RentalSight: An Online Movie Rental Warehouse

Introduction

In the dynamic landscape of online movie rentals, efficient data management is pivotal for success. Introducing Rentalsight, a comprehensive data warehouse crafted to empower online movie rental stores with insightful analytics, seamless inventory management, and strategic rental oversight.

Deliverables

Revenue Trend Analysis:

- Explore revenue trends over specified time periods.
- Visualize revenue patterns based on movie genres, or ratings.
- Identify peak revenue periods and potential areas for revenue growth.

Inventory Management:

- Track inventory levels in real-time.
- Identify slow-moving or obsolete inventory for strategic decision-making.
- Measure the efficiency of inventory management and return policy

Rental Oversight:

- Analyze rental patterns.
- Optimize rental pricing based on demand and historical data.

User-Friendly Interface:

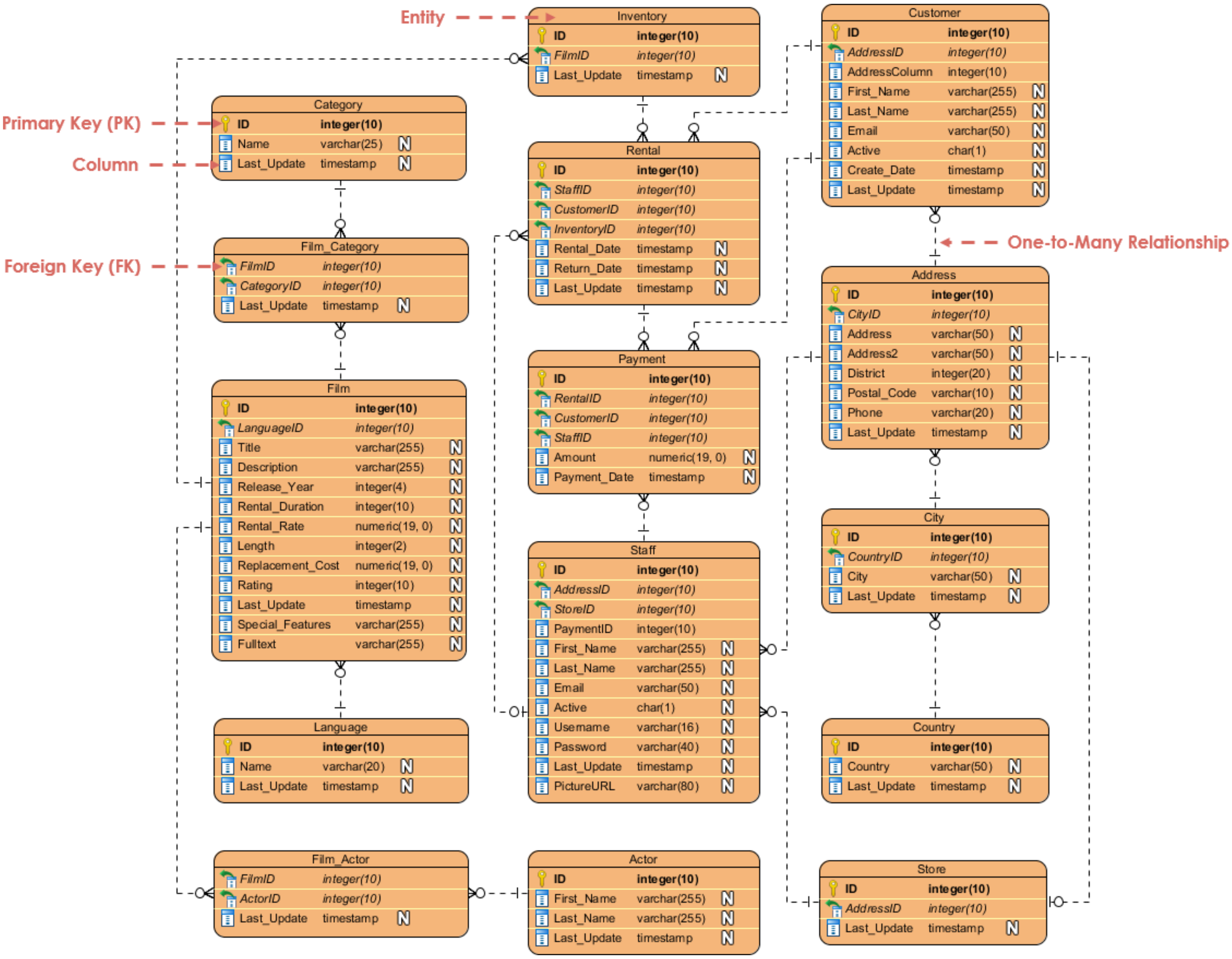
- Intuitive dashboards for easy navigation and quick access to key metrics.
- Customizable reports to tailor insights to specific business requirements.
- Real-time data updates for timely decision-making.

ETL phase

Extraction:

Data was extracted from an online dataset comprised of many individual tables in csv and JSON formats.

Original data can be found in source_data folder. The schema for the original database can be seen here



Inventory

ID	integer(10)	
FilmID	integer(10)	
Last_Update	timestamp	N

Category

ID	integer(10)	
Name	varchar(25)	N
Last_Update	timestamp	N

Film_Category

FilmID	integer(10)	
CategoryID	integer(10)	
Last_Update	timestamp	N

Film

ID	integer(10)	
LanguageID	integer(10)	
Title	varchar(255)	N
Description	varchar(255)	N
Release_Year	integer(4)	N
Rental_Duration	integer(10)	N
Rental_Rate	numeric(19, 0)	N
Length	integer(2)	N
Replacement_Cost	numeric(19, 0)	N
Rating	integer(10)	N
Last_Update	timestamp	N
Special_Features	varchar(255)	N
Fulltext	varchar(255)	N

Language

ID	integer(10)	
Name	varchar(20)	N
Last_Update	timestamp	N

Film_Actor

FilmID	integer(10)	
ActorID	integer(10)	
Last_Update	timestamp	N

Rental

ID	integer(10)	
StaffID	integer(10)	
CustomerID	integer(10)	
InventoryID	integer(10)	
Rental_Date	timestamp	N
Return_Date	timestamp	N
Last_Update	timestamp	N

Payment

ID	integer(10)	
RentalID	integer(10)	
CustomerID	integer(10)	
StaffID	integer(10)	
Amount	numeric(19, 0)	N
Payment_Date	timestamp	N

Staff

ID	integer(10)	
AddressID	integer(10)	
StoreID	integer(10)	
PaymentID	integer(10)	
First_Name	varchar(255)	N
Last_Name	varchar(255)	N
Email	varchar(50)	N
Active	char(1)	N
Username	varchar(16)	N
Password	varchar(40)	N
Last_Update	timestamp	N
PictureURL	varchar(80)	N

Actor

ID	integer(10)	
First_Name	varchar(255)	N
Last_Name	varchar(255)	N
Last_Update	timestamp	N

Customer

ID	integer(10)	
AddressID	integer(10)	
AddressColumn	integer(10)	
First_Name	varchar(255)	N
Last_Name	varchar(255)	N
Email	varchar(50)	N
Active	char(1)	N
Create_Date	timestamp	N
Last_Update	timestamp	N

Address

ID	integer(10)	
CityID	integer(10)	
Address	varchar(50)	N
Address2	varchar(50)	N
District	integer(20)	N
Postal_Code	varchar(10)	N
Phone	varchar(20)	N
Last_Update	timestamp	N

City

ID	integer(10)	
CountryID	integer(10)	
City	varchar(50)	N
Last_Update	timestamp	N

Country

ID	integer(10)	
Country	varchar(50)	N
Last_Update	timestamp	N

Store

ID	integer(10)	
AddressID	integer(10)	
Last_Update	timestamp	N

Transformation:

Data is then transformed using Python and the Pandas library.

The following files were used to individually filter out and transform certain data and save them into csv files located in the main folder:

- transform_address.py: This script reads 3 data files in CSV format as pandas dataframes. It then joins the addresses with the corresponding countries using country_id. “last_update”, “phone”, “address2” are not needed therefore they are discarded from the dataframe. It then joins addresses with cities using city_id and fills in two missing postal codes for “Alberta” and “QLD”. it then saves the output to address.csv
- transform_film.py: reads 4 CSV files, proceeds to drop unneeded columns then joins each film to the corresponding language and then to the correct genre (film_category). It then removes rental_rate column and saves the file.
- transform_customer.csv: reads addresses and customer information. Removes unneeded columns and joins customers to their addresses. formats the strings in “first_name”, “last_name”, and “email”
- transform_rental.csv: reads in rental information, removes unneeded columns, formats rental_date columns, saves day,month, and year information in separate columns. It then establishes primary keys for day_df,month_df,year_df. Calendar library is then imported to encode the month name in the month dataframe.
- transform_store.py: reads in store information and drops many unneeded columns. It then gives a name to the two stores and saves the file.
- The files create_inventory_snapshot.py and create_rental_transactions.py are used to create the two fact tables that will be employed in our analysis later on. They use
- the ETL.ipynb is a jupyter notebook created to execute all transformations, yielding the following csv files found in the main folder: address.csv, customer.csv, store.csv, film.csv, rental.csv, day.csv, year.csv, month.csv, rental_transactions.csv, inventory_snapshot.csv

Loading:

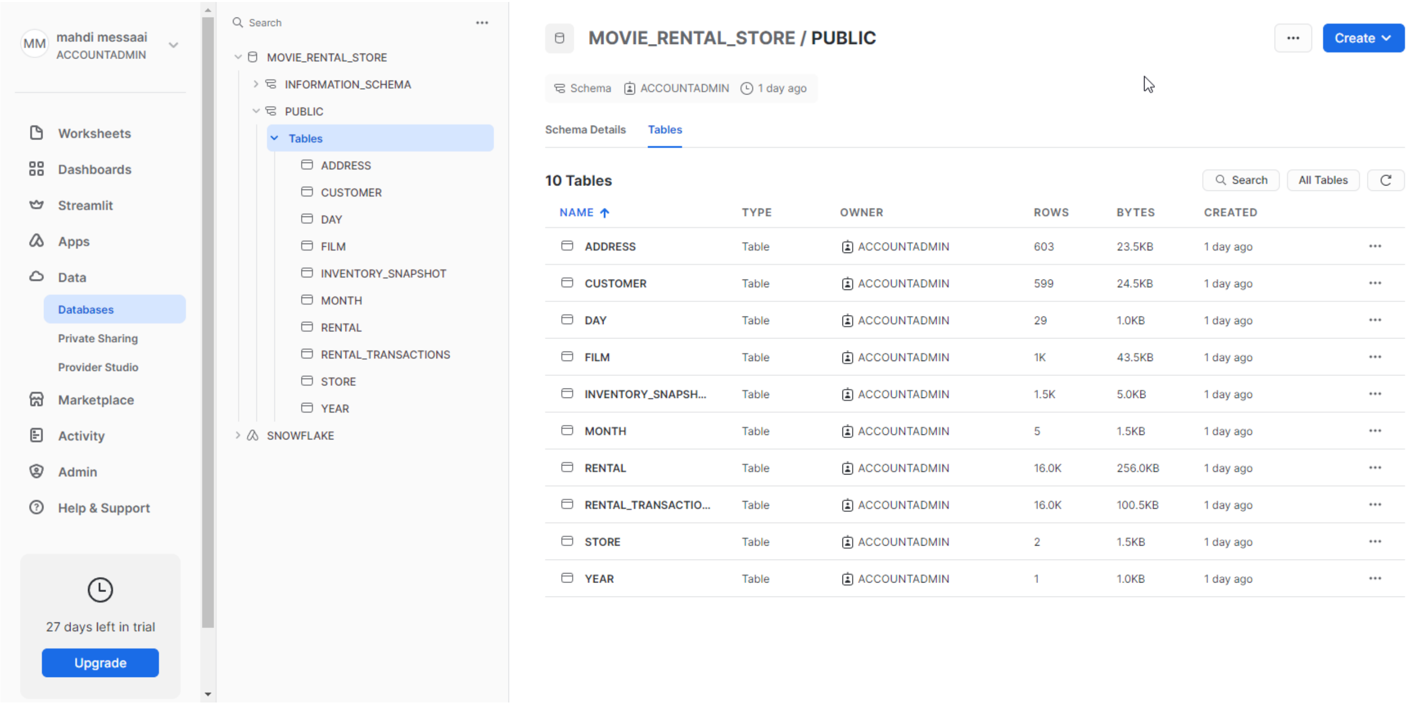
Data is then loaded onto a snowflake ROLAP database.

This database can be accessed through this link:

https://vn58684.west-europe.azure.snowflakecomputing.com

username: slimboyfu

password: Testingtesting1$

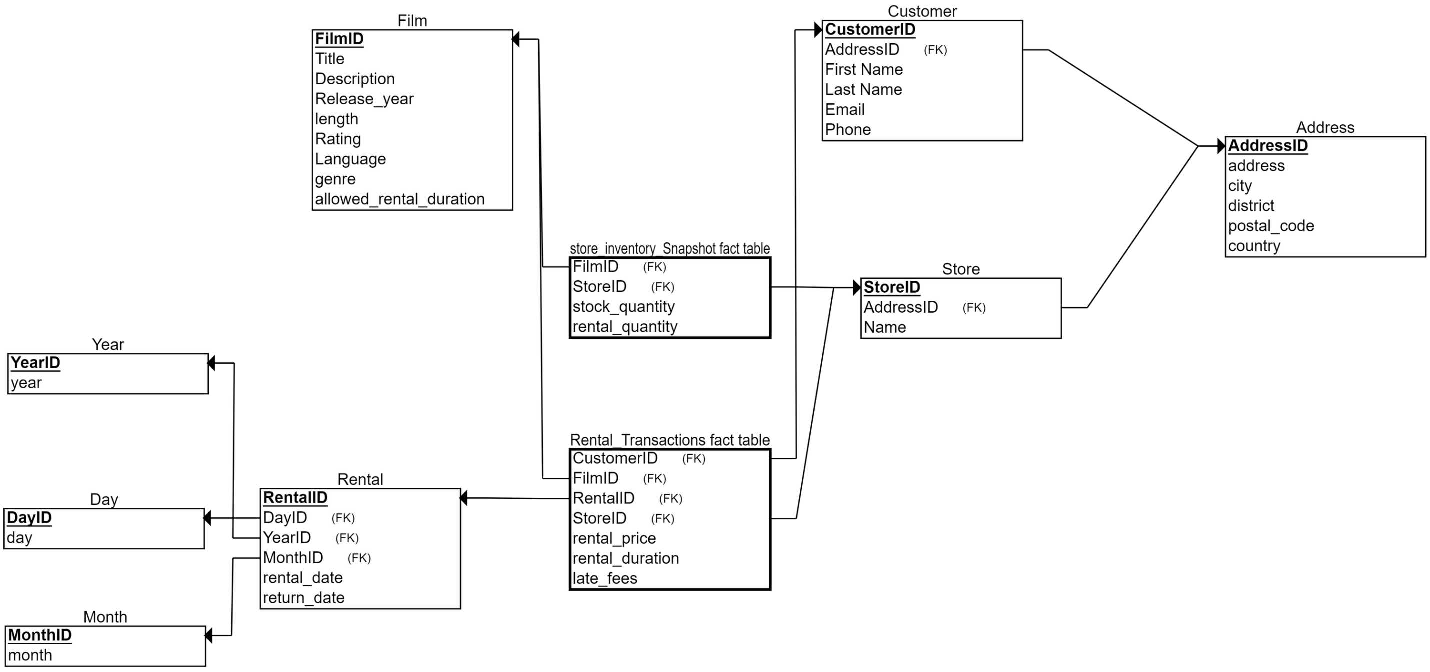


MOVIE_RENTAL_STORE / PUBLIC

10 Tables

NAME	TYPE	OWNER	ROWS	BYTES	CREATED
ADDRESS	Table	ACCOUNTADMIN	603	23.5KB	1 day ago
CUSTOMER	Table	ACCOUNTADMIN	599	24.5KB	1 day ago
DAY	Table	ACCOUNTADMIN	29	1.0KB	1 day ago
FILM	Table	ACCOUNTADMIN	1K	43.5KB	1 day ago
INVENTORY_SNAPSH...	Table	ACCOUNTADMIN	1.5K	5.0KB	1 day ago
MONTH	Table	ACCOUNTADMIN	5	1.5KB	1 day ago
RENTAL	Table	ACCOUNTADMIN	16.0K	256.0KB	1 day ago
RENTAL_TRANSACTIO...	Table	ACCOUNTADMIN	16.0K	100.5KB	1 day ago
STORE	Table	ACCOUNTADMIN	2	1.5KB	1 day ago
YEAR	Table	ACCOUNTADMIN	1	1.0KB	1 day ago

Data Warehousing



Store_inventory_snapshot: used to store historical data on stock and rented quantities by store and movie.

Rental_Transactions: used to store historical data on all transactions occurring in all stores. Its measures are the following:

- rental_price: price to rent a certain movie.
- rental_duration: actual rental duration of a movie by a customer

- late_fees: any incurred late return fees. (By default, 1$ is charged to the customer for every 1 day delay)

We have also identified the following dimensions:

- Film
- Rental
- Year
- Day
- Month
- Customer
- Store
- Address

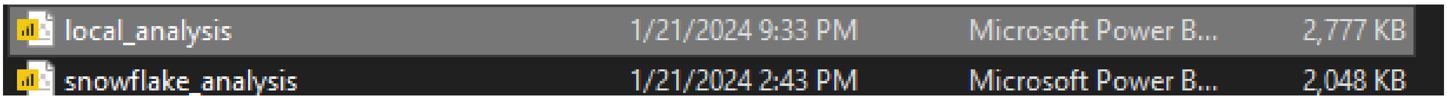
Analysis

Analysis was done using Power BI.

Two PowerBI files are included in the main folder: snowflake_analysis and local_analysis.

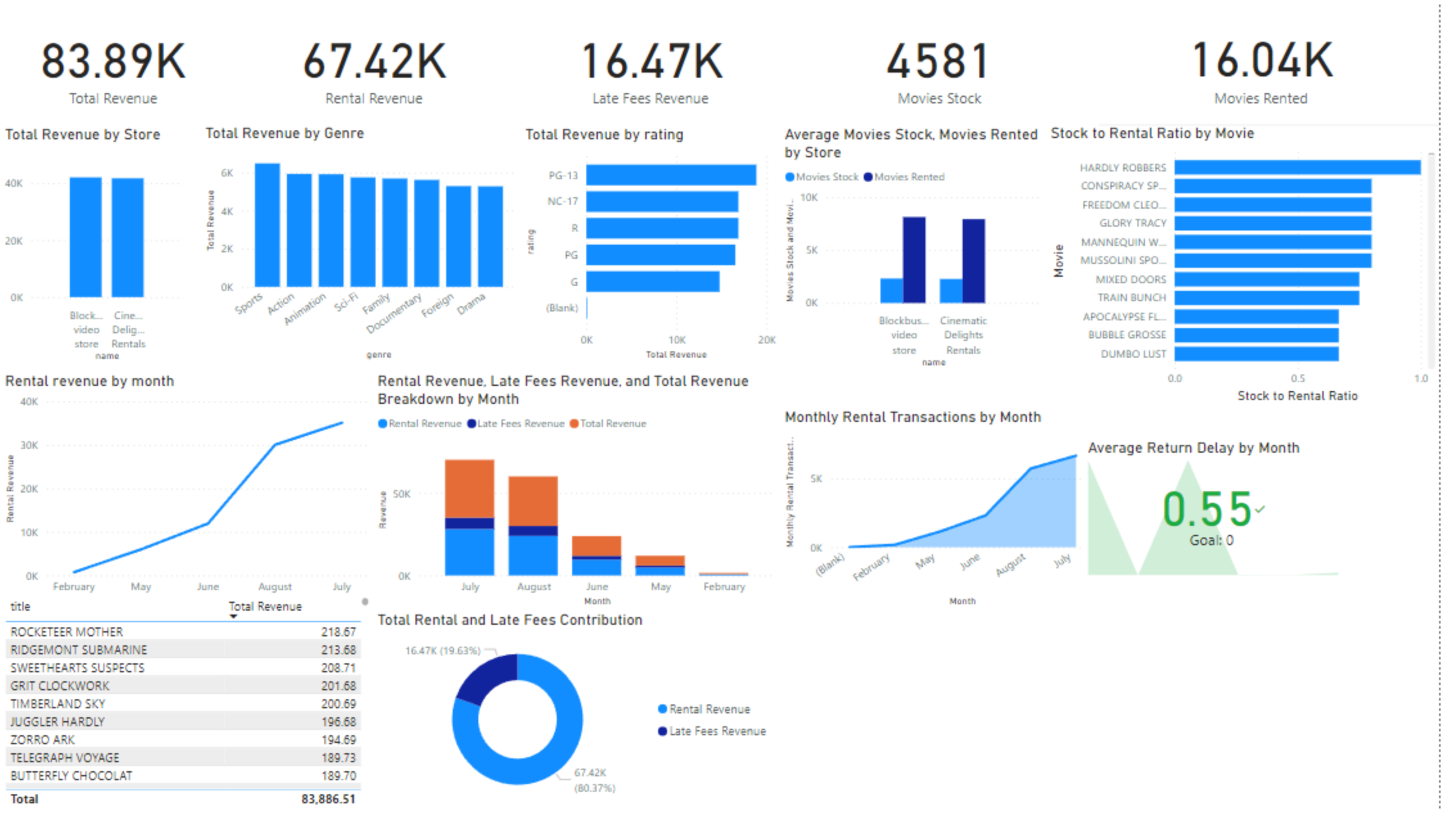
The two are identical. The only difference is snowflake_analysis gets its data from snowflake directly. That requires authentication or else it would fail to load any visualizations.

local_analysis uses the local files in the folder. This one should be used if you wish not to connect to snowflake.



local_analysis	1/21/2024 9:33 PM	Microsoft Power B...	2,777 KB
snowflake_analysis	1/21/2024 2:43 PM	Microsoft Power B...	2,048 KB

Upon loading up local_analysis.pbix. You are introduced to the dashboard.



title	Total Revenue
ROCKETEER MOTHER	218.67
RIDGEMONT SUBMARINE	213.68
SWEETHEARTS SUSPECTS	208.71
GRIT CLOCKWORK	201.68
TIMBERLAND SKY	200.69
JUGGLER HARDLY	196.68
ZORRO ARK	194.69
TELEGRAPH VOYAGE	189.73
BUTTERFLY CHOCOLAT	189.70
Total	**83,886.51**

Conclusion

Few things to note about the project:

- Data was not of the highest quality as can be seen from the visualizations. Barely any difference between the two stores. That made for some rather disappointing visualizations.
- Project could have been much more informative had I added more data and probably added a couple more dimensions.
- Working on this alone was a bit tough
- More complex and interesting KPIs were supposed to be done but were cut out due to lack of time/expertise.