film_category_category_id_fkey" for relati
psql:Data/pagila-schema.sql:1295: ERROR:  constraint "film_category_film_id_fkey" for relation "
psql:Data/pagila-schema.sql:1303: ERROR:  constraint "film_language_id_fkey" for relation "film"
psql:Data/pagila-schema.sql:1311: ERROR:  constraint "film_original_language_id_fkey" for relati
psql:Data/pagila-schema.sql:1319: ERROR:  constraint "inventory_film_id_fkey" for relation "inve
psql:Data/pagila-schema.sql:1327: ERROR:  constraint "inventory_store_id_fkey" for relation "inv
psql:Data/pagila-schema.sql:1334: ERROR:  constraint "rental_customer_id_fkey" for relation "ren
psql:Data/pagila-schema.sql:1342: ERROR:  constraint "rental_inventory_id_fkey" for relation "re
psql:Data/pagila-schema.sql:1350: ERROR:  constraint "rental_staff_id_fkey" for relation "rental
psql:Data/pagila-schema.sql:1358: ERROR:  constraint "staff_address_id_fkey" for relation "staff
psql:Data/pagila-schema.sql:1366: ERROR:  constraint "staff_store_id_fkey" for relation "staff"
psql:Data/pagila-schema.sql:1374: ERROR:  constraint "store_address_id_fkey" for relation "store
psql:Data/pagila-schema.sql:1384: ERROR:  constraint "payment_customer_id_fkey" for relation "pa
psql:Data/pagila-data.sql:224: ERROR:  duplicate key value violates unique constraint "actor_pke
DETAIL:  Key (actor_id)=(1) already exists.
CONTEXT:  COPY actor, line 1
psql:Data/pagila-data.sql:341: ERROR:  duplicate key value violates unique constraint "country_p
DETAIL:  Key (country_id)=(1) already exists.
CONTEXT:  COPY country, line 1
psql:Data/pagila-data.sql:949: ERROR:  duplicate key value violates unique constraint "city_pkey
DETAIL:  Key (city_id)=(1) already exists.
CONTEXT:  COPY city, line 1
psql:Data/pagila-data.sql:1560: ERROR:  duplicate key value violates unique constraint "address_
DETAIL:  Key (address_id)=(1) already exists.
CONTEXT:  COPY address, line 1
psql:Data/pagila-data.sql:1584: ERROR:  duplicate key value violates unique constraint "category
DETAIL:  Key (category_id)=(1) already exists.
CONTEXT:  COPY category, line 1
psql:Data/pagila-data.sql:1594: ERROR:  duplicate key value violates unique constraint "store_pk
DETAIL:  Key (store_id)=(1) already exists.
CONTEXT:  COPY store, line 1
psql:Data/pagila-data.sql:2201: ERROR:  duplicate key value violates unique constraint "customer
DETAIL:  Key (customer_id)=(1) already exists.
CONTEXT:  COPY customer, line 1
psql:Data/pagila-data.sql:2215: ERROR:  duplicate key value violates unique constraint "language
DETAIL:  Key (language_id)=(1) already exists.
CONTEXT:  COPY language, line 1
psql:Data/pagila-data.sql:3223: ERROR:  duplicate key value violates unique constraint "film_pke
DETAIL:  Key (film_id)=(1) already exists.
CONTEXT:  COPY film, line 1: "1        ACADEMY DINOSAUR        A Epic Drama of a Feminist And a
psql:Data/pagila-data.sql:8693: ERROR:  duplicate key value violates unique constraint "film_act
DETAIL:  Key (actor_id, film_id)=(1, 1) already exists.
CONTEXT:  COPY film_actor, line 1
psql:Data/pagila-data.sql:9701: ERROR:  duplicate key value violates unique constraint "film_cat
```

```
DETAIL:  Key (film_id, category_id)=(1, 6) already exists.
CONTEXT:  COPY film_category, line 1
psql:Data/pagila-data.sql:14290: ERROR:  duplicate key value violates unique constraint "inventc
DETAIL:  Key (inventory_id)=(1) already exists.
CONTEXT:  COPY inventory, line 1
psql:Data/pagila-data.sql:14300: ERROR:  duplicate key value violates unique constraint "staff_p
DETAIL:  Key (staff_id)=(1) already exists.
CONTEXT:  COPY staff, line 1
psql:Data/pagila-data.sql:30352: ERROR:  duplicate key value violates unique constraint "rental_
DETAIL:  Key (rental_id)=(2) already exists.
CONTEXT:  COPY rental, line 1
 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
--------
   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)
```

3.2 1.2 Connect to the newly created db

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

```
In [3]: DB_ENDPOINT = "127.0.0.1"
        DB = 'pagila'
        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)
```

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

```
In [4]: %sql $conn_string
```

```
Out[4]: 'Connected: student@pagila'
```

4 STEP 1.1: Create and populate the star schema

5 STEP2 : Explore the 3NF Schema

5.1 2.1 How much? What data sizes are we looking at?

```
In [5]: nStores = %sql select count(*) from store;
        nFilms = %sql select count(*) from film;
        nCustomers = %sql select count(*) from customer;
        nRentals = %sql select count(*) from rental;
        nPayment = %sql select count(*) from payment;
        nStaff = %sql select count(*) from staff;
        nCity = %sql select count(*) from city;
        nCountry = %sql select count(*) from country;

        print("nFilms\t\t=", nFilms[0][0])
        print("nCustomers\t=", nCustomers[0][0])
        print("nRentals\t=", nRentals[0][0])
        print("nPayment\t=", nPayment[0][0])
        print("nStaff\t\t=", nStaff[0][0])
        print("nStores\t\t=", nStores[0][0])
        print("nCities\t\t=", nCity[0][0])
        print("nCountry\t\t=", nCountry[0][0])
```

```
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
nFilms                = 1000
nCustomers        = 599
nRentals        = 16044
nPayment        = 32098
```

```
nStaff                = 2
nStores                = 2
nCities                = 600
nCountry                = 109
```

5.2 2.2 When? What time period are we talking about?

```
In [6]: %%sql
        select min(payment_date) as start, max(payment_date) as end from payment;
```

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

```
Out[6]: [(datetime.datetime(2017, 1, 24, 21, 21, 56, 996577, tzinfo=psycopg2.tz.FixedOffsetTimez
```

5.3 2.3 Where? Where do events in this database occur?

```
In [7]: %%sql
        select district,  sum(city_id) as n
        from address
        group by district
        order by n desc
        limit 10;
```

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

```
Out[7]: [('Shandong', 3237),
         ('England', 2974),
         ('So Paulo', 2952),
         ('West Bengali', 2623),
         ('Buenos Aires', 2572),
         ('Uttar Pradesh', 2462),
         ('California', 2444),
         ('Southern Tagalog', 1931),
         ('Tamil Nadu', 1807),
         ('Hubei', 1790)]
```

6 STEP3: Perform some simple data analysis

6.1 3.1 Insight 1: Top Grossing Movies

- Payments amounts are in table `payment`
- Movies are in table `film`
- They are not directly linked, `payment` refers to a `rental`, `rental` refers to an `inventory` item and `inventory` item refers to a `film`
- `payment rental inventory film`

6.1.1 3.1.1 Films

```
In [8]: %%sql
        select film_id, title, release_year, rental_rate, rating  from film limit 5;

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


Out[8]: [(1, 'ACADEMY DINOSAUR', 2006, Decimal('0.99'), 'PG'),
         (2, 'ACE GOLDFINGER', 2006, Decimal('4.99'), 'G'),
         (3, 'ADAPTATION HOLES', 2006, Decimal('2.99'), 'NC-17'),
         (4, 'AFFAIR PREJUDICE', 2006, Decimal('2.99'), 'G'),
         (5, 'AFRICAN EGG', 2006, Decimal('2.99'), 'G')]
```

6.1.2 3.1.2 Payments

```
In [9]: %%sql
        select * from payment limit 5;

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


Out[9]: [(16050, 269, 2, 7, Decimal('1.99'), datetime.datetime(2017, 1, 24, 21, 40, 19, 996577,
         (16051, 269, 1, 98, Decimal('0.99'), datetime.datetime(2017, 1, 25, 15, 16, 50, 996577,
         (16052, 269, 2, 678, Decimal('6.99'), datetime.datetime(2017, 1, 28, 21, 44, 14, 996577
         (16053, 269, 2, 703, Decimal('0.99'), datetime.datetime(2017, 1, 29, 0, 58, 2, 996577,
         (16054, 269, 1, 750, Decimal('4.99'), datetime.datetime(2017, 1, 29, 8, 10, 6, 996577,
```

6.1.3 3.1.3 Inventory

```
In [10]: %%sql
         select * from inventory limit 5;

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


Out[10]: [(1, 1, 1, datetime.datetime(2017, 2, 15, 10, 9, 17, tzinfo=psycopg2.tz.FixedOffsetTime
          (2, 1, 1, datetime.datetime(2017, 2, 15, 10, 9, 17, tzinfo=psycopg2.tz.FixedOffsetTime
          (3, 1, 1, datetime.datetime(2017, 2, 15, 10, 9, 17, tzinfo=psycopg2.tz.FixedOffsetTime
          (4, 1, 1, datetime.datetime(2017, 2, 15, 10, 9, 17, tzinfo=psycopg2.tz.FixedOffsetTime
          (5, 1, 2, datetime.datetime(2017, 2, 15, 10, 9, 17, tzinfo=psycopg2.tz.FixedOffsetTime
```

6.1.4 3.1.4 Get the movie of every payment

```
In [11]: %%sql
         SELECT f.title, p.amount, p.payment_date, p.customer_id
         FROM payment p
```

```
        JOIN rental r    ON ( p.rental_id = r.rental_id )
        JOIN inventory i ON ( r.inventory_id = i.inventory_id )
        JOIN film f ON ( i.film_id = f.film_id)
        limit 5;
```

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

```
Out[11]: [('SWARM GOLD', Decimal('1.99'), datetime.datetime(2017, 1, 24, 21, 40, 19, 996577, tzi
          ('PACKER MADIGAN', Decimal('0.99'), datetime.datetime(2017, 1, 25, 15, 16, 50, 996577,
          ('SOMETHING DUCK', Decimal('6.99'), datetime.datetime(2017, 1, 28, 21, 44, 14, 996577,
          ('DRACULA CRYSTAL', Decimal('0.99'), datetime.datetime(2017, 1, 29, 0, 58, 2, 996577,
          ('CLOSER BANG', Decimal('4.99'), datetime.datetime(2017, 1, 29, 8, 10, 6, 996577, tzin
```

6.1.5 3.1.5 sum movie rental revenue

```
In [12]: %%sql
         SELECT f.title, sum(p.amount) as revenue
         FROM payment p
         JOIN rental r    ON ( p.rental_id = r.rental_id )
         JOIN inventory i ON ( r.inventory_id = i.inventory_id )
         JOIN film f ON ( i.film_id = f.film_id)
         GROUP BY title
         ORDER BY revenue desc
         limit 10;
```

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

```
Out[12]: [('TELEGRAPH VOYAGE', Decimal('463.46')),
          ('WIFE TURN', Decimal('447.38')),
          ('ZORRO ARK', Decimal('429.38')),
          ('GOODFELLAS SALUTE', Decimal('419.38')),
          ('SATURDAY LAMBS', Decimal('409.44')),
          ('TITANS JERK', Decimal('403.42')),
          ('TORQUE BOUND', Decimal('397.44')),
          ('HARRY IDAHO', Decimal('391.40')),
          ('INNOCENT USUAL', Decimal('383.48')),
          ('HUSTLER PARTY', Decimal('381.56'))]
```

6.2 3.2 Insight 2: Top grossing cities

- Payments amounts are in table `payment`
- Cities are in table `cities`
- `payment customer address city`

6.2.1 3.2.1 Get the city of each payment

```
In [13]: %%sql
         SELECT p.customer_id, p.rental_id, p.amount, ci.city
         FROM payment p
         JOIN customer c  ON ( p.customer_id = c.customer_id )
         JOIN address a ON ( c.address_id = a.address_id )
         JOIN city ci ON ( a.city_id = ci.city_id )
         order by p.payment_date
         limit 10;

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


Out[13]: [(130, 1, Decimal('2.99'), 'guas Lindas de Gois'),
          (130, 1, Decimal('2.99'), 'guas Lindas de Gois'),
          (459, 2, Decimal('2.99'), 'Qomsheh'),
          (459, 2, Decimal('2.99'), 'Qomsheh'),
          (408, 3, Decimal('3.99'), 'Jaffna'),
          (408, 3, Decimal('3.99'), 'Jaffna'),
          (333, 4, Decimal('4.99'), 'Baku'),
          (333, 4, Decimal('4.99'), 'Baku'),
          (222, 5, Decimal('6.99'), 'Jaroslavl'),
          (222, 5, Decimal('6.99'), 'Jaroslavl')]
```

6.2.2 3.2.2 Top grossing cities

```
In [14]: %%sql
         SELECT ci.city ,  sum(p.amount) as revenue
         FROM payment p
         JOIN customer c  ON ( p.customer_id = c.customer_id )
         JOIN address a ON ( c.address_id = a.address_id )
         JOIN city ci ON ( a.city_id = ci.city_id )
         group by ci.city
         order by revenue desc
         limit 10;

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


Out[14]: [('Cape Coral', Decimal('443.10')),
          ('Saint-Denis', Decimal('433.08')),
          ('Aurora', Decimal('397.00')),
          ('Molodetno', Decimal('391.16')),
          ('Apeldoorn', Decimal('389.22')),
          ('Santa Brbara dOeste', Decimal('389.22')),
          ('Qomsheh', Decimal('373.24')),
```

```
          ('London', Decimal('361.04')),
          ('Ourense (Orense)', Decimal('355.20')),
          ('Bijapur', Decimal('351.22'))]
```

6.3 3.3 Insight 3 : Revenue of a movie by customer city and by month

6.3.1 3.3.1 Total revenue by month

```
In [15]: %%sql
         SELECT sum(p.amount) as revenue, EXTRACT(month FROM p.payment_date) as month
         from payment p
         group by month
         order by revenue desc
         limit 10;

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


Out[15]: [(Decimal('57118.92'), 4.0),
          (Decimal('47773.12'), 3.0),
          (Decimal('19263.76'), 2.0),
          (Decimal('9648.86'), 1.0),
          (Decimal('1028.36'), 5.0)]
```

6.3.2 3.3.2 Each movie by customer city and by month (data cube)

```
In [16]: %%sql
         SELECT f.title, p.amount, p.customer_id, ci.city, p.payment_date,EXTRACT(month FROM p.p
         FROM payment p
         JOIN rental r    ON ( p.rental_id = r.rental_id )
         JOIN inventory i ON ( r.inventory_id = i.inventory_id )
         JOIN film f ON ( i.film_id = f.film_id)
         JOIN customer c  ON ( p.customer_id = c.customer_id )
         JOIN address a ON ( c.address_id = a.address_id )
         JOIN city ci ON ( a.city_id = ci.city_id )
         order by p.payment_date
         limit 10;

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


Out[16]: [('BLANKET BEVERLY', Decimal('2.99'), 130, 'guas Lindas de Gois', datetime.datetime(201
          ('BLANKET BEVERLY', Decimal('2.99'), 130, 'guas Lindas de Gois', datetime.datetime(201
          ('FREAKY POCUS', Decimal('2.99'), 459, 'Qomsheh', datetime.datetime(2017, 1, 24, 21, 2
          ('FREAKY POCUS', Decimal('2.99'), 459, 'Qomsheh', datetime.datetime(2017, 1, 24, 21, 2
          ('GRADUATE LORD', Decimal('3.99'), 408, 'Jaffna', datetime.datetime(2017, 1, 24, 21, 3
          ('GRADUATE LORD', Decimal('3.99'), 408, 'Jaffna', datetime.datetime(2017, 1, 24, 21, 3
```

```
 ('LOVE SUICIDES', Decimal('4.99'), 333, 'Baku', datetime.datetime(2017, 1, 24, 21, 33,
 ('LOVE SUICIDES', Decimal('4.99'), 333, 'Baku', datetime.datetime(2017, 1, 24, 21, 33,
 ('IDOLS SNATCHERS', Decimal('6.99'), 222, 'Jaroslavl', datetime.datetime(2017, 1, 24,
 ('IDOLS SNATCHERS', Decimal('6.99'), 222, 'Jaroslavl', datetime.datetime(2017, 1, 24,
```

6.3.3 3.3.3 Sum of revenue of each movie by customer city and by month

```
In [17]: %%sql
         SELECT f.title, ci.city,EXTRACT(month FROM p.payment_date) as month, sum(p.amount) as r
         FROM payment p
         JOIN rental r    ON ( p.rental_id = r.rental_id )
         JOIN inventory i ON ( r.inventory_id = i.inventory_id )
         JOIN film f ON ( i.film_id = f.film_id)
         JOIN customer c  ON ( p.customer_id = c.customer_id )
         JOIN address a ON ( c.address_id = a.address_id )
         JOIN city ci ON ( a.city_id = ci.city_id )
         group by (f.title, ci.city, month)
         order by month, revenue desc
         limit 10;

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


Out[17]: [('SHOW LORD', 'Mannheim', 1.0, Decimal('23.98')),
          ('AMERICAN CIRCUS', 'Callao', 1.0, Decimal('21.98')),
          ('CASUALTIES ENCINO', 'Warren', 1.0, Decimal('21.98')),
          ('TELEGRAPH VOYAGE', 'Naala-Porto', 1.0, Decimal('21.98')),
          ('KISSING DOLLS', 'Toulon', 1.0, Decimal('21.98')),
          ('MILLION ACE', 'Bergamo', 1.0, Decimal('19.98')),
          ('TITANS JERK', 'Kimberley', 1.0, Decimal('19.98')),
          ('DARKO DORADO', 'Bhilwara', 1.0, Decimal('19.98')),
          ('SUNRISE LEAGUE', 'Nagareyama', 1.0, Decimal('19.98')),
          ('MILLION ACE', 'Gaziantep', 1.0, Decimal('19.98'))]
```

7 STEP 4 : Creating Facts & Dimensions

```
In [18]: %%sql
         CREATE TABLE dimDate
         (
           date_key integer NOT NULL PRIMARY KEY,
           date date NOT NULL,
           year smallint NOT NULL,
           quarter smallint NOT NULL,
           month smallint NOT NULL,
           day smallint NOT NULL,
           week smallint NOT NULL,
           is_weekend boolean
```

```
);

CREATE TABLE dimCustomer
(
  customer_key SERIAL PRIMARY KEY,
  customer_id  smallint NOT NULL,
  first_name   varchar(45) NOT NULL,
  last_name    varchar(45) NOT NULL,
  email        varchar(50),
  address      varchar(50) NOT NULL,
  address2     varchar(50),
  district     varchar(20) NOT NULL,
  city         varchar(50) NOT NULL,
  country      varchar(50) NOT NULL,
  postal_code  varchar(10),
  phone        varchar(20) NOT NULL,
  active       smallint NOT NULL,
  create_date  timestamp NOT NULL,
  start_date   date NOT NULL,
  end_date     date NOT NULL
);

CREATE TABLE dimMovie
(
  movie_key          SERIAL PRIMARY KEY,
  film_id            smallint NOT NULL,
  title              varchar(255) NOT NULL,
  description        text,
  release_year       year,
  language           varchar(20) NOT NULL,
  original_language  varchar(20),
  rental_duration    smallint NOT NULL,
  length             smallint NOT NULL,
  rating             varchar(5) NOT NULL,
  special_features   varchar(60) NOT NULL
);
CREATE TABLE dimStore
(
  store_key           SERIAL PRIMARY KEY,
  store_id            smallint NOT NULL,
  address             varchar(50) NOT NULL,
  address2            varchar(50),
  district            varchar(20) NOT NULL,
  city                varchar(50) NOT NULL,
  country             varchar(50) NOT NULL,
  postal_code         varchar(10),
  manager_first_name  varchar(45) NOT NULL,
  manager_last_name   varchar(45) NOT NULL,
```

```
  start_date          date NOT NULL,
  end_date            date NOT NULL
);
CREATE TABLE factSales
(
  sales_key        SERIAL PRIMARY KEY,
  date_key         INT NOT NULL REFERENCES dimDate(date_key),
  customer_key     INT NOT NULL REFERENCES dimCustomer(customer_key),
  movie_key        INT NOT NULL REFERENCES dimMovie(movie_key),
  store_key        INT NOT NULL REFERENCES dimStore(store_key),
  sales_amount     decimal(5,2) NOT NULL
);
```

```
 * postgresql://student:***@127.0.0.1:5432/pagila
(psycopg2.ProgrammingError) relation "dimdate" already exists
 [SQL: 'CREATE TABLE dimDate\n(\n  date_key integer NOT NULL PRIMARY KEY,\n  date date NOT NULL,
```

8 STEP 5: ETL the data from 3NF tables to Facts & Dimension Tables

In [19]:
```
%%sql
INSERT INTO dimDate (date_key, date, year, quarter, month, day, week, is_weekend)
SELECT DISTINCT(TO_CHAR(payment_date :: DATE, 'yyyyMMDD')::integer) AS date_key,
       date(payment_date)                                           AS date,
       EXTRACT(year FROM payment_date)                              AS year,
       EXTRACT(quarter FROM payment_date)                           AS quarter,
       EXTRACT(month FROM payment_date)                             AS month,
       EXTRACT(day FROM payment_date)                               AS day,
       EXTRACT(week FROM payment_date)                              AS week,
       CASE WHEN EXTRACT(ISODOW FROM payment_date) IN (6, 7) THEN true ELSE false END A
FROM payment;


INSERT INTO dimCustomer (customer_key, customer_id, first_name, last_name, email, addre
SELECT c.customer_id AS customer_key,
       c.customer_id,
       c.first_name,
       c.last_name,
       c.email,
       a.address,
       a.address2,
       a.district,
       ci.city,
       co.country,
       a.postal_code,
       a.phone,
       c.active,
```

```
       c.create_date,
       now()         AS start_date,
       now()         AS end_date
FROM customer c
JOIN address a  ON (c.address_id = a.address_id)
JOIN city ci    ON (a.city_id = ci.city_id)
JOIN country co ON (ci.country_id = co.country_id);

INSERT INTO dimMovie (movie_key, film_id, title, description, release_year, language, c
SELECT f.film_id      AS movie_key,
       f.film_id,
       f.title,
       f.description,
       f.release_year,
       l.name         AS language,
       orig_lang.name AS original_language,
       f.rental_duration,
       f.length,
       f.rating,
       f.special_features
FROM film f
JOIN language l              ON (f.language_id=l.language_id)
LEFT JOIN language orig_lang ON (f.original_language_id = orig_lang.language_id);

INSERT INTO dimStore (store_key, store_id, address, address2, district, city, country,
SELECT s.store_id    AS store_key,
       s.store_id,
       a.address,
       a.address2,
       a.district,
       c.city,
       co.country,
       a.postal_code,
       st.first_name AS manager_first_name,
       st.last_name  AS manager_last_name,
       now()         AS start_date,
       now()         AS end_date
FROM store s
JOIN staff st   ON (s.manager_staff_id = st.staff_id)
JOIN address a  ON (s.address_id = a.address_id)
JOIN city c     ON (a.city_id = c.city_id)
JOIN country co ON (c.country_id = co.country_id);

INSERT INTO factSales (date_key, customer_key, movie_key, store_key, sales_amount)
SELECT TO_CHAR(p.payment_date :: DATE, 'yyyyMMDD')::integer AS date_key ,
       p.customer_id                                        AS customer_key,
       i.film_id                                            AS movie_key,
       i.store_id                                           AS store_key,
```

```
        p.amount                                                AS sales_amount
 FROM payment p
 JOIN rental r    ON ( p.rental_id = r.rental_id )
 JOIN inventory i ON ( r.inventory_id = i.inventory_id );
```

```
* postgresql://student:***@127.0.0.1:5432/pagila
```

```
---------------------------------------------------------------------------

IntegrityError                            Traceback (most recent call last)

/opt/conda/lib/python3.6/site-packages/sqlalchemy/engine/base.py in _execute_context(sel
   1181                         parameters,
-> 1182                         context)
   1183         except BaseException as e:


/opt/conda/lib/python3.6/site-packages/sqlalchemy/engine/default.py in do_execute(self,
    469     def do_execute(self, cursor, statement, parameters, context=None):
--> 470         cursor.execute(statement, parameters)
    471


IntegrityError: duplicate key value violates unique constraint "dimdate_pkey"
DETAIL:  Key (date_key)=(20170407) already exists.



The above exception was the direct cause of the following exception:


IntegrityError                            Traceback (most recent call last)

<ipython-input-19-f4af0de3925d> in <module>()
----> 1 get_ipython().run_cell_magic('sql', '', "INSERT INTO dimDate (date_key, date, year,


/opt/conda/lib/python3.6/site-packages/IPython/core/interactiveshell.py in run_cell_magi
   2165             magic_arg_s = self.var_expand(line, stack_depth)
   2166             with self.builtin_trap:
-> 2167                 result = fn(magic_arg_s, cell)
   2168             return result
   2169


<decorator-gen-126> in execute(self, line, cell, local_ns)
```

```
    /opt/conda/lib/python3.6/site-packages/IPython/core/magic.py in <lambda>(f, *a, **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):


    <decorator-gen-125> in execute(self, line, cell, local_ns)


    /opt/conda/lib/python3.6/site-packages/IPython/core/magic.py in <lambda>(f, *a, **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):


    /opt/conda/lib/python3.6/site-packages/sql/magic.py in execute(self, line, cell, local_n
     93
     94         try:
---> 95             result = sql.run.run(conn, parsed['sql'], self, user_ns)
     96
     97             if result is not None and not isinstance(result, str) and self.column_lo


    /opt/conda/lib/python3.6/site-packages/sql/run.py in run(conn, sql, config, user_namespa
    338             else:
    339                 txt = sqlalchemy.sql.text(statement)
--> 340                 result = conn.session.execute(txt, user_namespace)
    341             _commit(conn=conn, config=config)
    342             if result and config.feedback:


    /opt/conda/lib/python3.6/site-packages/sqlalchemy/engine/base.py in execute(self, object
    943             raise exc.ObjectNotExecutableError(object)
    944         else:
--> 945             return meth(self, multiparams, params)
    946
    947     def _execute_function(self, func, multiparams, params):


    /opt/conda/lib/python3.6/site-packages/sqlalchemy/sql/elements.py in _execute_on_connect
    261     def _execute_on_connection(self, connection, multiparams, params):
    262         if self.supports_execution:
```

```
--> 263             return connection._execute_clauseelement(self, multiparams, params)
    264         else:
    265             raise exc.ObjectNotExecutableError(self)


    /opt/conda/lib/python3.6/site-packages/sqlalchemy/engine/base.py in _execute_clauseeleme
   1051             compiled_sql,
   1052             distilled_params,
-> 1053             compiled_sql, distilled_params
   1054         )
   1055         if self._has_events or self.engine._has_events:


    /opt/conda/lib/python3.6/site-packages/sqlalchemy/engine/base.py in _execute_context(sel
   1187                 parameters,
   1188                 cursor,
-> 1189                 context)
   1190
   1191         if self._has_events or self.engine._has_events:


    /opt/conda/lib/python3.6/site-packages/sqlalchemy/engine/base.py in _handle_dbapi_except
   1400                 util.raise_from_cause(
   1401                     sqlalchemy_exception,
-> 1402                     exc_info
   1403                 )
   1404             else:


    /opt/conda/lib/python3.6/site-packages/sqlalchemy/util/compat.py in raise_from_cause(exc
    201     exc_type, exc_value, exc_tb = exc_info
    202     cause = exc_value if exc_value is not exception else None
--> 203     reraise(type(exception), exception, tb=exc_tb, cause=cause)
    204
    205 if py3k:


    /opt/conda/lib/python3.6/site-packages/sqlalchemy/util/compat.py in reraise(tp, value, t
    184             value.__cause__ = cause
    185         if value.__traceback__ is not tb:
--> 186             raise value.with_traceback(tb)
    187         raise value
    188


    /opt/conda/lib/python3.6/site-packages/sqlalchemy/engine/base.py in _execute_context(sel
   1180                         statement,
   1181                         parameters,
```

```
-> 1182                         context)
   1183         except BaseException as e:
   1184             self._handle_dbapi_exception(


    /opt/conda/lib/python3.6/site-packages/sqlalchemy/engine/default.py in do_execute(self,
    468
    469     def do_execute(self, cursor, statement, parameters, context=None):
--> 470         cursor.execute(statement, parameters)
    471
    472     def do_execute_no_params(self, cursor, statement, context=None):


    IntegrityError: (psycopg2.IntegrityError) duplicate key value violates unique constraint
DETAIL:  Key (date_key)=(20170407) already exists.
 [SQL: "INSERT INTO dimDate (date_key, date, year, quarter, month, day, week, is_weekend)\nS
```

9 STEP 6: Repeat the computation from the facts & dimension table

9.1 6.1 Facts Table has all the needed dimensions, no need for deep joins

In [ ]:
```
%%time
%%sql
SELECT movie_key, date_key, customer_key, sales_amount
FROM factSales
limit 5;
```

9.2 6.2 Join fact table with dimensions to replace keys with attributes

In [ ]:
```
%%time
%%sql
SELECT dimMovie.title, dimDate.month, dimCustomer.city, sales_amount
FROM factSales
JOIN dimMovie on (dimMovie.movie_key = factSales.movie_key)
JOIN dimDate on (dimDate.date_key = factSales.date_key)
JOIN dimCustomer on (dimCustomer.customer_key = factSales.customer_key)
limit 5;
```

In [ ]:
```
%%time
%%sql
SELECT dimMovie.title, dimDate.month, dimCustomer.city, sum(sales_amount) as revenue
FROM factSales
JOIN dimMovie    on (dimMovie.movie_key      = factSales.movie_key)
JOIN dimDate     on (dimDate.date_key         = factSales.date_key)
JOIN dimCustomer on (dimCustomer.customer_key = factSales.customer_key)
group by (dimMovie.title, dimDate.month, dimCustomer.city)
order by dimMovie.title, dimDate.month, dimCustomer.city, revenue desc;
```

```
In [ ]: %%time
        %%sql
        SELECT f.title, EXTRACT(month FROM p.payment_date) as month, ci.city, sum(p.amount) as r
        FROM payment p
        JOIN rental r    ON ( p.rental_id = r.rental_id )
        JOIN inventory i ON ( r.inventory_id = i.inventory_id )
        JOIN film f ON ( i.film_id = f.film_id)
        JOIN customer c  ON ( p.customer_id = c.customer_id )
        JOIN address a ON ( c.address_id = a.address_id )
        JOIN city ci ON ( a.city_id = ci.city_id )
        group by (f.title, month, ci.city)
        order by f.title, month, ci.city, revenue desc;
```

10 Conclusion

- We were able to show that a start schema is easier to understand
- Evidence that is more performant

```
In [ ]: !PGPASSWORD=student pg_dump -h 127.0.0.1 -U student pagila > Data/pagila-star.sql
```