# DVD RENTAL DATABASE

Milestone 7

Group 6
Vikrant Shelke (shelke.v@northeastern.edu)
Prabhat Chanda (chanda.p@northeastern.edu)

Submission Date: 08th December 2023

## Problem Definition:

A DVD rental database is a structured collection of information that stores data related toDVD rentals. It is designed to efficiently manage and organize various aspects of a DVDrental business, such as inventory management, customer information, rental transactions, and more. This type of database enables rental businesses to streamline their operations, track rentals and returns, and provide better customer service.
The database consists of 15 tables that store information related to actors, films, categories, stores, inventory, rentals, payments, staff, customers, addresses, cities, andcountries. The problem at hand involves efficiently managing and extracting valuable insights from this database to address various business needs and challenges.

## Operational Business Context:

- **Inventory Management:** The database manages the store's inventory of DVDs. Ittracks which DVDs are available for rent, and the number of copies of each DVD in stock. This ensures that customers can easily browse and rent DVDs that are instock.
- **Customer Information:** The database stores customer data, including names, contact information, and rental history. This allows the store to maintain a record of its customers, their preferences, and their rental activities.
- **Rental Transactions:** The core function of the database is to record rental transactions. When a customer rents a DVD, the database records the rental details, including the rental date, and rental duration. This information is crucial forbilling and tracking overdue rentals.
- **Payment Processing:** The database handles payment transactions made bycustomers. It records payment details, such as the amount paid, payment date etc..This information is used for financial reporting and tracking customer payments.
- **Staff Management:** Information about store staff is stored in the database. This allows the store to manage its workforce and assign roles and responsibilities.
- **Film Catalog:** The database maintains a comprehensive catalog of available films.It includes details about each film's title, release year, length, rating, and category.This catalog helps customers browse and select films for rental.
- **Relationships:** The database establishes relationships between films, actors, and categories. For example, it tracks which actors are associated with each film and which categories each film belongs to. This helps customers discover films basedon their favorite actors or genres.

## Analysis in Future:

**Customer Behavior Analysis:**

- Analyze which DVDs are most frequently rented and when.
- Understand the age, location, and preferences of customers.
- Examine how long customers typically keep DVDs.

**Inventory Management:**

- Monitor inventory to prevent stockouts or overstock situations.
- Identify which genres of DVDs are in high demand.

**Financial Analysis:**

- Track monthly and yearly revenue to identify peak seasons.

**DVD Preferences:**

- Identify highly rated films and analyze their rental frequency.
- Determine if certain directors or actors influence rental choices.

**Geographic Analysis:**

- Analyze rental preferences by city or region.

**Time-Series Analysis:**

- Identify trends and seasonality in DVD rentals over time.

**Operational KPIs:**

- Monitor key performance indicators such as rental turnover rate, average rentalduration, and customer satisfaction.

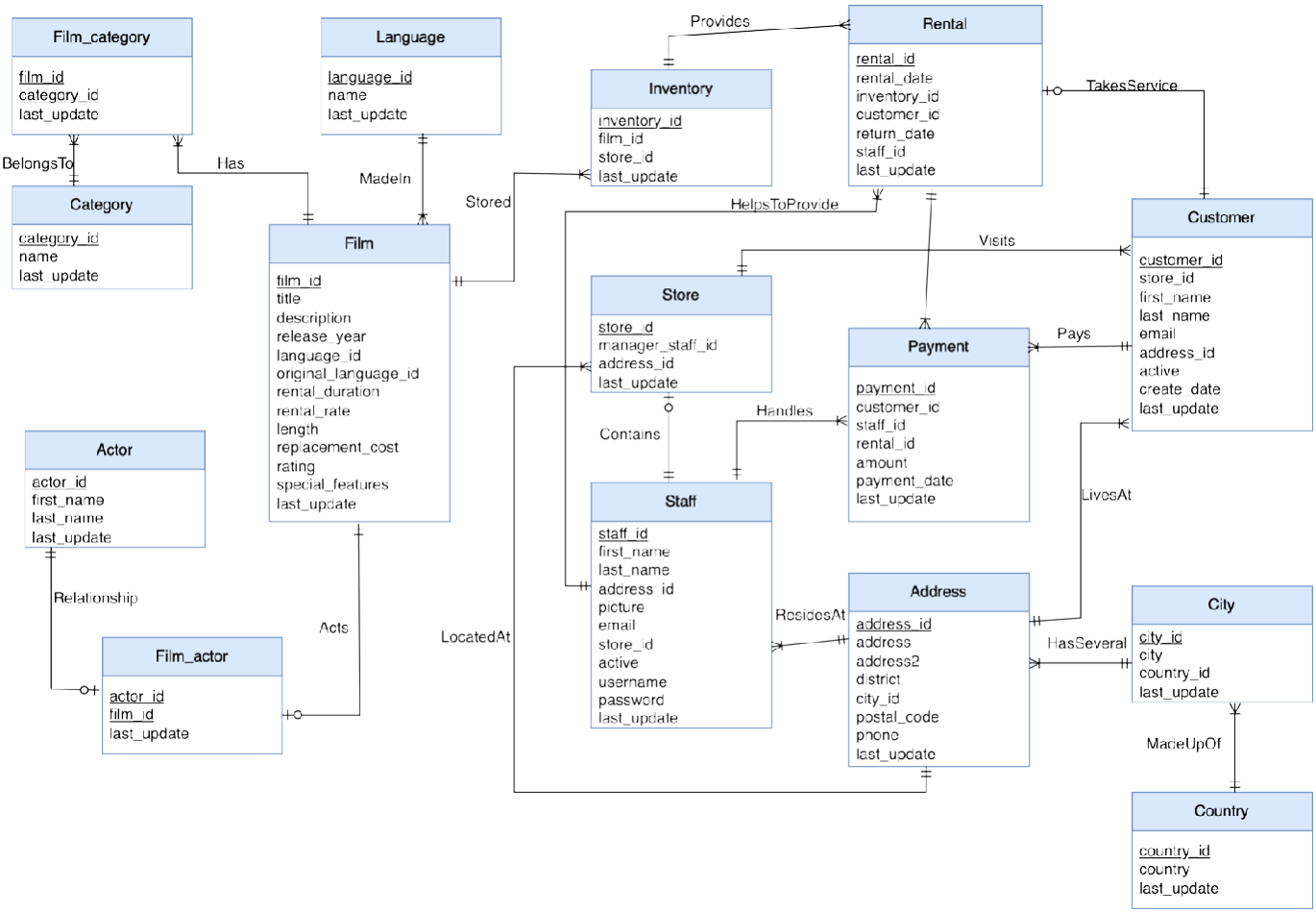
**Inventory Aging:**

- Determine which DVDs are not being rented and consider discounting or retiringthem.

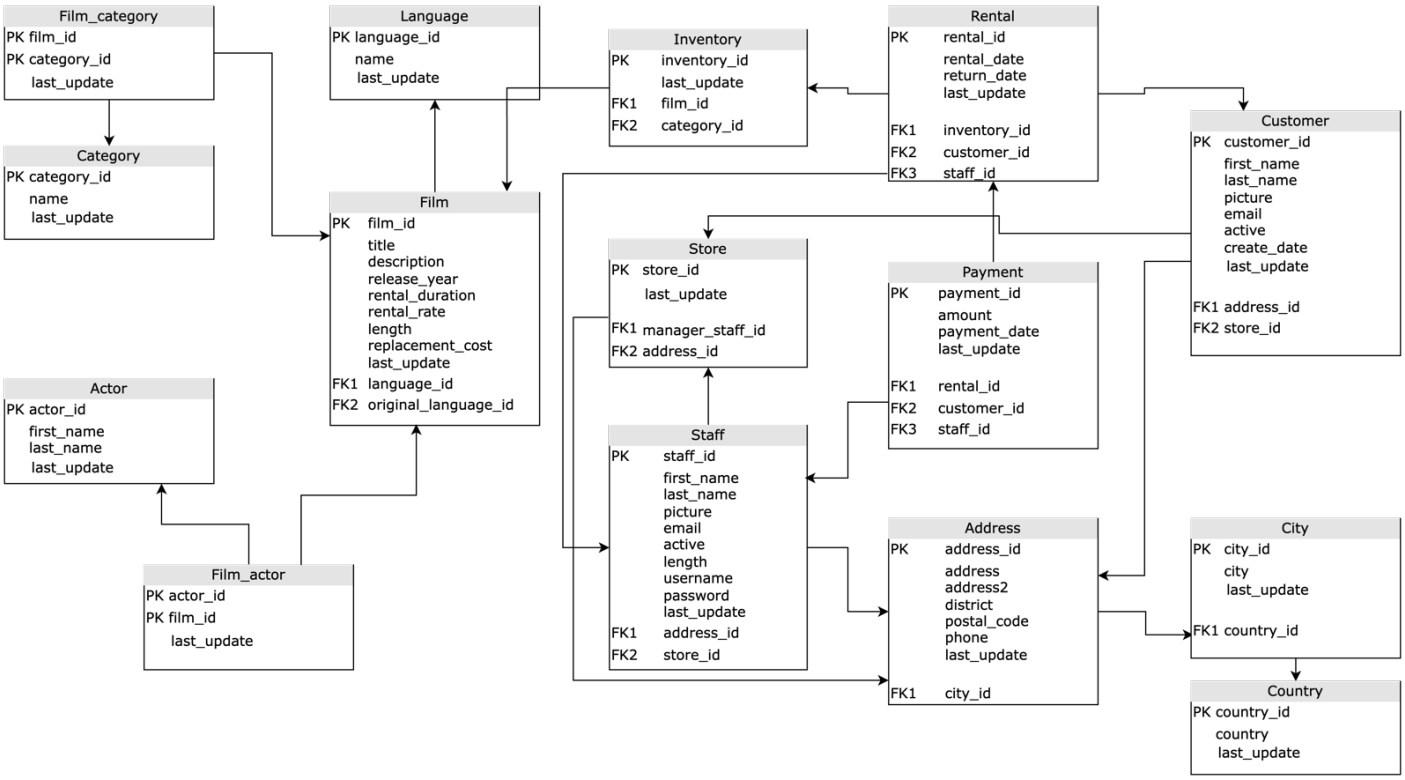
## Data Source:

https://www.kaggle.com/datasets/atanaskanev/sqlite-sakila-sample-database/

# Entity Relationship Diagram:

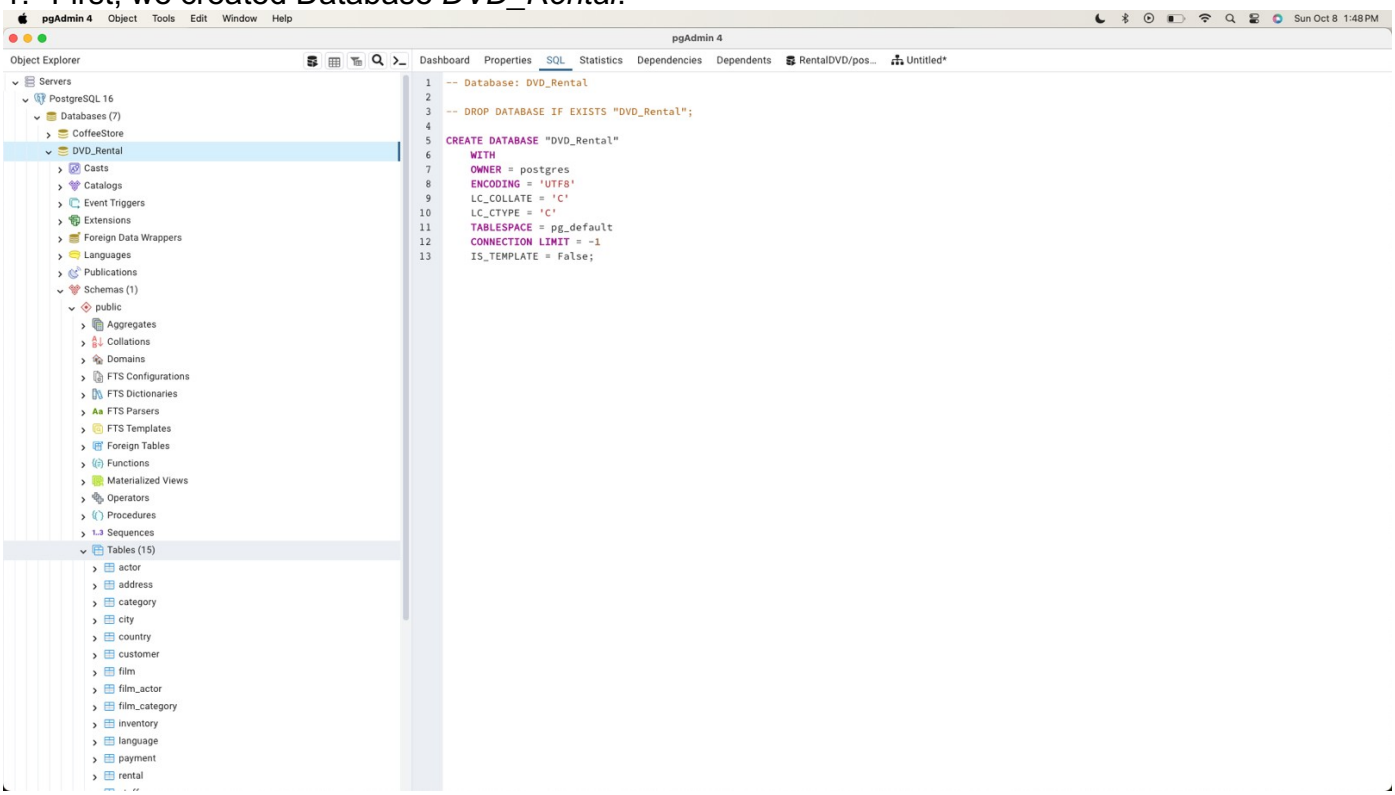


# Relational Model:



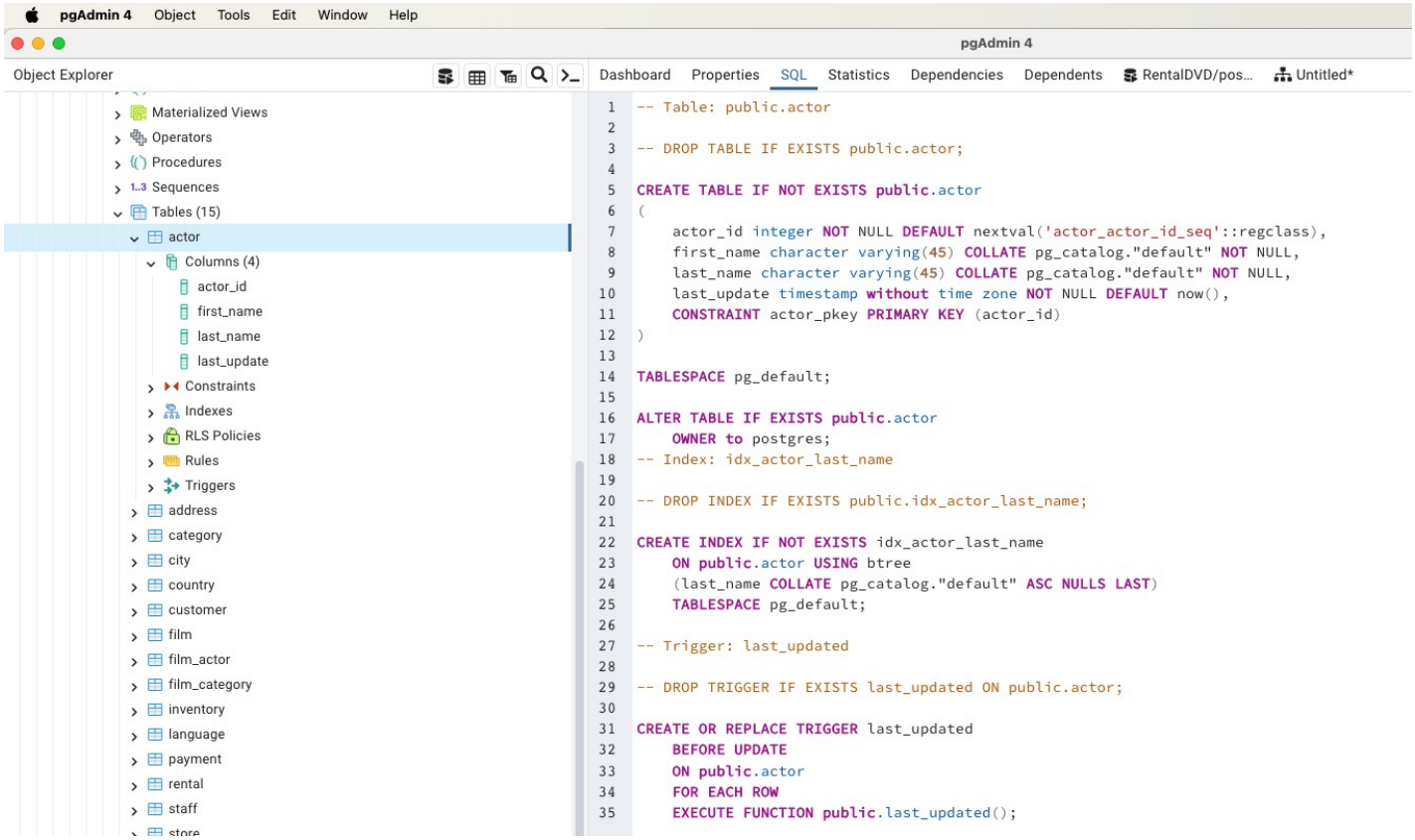

# PostgreSQL Implementation:

1. First, we created Database *DVD_Rental*.

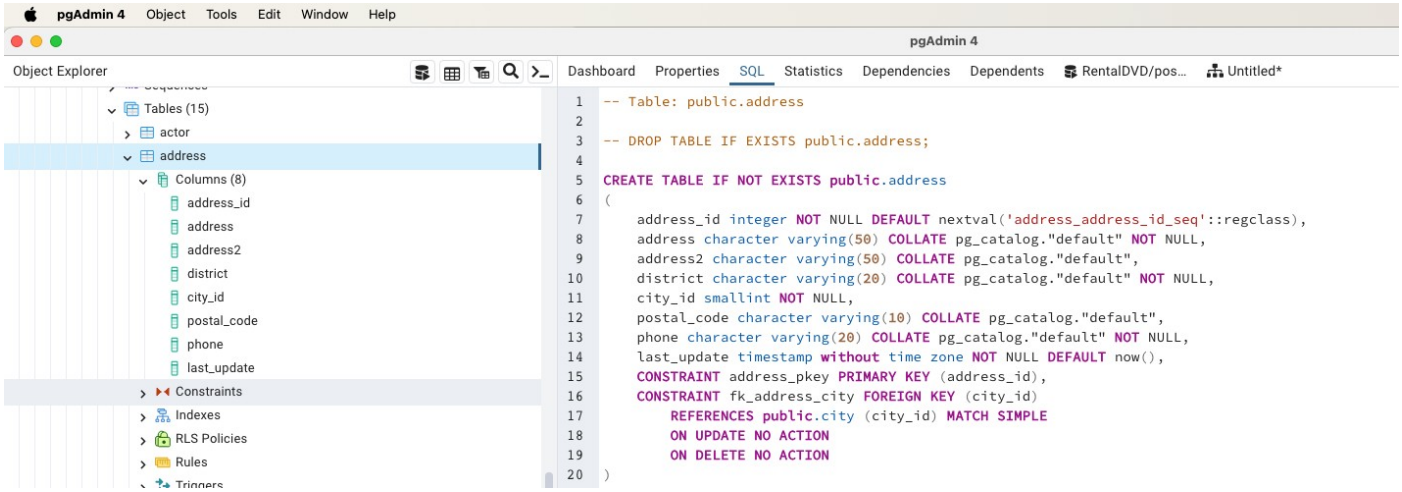


2. Then we created 15 Tables.

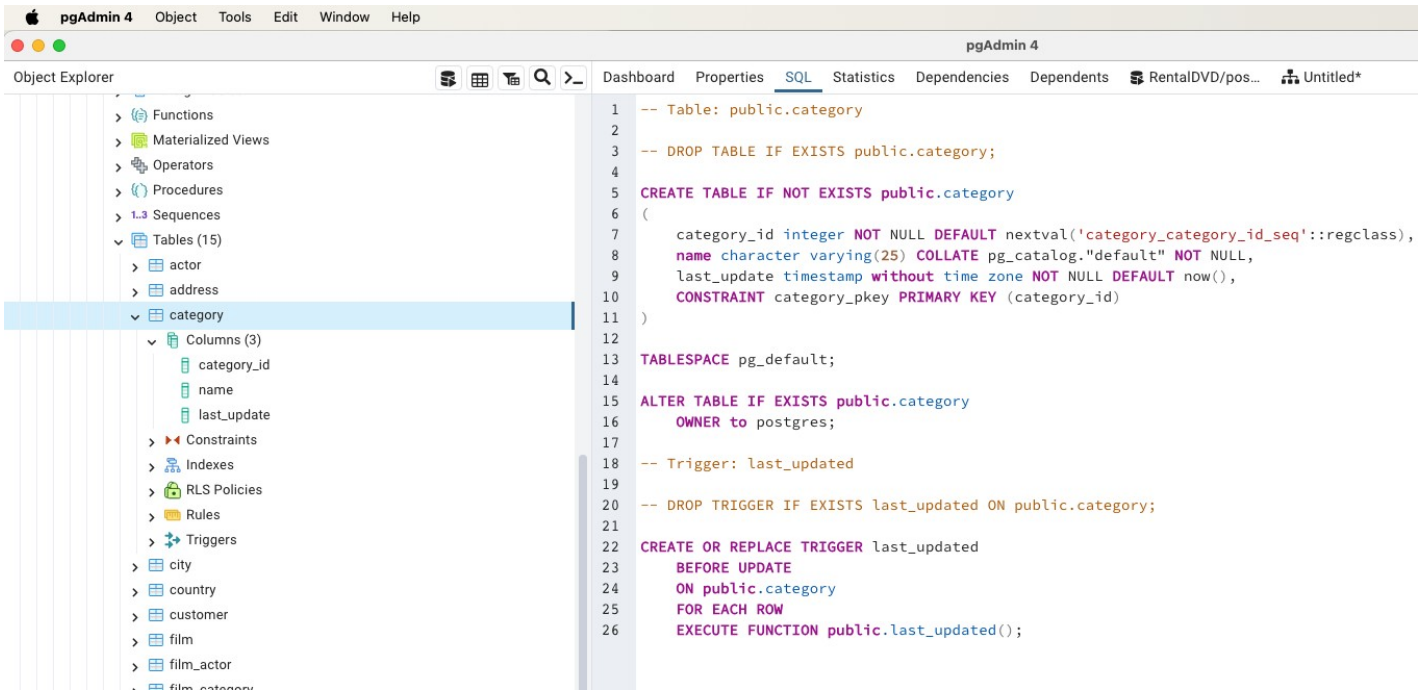
Actor:



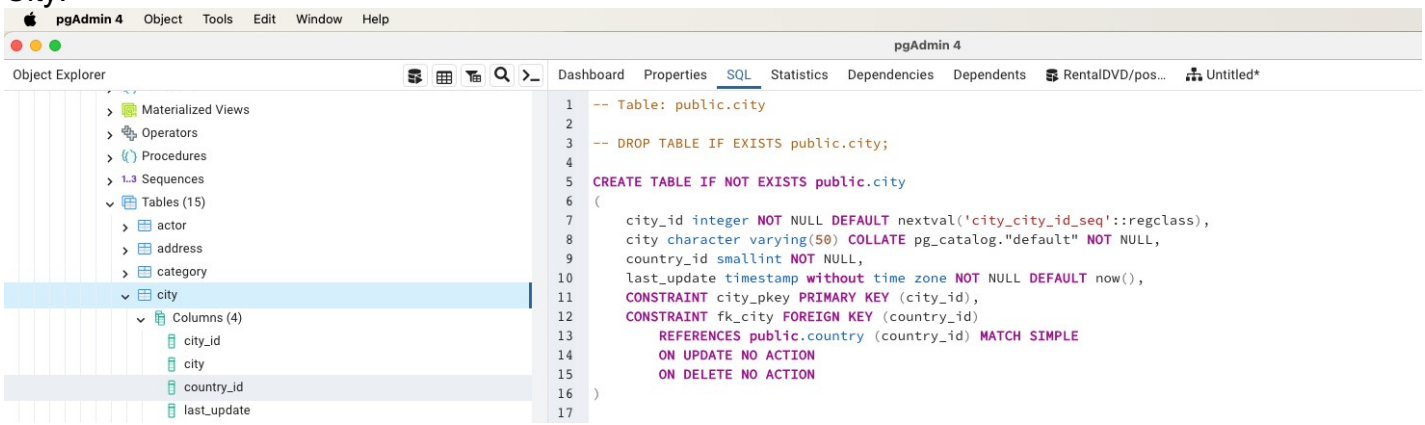

Address:



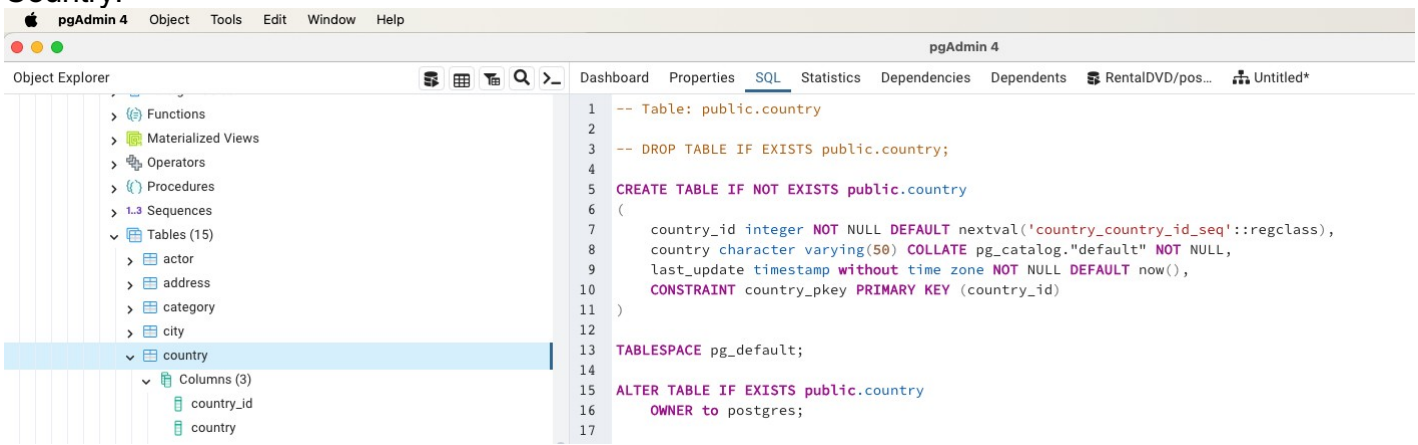

Category:



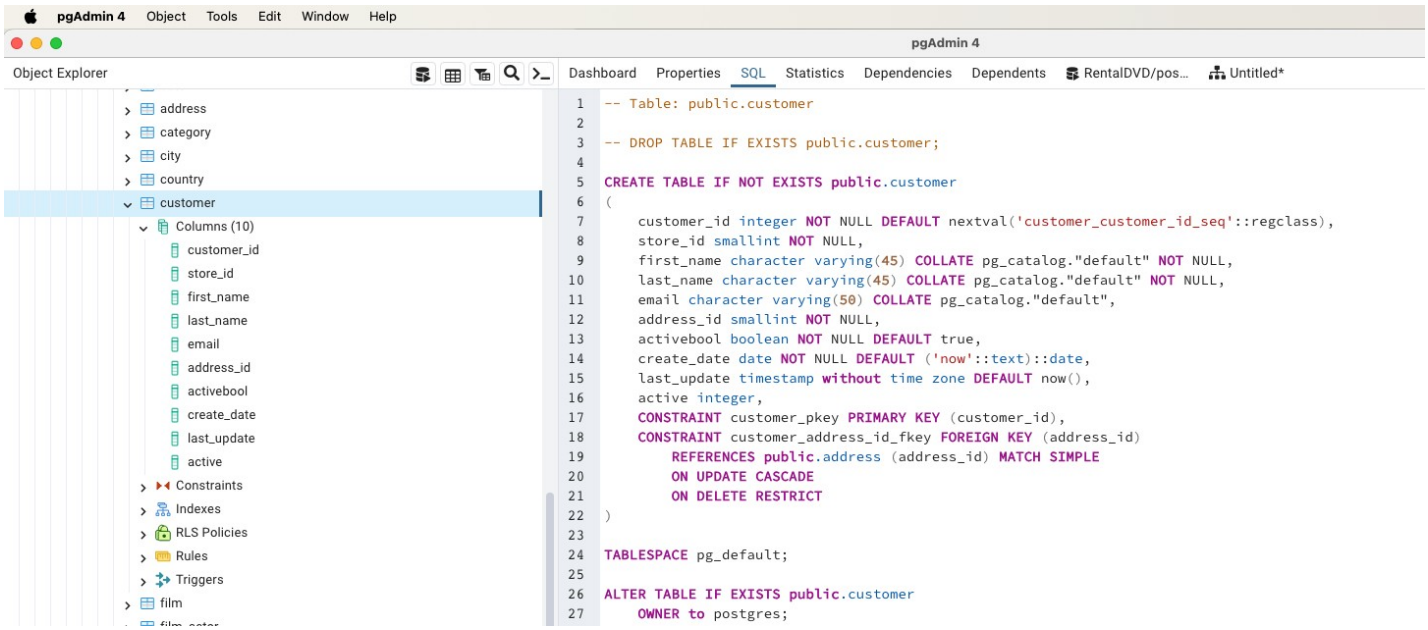

City:



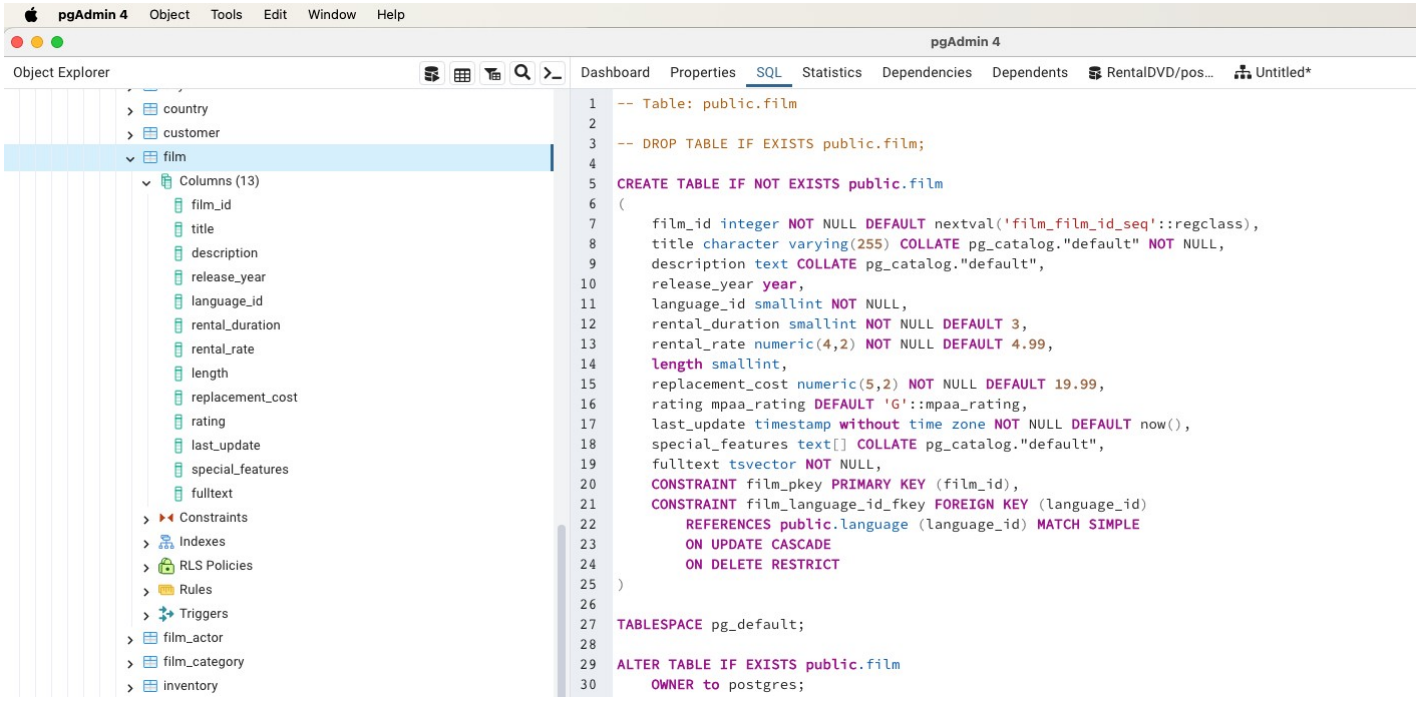

Country:



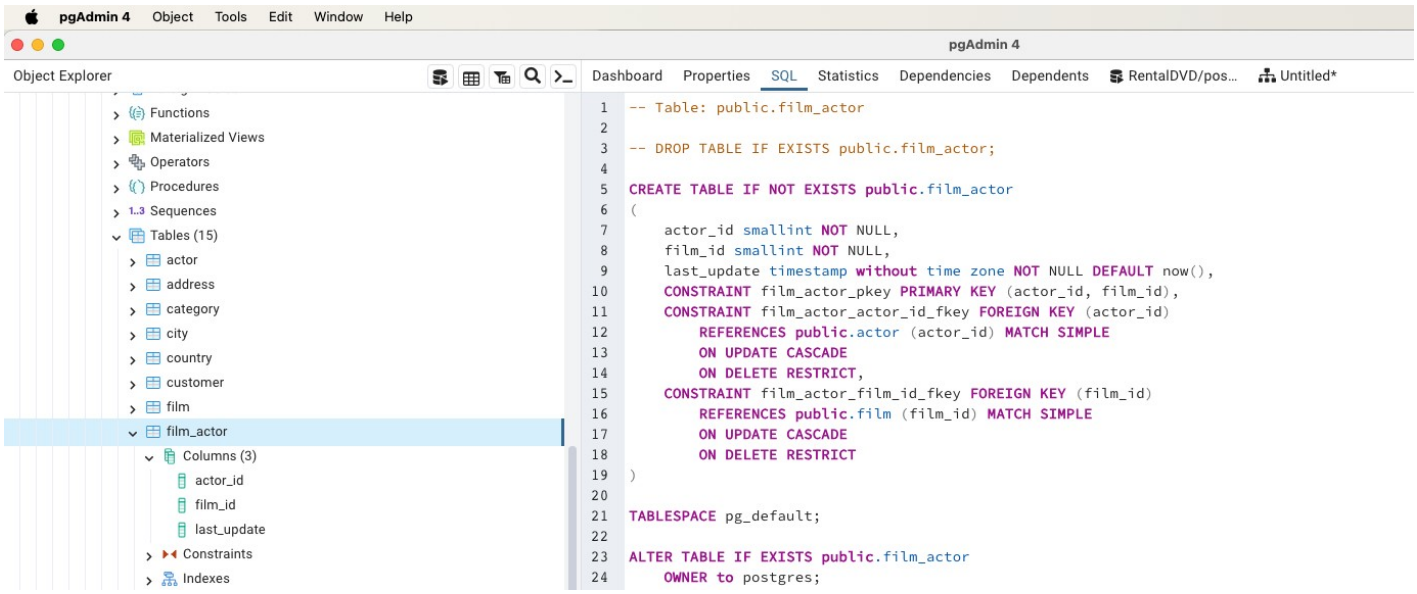

Customer:



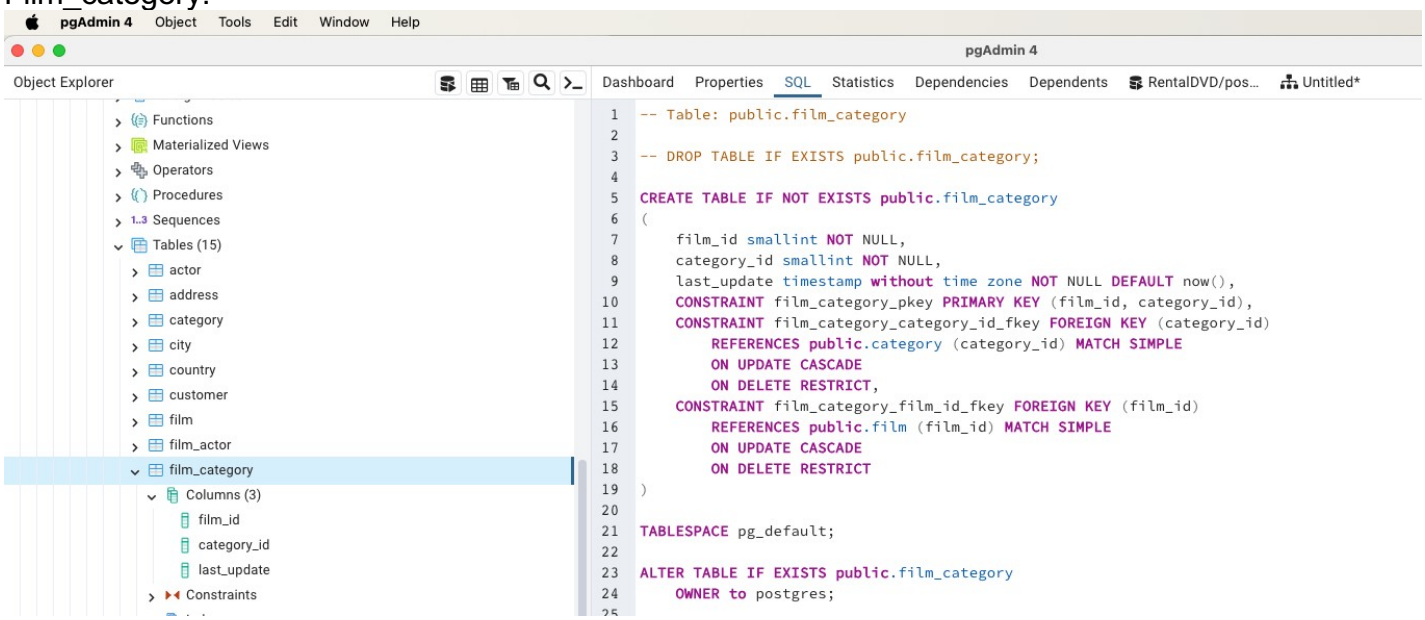

Film:



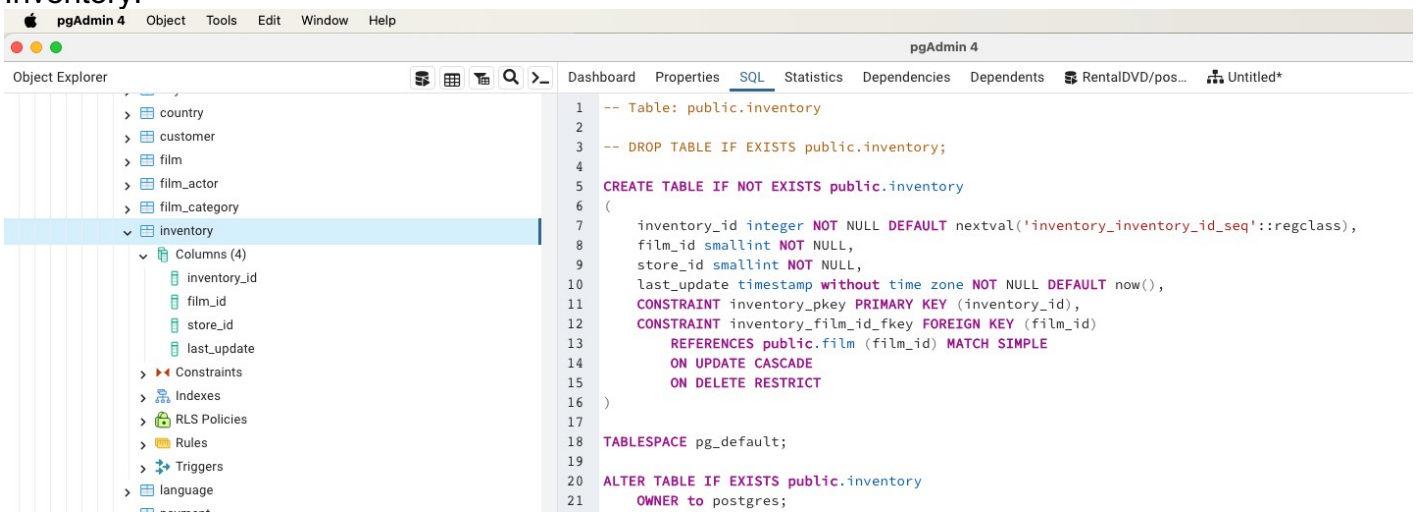

Film_actor:



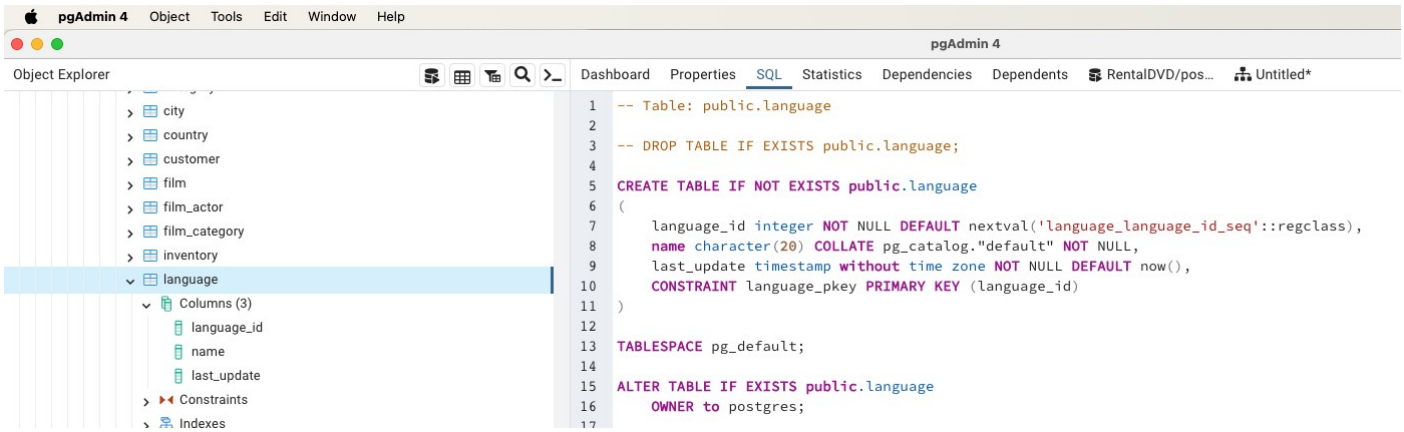

Film_category:



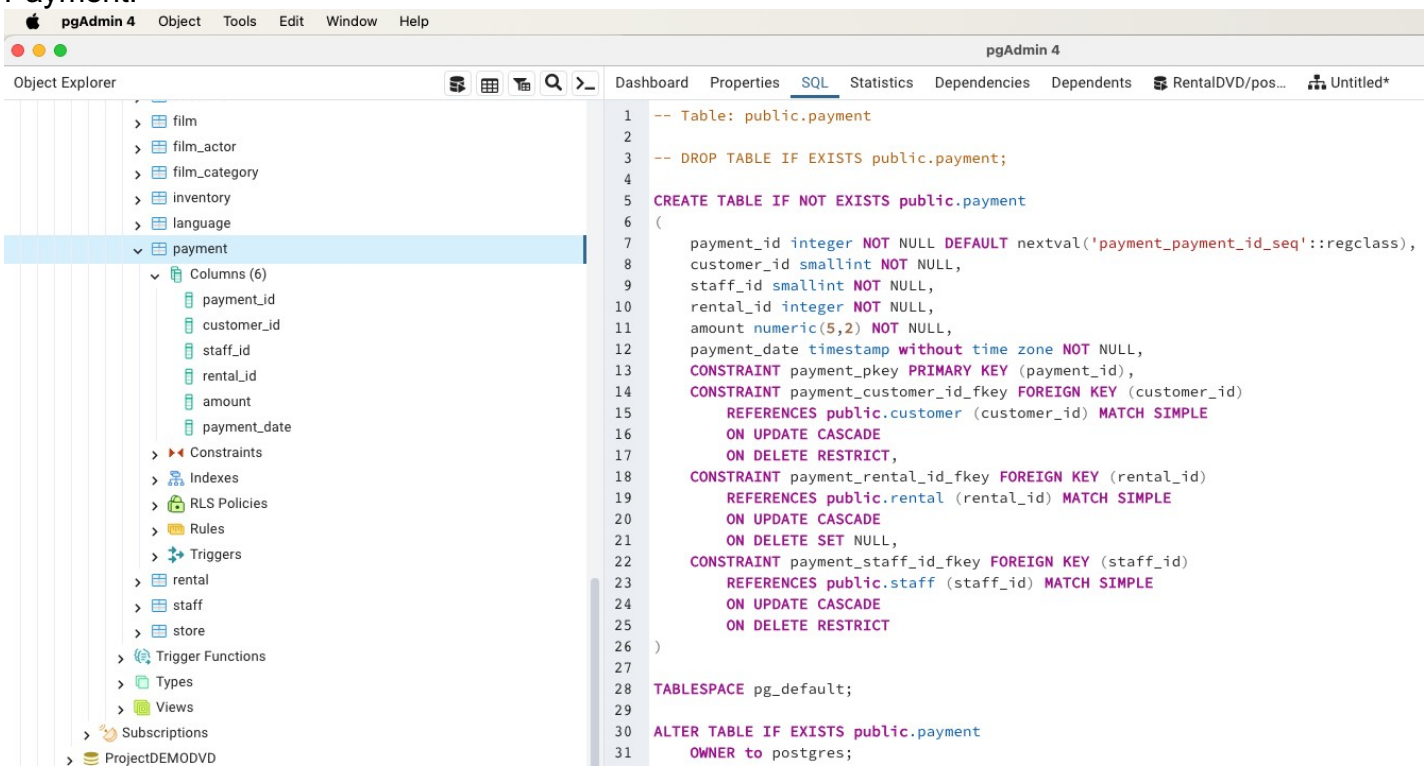

Inventory:



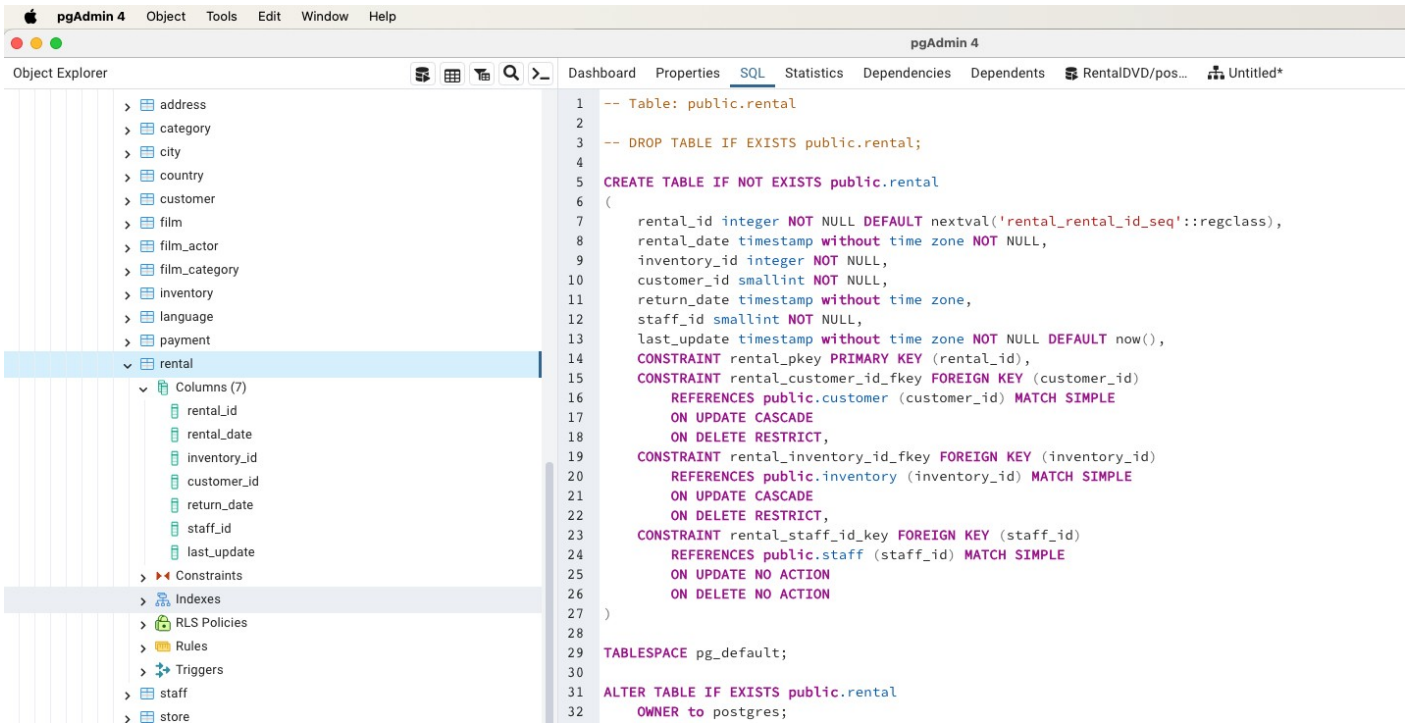

Language:



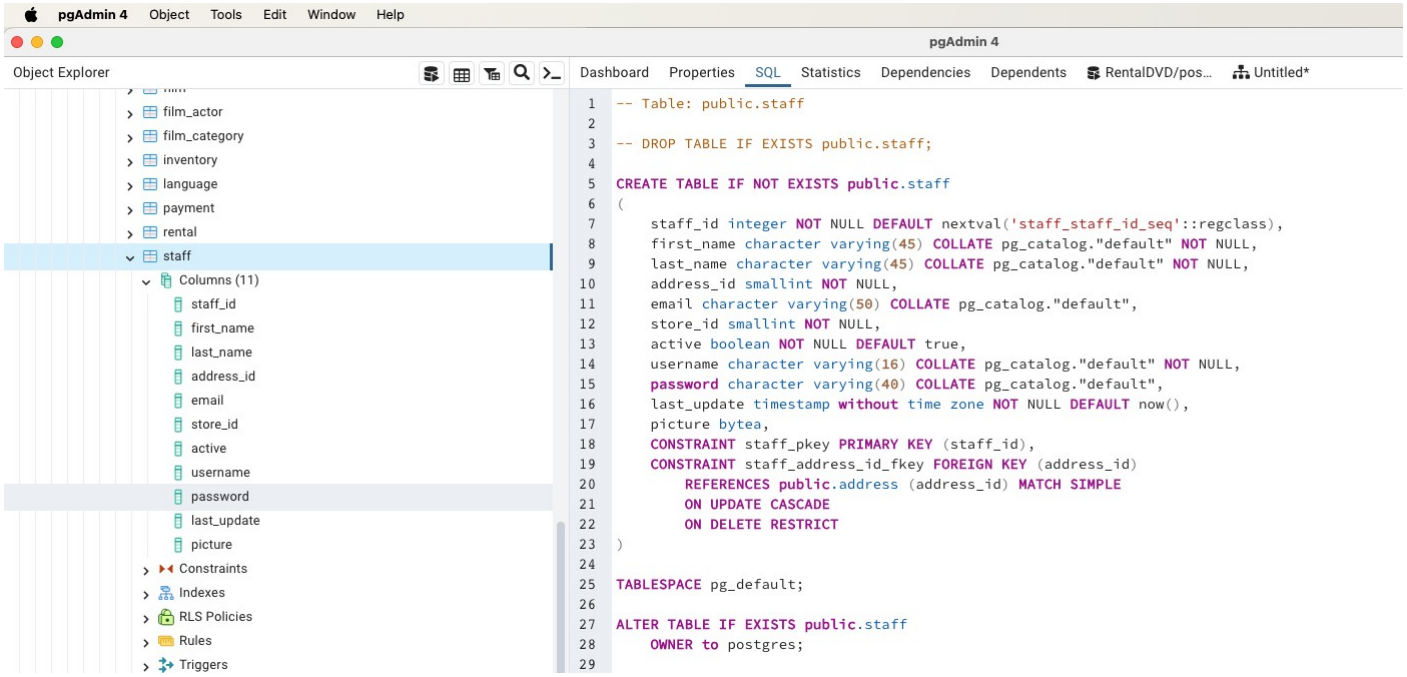

Payment:



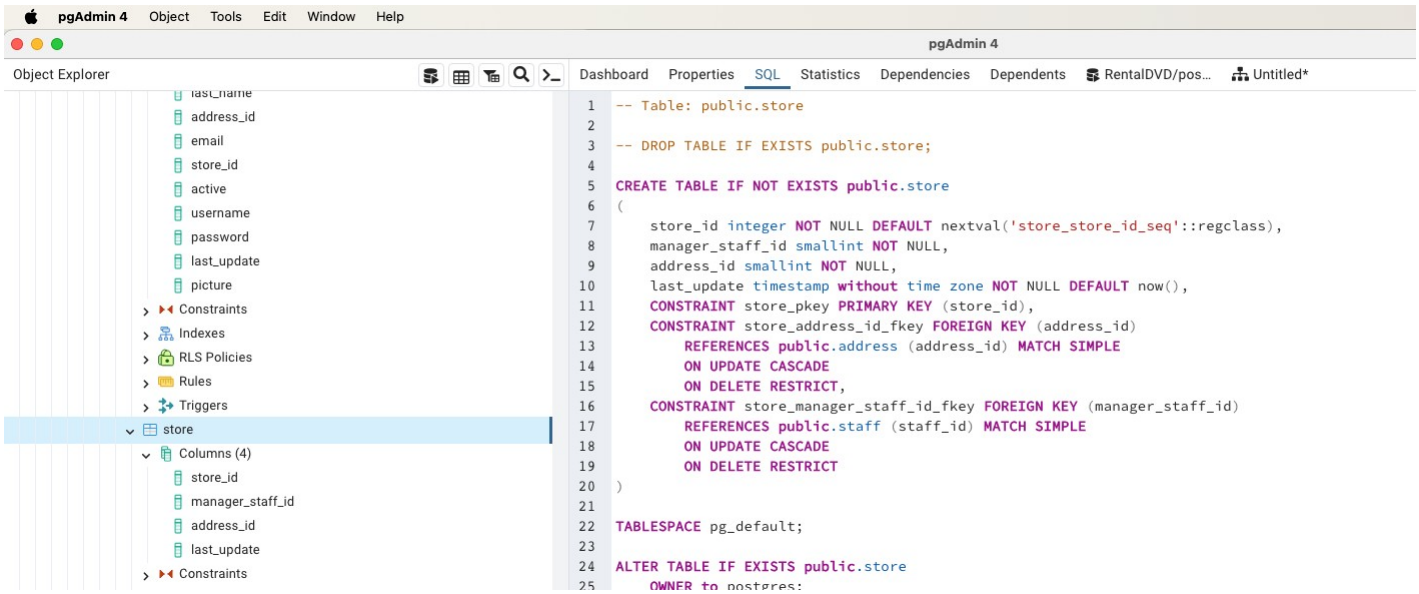

Store:


pgAdmin 4
Object Tools Edit Window Help
pgAdmin 4
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents RentalDVD/pos... Untitled*
address
category
city
country
customer
film
film_actor
film_category
inventory
language
payment
rental
Columns (7)
rental_id
rental_date
inventory_id
customer_id
return_date
staff_id
last_update
Constraints
Indexes
RLS Policies
Rules
Triggers
staff
store
-- Table: public.rental
-- DROP TABLE IF EXISTS public.rental;
CREATE TABLE IF NOT EXISTS public.rental
(
rental_id integer NOT NULL DEFAULT nextval('rental_rental_id_seq'::regclass),
rental_date timestamp without time zone NOT NULL,
inventory_id integer NOT NULL,
customer_id smallint NOT NULL,
return_date timestamp without time zone,
staff_id smallint NOT NULL,
last_update timestamp without time zone NOT NULL DEFAULT now(),
CONSTRAINT rental_pkey PRIMARY KEY (rental_id),
CONSTRAINT rental_customer_id_fkey FOREIGN KEY (customer_id)
REFERENCES public.customer (customer_id) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT,
CONSTRAINT rental_inventory_id_fkey FOREIGN KEY (inventory_id)
REFERENCES public.inventory (inventory_id) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT,
CONSTRAINT rental_staff_id_key FOREIGN KEY (staff_id)
REFERENCES public.staff (staff_id) MATCH SIMPLE
ON UPDATE NO ACTION
ON DELETE NO ACTION
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public.rental
OWNER to postgres;

Staff:


pgAdmin 4
Object Tools Edit Window Help
pgAdmin 4
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents RentalDVD/pos... Untitled*
film_actor
film_category
inventory
language
payment
rental
staff
Columns (11)
staff_id
first_name
last_name
address_id
email
store_id
active
username
password
last_update
picture
Constraints
Indexes
RLS Policies
Rules
Triggers
-- Table: public.staff
-- DROP TABLE IF EXISTS public.staff;
CREATE TABLE IF NOT EXISTS public.staff
(
staff_id integer NOT NULL DEFAULT nextval('staff_staff_id_seq'::regclass),
first_name character varying(45) COLLATE pg_catalog."default" NOT NULL,
last_name character varying(45) COLLATE pg_catalog."default" NOT NULL,
address_id smallint NOT NULL,
email character varying(50) COLLATE pg_catalog."default",
store_id smallint NOT NULL,
active boolean NOT NULL DEFAULT true,
username character varying(16) COLLATE pg_catalog."default" NOT NULL,
password character varying(40) COLLATE pg_catalog."default",
last_update timestamp without time zone NOT NULL DEFAULT now(),
picture bytea,
CONSTRAINT staff_pkey PRIMARY KEY (staff_id),
CONSTRAINT staff_address_id_fkey FOREIGN KEY (address_id)
REFERENCES public.address (address_id) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public.staff
OWNER to postgres;

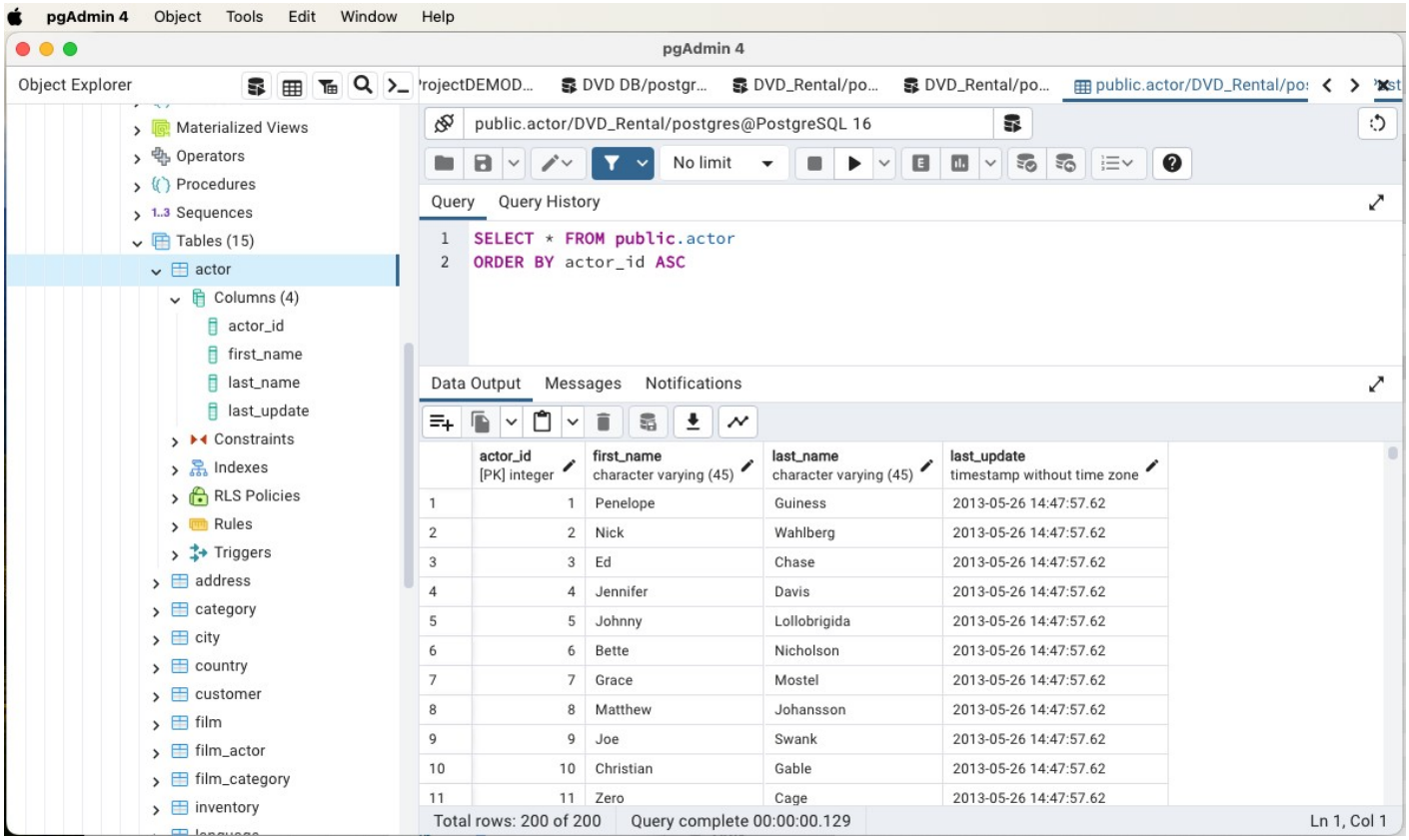
Store:



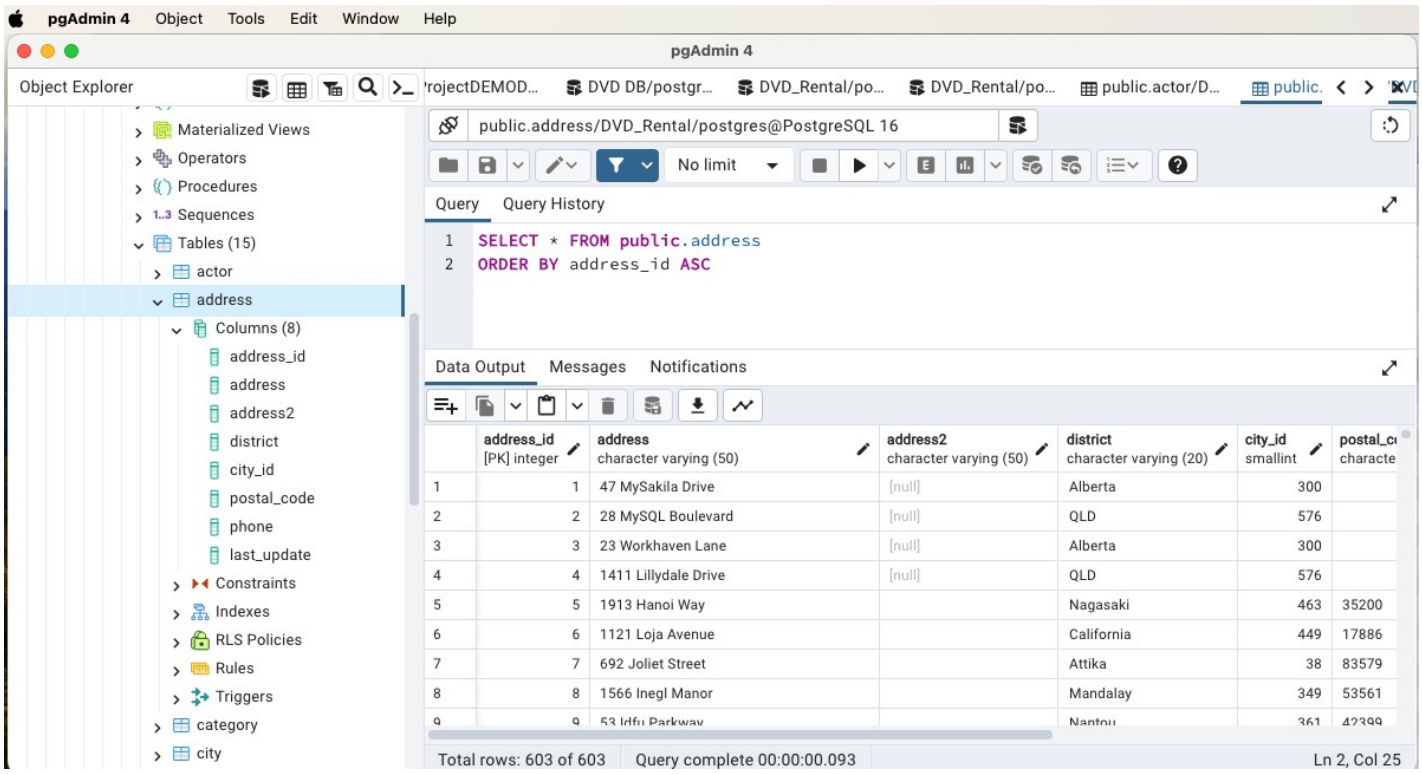

# Data Population Methodology:

Using the insert queries from the data source mentioned above, we have inserted the records into the tables.

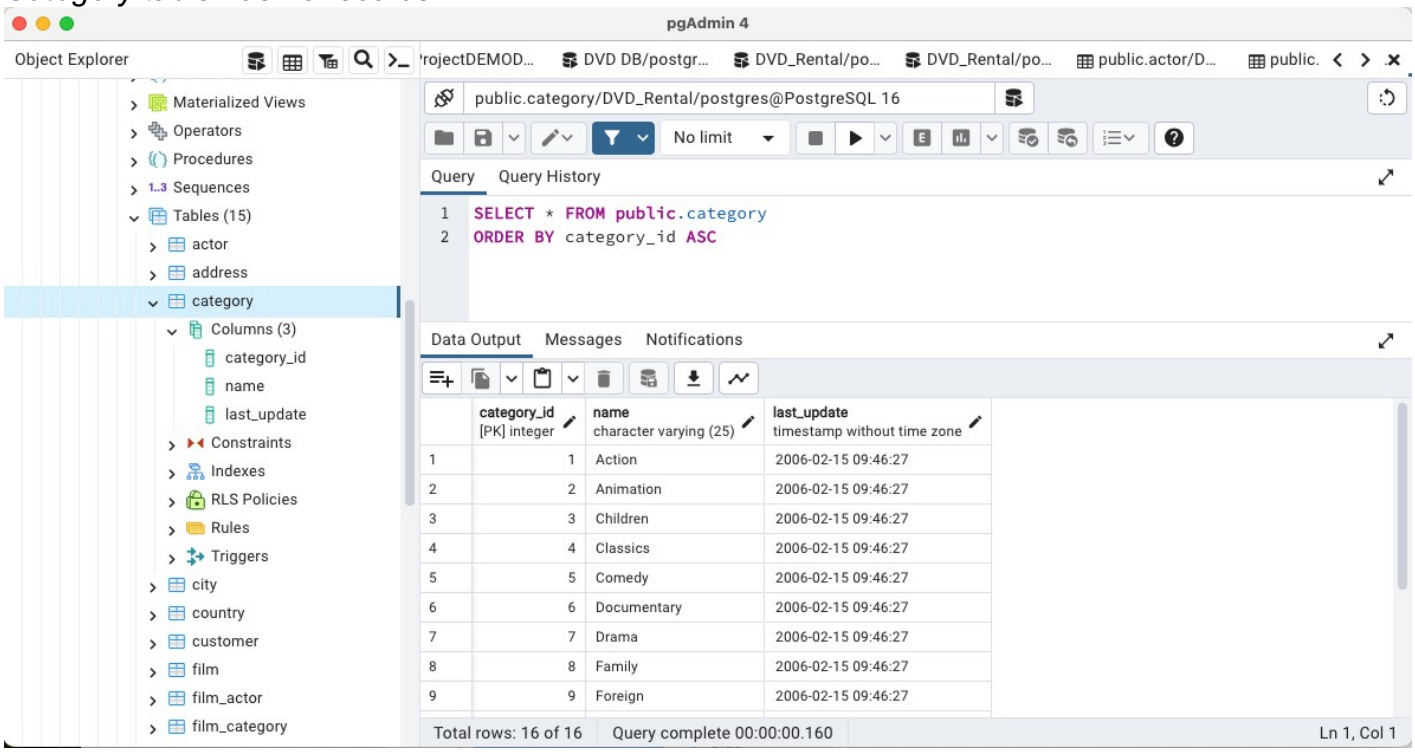
*Actor* table has 200 records.



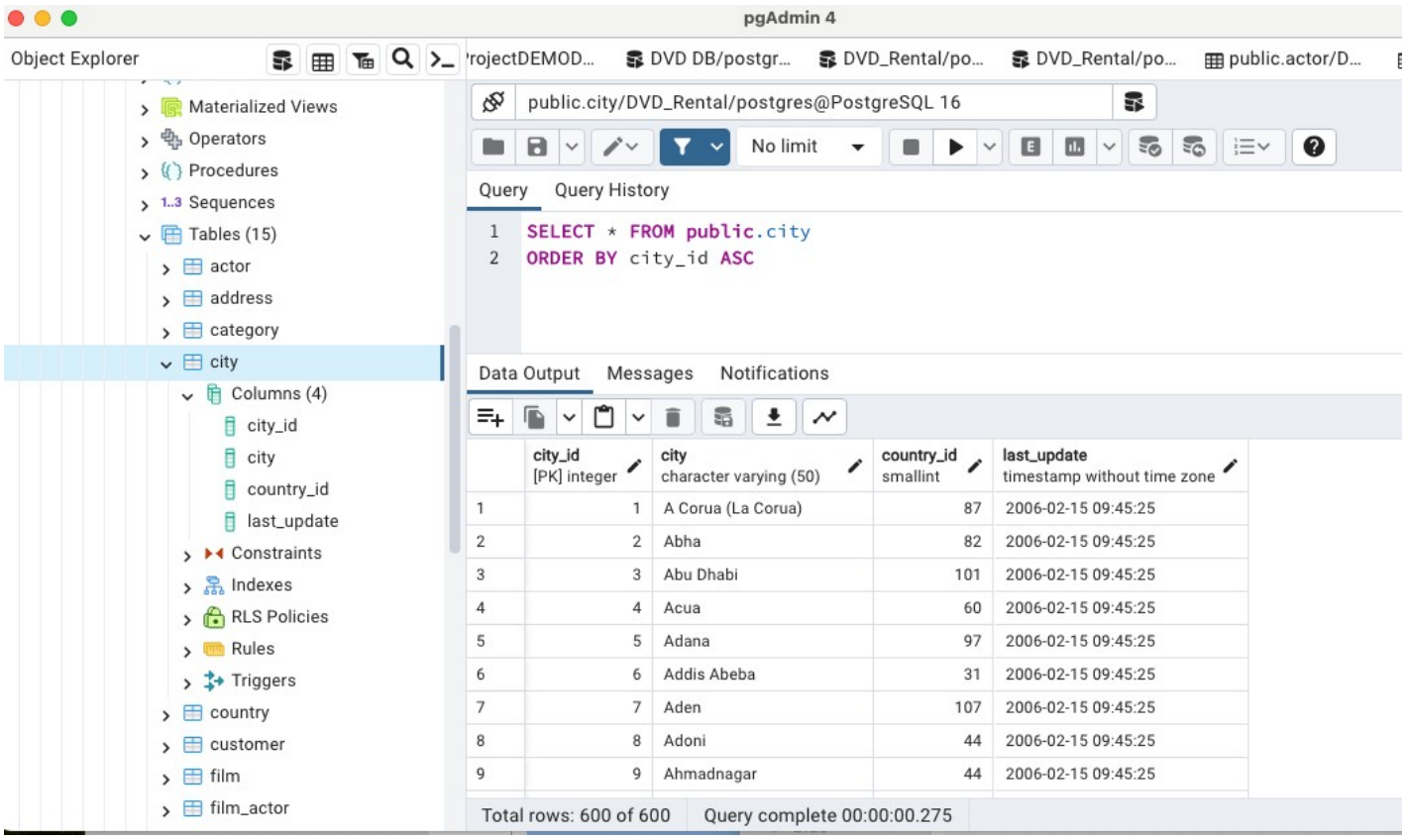

*Address* table has 603 records.



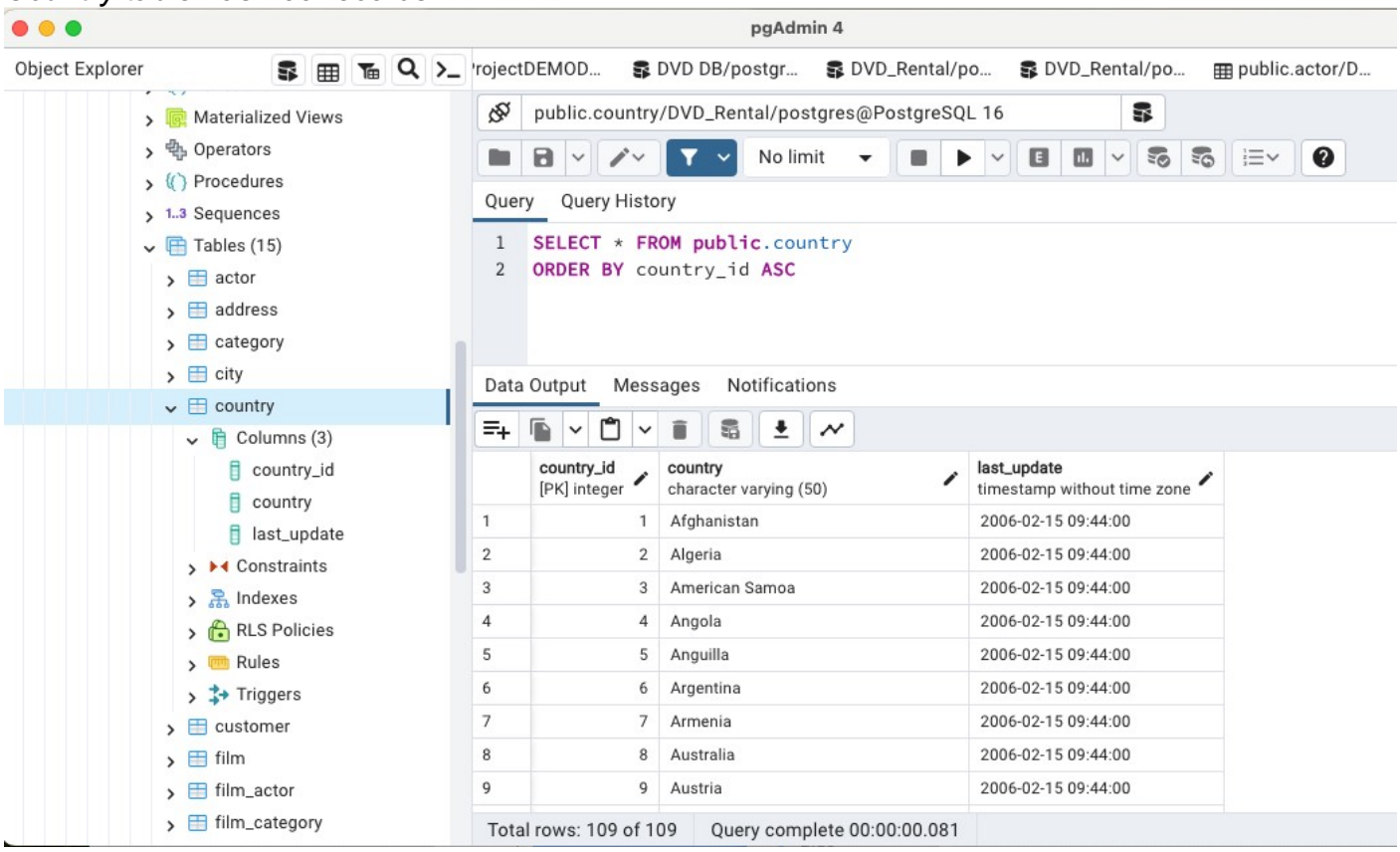

*Category* table has 16 records.



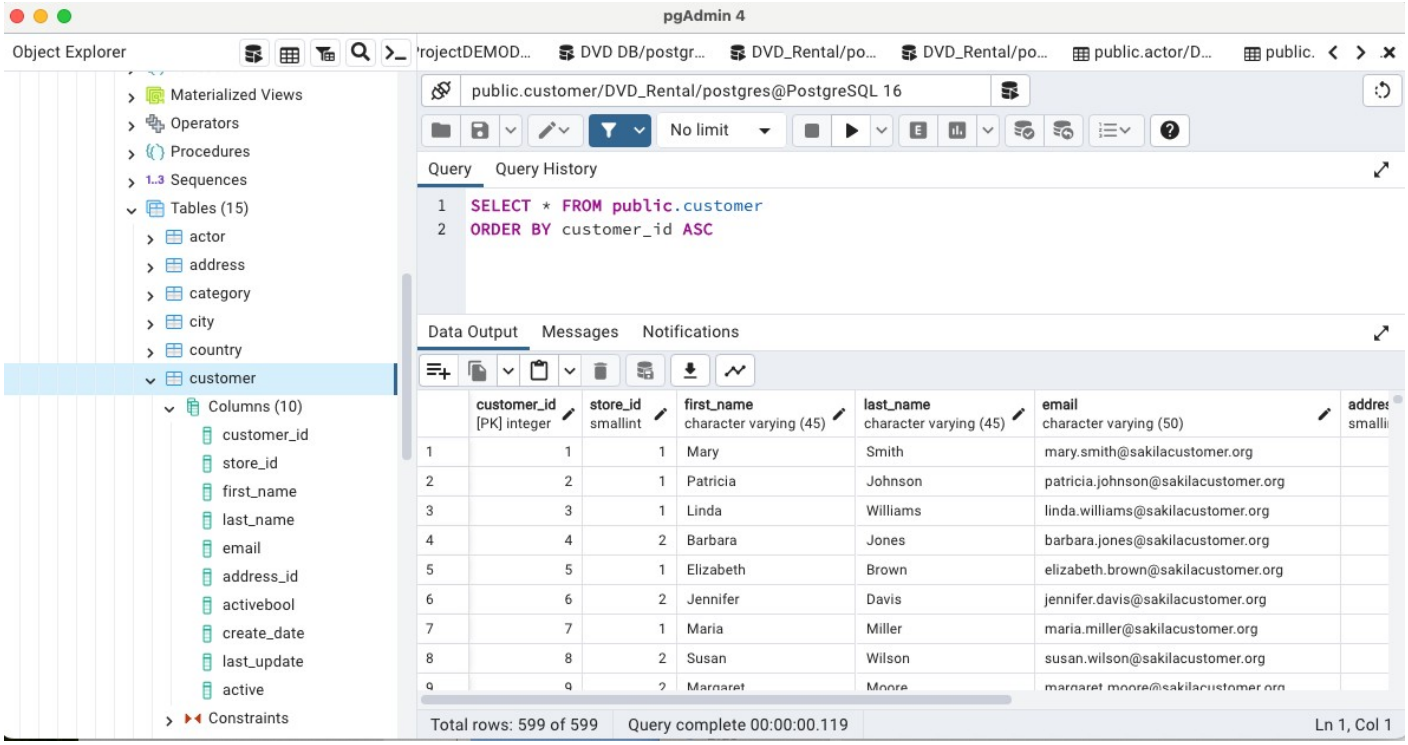

*City* table has 600 records.



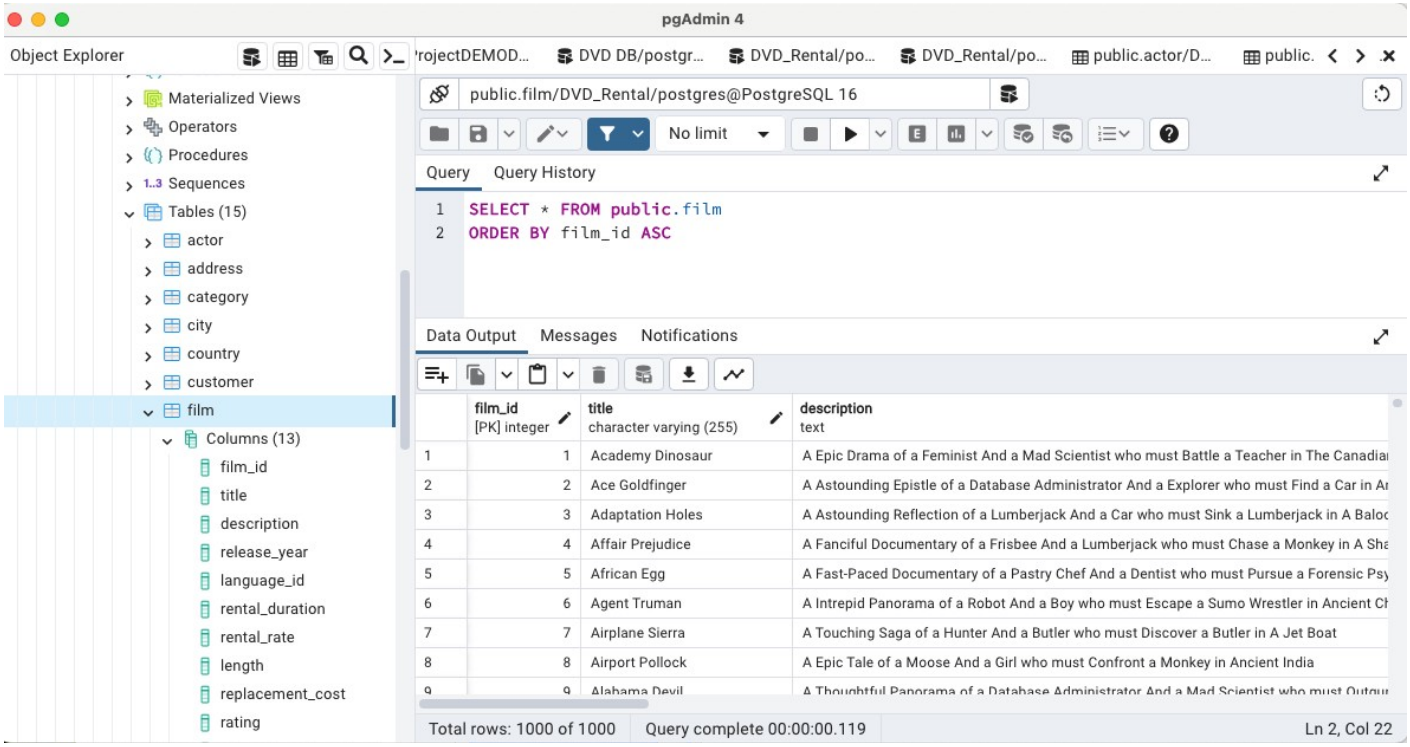

*Country* table has 109 records.



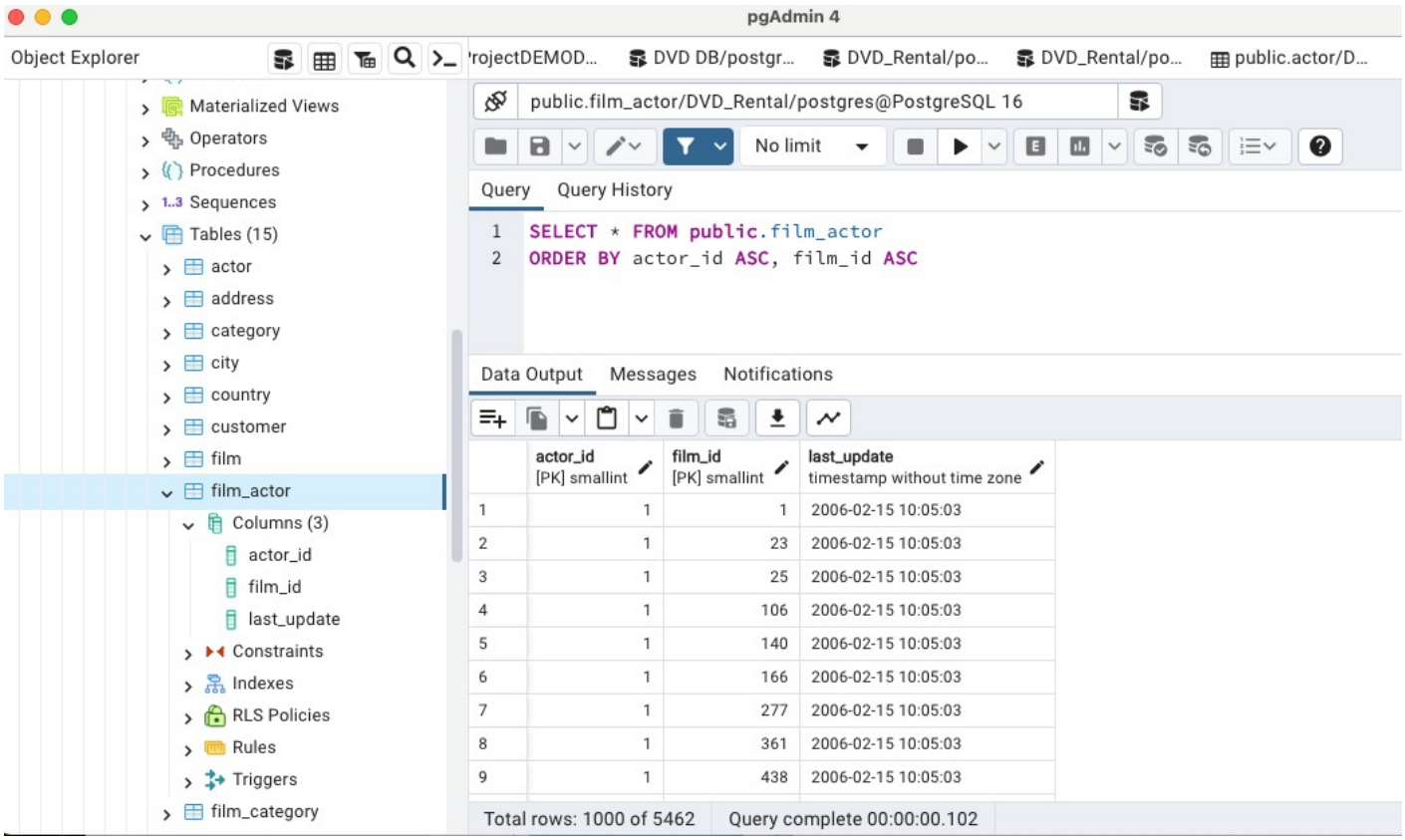

*Customer* table has 599 records.



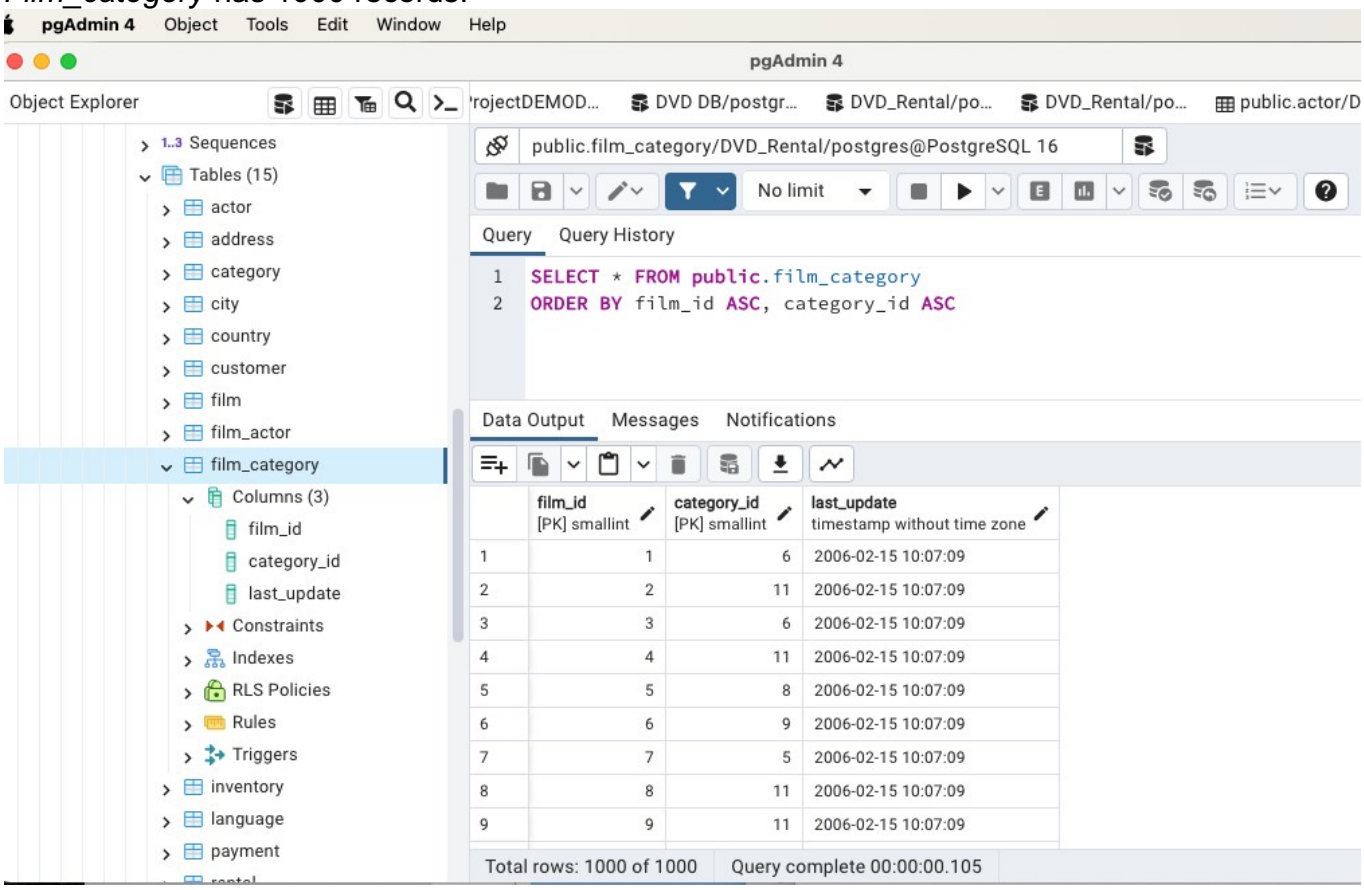

*Film* table has 1000 records.



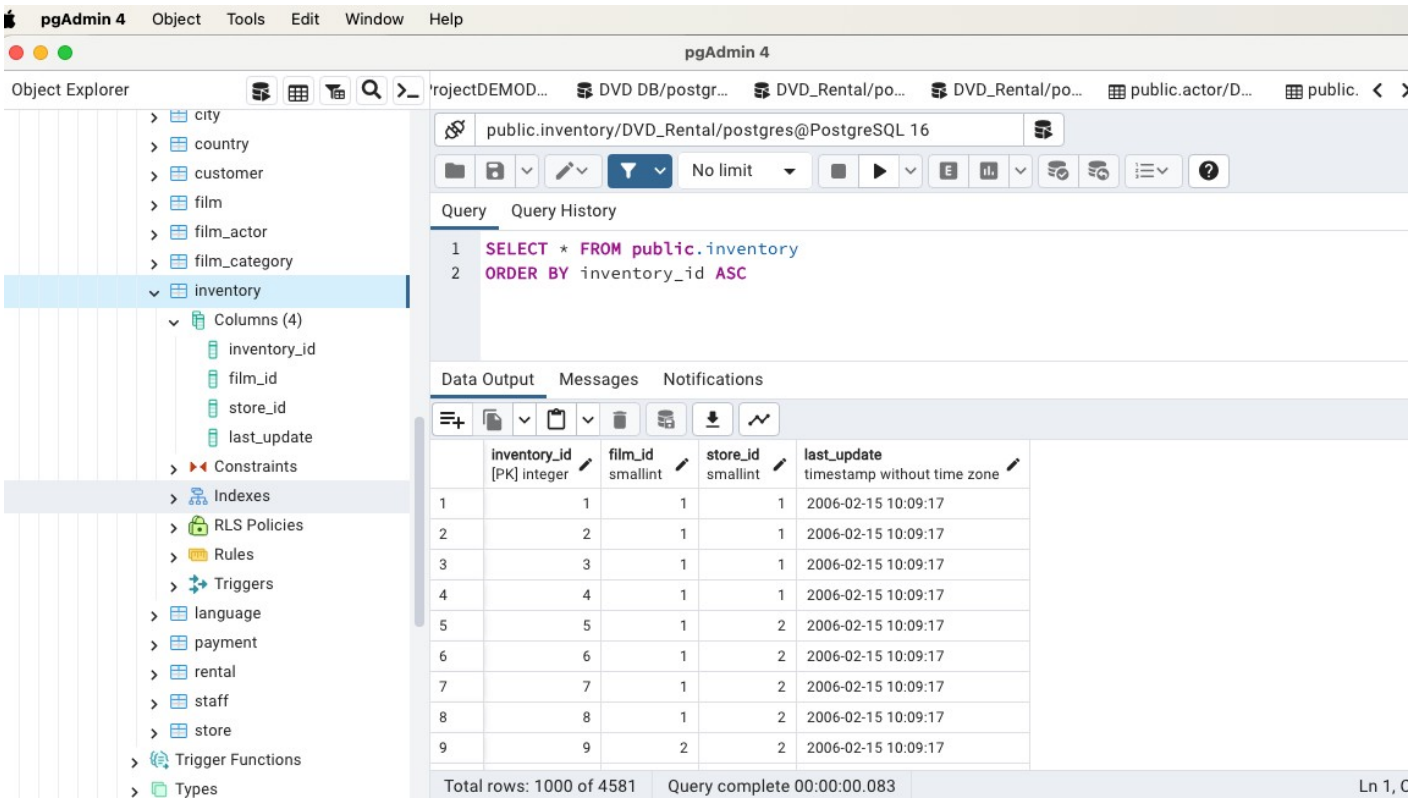

*Film_actor* table has 5462 records.



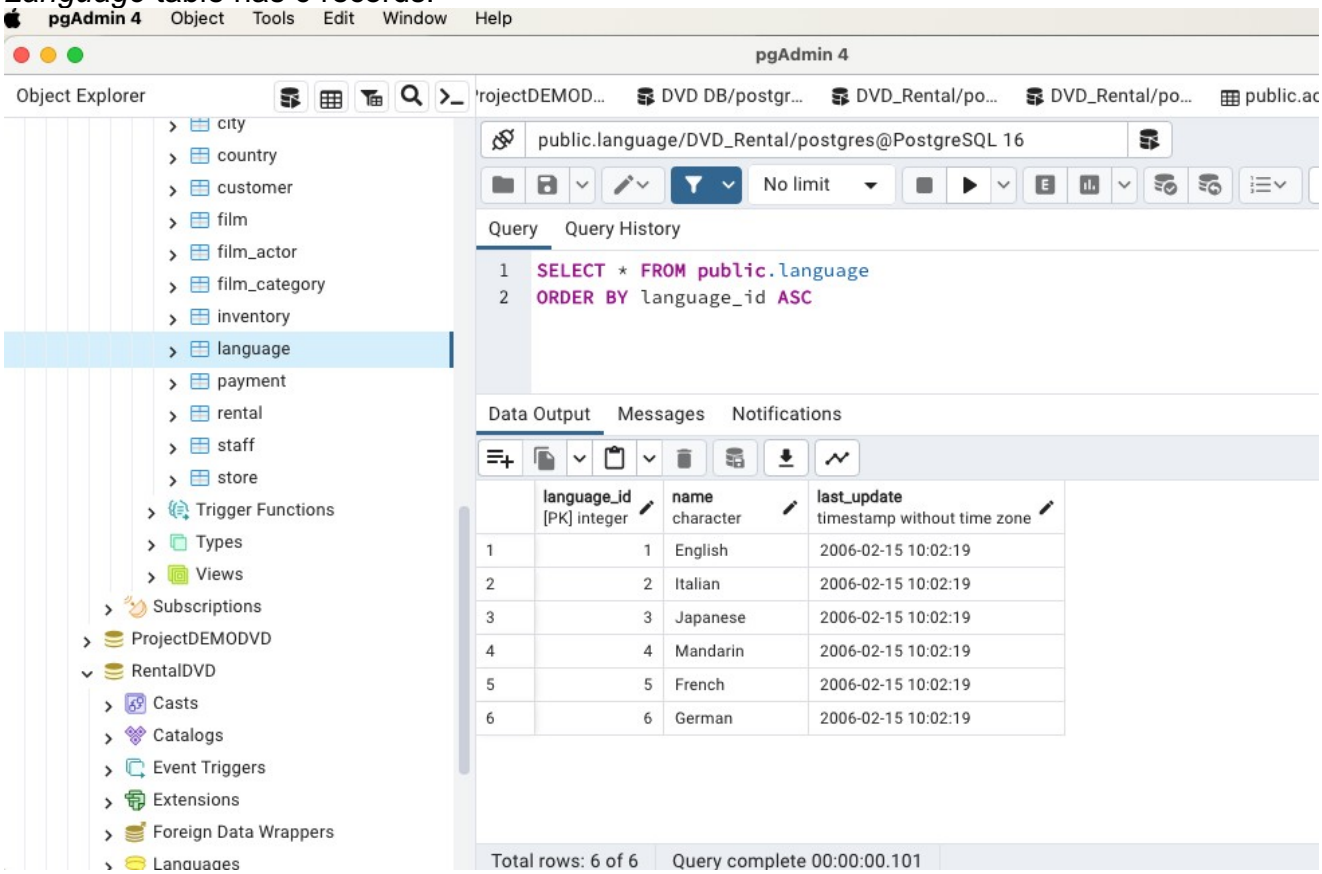

*Film_category* has 1000 records.



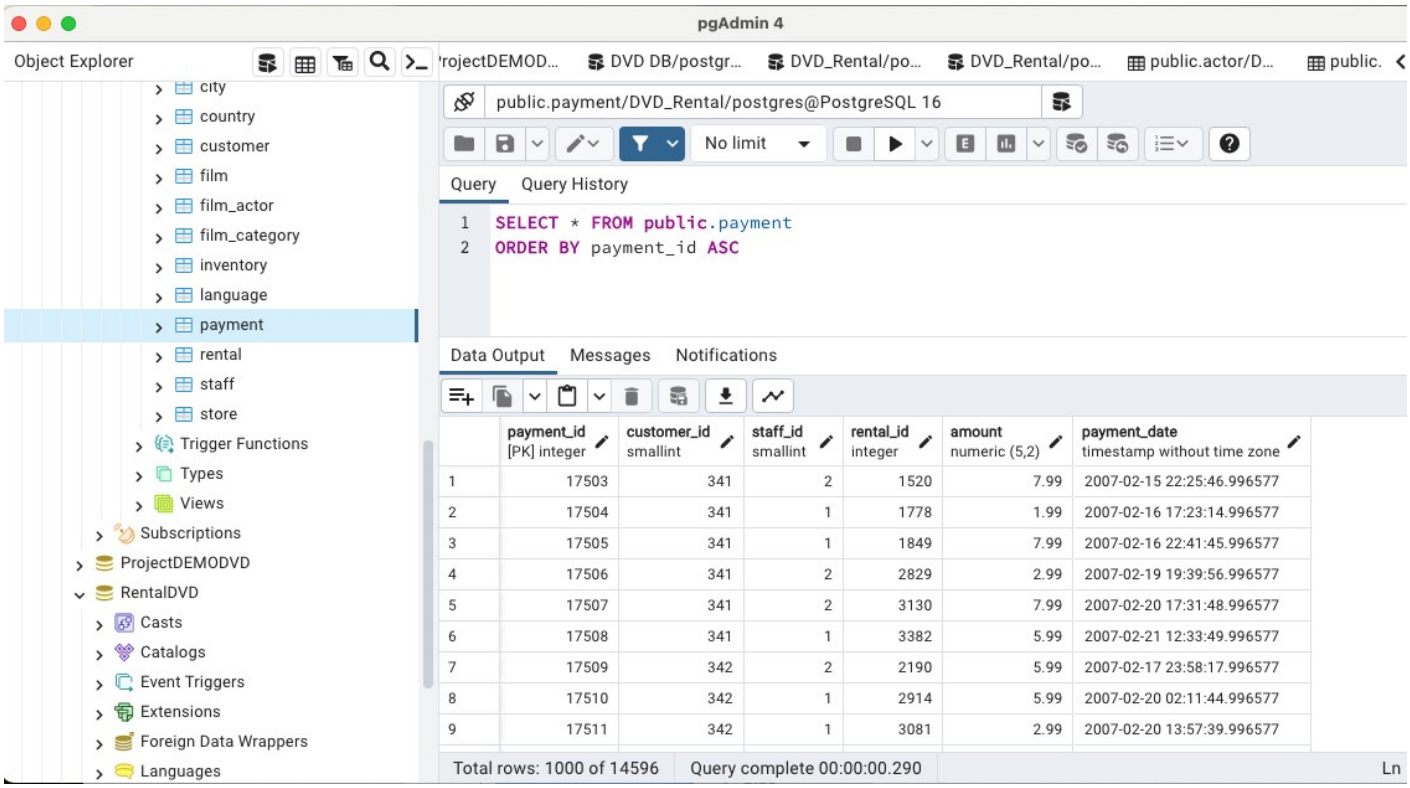

*Inventory* table has 4581 records.



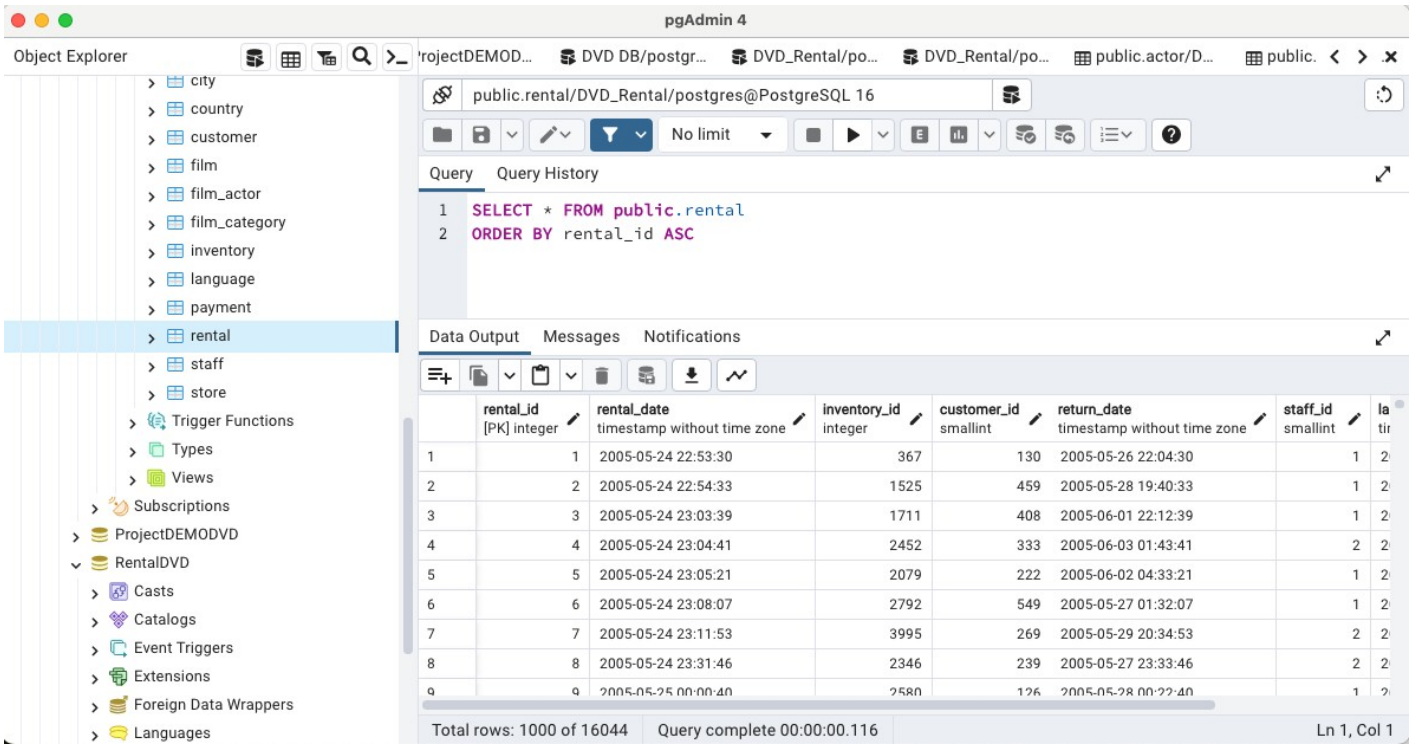

*Language* table has 6 records.



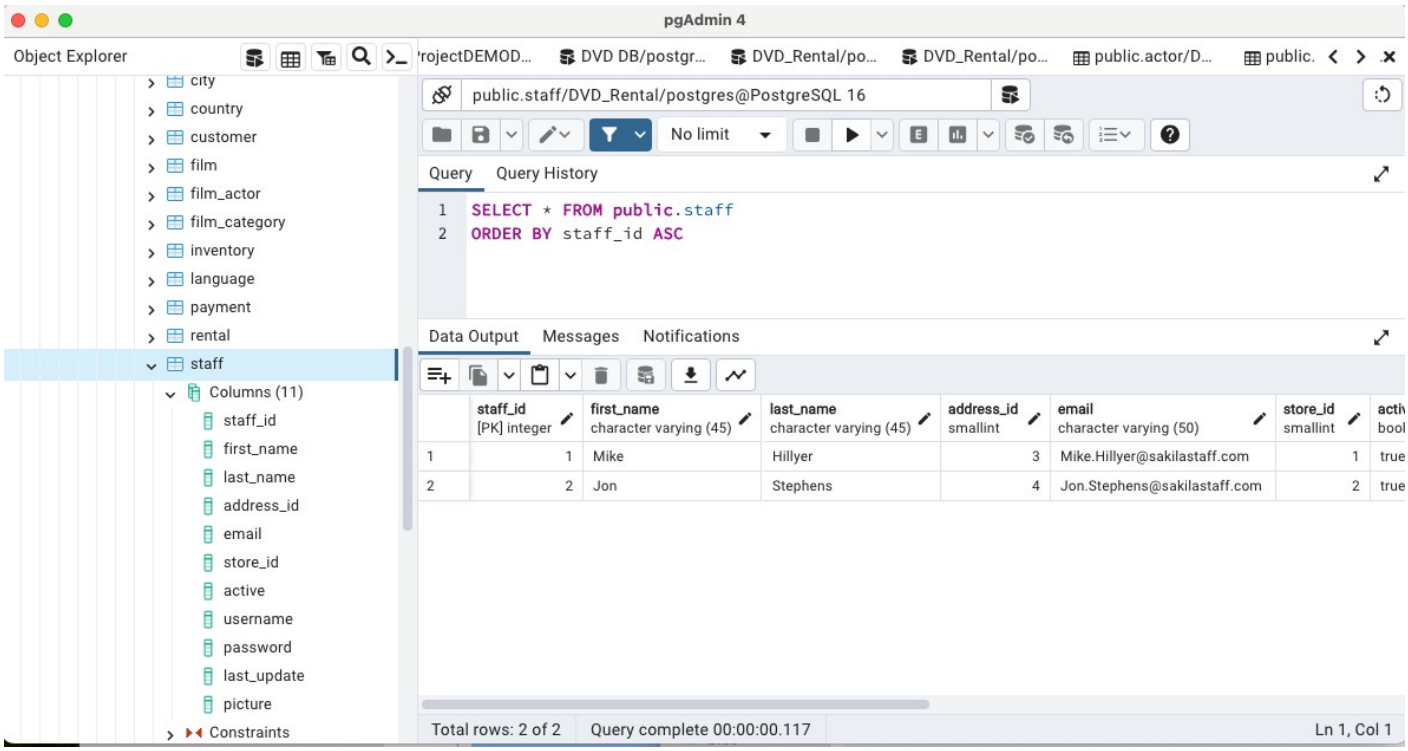

*Payment* table has 14596 records.



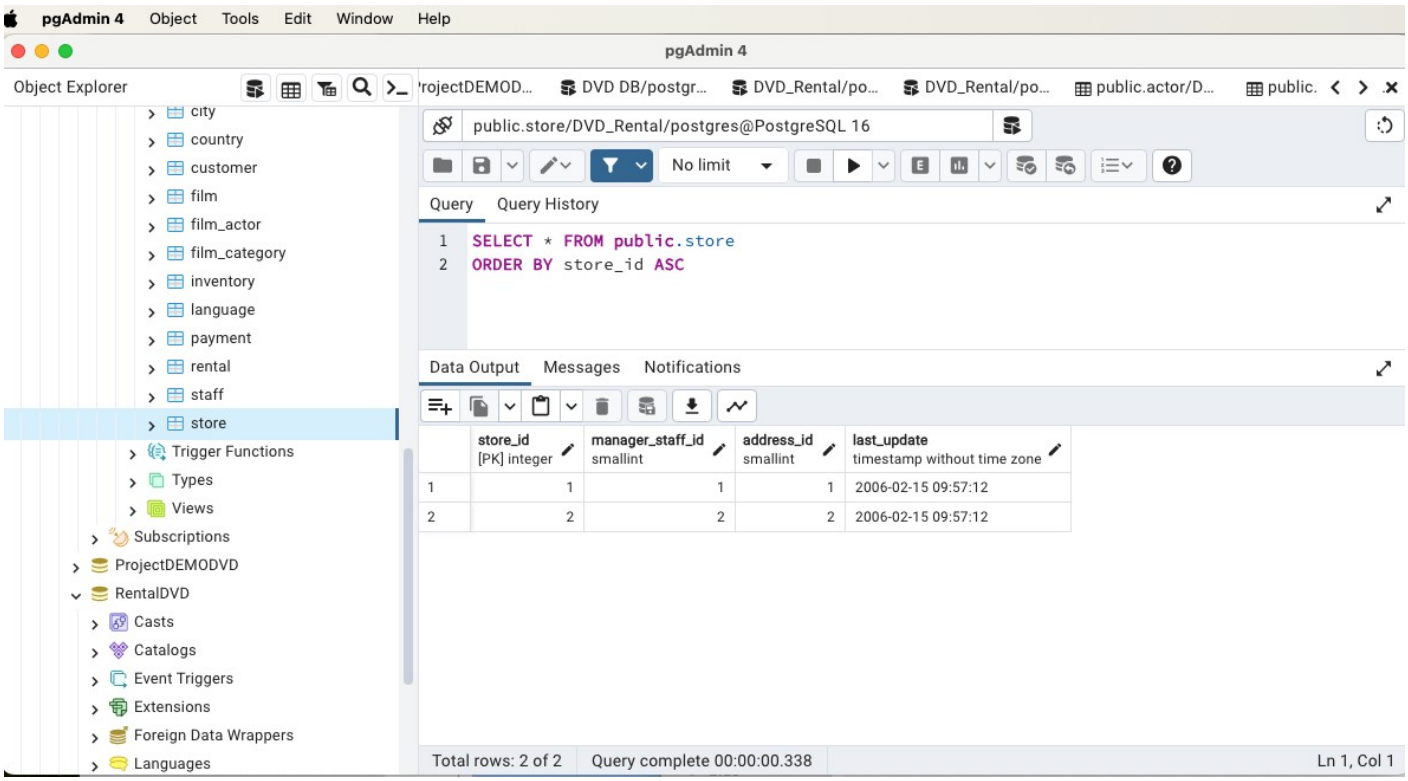

*Rental* table has 16044 records.




*Staff* table has 2 records.




*Store* table has 2 records.



## Warehouse Design Proposal:

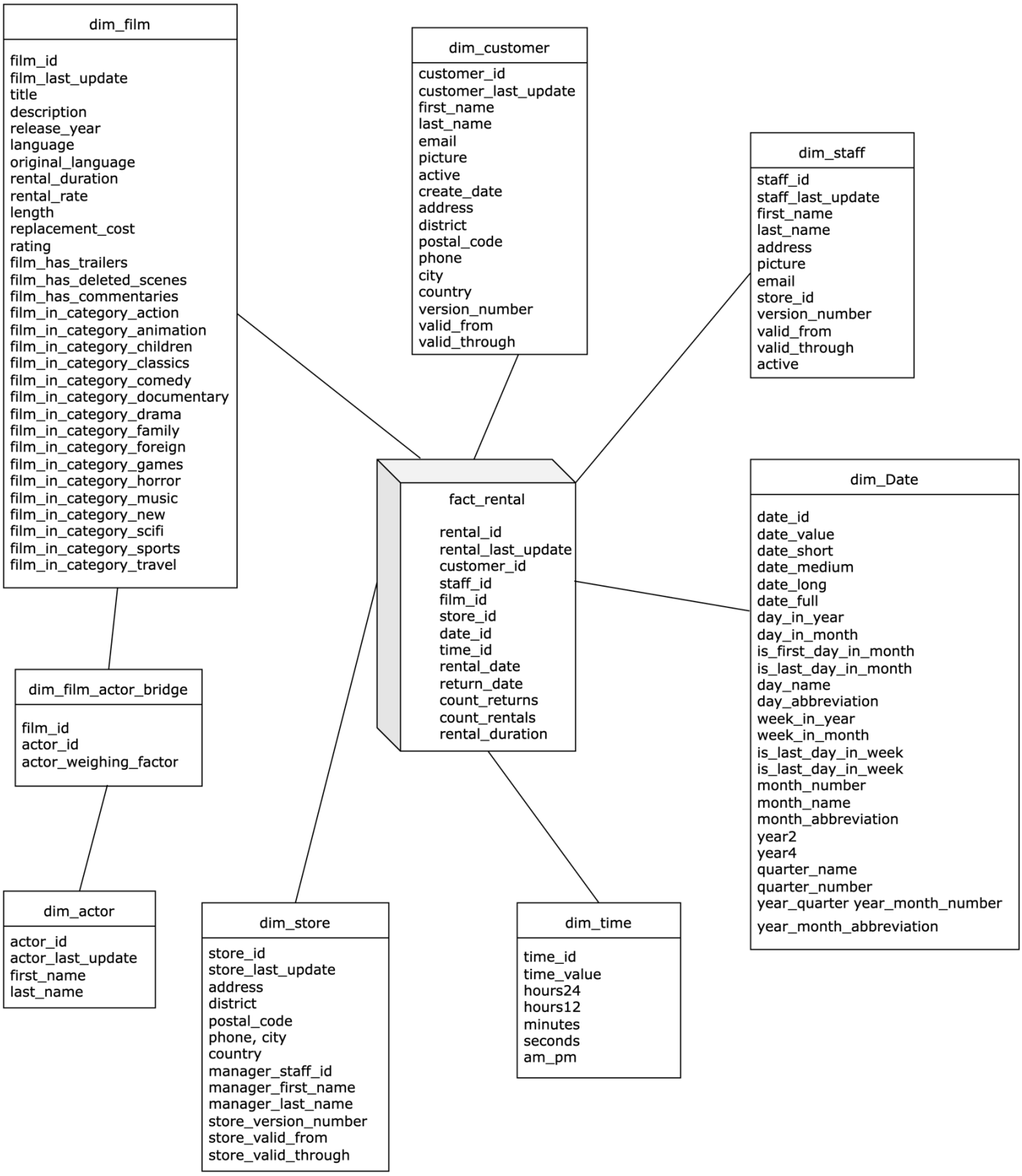
The data warehouse schema could have the following structure:

Dimensions:

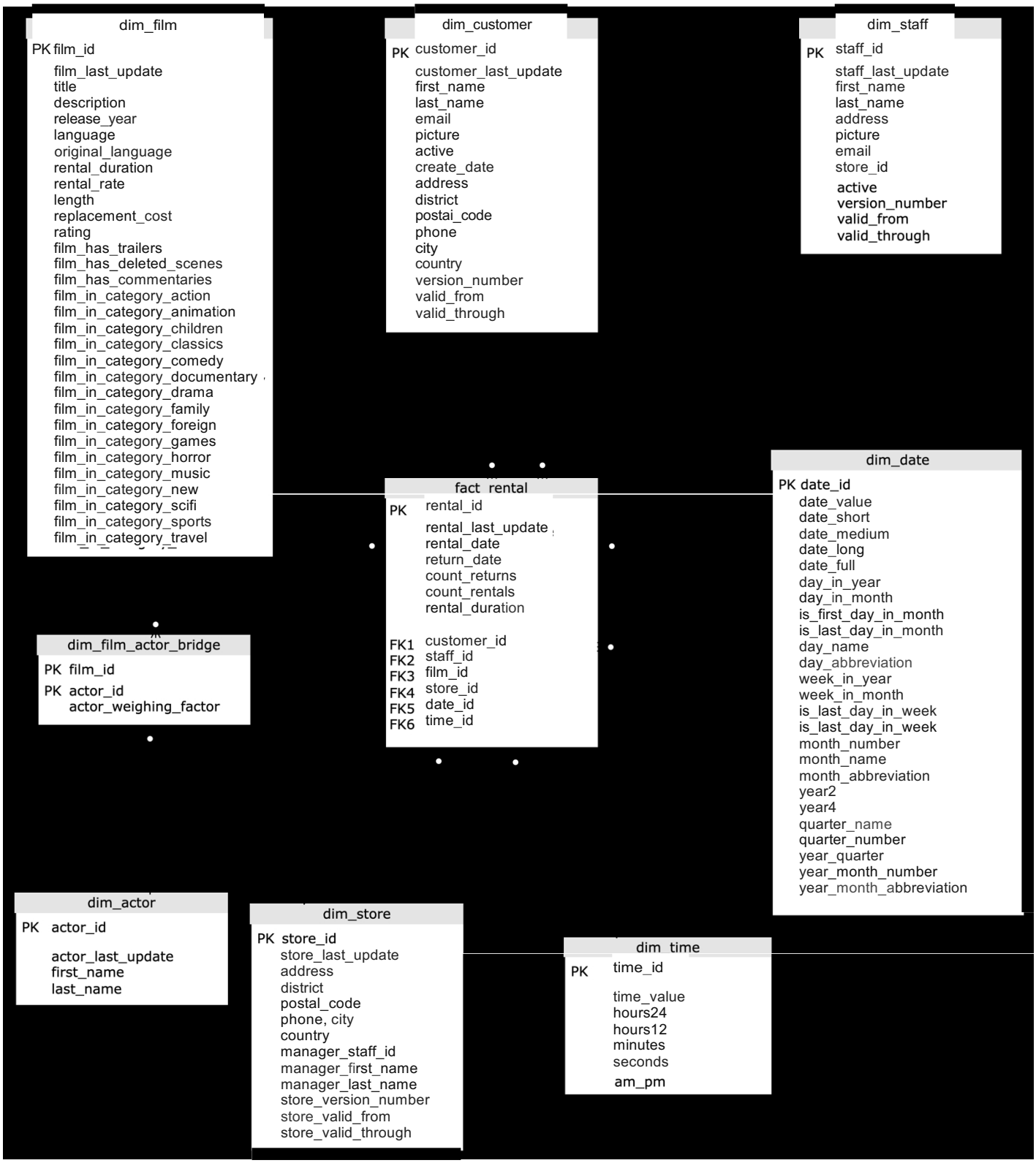
- dim_customer(customer_id, customer_last_update ,first_name, last_name, email, picture, active, create_date, address, address2, district, postal_code, phone, city, country, version_number, valid_from, valid_through)
- dim_staff(staff_id, staff_last_update, first_name, last_name, address, picture, email, store_id, active, version_number, valid_from, valid_through)
- dim_date(date_id, date_short, date_medium, date_long, date_full, day_in_year, day_in_month, is_first_day_in_month, is_last_day_in_month, day_name, day_abbreviation, week_in_year, week_in_month, is_last_day_in_week, is_last_day_in_week, month_number, month_name, month_abbreviation, year2, year4, quarter_name, quarter_number, year_quarter, year_month_number, year_month_abbreviation)
- dim_time(time_id, hours24, hours12, minutes, seconds, am_pm)
- dim_store(store_id, address, district, postal_code, phone, city, country, manager_staff_id, manager_first_name, manager_last_name, store_version_number, store_valid_from, store_valid_through)
- dim_film(film_id, film_last_update,title, description, release_year, language, original_language, rental_duration, rental_rate, length, replacement_cost, rating, film_has_trailers, film_has_deleted_scenes, film_has_commentaries, film_in_category_action, film_in_category_animation, film_in_category_children, film_in_category_classics, film_in_category_comedy, film_in_category_documentary, film_in_category_drama, film_in_category_family, film_in_category_foreign, film_in_category_games, film_in_category_horror, film_in_category_music, film_in_category_new, film_in_category_scifi, film_in_category_sports, film_in_category_travel)
- dim_actor(actor_id, actor_last_update, first_name, last_name)
- dim_film_actor_bridge(film_id, actor_id, actor_weighing_factor )

Measures:
rental_last_update
customer_id
staff_id
film_id
store_id
rental_date
return_date
count_returns
count_rentals
rental_duration

## Conceptual Model:




## Logical Model:


dim_film
PK film_id
film_last_update
title
description
release_year
language
original_language
rental_duration
rental_rate
length
replacement_cost
rating
film_has_trailers
film_has_deleted_scenes
film_has_commentaries
film_in_category_action
film_in_category_animation
film_in_category_children
film_in_category_classics
film_in_category_comedy
film_in_category_documentary
film_in_category_drama
film_in_category_family
film_in_category_foreign
film_in_category_games
film_in_category_horror
film_in_category_music
film_in_category_new
film_in_category_scifi
film_in_category_sports
film_in_category_travel
dim_customer
PK customer_id
customer_last_update
first_name
last_name
email
picture
active
create_date
address
district
postai_code
phone
city
country
version_number
valid_from
valid_through
dim_staff
PK staff_id
staff_last_update
first_name
last_name
address
picture
email
store_id
active
version_number
valid_from
valid_through
fact_rental
PK rental_id
rental_last_update
rental_date
return_date
count_returns
count_rentals
rental_duration
FK1 customer_id
FK2 staff_id
FK3 film_id
FK4 store_id
FK5 date_id
FK6 time_id
dim_date
PK date_id
date_value
date_short
date_medium
date_long
date_full
day_in_year
day_in_month
is_first_day_in_month
is_last_day_in_month
day_name
day_abbreviation
week_in_year
week_in_month
is_last_day_in_week
is_last_day_in_week
month_number
month_name
month_abbreviation
year2
year4
quarter_name
quarter_number
year_quarter
year_month_number
year_month_abbreviation
dim_film_actor_bridge
PK film_id
PK actor_id
actor_weighing_factor
dim_actor
PK actor_id
actor_last_update
first_name
last_name
dim_store
PK store_id
store_last_update
address
district
postal_code
phone, city
country
manager_staff_id
manager_first_name
manager_last_name
store_version_number
store_valid_from
store_valid_through
dim_time
PK time_id
time_value
hours24
hours12
minutes
seconds
am_pm

## SCHEMA IMPLEMENTATION:

Table structure for table `fact_rental`

```
CREATE TABLE `fact_rental` (
  `rental_last_update` datetime DEFAULT NULL,
  `rental_id` int(11) DEFAULT NULL,
  `rental_date` int(8) NOT NULL,
  `return_date` int(10) NOT NULL,
  `count_returns` int(10) NOT NULL,
  `count_rentals` int(8) NOT NULL,
  `rental_duration` int(11) DEFAULT NULL,
  `customer_id` int(8) NOT NULL,
  `staff_id` int(8) NOT NULL,
  `film_id` int(8) NOT NULL,
  `store_id` int(8) NOT NULL,
  ‘date_id’ int(10) NOT NULL,
  `time_id` int(8) NOT NULL,
  PRIMARY KEY (‘rental_id’),
  FOREIGN KEY (‘customer_id’) REFERENCES customer(`customer_id`),
  FOREIGN KEY (‘staff_id’) REFERENCES staff(`staff_id`),
  FOREIGN KEY (‘film_id’) REFERENCES film(`film_id`),
  FOREIGN KEY (‘store_id’) REFERENCES store(`store_id`),
  FOREIGN KEY (‘date_id’) REFERENCES date(`date_id`),
  FOREIGN KEY (‘time_id’) REFERENCES time(`time_id`)
);
```

Table structure for `dim_actor`

```
CREATE TABLE IF NOT EXISTS public.dim_actor (
  `actor_last_update` datetime NOT NULL,
  `last_name` varchar(45) NOT NULL,
  `first_name` varchar(45) NOT NULL,
  `actor_id` int(11) NOT NULL,
  CONSTRAINT pk_actor PRIMARY KEY (`actor_id`)
);
```

Table structure for table `dim_customer`

```
CREATE TABLE IF NOT EXISTS public.dim_customer (
  `customer_last_update` datetime NOT NULL DEFAULT '1970-01-01 00:00:00',
  `customer_id` int(8) DEFAULT NULL,
  `first_name` varchar(45) DEFAULT NULL,
  `last_name` varchar(45) DEFAULT NULL,
  `email` varchar(50) DEFAULT NULL,
  ‘picture’ image DEFAULT NULL,
  `active` char(3) DEFAULT NULL,
  `create_date` date DEFAULT NULL,
  `address` varchar(64) DEFAULT NULL,
  `district` varchar(20) DEFAULT NULL,
  `postal_code` varchar(10) DEFAULT NULL,
  `phone` varchar(20) DEFAULT NULL,
```

```
  `city` varchar(50) DEFAULT NULL,
  `country` varchar(50) DEFAULT NULL,
  `version_number` smallint(5) DEFAULT NULL,
  `valid_from` date DEFAULT NULL,
  `valid_through` date DEFAULT NULL,
  CONSTRAINT pk_customer PRIMARY KEY (`customer_id`)
);
```

Table structure for table `dim_date`

```
CREATE TABLE IF NOT EXISTS public.dim_date (
  ‘date_id’ int(10) NOT NULL,
  `date_value` date NOT NULL,
  `date_short` char(12) NOT NULL,
  `date_medium` char(16) NOT NULL,
  `date_long` char(24) NOT NULL,
  `date_full` char(32) NOT NULL,
  `day_in_year` smallint(5) NOT NULL,
  `day_in_month` tinyint(3) NOT NULL,
  `is_first_day_in_month` char(10) NOT NULL,
  `is_last_day_in_month` char(10) NOT NULL,
  `day_abbreviation` char(3) NOT NULL,
  `day_name` char(12) NOT NULL,
  `week_in_year` tinyint(3) NOT NULL,
  `week_in_month` tinyint(3) NOT NULL,
  `is_first_day_in_week` char(10) NOT NULL,
  `is_last_day_in_week` char(10) NOT NULL,
  `month_number` tinyint(3) NOT NULL,
  `month_abbreviation` char(3) NOT NULL,
  `month_name` char(12) NOT NULL,
  `year2` char(2) NOT NULL,
  ‘year4` smallint(5) NOT NULL,
  `quarter_name` char(2) NOT NULL,
  `quarter_number` tinyint(3) NOT NULL,
  `year_quarter` char(7) NOT NULL,
  `year_month_number` char(7) NOT NULL,
  `year_month_abbreviation` char(8) NOT NULL,
  CONSTRAINT pk_date PRIMARY KEY (`date_id`),
);
```

Table structure for table `dim_film`

```
CREATE TABLE IF NOT EXISTS public.dim_film (
  `film_id` int(11) NOT NULL,
  `film_last_update` datetime NOT NULL,
  `title` varchar(64) NOT NULL,
  `description` text NOT NULL,
  `release_year` smallint(5) NOT NULL,
  `language` varchar(20) NOT NULL,
  `original_language` varchar(20) NOT NULL,
  `rental_duration` tinyint(3) DEFAULT NULL,
  `rental_rate` decimal(4,2) DEFAULT NULL,
  `length` int(8) DEFAULT NULL,
```

```
  `replacement_cost` decimal(5,2) DEFAULT NULL,
  `rating` char(5) DEFAULT NULL,
  `film_has_trailers` char(4) DEFAULT NULL,
  `film_has_commentaries` char(4) DEFAULT NULL,
  `film_has_deleted_scenes` char(4) DEFAULT NULL,
  `film_in_category_action` char(4) DEFAULT NULL,
  `film_in_category_animation` char(4) DEFAULT NULL,
  `film_in_category_children` char(4) DEFAULT NULL,
  `film_in_category_classics` char(4) DEFAULT NULL,
  `film_in_category_comedy` char(4) DEFAULT NULL,
  `film_in_category_documentary` char(4) DEFAULT NULL,
  `film_in_category_drama` char(4) DEFAULT NULL,
  `film_in_category_family` char(4) DEFAULT NULL,
  `film_in_category_foreign` char(4) DEFAULT NULL,
  `film_in_category_games` char(4) DEFAULT NULL,
  `film_in_category_horror` char(4) DEFAULT NULL,
  `film_in_category_music` char(4) DEFAULT NULL,
  `film_in_category_new` char(4) DEFAULT NULL,
  `film_in_category_scifi` char(4) DEFAULT NULL,
  `film_in_category_sports` char(4) DEFAULT NULL,
  `film_in_category_travel` char(4) DEFAULT NULL,
  CONSTRAINT pk_film PRIMARY KEY (`film_id`)
);
```

Table structure for table `dim_film_actor_bridge`

```
CREATE TABLE IF NOT EXISTS public.dim_film_actor_bridge (
  `film_id` int(8) NOT NULL,
  `actor_id` int(10) NOT NULL,
  `actor_weighting_factor` decimal(3,2) NOT NULL,
  CONSTRAINT pk_film_actor_bridge PRIMARY KEY (`film_id`, `actor_id`),
);
```

Table structure for table `dim_staff`

```
CREATE TABLE IF NOT EXISTS public.dim_staff (
  `staff_id` int(8) DEFAULT NULL,
  `staff_last_update` datetime NOT NULL DEFAULT '1970-01-01 00:00:00',
  `first_name` varchar(45) DEFAULT NULL,
  `last_name` varchar(45) DEFAULT NULL,
  `email` varchar(50) DEFAULT NULL,
  ‘picture’ image DEFAULT NULL,
  `address` varchar(64) DEFAULT NULL,
  `store_id` int(8) DEFAULT NULL,
  `version_number` smallint(5) DEFAULT NULL,
  `valid_from` date DEFAULT NULL,
  `valid_through` date DEFAULT NULL,
  `active` char(3) DEFAULT NULL,
  CONSTRAINT pk_staff PRIMARY KEY (`staff_id`),
);
```

Table structure for table `dim_store`

```
CREATE TABLE IF NOT EXISTS public.dim_store (
  `store_last_update` datetime NOT NULL DEFAULT '1970-01-01 00:00:00',
  `store_id` int(8) DEFAULT NULL,
  `address` varchar(64) DEFAULT NULL,
  `district` varchar(20) DEFAULT NULL,
  `postal_code` varchar(10) DEFAULT NULL,
  `phone_number` varchar(20) DEFAULT NULL,
  `city` varchar(50) DEFAULT NULL,
  `country` varchar(50) DEFAULT NULL,
  `manager_staff_id` int(8) DEFAULT NULL,
  `manager_first_name` varchar(45) DEFAULT NULL,
  `manager_last_name` varchar(45) DEFAULT NULL,
  `store_version_number` smallint(5) DEFAULT NULL,
  `store_valid_from` date DEFAULT NULL,
  `store_valid_through` date DEFAULT NULL,
  CONSTRAINT pk_store PRIMARY KEY (`store_id`),
);
```

Table structure for table `dim_time`

```
CREATE TABLE IF NOT EXISTS public.dim_time (
  `time_id` int(8) NOT NULL,
  `time_value` time NOT NULL,
  `hours24` tinyint(3) NOT NULL,
  `hours12` tinyint(3) DEFAULT NULL,
  `minutes` tinyint(3) DEFAULT NULL,
  `seconds` tinyint(3) DEFAULT NULL,
  `am_pm` char(3) DEFAULT NULL,
  CONSTRAINT pk_time PRIMARY KEY (`time_id`),
);
```

## Primary events:

The primary events in the DVD rental database schema are actions related to DVD rentals. These events can include the following:

1. **Rental**: Customers renting DVDs from a rental store.
2. **Return**: Customers returning rented DVDs.
3. **Customer Activity**: Customer interactions and details, such as their contact information and rental history.
4. **Film**: Information related to films available for rent.
5. **Staff Activity**: Actions taken by staff members.
6. **Store Information**: Details about the rental store locations.
7. **Time and Date**: Time and date-related events, including rental and return timestamps.
8. **Actor and Film Relationships**: Associations between actors and films, including weighting factors.
9. **Category**: Categorization of films into various genres.
10. **Manager Information**: Details about store managers.

## OLAP (Online Analytical Processing) Operations:

1. Total payment amount per customer, per year.

   ROLLUP*(Rental, Customer ➔ Customer, Payment Date ➔ Year, SUM(Payment.amount) AS Total Payment Amount)

2. Monthly rentals by cities of the customers.

   Rental_1 ⇒ ROLLUP*(Rental, Rental Date ➔ Month, Address ➔ City, COUNT(Rental.Rental ID) AS Monthly Rentals)

3. 3 Customers who paid the most money for rentals.

   Rental_1 ⇒ ROLLUP*(Rental, Customer ➔ Customer, SUM(Payment.amount)) MAX(Rental_1, Payment.amount, 3)

4. Inventories with the number of rentals per month.

   Rental_1 ⇒ ROLLUP*(Rental, Rental Date ➔ Month) COUNT(Rental_1, Rental.Rental ID) BY Rental.Inventory ID

5. Duration of films based on Categories.

   Rental_1 ⇒ ROLLUP*(Rental, Film ➔ Length, Film Category ➔ Film D) SUM(Rental_1, Sum(Film.Length)) BY Film Category.Category ID

6. Top Films by Total Rental Revenue:

   Rental_1 <- ROLLUP*(Rental, Film -> Film, SUM(Rental.Rental Duration * Film.Rental Rate) AS Total_Revenue) MAX(Rental_1, Total_Revenue, 10)

7. Late Returns by Customer and Month:

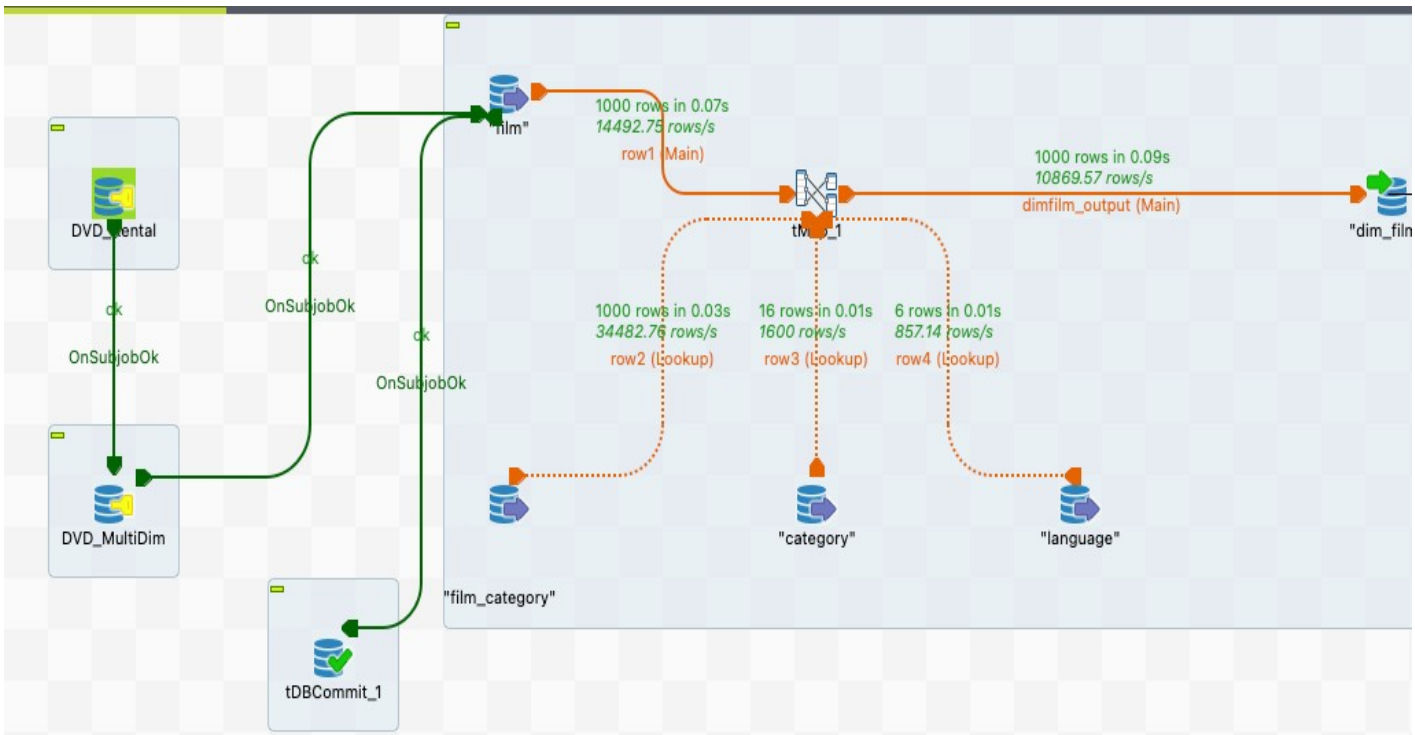
   Rental_1 <- ROLLUP*(Rental, Rental Date -> Month, Customer -> Customer, SUM(CASE WHEN Rental.Return Date > Rental.Rental Date THEN 1 ELSE 0 END) AS Late_Returns)

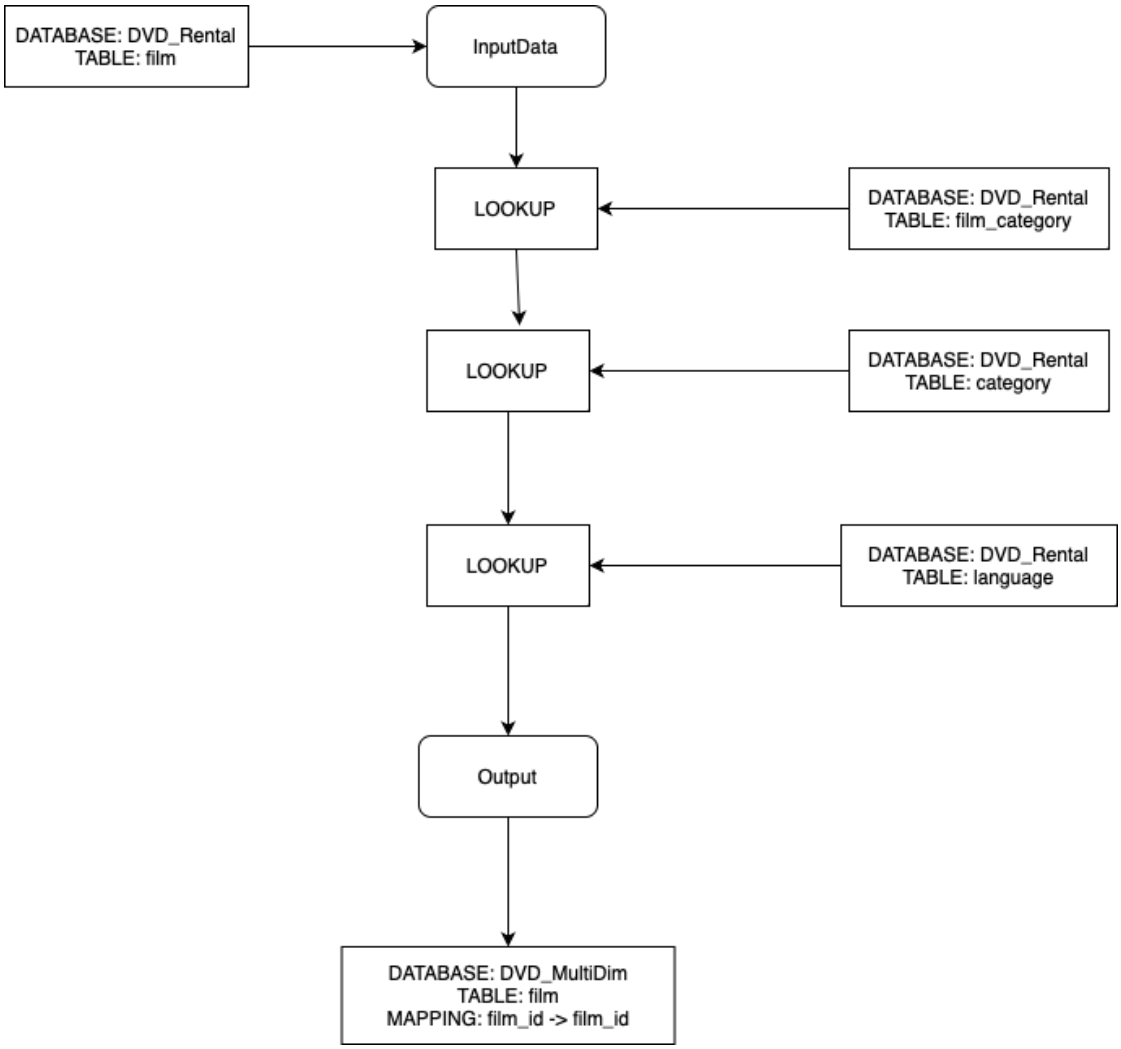
8. Total Payments by Store and Month:

   Rental_1 <- ROLLUP*(Rental, Payment.Date -> Month, Store.City, SUM(Payment.Amount) AS Total_Payment_Amount)

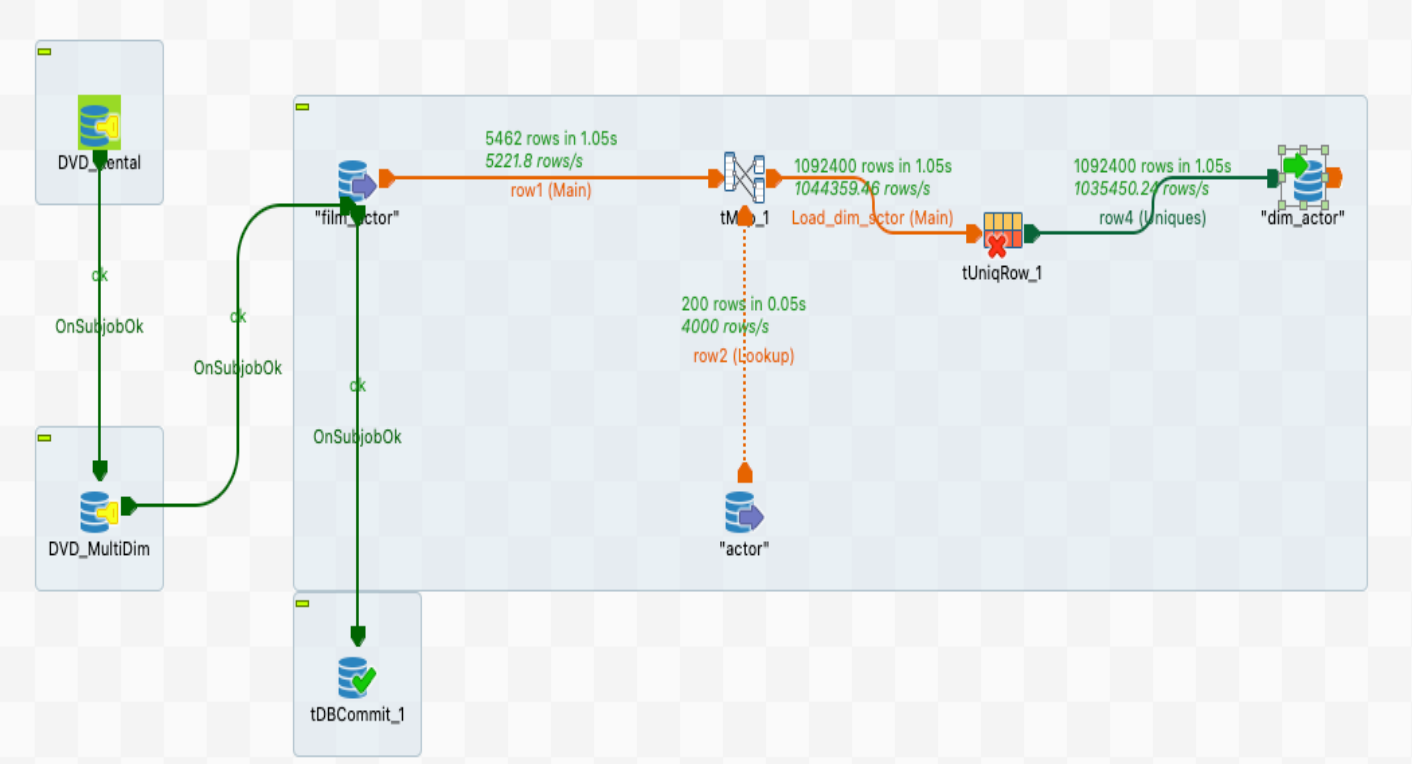
# CONTROL AND DATA FLOWS:

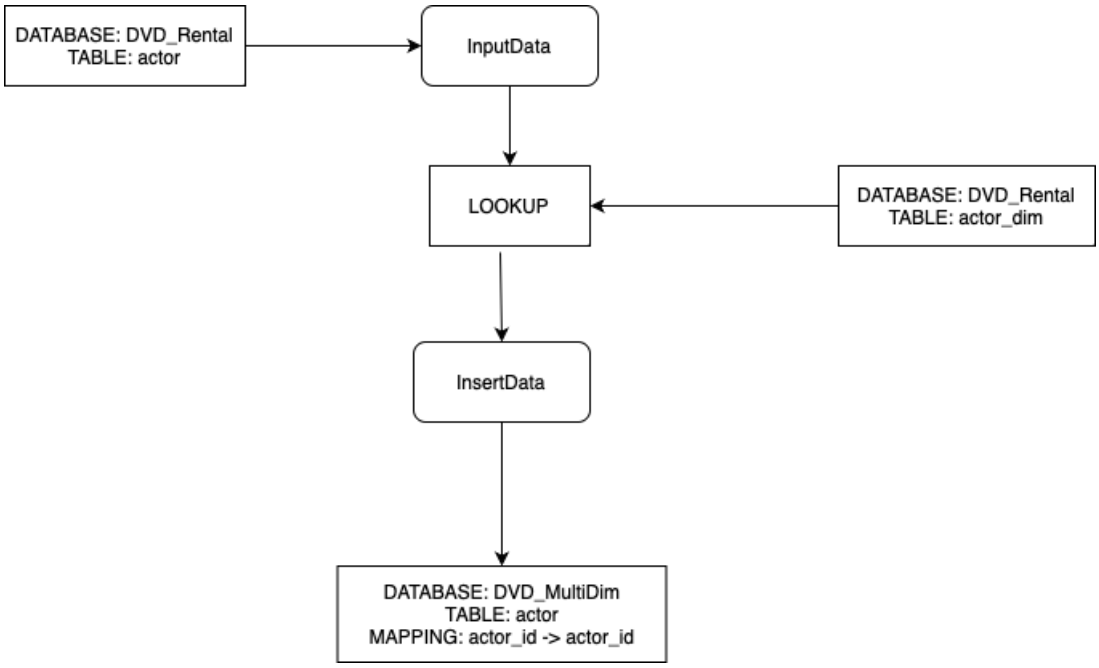
Loading data into dimension table dim_film:



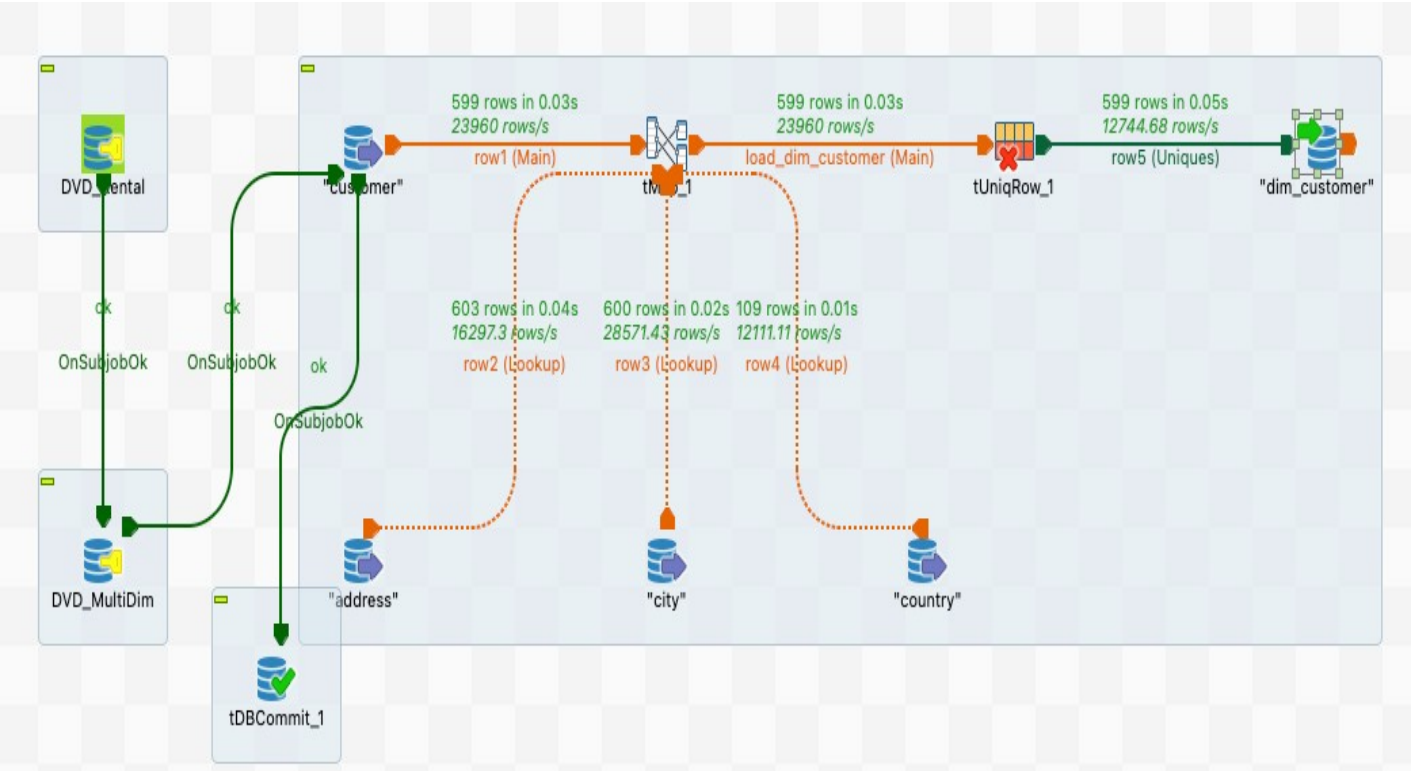
Loading data into dim_actor:



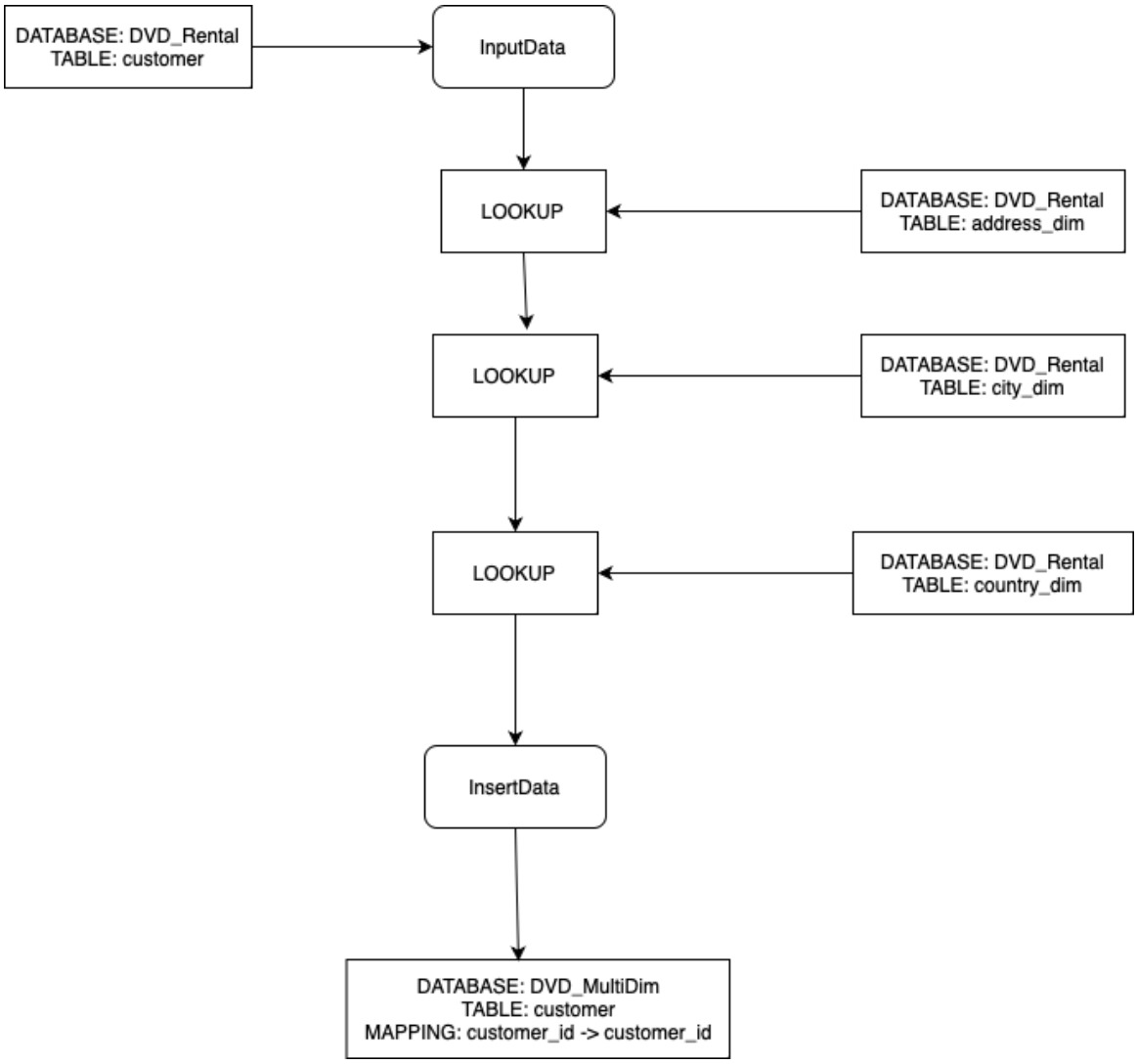




We have used tUniqRow component to ensure the quality of data in a job's input or output flow.

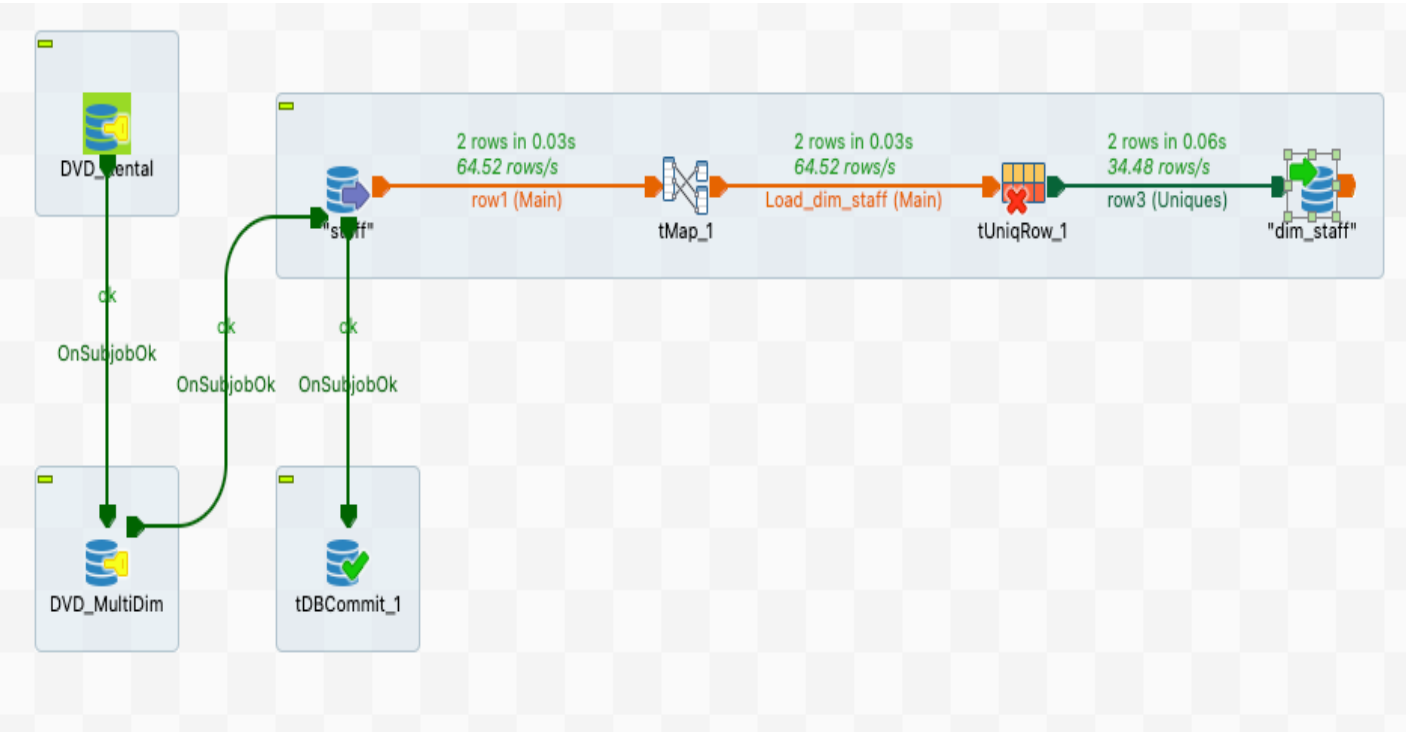
Loading Data into dim_customer:



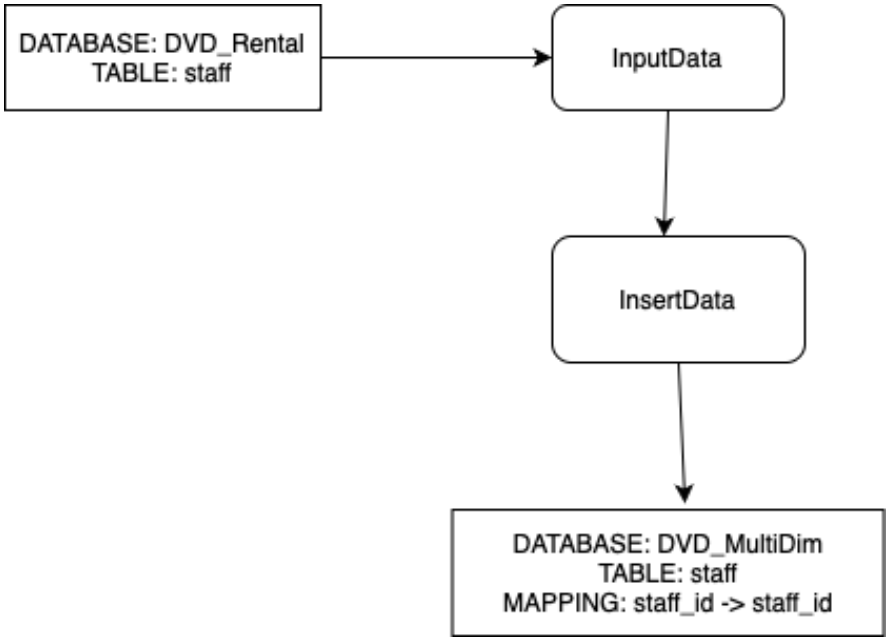

We have used tUniqRow component to ensure the quality of data in a job's input or output flow.



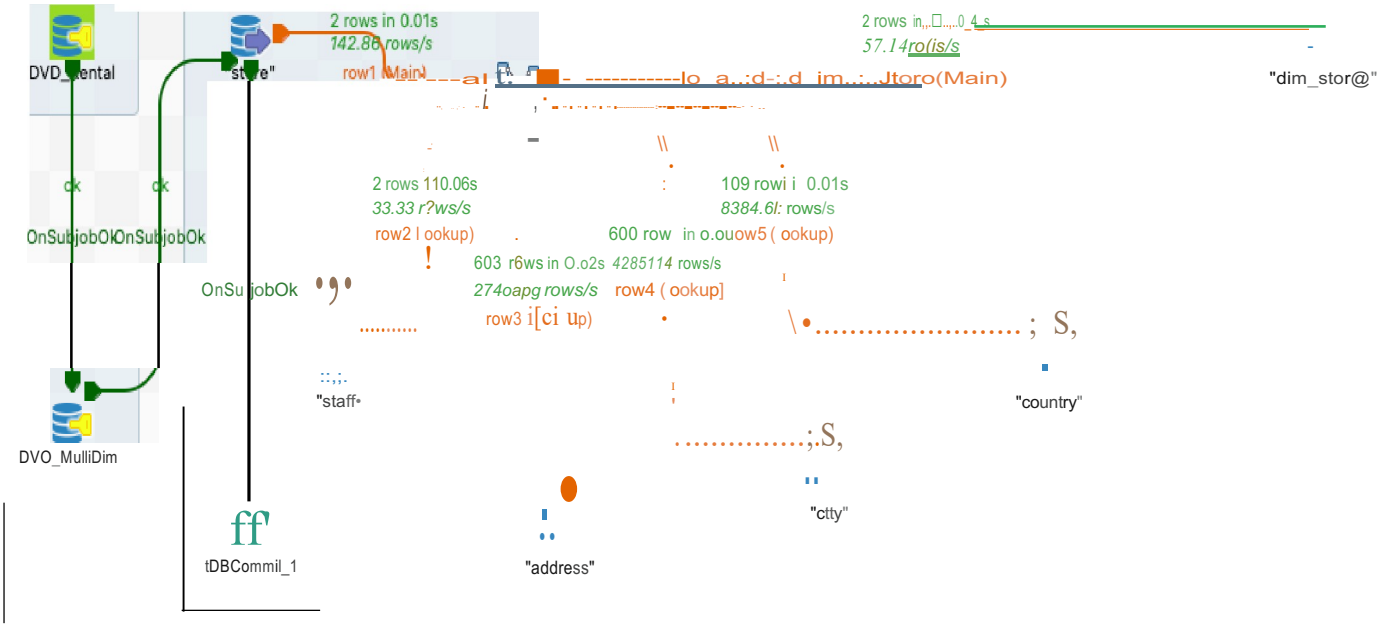

Loading data into dim_staff

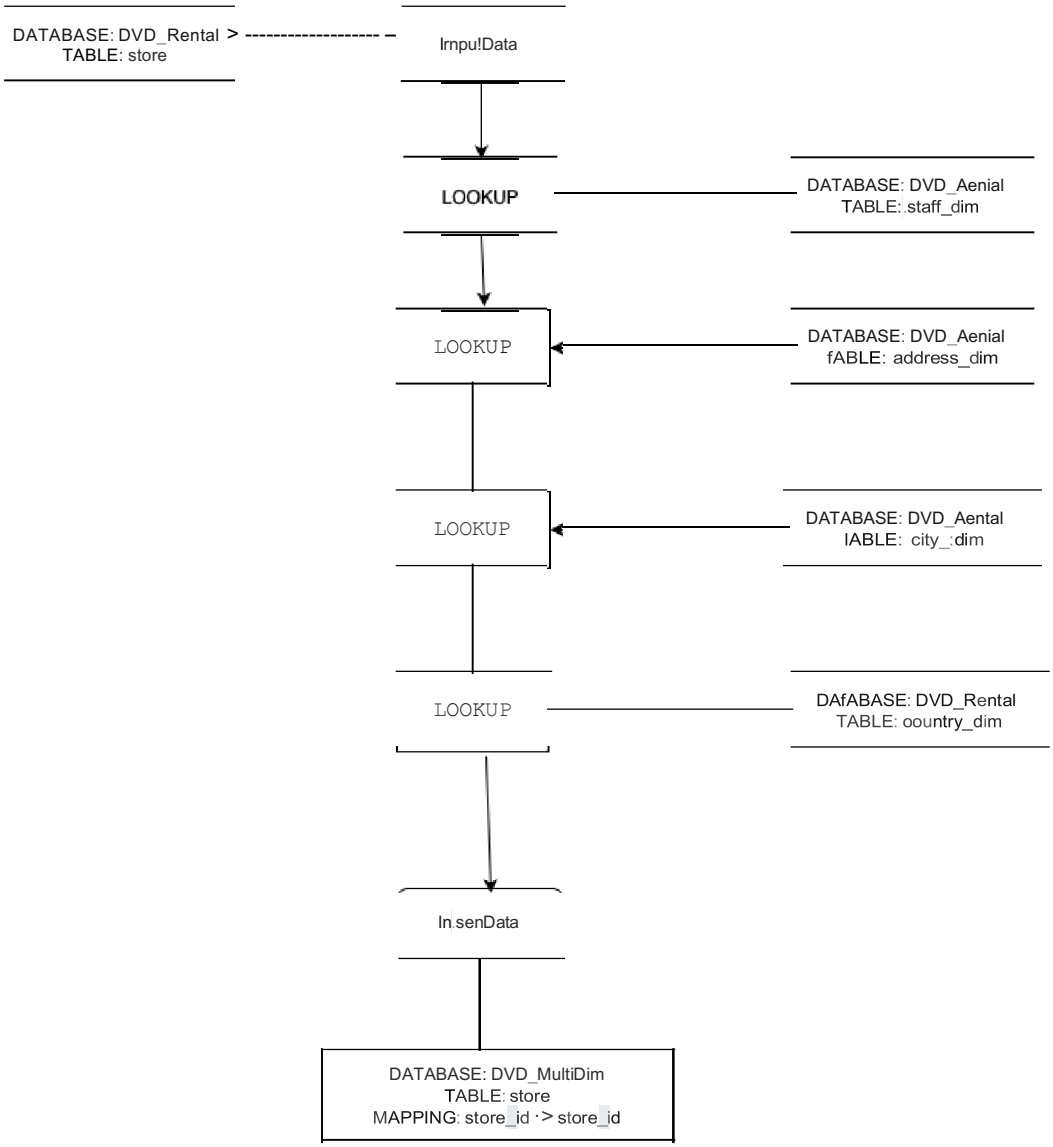


We have used tUniqRow component to ensure the quality of data in a job's input or output flow.

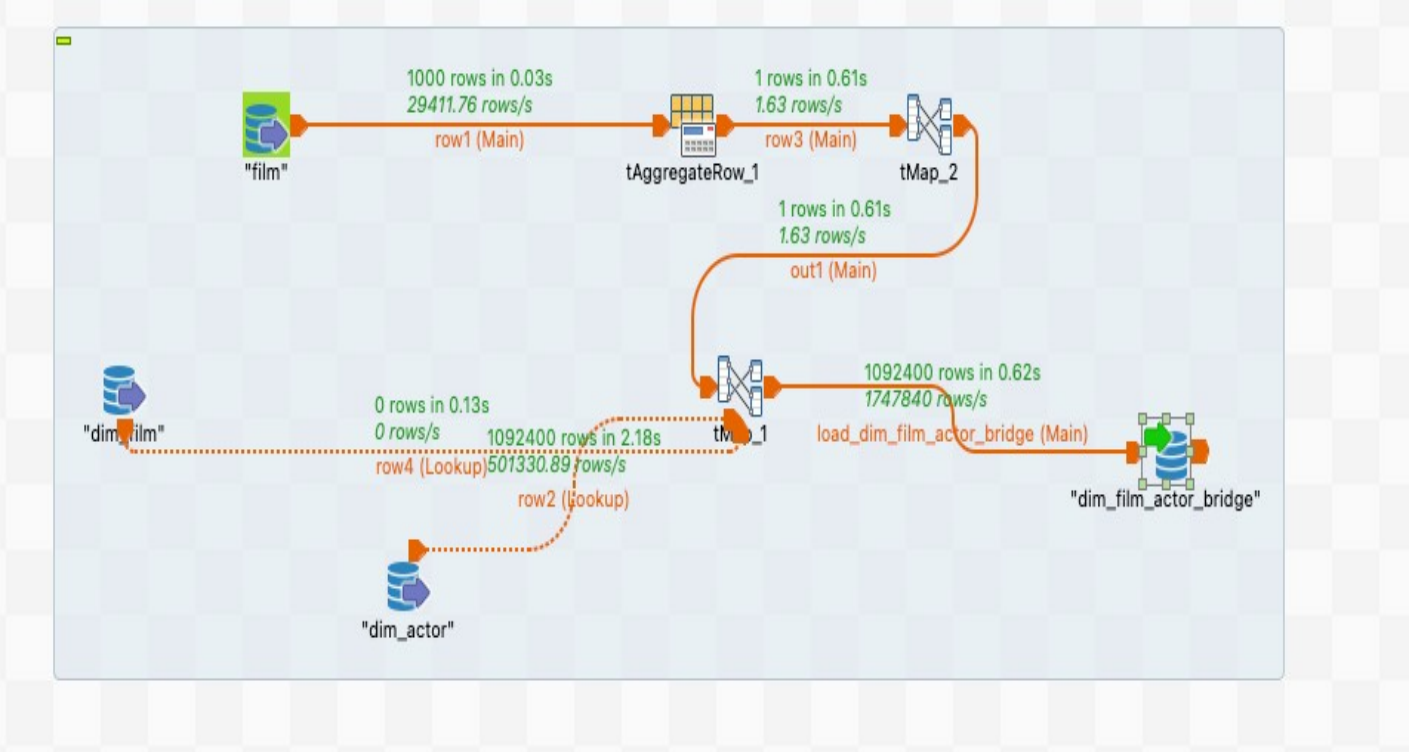


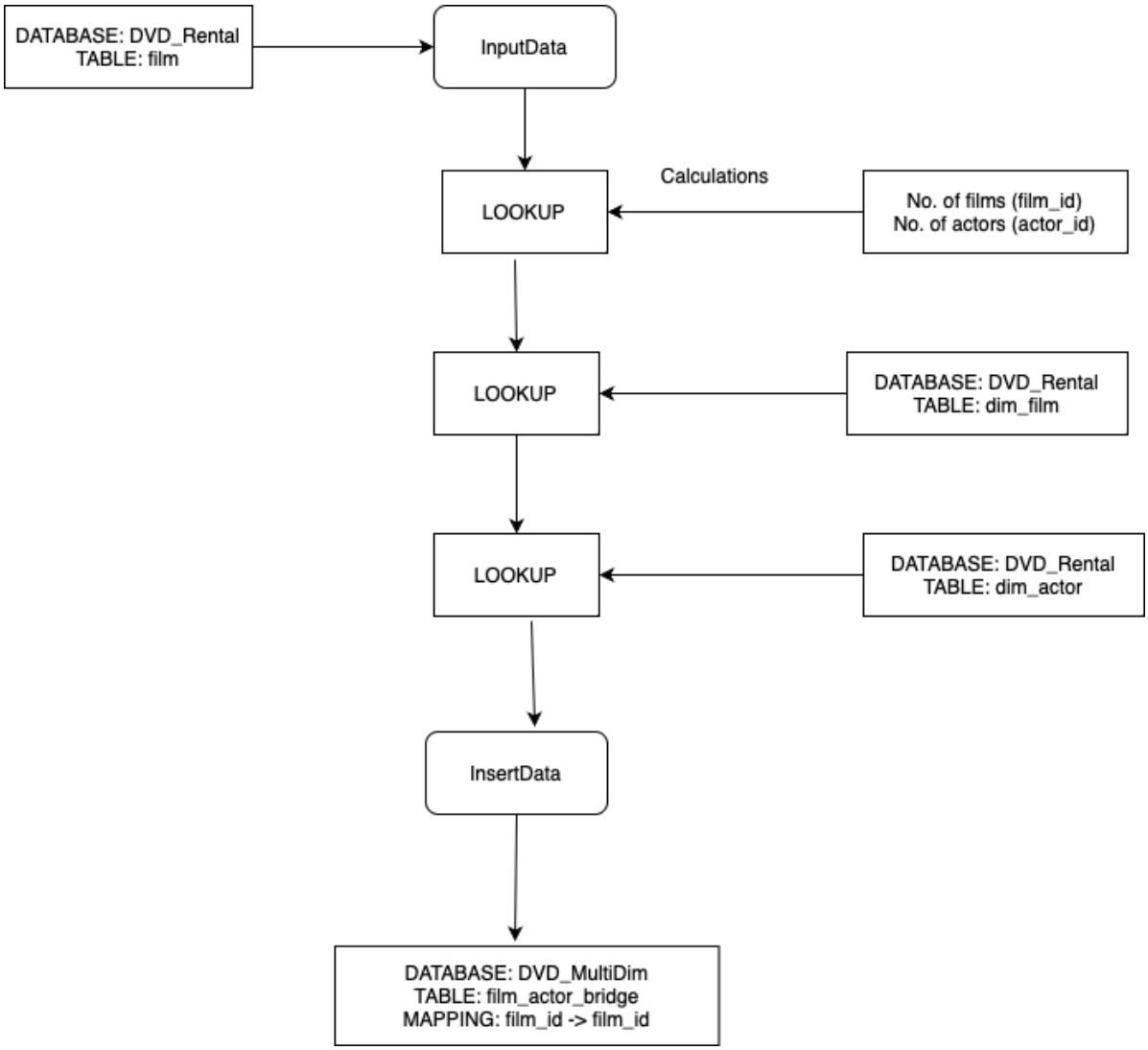

Loading data into dim_store:


2 rows in 0.01s
142.88 rows/s
row1 (Main)
2 rows in 0.4 s
57.14 rows/s
"dim_stor@"
DVD_Rental
"store"
ok
ok
OnSubjobOk
OnSubjobOk
OnSubjobOk
2 rows 110.06s
33.33 rows/s
row2 (Lookup)
109 rows 0.01s
8384.61 rows/s
600 rows in 0.0005 s (Lookup)
603 rows in 0.02s 4285114 rows/s
274 rows/s
row4 (Lookup)
row3 (Lookup)
"staff"
"country"
DVD_MultiDim
tDBCommit_1
"address"
"city"


DATABASE: DVD_Rental
TABLE: store
Input Data
LOOKUP
DATABASE: DVD_Rental
TABLE: staff_dim
LOOKUP
DATABASE: DVD_Rental
TABLE: address_dim
LOOKUP
DATABASE: DVD_Rental
TABLE: city_dim
LOOKUP
DATABASE: DVD_Rental
TABLE: country_dim
Insert Data
DATABASE: DVD_MultiDim
TABLE: store
MAPPING: store_id -> store_id

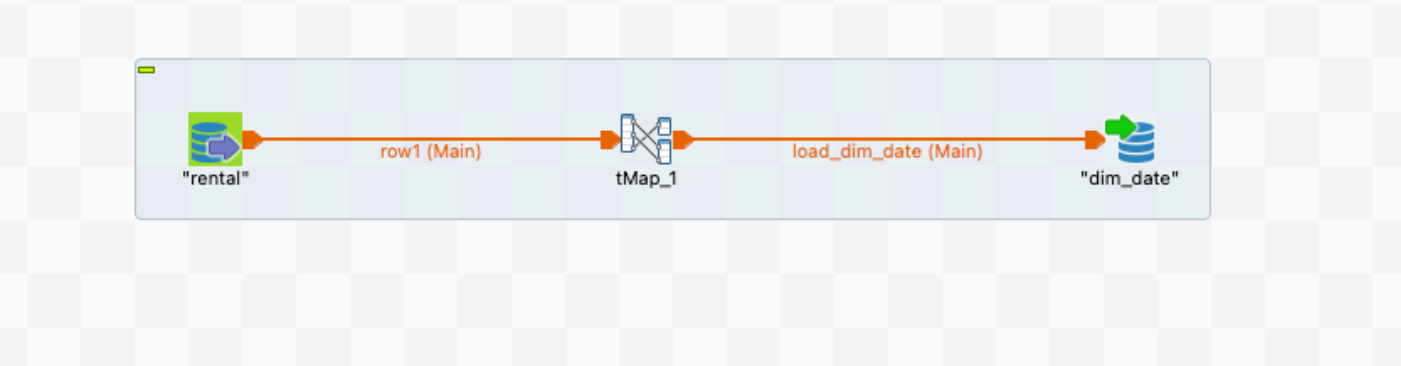
Loading data into dim_film_actor_bridge:



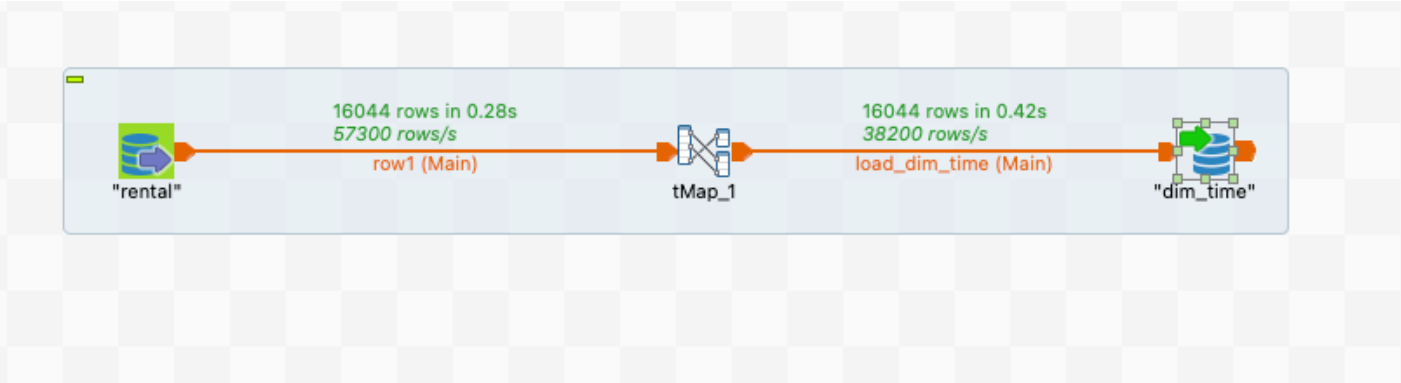




Here, we have used tAggregateRow component to calculate the sum of film_rental_duration.

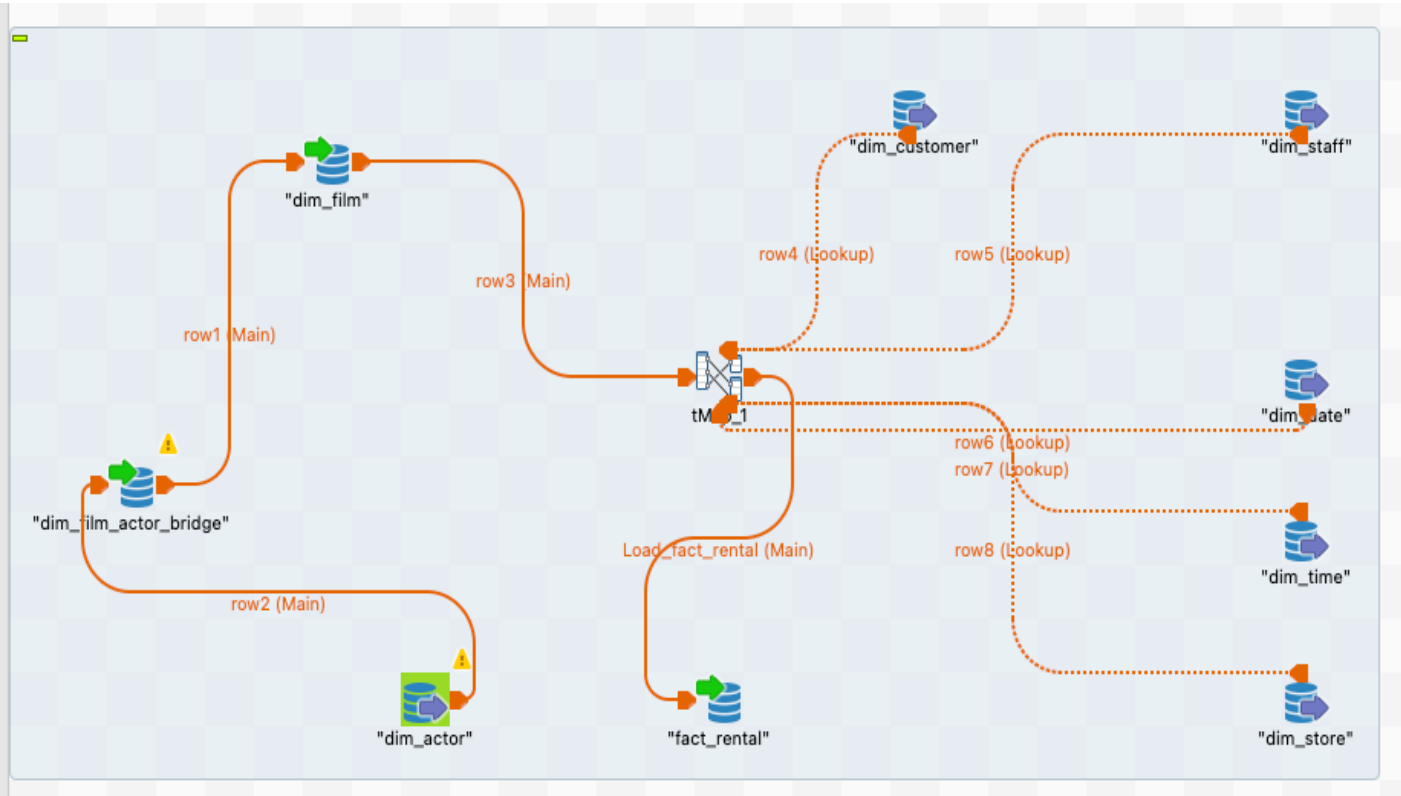
Loading data into dim_date:



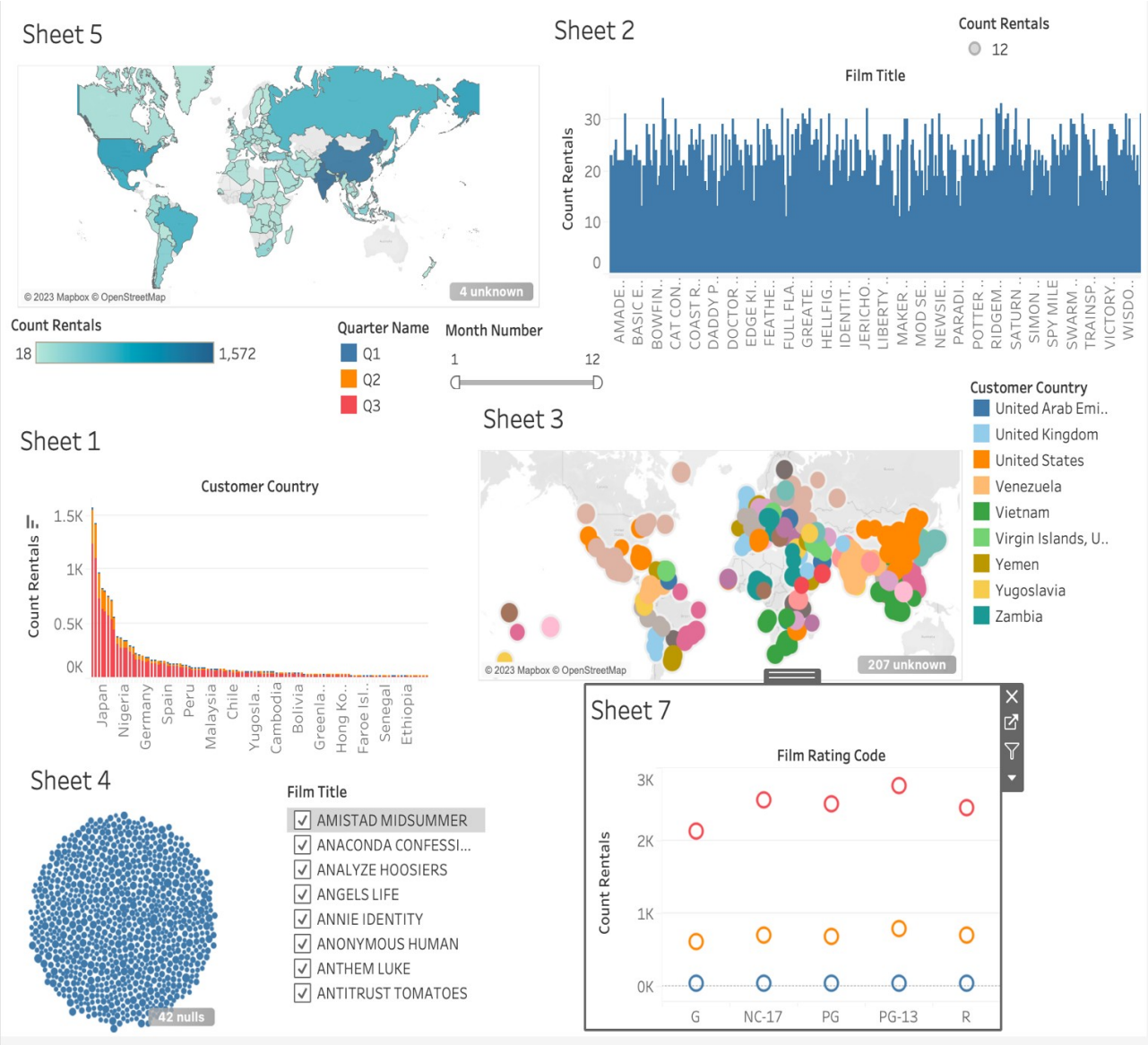

Loading data into dim_time:




Loading data into fact table: fact_rental



## ANALYSIS IN TABLEAU:




- Figuring out the quarter-wise number of rentals in each Country.
- We can see, for almost all the countries, Quarter 3 had the highest rentals.

- Figuring out the total number of rentals for each film across the globe. There is a slider, and we can check total monthly rentals for a film in each country as well.

- This world map shows the number of rentals for each city in each country. The size of the dots represents the number of rentals.
- We can see percentage of cities where rentals happened was the highest in India.

- This visualization shows the total duration of a film, for which it was rented. The check boxes on the side allow us to check for a particular film or many films at once.

- This world heat map shows the total rentals in each country. If we hover over a particular country, we can see the monthly trends of rentals for that country.

- Displaying the total rentals for each rating category, quarterly. We can see that in Quarter 1, the lowest rentals took place for each of the category.

## CONCLUSION:

The completion of this project marks the successful integration of our DVD Rental Database into Talend, we loaded our database from Postgres. The implemented ETL pipelines have significantly enhanced data processing efficiency, streamlining the flow of information throughout our system. The project's architecture is designed with scalability in mind, allowing for future expansion and adaptation to accommodate growing data volumes and evolving business needs. The ETL processes have been fine-tuned to deliver optimal performance, ensuring timely data extraction, transformation, and loading operations.