**# \*\*DVD Rental’s Analysis Project\*\***

**### \*\*Difficulty Level: Advanced\*\***

**## \*\*Project Overview\*\***

I have worked on analyzing a dataset of over 16,000 DVD rental records from a DVD Rentals dataset. This project involves extensive querying of customer transactions, actors and films performance, and sales trends by stores, customers, and countries

using PostgreSQL. Through this project, I have tackled various SQL problems, including revenue analysis, customer segmentation, and inventory management.

The project also focuses on solving real-world business problems using structured queries and functions such as window functions, date functions, aggregates, grouping, case statements, string functions, joining tables and much more.

An ERD diagram is included to visually represent the database schema and relationships between tables.

**## \*\*Database Setup & Design\*\***

**### \*\*Schema Structure\*\***

BEGIN;

CREATE TABLE IF NOT EXISTS public.actor

(

actor\_id integer NOT NULL DEFAULT nextval('actor\_actor\_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,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT actor\_pkey PRIMARY KEY (actor\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.address

(

address\_id integer NOT NULL DEFAULT nextval('address\_address\_id\_seq'::regclass),

address character varying(50) COLLATE pg\_catalog."default" NOT NULL,

address2 character varying(50) COLLATE pg\_catalog."default",

district character varying(20) COLLATE pg\_catalog."default" NOT NULL,

city\_id smallint NOT NULL,

postal\_code character varying(10) COLLATE pg\_catalog."default",

phone character varying(20) COLLATE pg\_catalog."default" NOT NULL,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT address\_pkey PRIMARY KEY (address\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.category

(

category\_id integer NOT NULL DEFAULT nextval('category\_category\_id\_seq'::regclass),

name character varying(25) COLLATE pg\_catalog."default" NOT NULL,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT category\_pkey PRIMARY KEY (category\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.city

(

city\_id integer NOT NULL DEFAULT nextval('city\_city\_id\_seq'::regclass),

city character varying(50) COLLATE pg\_catalog."default" NOT NULL,

country\_id smallint NOT NULL,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT city\_pkey PRIMARY KEY (city\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.country

(

country\_id integer NOT NULL DEFAULT nextval('country\_country\_id\_seq'::regclass),

country character varying(50) COLLATE pg\_catalog."default" NOT NULL,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT country\_pkey PRIMARY KEY (country\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.customer

(

customer\_id integer NOT NULL DEFAULT nextval('customer\_customer\_id\_seq'::regclass),

store\_id smallint NOT NULL,

first\_name character varying(45) COLLATE pg\_catalog."default" NOT NULL,

last\_name character varying(45) COLLATE pg\_catalog."default" NOT NULL,

email character varying(50) COLLATE pg\_catalog."default",

address\_id smallint NOT NULL,

activebool boolean NOT NULL DEFAULT true,

create\_date date NOT NULL DEFAULT ('now'::text)::date,

last\_update timestamp without time zone DEFAULT now(),

active integer,

CONSTRAINT customer\_pkey PRIMARY KEY (customer\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.film

(

film\_id integer NOT NULL DEFAULT nextval('film\_film\_id\_seq'::regclass),

title character varying(255) COLLATE pg\_catalog."default" NOT NULL,

description text COLLATE pg\_catalog."default",

release\_year year,

language\_id smallint NOT NULL,

original\_language\_id smallint,

rental\_duration smallint NOT NULL DEFAULT 3,

rental\_rate numeric(4, 2) NOT NULL DEFAULT 4.99,

length smallint,

replacement\_cost numeric(5, 2) NOT NULL DEFAULT 19.99,

rating mpaa\_rating DEFAULT 'G'::mpaa\_rating,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

special\_features text[] COLLATE pg\_catalog."default",

fulltext tsvector NOT NULL,

CONSTRAINT film\_pkey PRIMARY KEY (film\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.film\_actor

(

actor\_id smallint NOT NULL,

film\_id smallint NOT NULL,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT film\_actor\_pkey PRIMARY KEY (actor\_id, film\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.film\_category

(

film\_id smallint NOT NULL,

category\_id smallint NOT NULL,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT film\_category\_pkey PRIMARY KEY (film\_id, category\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.inventory

(

inventory\_id integer NOT NULL DEFAULT nextval('inventory\_inventory\_id\_seq'::regclass),

film\_id smallint NOT NULL,

store\_id smallint NOT NULL,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT inventory\_pkey PRIMARY KEY (inventory\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.language

(

language\_id integer NOT NULL DEFAULT nextval('language\_language\_id\_seq'::regclass),

name character(20) COLLATE pg\_catalog."default" NOT NULL,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT language\_pkey PRIMARY KEY (language\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.nyweather

(

station character varying(255) COLLATE pg\_catalog."default",

station\_name character varying(255) COLLATE pg\_catalog."default",

date date,

prcp numeric(5, 4)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.payment

(

payment\_id integer NOT NULL DEFAULT nextval('payment\_payment\_id\_seq'::regclass),

customer\_id smallint NOT NULL,

staff\_id smallint NOT NULL,

rental\_id integer NOT NULL,

amount numeric(5, 2) NOT NULL,

payment\_date timestamp without time zone NOT NULL,

CONSTRAINT payment\_pkey PRIMARY KEY (payment\_id)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.payment\_p2007\_01

(

payment\_id integer NOT NULL DEFAULT nextval('payment\_payment\_id\_seq'::regclass),

customer\_id smallint NOT NULL,

staff\_id smallint NOT NULL,

rental\_id integer NOT NULL,

amount numeric(5, 2) NOT NULL,

payment\_date timestamp without time zone NOT NULL

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.payment\_p2007\_02

(

payment\_id integer NOT NULL DEFAULT nextval('payment\_payment\_id\_seq'::regclass),

customer\_id smallint NOT NULL,

staff\_id smallint NOT NULL,

rental\_id integer NOT NULL,

amount numeric(5, 2) NOT NULL,

payment\_date timestamp without time zone NOT NULL

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.payment\_p2007\_03

(

payment\_id integer NOT NULL DEFAULT nextval('payment\_payment\_id\_seq'::regclass),

customer\_id smallint NOT NULL,

staff\_id smallint NOT NULL,

rental\_id integer NOT NULL,

amount numeric(5, 2) NOT NULL,

payment\_date timestamp without time zone NOT NULL

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.payment\_p2007\_04

(

payment\_id integer NOT NULL DEFAULT nextval('payment\_payment\_id\_seq'::regclass),

customer\_id smallint NOT NULL,

staff\_id smallint NOT NULL,

rental\_id integer NOT NULL,

amount numeric(5, 2) NOT NULL,

payment\_date timestamp without time zone NOT NULL

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.payment\_p2007\_05

(

payment\_id integer NOT NULL DEFAULT nextval('payment\_payment\_id\_seq'::regclass),

customer\_id smallint NOT NULL,

staff\_id smallint NOT NULL,

rental\_id integer NOT NULL,

amount numeric(5, 2) NOT NULL,

payment\_date timestamp without time zone NOT NULL

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.payment\_p2007\_06

(

payment\_id integer NOT NULL DEFAULT nextval('payment\_payment\_id\_seq'::regclass),

customer\_id smallint NOT NULL,

staff\_id smallint NOT NULL,

rental\_id integer NOT NULL,

amount numeric(5, 2) NOT NULL,

payment\_date timestamp without time zone NOT NULL

)

WITH (

OIDS = FALSE

);

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)

)

WITH (

OIDS = FALSE

);

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)

)

WITH (

OIDS = FALSE

);

CREATE TABLE IF NOT EXISTS public.store

(

store\_id integer NOT NULL DEFAULT nextval('store\_store\_id\_seq'::regclass),

manager\_staff\_id smallint NOT NULL,

address\_id smallint NOT NULL,

last\_update timestamp without time zone NOT NULL DEFAULT now(),

CONSTRAINT store\_pkey PRIMARY KEY (store\_id)

)

WITH (

OIDS = FALSE

);

ALTER TABLE IF EXISTS public.address

ADD CONSTRAINT address\_city\_id\_fkey FOREIGN KEY (city\_id)

REFERENCES public.city (city\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_city\_id

ON public.address(city\_id);

ALTER TABLE IF EXISTS public.city

ADD CONSTRAINT city\_country\_id\_fkey FOREIGN KEY (country\_id)

REFERENCES public.country (country\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_country\_id

ON public.city(country\_id);

ALTER TABLE IF EXISTS public.customer

ADD CONSTRAINT customer\_address\_id\_fkey FOREIGN KEY (address\_id)

REFERENCES public.address (address\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_address\_id

ON public.customer(address\_id);

ALTER TABLE IF EXISTS public.customer

ADD CONSTRAINT customer\_store\_id\_fkey FOREIGN KEY (store\_id)

REFERENCES public.store (store\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_store\_id

ON public.customer(store\_id);

ALTER TABLE IF EXISTS public.film

ADD CONSTRAINT film\_language\_id\_fkey FOREIGN KEY (language\_id)

REFERENCES public.language (language\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_language\_id

ON public.film(language\_id);

ALTER TABLE IF EXISTS public.film

ADD CONSTRAINT film\_original\_language\_id\_fkey FOREIGN KEY (original\_language\_id)

REFERENCES public.language (language\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_original\_language\_id

ON public.film(original\_language\_id);

ALTER TABLE IF EXISTS public.film\_actor

ADD CONSTRAINT film\_actor\_actor\_id\_fkey FOREIGN KEY (actor\_id)

REFERENCES public.actor (actor\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.film\_actor

ADD CONSTRAINT film\_actor\_film\_id\_fkey FOREIGN KEY (film\_id)

REFERENCES public.film (film\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_film\_id

ON public.film\_actor(film\_id);

ALTER TABLE IF EXISTS public.film\_category

ADD CONSTRAINT film\_category\_category\_id\_fkey FOREIGN KEY (category\_id)

REFERENCES public.category (category\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.film\_category

ADD CONSTRAINT film\_category\_film\_id\_fkey FOREIGN KEY (film\_id)

REFERENCES public.film (film\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.inventory

ADD CONSTRAINT inventory\_film\_id\_fkey FOREIGN KEY (film\_id)

REFERENCES public.film (film\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.inventory

ADD CONSTRAINT inventory\_store\_id\_fkey FOREIGN KEY (store\_id)

REFERENCES public.store (store\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.payment

ADD CONSTRAINT payment\_customer\_id\_fkey FOREIGN KEY (customer\_id)

REFERENCES public.customer (customer\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_customer\_id

ON public.payment(customer\_id);

ALTER TABLE IF EXISTS public.payment

ADD CONSTRAINT payment\_rental\_id\_fkey FOREIGN KEY (rental\_id)

REFERENCES public.rental (rental\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE SET NULL;

ALTER TABLE IF EXISTS public.payment

ADD CONSTRAINT payment\_staff\_id\_fkey FOREIGN KEY (staff\_id)

REFERENCES public.staff (staff\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_staff\_id

ON public.payment(staff\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_01

ADD CONSTRAINT payment\_p2007\_01\_customer\_id\_fkey FOREIGN KEY (customer\_id)

REFERENCES public.customer (customer\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_01\_customer\_id

ON public.payment\_p2007\_01(customer\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_01

ADD CONSTRAINT payment\_p2007\_01\_rental\_id\_fkey FOREIGN KEY (rental\_id)

REFERENCES public.rental (rental\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.payment\_p2007\_01

ADD CONSTRAINT payment\_p2007\_01\_staff\_id\_fkey FOREIGN KEY (staff\_id)

REFERENCES public.staff (staff\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_01\_staff\_id

ON public.payment\_p2007\_01(staff\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_02

ADD CONSTRAINT payment\_p2007\_02\_customer\_id\_fkey FOREIGN KEY (customer\_id)

REFERENCES public.customer (customer\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_02\_customer\_id

ON public.payment\_p2007\_02(customer\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_02

ADD CONSTRAINT payment\_p2007\_02\_rental\_id\_fkey FOREIGN KEY (rental\_id)

REFERENCES public.rental (rental\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.payment\_p2007\_02

ADD CONSTRAINT payment\_p2007\_02\_staff\_id\_fkey FOREIGN KEY (staff\_id)

REFERENCES public.staff (staff\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_02\_staff\_id

ON public.payment\_p2007\_02(staff\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_03

ADD CONSTRAINT payment\_p2007\_03\_customer\_id\_fkey FOREIGN KEY (customer\_id)

REFERENCES public.customer (customer\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_03\_customer\_id

ON public.payment\_p2007\_03(customer\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_03

ADD CONSTRAINT payment\_p2007\_03\_rental\_id\_fkey FOREIGN KEY (rental\_id)

REFERENCES public.rental (rental\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.payment\_p2007\_03

ADD CONSTRAINT payment\_p2007\_03\_staff\_id\_fkey FOREIGN KEY (staff\_id)

REFERENCES public.staff (staff\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_03\_staff\_id

ON public.payment\_p2007\_03(staff\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_04

ADD CONSTRAINT payment\_p2007\_04\_customer\_id\_fkey FOREIGN KEY (customer\_id)

REFERENCES public.customer (customer\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_04\_customer\_id

ON public.payment\_p2007\_04(customer\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_04

ADD CONSTRAINT payment\_p2007\_04\_rental\_id\_fkey FOREIGN KEY (rental\_id)

REFERENCES public.rental (rental\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.payment\_p2007\_04

ADD CONSTRAINT payment\_p2007\_04\_staff\_id\_fkey FOREIGN KEY (staff\_id)

REFERENCES public.staff (staff\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_04\_staff\_id

ON public.payment\_p2007\_04(staff\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_05

ADD CONSTRAINT payment\_p2007\_05\_customer\_id\_fkey FOREIGN KEY (customer\_id)

REFERENCES public.customer (customer\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_05\_customer\_id

ON public.payment\_p2007\_05(customer\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_05

ADD CONSTRAINT payment\_p2007\_05\_rental\_id\_fkey FOREIGN KEY (rental\_id)

REFERENCES public.rental (rental\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.payment\_p2007\_05

ADD CONSTRAINT payment\_p2007\_05\_staff\_id\_fkey FOREIGN KEY (staff\_id)

REFERENCES public.staff (staff\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_05\_staff\_id

ON public.payment\_p2007\_05(staff\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_06

ADD CONSTRAINT payment\_p2007\_06\_customer\_id\_fkey FOREIGN KEY (customer\_id)

REFERENCES public.customer (customer\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_06\_customer\_id

ON public.payment\_p2007\_06(customer\_id);

ALTER TABLE IF EXISTS public.payment\_p2007\_06

ADD CONSTRAINT payment\_p2007\_06\_rental\_id\_fkey FOREIGN KEY (rental\_id)

REFERENCES public.rental (rental\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.payment\_p2007\_06

ADD CONSTRAINT payment\_p2007\_06\_staff\_id\_fkey FOREIGN KEY (staff\_id)

REFERENCES public.staff (staff\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS idx\_fk\_payment\_p2007\_06\_staff\_id

ON public.payment\_p2007\_06(staff\_id);

ALTER TABLE IF EXISTS public.rental

ADD CONSTRAINT rental\_customer\_id\_fkey FOREIGN KEY (customer\_id)

REFERENCES public.customer (customer\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.rental

ADD CONSTRAINT rental\_inventory\_id\_fkey FOREIGN KEY (inventory\_id)

REFERENCES public.inventory (inventory\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_fk\_inventory\_id

ON public.rental(inventory\_id);

ALTER TABLE IF EXISTS public.rental

ADD CONSTRAINT rental\_staff\_id\_fkey FOREIGN KEY (staff\_id)

REFERENCES public.staff (staff\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.staff

ADD CONSTRAINT staff\_address\_id\_fkey FOREIGN KEY (address\_id)

REFERENCES public.address (address\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.staff

ADD CONSTRAINT staff\_store\_id\_fkey FOREIGN KEY (store\_id)

REFERENCES public.store (store\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.store

ADD CONSTRAINT store\_address\_id\_fkey FOREIGN KEY (address\_id)

REFERENCES public.address (address\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.store

ADD CONSTRAINT store\_manager\_staff\_id\_fkey FOREIGN KEY (manager\_staff\_id)

REFERENCES public.staff (staff\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS idx\_unq\_manager\_staff\_id

ON public.store(manager\_staff\_id);

END;

**## \*\*Solving Business Problems\*\***

### Solutions Implemented:

**/\* 1. Name the top 5 actors that are in movies with the most rentals. How much revenue did each actor generate in total.**

**Out of that total, get the percentage of total each actor contributed to total revenue.**

**For the Actor who has the highest percentage give He or She the title "Best Selling Actor”\*/**

-- join tables and get the number of orders for each actor and the amount generated from each actor in sales

WITH actor\_count AS (

SELECT A.actor\_id, CONCAT(A.first\_name,' ' ,A.last\_name ) full\_name, A.last\_name, COUNT(A.actor\_id) as actor\_cnt, SUM(amount) actor\_revenue

FROM customer as C INNER JOIN rental as R ON C.customer\_id = R.customer\_id

INNER JOIN inventory as I on I.inventory\_id = R.inventory\_id

INNER JOIN film\_actor as FA ON FA.film\_id = I.film\_id INNER JOIN

actor as A ON A.actor\_id = FA.actor\_id INNER JOIN payment as P ON

R.rental\_id = P.rental\_id

GROUP BY A.actor\_id, A.first\_name, A.last\_name),

--Rank actors based on number of rentals by customers and get the total revenue of all actors

ranking AS(

SELECT full\_name, actor\_cnt, actor\_revenue, RANK() OVER( ORDER BY actor\_cnt DESC) as actor\_rank

FROM actor\_count),

--get the top 5 actors and total\_revenue

total\_revenue AS (

SELECT full\_name, actor\_cnt, actor\_revenue, SUM(actor\_revenue) OVER() AS total\_revenue

FROM ranking

WHERE actor\_rank <=5)

--get the percent of total for total revenue and get the rank of each actor

SELECT full\_name, actor\_cnt, actor\_revenue, CONCAT(ROUND((actor\_revenue\*100/total\_revenue)::decimal,2),'%') as percent\_of\_total,

CASE WHEN RANK() OVER(ORDER BY (actor\_revenue\*100/total\_revenue) DESC) = 1 THEN 'Best Selling Actor'

WHEN RANK() OVER(ORDER BY (actor\_revenue\*100/total\_revenue) DESC) = 2 THEN 'Second Best Selling Actor'

WHEN RANK() OVER(ORDER BY (actor\_revenue\*100/total\_revenue) DESC) = 3 THEN 'Third Best Selling Actor'

WHEN RANK() OVER(ORDER BY (actor\_revenue\*100/total\_revenue) DESC) = 4 THEN 'Fourth Best Selling Actor'

WHEN RANK() OVER(ORDER BY (actor\_revenue\*100/total\_revenue) DESC) = 5 THEN 'Fifth Best Selling Actor'

END as selling\_rank

FROM total\_revenue

SELECT \*

FROM rental

ORDER BY rental\_date

**/\* 2.**

**Calculate the monthly percentage change of total revenue over the last 5 months for each category.**

**Retrieve the month for each category that had the highest month-over-month percentage change.**

**\*/**

--filter data to last 5 months, get the total revenue for each category and previous month revenue

WITH cte AS(

SELECT CA.category\_id as category\_id, name as Category\_name,

TO\_CHAR(payment\_date, 'Month') as month, SUM(amount) as Total\_revenue,

EXTRACT(month FROM payment\_date) as month\_num,

LAG(SUM(amount)) OVER(PARTITION BY name order by EXTRACT(month FROM payment\_date)) previous\_amount

FROM inventory as I INNER JOIN rental as R

ON I.inventory\_id = R.inventory\_id INNER JOIN payment as P

ON P.rental\_id = R.rental\_id INNER JOIN film\_category as FA

ON FA.film\_id = I.film\_id INNER JOIN category as CA on

FA.category\_id = CA.category\_id

WHERE payment\_date BETWEEN

(SELECT max(payment\_date) FROM payment) - INTERVAL '5 MONTHS' AND

(SELECT max(payment\_date) FROM payment)

GROUP BY CA.category\_id, name, TO\_CHAR(payment\_date, 'Month'), EXTRACT(month FROM payment\_date)),

--get the month over month percentage of each category and rank the month over month percentage for each month

ranking as (

SELECT category\_name, month,month\_num,SUM(Total\_revenue) as Total\_revenue,

COALESCE(SUM(previous\_amount),0) previous\_amount,

COALESCE((SUM(total\_revenue)-SUM(previous\_amount))/SUM(previous\_amount)\*100,0) percentage\_change\_over\_time,

RANK() OVER(PARTITION BY category\_name ORDER BY (SUM(total\_revenue)-SUM(previous\_amount))/SUM(previous\_amount)\*100 DESC)

FROM cte

GROUP BY category\_id,category\_name,month, month\_num

ORDER BY category\_name, month\_num)

--select the month with the highest month over month percentage change

SELECT category\_name,month,

CONCAT('$',Total\_revenue) as Total\_revenue, CONCAT('$',previous\_amount) as Previous\_amount,

CONCAT(ROUND(percentage\_change\_over\_time::decimal,2), '%') as percentage\_change\_over\_time

FROM ranking

WHERE rank = 2

**/\* 3.**

**What is the average duration of all DVD's that were rented for each rating.**

**If the Average duration is above a hour and a half say "Customers prefer extended movies"**

**if below an hour and a half say "Customers prefer short films"**

**\*/**

--get the avg duration of films for each rating

WITH rating AS(

SELECT rating, AVG(length/60) as avg\_duration\_of\_film

FROM film as F INNER JOIN inventory as I

ON F.film\_id = I.film\_id INNER JOIN rental as R

ON I.inventory\_id = R.inventory\_id INNER JOIN

payment as P ON R.rental\_id = P.rental\_id

GROUP BY rating)

--determine if customers prefer short films or extended films for each rating

SELECT rating, ROUND(avg\_duration\_of\_film,3) as avg\_duration\_of\_film,

CASE WHEN avg\_duration\_of\_film < 1.5 THEN 'Customers prefer short films' ELSE 'Customers prefer extended movies' END

as customers\_preference

FROM rating

**/\* 4.**

**Create a pivot table that show the months in columns and the stores in rows.**

**The values within the columns should reflect the total payment amount of each month.**

**\*/**

--retrieve the columns from the table stores you created. Add all the months together to get the total amount of each store.

SELECT

stores.store\_id,

CONCAT('$',SUM(January)) as January,

CONCAT('$',SUM(February)) as February,

CONCAT('$',SUM(March)) as March,

CONCAT('$',SUM(April)) as April,

CONCAT('$',SUM(May)) as May,

CONCAT('$',SUM(January)+SUM(February)+SUM(March)+SUM(April)+SUM(May)) as Total

FROM

--get the total payment amount for each month, name each date number by the month date name, group by store\_id

(SELECT

ST.store\_id,

CASE WHEN EXTRACT(MONTH FROM payment\_date) = 1 THEN SUM(amount) END as January,

CASE WHEN EXTRACT(MONTH FROM payment\_date) = 2 THEN SUM(amount) END as February,

CASE WHEN EXTRACT(MONTH FROM payment\_date) = 3 THEN SUM(amount) END as March,

CASE WHEN EXTRACT(MONTH FROM payment\_date) = 4 THEN SUM(amount) END as April,

CASE WHEN EXTRACT(MONTH FROM payment\_date) = 5 THEN SUM(amount) END as May

FROM customer as C INNER JOIN address as A

ON C.address\_id = A.address\_id INNER JOIN city as CI

ON A.city\_id = CI.city\_id INNER JOIN country as CO

ON CI.country\_id = CO.country\_id INNER JOIN payment as P

ON C.customer\_id = P.customer\_id INNER JOIN rental AS R

ON P.rental\_id = R.rental\_id INNER Join staff as S ON

R.staff\_id = S.staff\_id INNER JOIN store as ST ON S.store\_id = ST.store\_id

GROUP BY ST.store\_id, payment\_date) as stores

GROUP BY stores.store\_id

ORDER BY stores.store\_id

**/\* 5.**

**Inventory Stock Alerts**

**Query products with stock levels below a certain threshold (e.g., less than 3 units).**

**Challenge: Include last restock date.**

**\*/**

--count # of films by title in each store. Find the last\_update date of the inventory, filter table by stock less than 3

SELECT title,store\_id, COUNT(I.film\_id) as inventory\_stock, MAX(I.last\_update) as last\_update

FROM inventory as I INNER JOIN film as F ON

I.film\_id = F.film\_id

GROUP BY store\_id, title

HAVING COUNT(I.film\_id) < 3

ORDER BY 2,3

**/\* 6.**

**get the total amount of customers for each country whose country name start with either an 'A' or 'S'**

**What country has the most customers?**

**\*/**

--get the count of customers with names that start with 'a' or 's', then rank countries by number of customers

WITH customers\_by\_country AS (

SELECT country, count(C.customer\_id) count\_of\_customers, RANK() OVER(ORDER BY count(C.customer\_id) DESC ) ranking

from customer as C INNER JOIN address as A

ON C.address\_id = A.address\_id INNER JOIN city as CI

ON A.city\_id = CI.city\_id INNER Join country as CO

ON CI.country\_id = CO.country\_id

WHERE country LIKE 'A%' OR country LIKE 'S%'

GROUP BY country

ORDER BY country)

**## \*\*Learning Outcomes\*\***

This project enabled me to:

- Design and implement a normalized database schema.

- Use advanced SQL techniques, including window functions, subqueries, joins, case statements, date functions, and more.

- Conduct in-depth business analysis using SQL.

- Optimize query performance and handle large datasets efficiently.

---

**## \*\*Conclusion\*\***

This advanced SQL project successfully demonstrates my ability to solve real-world problems using structured queries. From analyzing top performing countries and stores to optimizing inventory and logistics, the project provides valuable insights into operational challenges and solutions.

By completing this project, I have gained a deeper understanding of how SQL can be used to tackle complex data problems and drive business decision-making.