Table of contents

Project overview

The Sakila database is designed to represent a typical film rental store. It includes data about films, actors, customers, rentals, payments, staff, and stores. The objective is to build a data warehouse to serve analytical purposes, understand the business better, and support decision-making. The main goal is to deliver data quickly and accurately.

Modeling

a. Business Process

In the business process, the customer decides to rent a film and visits the store. Our staff assists the customer in selecting a film, and the customer proceeds to rent the film. After the agreed rental period, the customer returns the film to the store, completing the rental transaction.

b. Granuality

The level of granularity is such that each row in the fact table represents one film rental.

c. Dimensions

- **Dim customer**
  (Slowly Changing Dimension Type 2)
  contains customer information, details of the customer's address, and three columns to handle slowly changing dimensions (Type 2): valid_from, valid_through, and a version number.
- **Dim staff**
  contains the staff information and their addresses.
- **Dim store**
  contains the store information and their addresses.
- **Dim film**
  The Film Dimension contains all film information and transforms the many-to-many relationship with the Category table by adding a column for each category indicating 'yes' or 'no'.

- **Film to actor bridge**
  (Bridge table)
  The Film to Actor Bridge manages the many-to-many relationship between the Film Dimension and the Actor Dimension.
- **Dim actor**
  (Multivalued Dimension)
  Contains actor information.
- **Dim date**
  (Role Playing Dimension)
  contains all information about a specific date, such as the full date, year, quarter, month_name, month_of_year, week_of_year, day_of_week, and day_name.
- **Dim time**
  (Role Playing Dimension)
  contains all information about a specific time, including the time key (primary key), full time, hour, and minute.

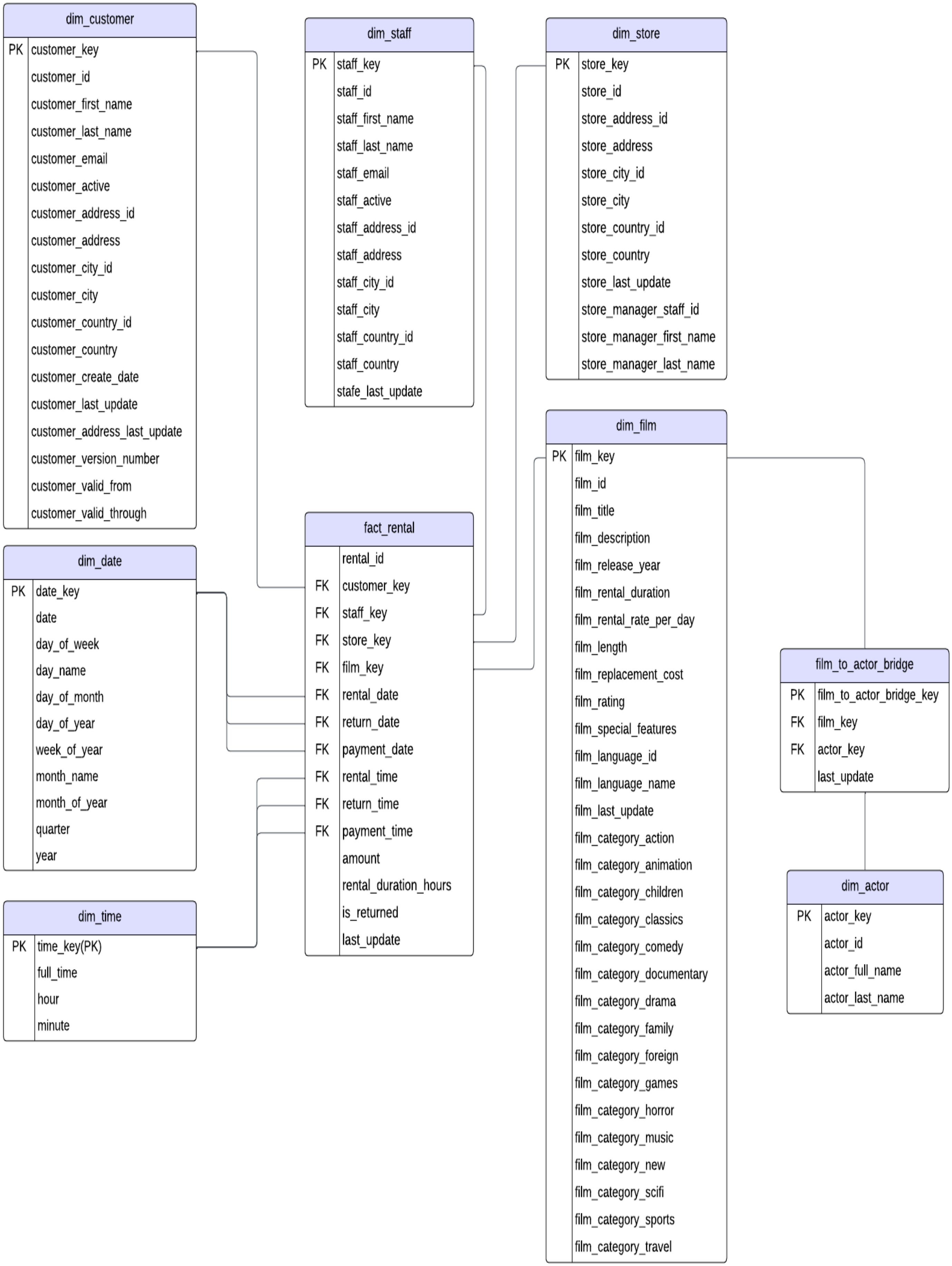
In each dimension table, there is a surrogate key (tablename_key) separate from the primary key, which serves as the primary key of the table.

c. Facts

(fact rental)

The fact table contains one row for each rental, whether the film is returned or not, and includes measures related to the rental such as amount, rental duration, and return status. It also contains foreign keys related to this rental, such as customer_key, staff_key, and film_key.

each table contains a 'last update' column to facilitate incremental loading.



dim_customer	
PK	customer_key
	customer_id
	customer_first_name
	customer_last_name
	customer_email
	customer_active
	customer_address_id
	customer_address
	customer_city_id
	customer_city
	customer_country_id
	customer_country
	customer_create_date
	customer_last_update
	customer_address_last_update
	customer_version_number
	customer_valid_from
	customer_valid_through

dim_staff	
PK	staff_key
	staff_id
	staff_first_name
	staff_last_name
	staff_email
	staff_active
	staff_address_id
	staff_address
	staff_city_id
	staff_city
	staff_country_id
	staff_country
	stafe_last_update

dim_store	
PK	store_key
	store_id
	store_address_id
	store_address
	store_city_id
	store_city
	store_country_id
	store_country
	store_last_update
	store_manager_staff_id
	store_manager_first_name
	store_manager_last_name

dim_date	
PK	date_key
	date
	day_of_week
	day_name
	day_of_month
	day_of_year
	week_of_year
	month_name
	month_of_year
	quarter
	year

fact_rental	
	rental_id
FK	customer_key
FK	staff_key
FK	store_key
FK	film_key
FK	rental_date
FK	return_date
FK	payment_date
FK	rental_time
FK	return_time
FK	payment_time
	amount
	rental_duration_hours
	is_returned
	last_update

dim_time	
PK	time_key(PK)
	full_time
	hour
	minute

dim_film	
PK	film_key
	film_id
	film_title
	film_description
	film_release_year
	film_rental_duration
	film_rental_rate_per_day
	film_length
	film_replacement_cost
	film_rating
	film_special_features
	film_language_id
	film_language_name
	film_last_update
	film_category_action
	film_category_animation
	film_category_children
	film_category_classics
	film_category_comedy
	film_category_documentary
	film_category_drama
	film_category_family
	film_category_foreign
	film_category_games
	film_category_horror
	film_category_music
	film_category_new
	film_category_scifi
	film_category_sports
	film_category_travel

film_to_actor_bridge	
PK	film_to_actor_bridge_key
FK	film_key
FK	actor_key
	last_update

dim_actor	
PK	actor_key
	actor_id
	actor_full_name
	actor_last_name

3. Query optimization

Create clustered columnstore indexes on all tables to store data in columnar format, which is an effective approach for read-only databases and delivers fast query performance. Columnstore indexes optimize query processing by accessing only the necessary columns, reducing I/O operations and benefiting from enhanced cache efficiency. Additionally, data compression within columns saves storage space and reduces disk I/O, making them ideal for analytical workloads involving large datasets.

4- ETL mapping

- **Dim customer**

Source column	Source tables	Target Column Type	Mapping Rules	Target column
customer_id	customer	int	As-Is	customer_id
first_name	customer	varchar(45)	As-Is (unknown if null)	customer_first_name
last_name	customer	varchar(45)	As-Is (unknown if null)	customer_last_name
email	customer	varchar(50)	As-Is (unknown if null)	customer_email
active	customer	char(2)	As-Is (unknown if null)	customer_active
address_id	customer	int	As-Is	cutomer_address_id
address	customer address	varchar(50)	Left join with address table (unknown if null)	customer_address
city_id	address	int	As-Is	customer_city_id
city	address city	varchar(50)	Left join with city table (unknown if null)	customer_city
country_id	city	smallint	As-Is	customer_country_id
country	city country	varchar(50)	Left join with country table (unknown if null)	customer_country
create_date	customer	datetime	As-Is (1900-1-1 if null)	customer_create_date

last_update	customer	datetime	As-Is (1900-1-1 if null)	customer_last_update

- **Dim staff**

Source column	Source tables	Target Column Type	Mapping Rules	Target column
staff_id	staff	int	As-Is	staff_id
first_name	staff	varchar(45)	As-Is (unknown if null)	staff_first_name
last_name	staff	varchar(45)	As-Is (unknown if null)	staff_last_name
email	staff	varchar(50)	As-Is (unknown if null)	staff_email
active	staff	char(2)	As-Is (unknown if null)	staff_active
address_id	staff	int	As-Is	staff_address_id
address	staff address	varchar(50)	Left join with address table (unknown if null)	staff_address
city_id	address	int	As-Is	staff_city_id
city	address city	varchar(50)	Left join with city table (unknown if null)	staff_city
country_id	city	smallint	As-Is	staff_country_id
country	city country	varchar(50)	Left join with country table (unknown if null)	staff_country
last_update	staff	datetime	As-Is (1900-1-1 if null)	staff_last_update

- **Dim store**

Source column	Source tables	Target Column Type	Mapping Rules	Target column
staff_id	store	int	As-Is	store_id
address_id	store	int	As-Is	store_address_id
address	store address	varchar(50)	Left join with address table	store_address

			(unknown if null)	
city_id	address	int	As-Is	store_city_id
city	address city	varchar(50)	Left join with city table (unknown if null)	store_city
country_id	city	smallint	As-Is	store_country_id
country	city country	varchar(50)	Left join with country table (unknown if null)	store_country
last_update	store	datetime	As-Is (1900-1-1 if null)	store_last_update
manager_staff_id	store	int	As-Is	store_manger_staff_id
first_name	store staff	varchar(45)	Left join with staff table(unknown if null)	store_manager_first_name
last_name	store staff	varchar(45)	Left join with staff table(unknown if null)	store_manager_last_name

- **Dim film**

Source column	Source tables	Target Column Type	Mapping Rules	Target column
film_id	film	int	As-Is	film_id
title	film	int	As-Is(unknown if null)	film_title
description	film	varchar(max)	As-Is (unknown if null)	film_description
release_year	film	varchar(4)	As-Is(‘xxx’ if null)	film_release_year
rental_duration	film	int	As-Is(-1 if null)	film_rental_duration
rental_duration rental_rate	film	decimal(4, 2)	rental_rate / rental_duration (-1 if null)	film_rental_rate_per_day
length	film	int	As-Is (-1 if null)	film_length

replacement_cost	film	decimal(5,2)	As-Is(-1 if null)	film_replacement_cost
rating	film	varchar(10)	As-Is(unknown if null)	film_rating
special_features	film	varchar(255)	As-Is(unknown if null)	film_special_features
language_id	language	int	As-Is	film_language_id
name	language	char(20)	As-Is(unknown if null)	film_language_name
last_update	film	datetime	As-Is(1900-1-1 if null)	film_last_update
name	film film_category category	char(3)	join with film_category fc and category c if c.name = action then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_action
name	film film_category category	char(3)	join with film_category fc and category c if c.name = animation then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_animation
name	film film_category category	char(3)	join with film_category fc and category c if c.name = children then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_children
name	film film_category category	char(3)	join with film_category fc and category c if c.name = classics then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_classics

name	film film_category category	char(3)	join with film_category fc and category c if c.name = comedy then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_comedy
name	film film_category category	char(3)	join with film_category fc and category c if c.name = documentary then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_documentary
name	film film_category category	char(3)	join with film_category fc and category c if c.name = drama then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_drama
name	film film_category category	char(3)	join with film_category fc and category c if c.name = family then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_family
name	film film_category category	char(3)	join with film_category fc and category c if c.name = foreign then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_foreign
name	film film_category category	char(3)	join with film_category fc and category c	film_category_games

			if c.name = games then ‘yes’ else ‘no’ (‘xxx’ if null)	
name	film film_category category	char(3)	join with film_category fc and category c if c.name = horror then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_horror
name	film film_category category	char(3)	join with film_category fc and category c if c.name = music then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_music
name	film film_category category	char(3)	join with film_category fc and category c if c.name = new then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_new
name	film film_category category	char(3)	join with film_category fc and category c if c.name = scifi then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_scifi
name	film film_category category	char(3)	join with film_category fc and category c if c.name = sports then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_sports

name	film film_category category	char(3)	join with film_category fc and category c if c.name = travel then ‘yes’ else ‘no’ (‘xxx’ if null)	film_category_travel

- **Film to actor bridge**

Source column	**Source tables**	**Target Column Type**	**Mapping Rules**	**Target column**
film_key	film_category dim_film	int	left join film_category with dim_film	film_key
actor_key	film_category dim_actor	int	film_category left join with dim_actor	actor_key
last_update	film_category	datetime	As-Is (1900-1-1 if null)	last_update

- **Dim actor**

Source column	**Source tables**	**Target Column Type**	**Mapping Rules**	**Target column**
actor_id	actor	int	As-Is	actor_id
first_name last_name	actor	varchat(100)	first_name + ‘ ‘ + last_name (unknown if null)	actor_full_name
last_update	actor	datetime	As-Is (1900-1-1 if null)	actor_last_update

- **Fact rental**

Source column	**Source tables**	**Target Column Type**	**Mapping Rules**	**Target column**
rental_id	rental	int	As-Is	renta_id
customer_key	rental	int	left join rental with dim_customer	customer_key

	dim_customer			
staff_key	rental dim_staff	int	left join rental with dim_staff	staff_key
store_key	rental inventory dim_store	int	left join(rental , inventory , dim_store)	store_key
film_key	rental inventory dim_film	int	left join(rental , inventory , dim_film)	film_key
date_key	rental dim_date	int	left join(rental ,dim_date)	rental_date
date_key	rental dim_date	int	left join(rental ,dim_date)	return_date
date_key	payment dim_date	int	left join(rental ,dim_date)	payment_date
time_key	rental dim_time	int	left join(rental , dim_time)	rental_time
time_key	rental dim_time	int	left join(rental , dim_time)	return_time
time_key	payment dim_time	int	left join(rental , dim_time)	payment_time
amount	payment	decimal(5,2)	As-Is	amount
rental_date return_date	rental	int	DATEDIFF("DAY", rental_date , return_date) * 24 + DATEDIFF("HOUR",rental_time, return_time)	rental_duration_hours
return_date	rental	char(3)	if return_date <= GETDATE() then 1 else 0	is_returned
last_update	payment	datetime	As-Is	last_update

5- Business Questions

1- What is the ratio of each rating in rentals?

2- Who are the top 10 actors whose films have the highest rentals?

3- What are the 5 most rented categories?

5- How does the rental behavior vary by customer country?

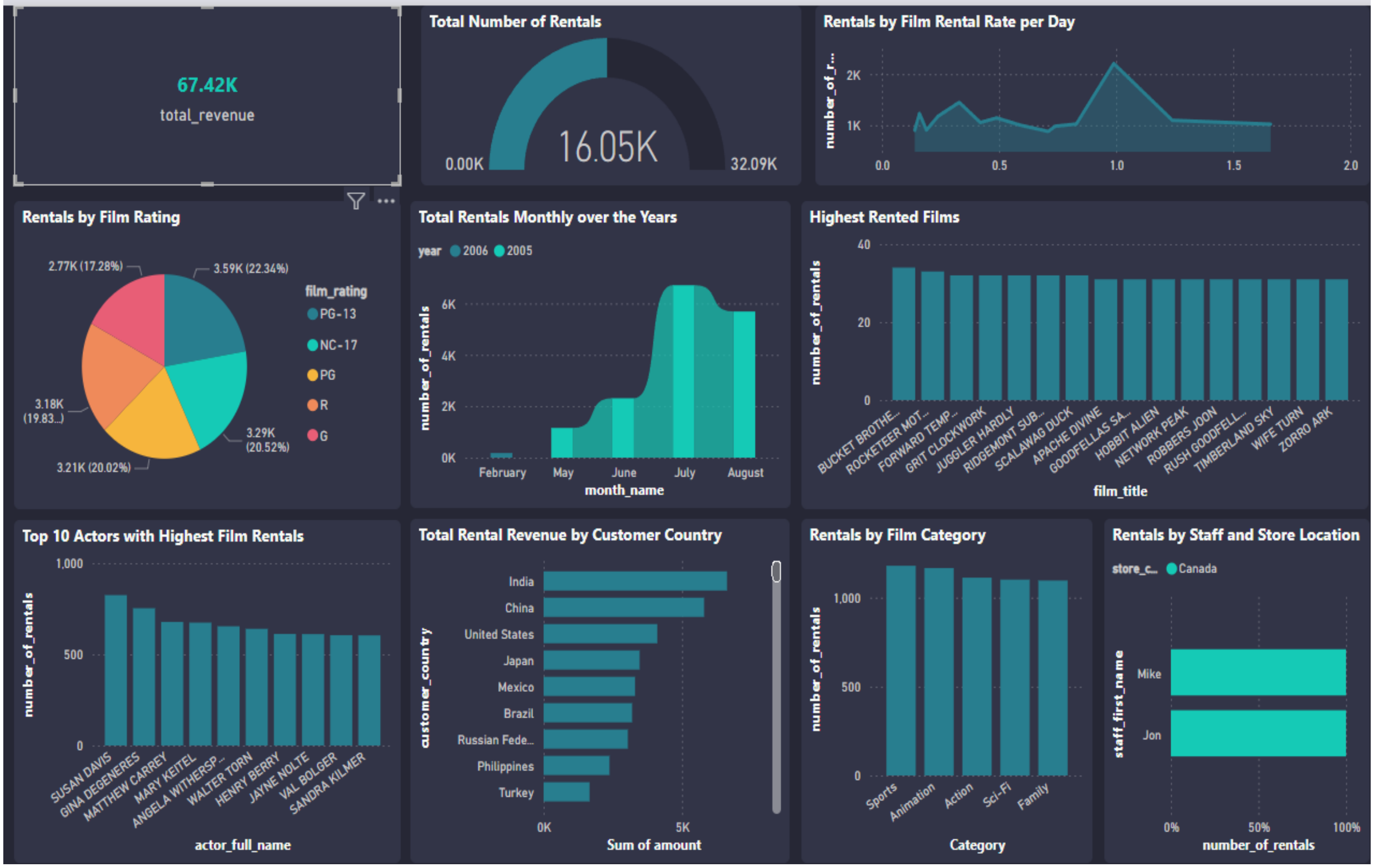
6- What is the trend of rentals over the years?

7- Which films have the highest number of rentals?

8 - Which staff members have the highest rental transactions, and in which store country?

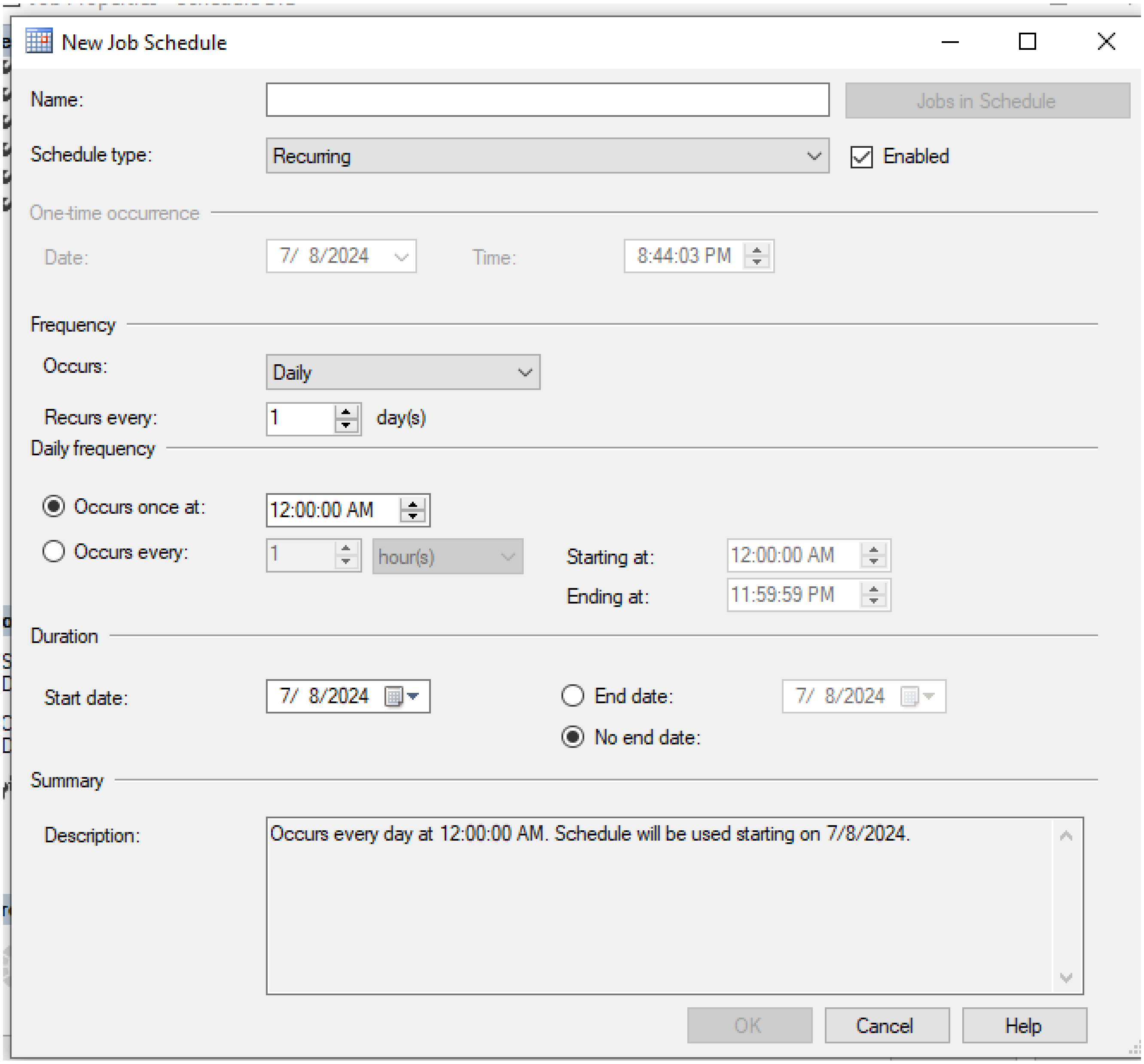
9 – what is the total revenue?

10- what is the total number of rentals?



6- create a SQL Server Agent job

Create a job in SSMS to schedule the main package (ETL_pipeline). Schedule the job to run every day at 12 AM.



New Job Schedule

Name:

Schedule type: Recurring

☑ Enabled

One-time occurrence

Date: 7/ 8/2024

Time: 8:44:03 PM

Frequency

Occurs: Daily

Recurs every: 1 day(s)

Daily frequency

● Occurs once at: 12:00:00 AM

○ Occurs every: 1 hour(s)

Starting at: 12:00:00 AM

Ending at: 11:59:59 PM

Duration

Start date: 7/ 8/2024

○ End date: 7/ 8/2024

● No end date:

Summary

Description: Occurs every day at 12:00:00 AM. Schedule will be used starting on 7/8/2024.