Project 1
----

1. **DML/DDL**: The dvdrental db already has a pre-populated data in it, but let's assume that the business is still running in which case we need to not only analyze existing data but also maintain the database mainly by INSERTing data for new rentals and UPDATEing the db for existing rentals--i.e implementing DML (Data Manipulation Language). To this effect,

    1. Write ALL the queries we need to **rent** out a given movie. (**Hint**: these are the business logics that go into this task: first confirm that the given movie is in stock, and then INSERT a row into the rental and the payment tables. You may also need to check whether the customer has an outstanding balance or an overdue rental before allowing him/her to rent a new DVD).
    2. write ALL the queries we need to process return of a rented movie. (**Hint**: update the rental table and add the return date by first identifying the rental_id to update based on the inventory_id of the movie being returned.)

1. **DQL**: Now that we have an up-to-date database, let's write some queries and analyze the data to understand how our DVD rental business is performing so far.

    1. Which movie genres are the most and least popular? And how much revenue have they each generated for the business?
    2. What are the top 10 most popular movies? And how many times have they each been rented out thus far?
    3. Which genres have the highest and the lowest average rental rate?
    4. How many rented movies were returned late? Is this somehow correlated with the genre of a movie?
    5. What are the top 5 cities that rent the most movies? How about in terms of total sales volume?
    6. let's say you want to give discounts as a reward to your loyal customers and those who return movies they rented on time. So, who are your 10 best customers in this respect?
    7. What are the 10 best rated movies? Is customer rating somehow correlated with revenue? Which actors have acted in most number of the most popular or highest rated movies?
    8. Rentals and hence revenues have been falling behind among young families. In order to reverse this, you wish to target all family movies for a promotion. Identify all movies categorized as *family* films.
    9. How much revenue has each store generated so far?
    10. As a data analyst for the DVD rental business, you would like to have an easy way of viewing the Top 5 genres by average revenue. Write the query to get list of the top 5 genres in average revenue in descending order and create a **view** for it?

Answers

```
--1A ****************************************************

WITH T1 AS (SELECT
F.FILM_ID,F.TITLE,I.INVENTORY_ID,R.RENTAL_ID,F.RENTAL_DURATION,EXTRACT(DAY
FROM (R.RETURN_DATE-R.RENTAL_DATE)) AS DIFF,R.RETURN_DATE

        FROM FILM F

        INNER JOIN INVENTORY I

        USING (FILM_ID)

        INNER JOIN RENTAL R

        USING (INVENTORY_ID)

                )

SELECT *

FROM T1

WHERE T1.RENTAL_DURATION >= DIFF

AND T1.TITLE = 'African Egg'

AND T1.RETURN_DATE IS NOT NULL

--existing cusomer_id=6

INSERT INTO RENTAL
(RENTAL_ID,RENTAL_DATE,INVENTORY_ID,CUSTOMER_ID,RETURN_DATE,STAFF_ID)

VALUES(16050,CURRENT_TIMESTAMP,25,6,NULL,2)

INSERT INTO PAYMENT
(PAYMENT_ID,CUSTOMER_ID,STAFF_ID,RENTAL_ID,AMOUNT,PAYMENT_DATE)

VALUES(32099,6,2,16050,4.99,CURRENT_TIMESTAMP)



--1B*******************************************************************************

SELECT *
```

```
FROM RENTAL

WHERE RENTAL_ID=16050

AND RETURN_DATE IS NULL


UPDATE RENTAL

SET RETURN_DATE = CURRENT_TIMESTAMP

WHERE INVENTORY_ID=25


--
2B********************************************************************************

WITH movie_rental_counts AS (

    SELECT

        f.film_id,

        f.title AS movie_title,

        COUNT(r.rental_id) AS rental_count,

        DENSE_RANK() OVER (ORDER BY COUNT(r.rental_id) DESC) AS rental_rank

    FROM

        film f

    JOIN

        inventory i ON f.film_id = i.film_id

    JOIN

        rental r ON i.inventory_id = r.inventory_id

    GROUP BY

        f.film_id, f.title
)
SELECT
   rental_rank,
          movie_title,
   rental_count
FROM
   movie_rental_counts
WHERE
   rental_rank <= 10
ORDER BY
   rental_rank;

--2C**************************************************************************

SELECT C.NAME AS GENRE,AVG(F.RENTAL_RATE) AS AVGR
FROM CATEGORY C
INNER JOIN FILM_CATEGORY FC
USING (CATEGORY_ID)
INNER JOIN FILM F
USING (FILM_ID)
GROUP BY 1
ORDER BY 2 DESC

--2E******************************************************************************
```

```
SELECT
    city.city AS city_name,
    COUNT(rental.rental_id) AS rental_count,
    SUM(payment.amount) AS total_sales
FROM
    city
JOIN
    address ON city.city_id = address.city_id
JOIN
    customer ON address.address_id = customer.address_id
JOIN
    rental ON customer.customer_id = rental.customer_id
JOIN
    payment ON rental.rental_id = payment.rental_id
GROUP BY
    city.city
ORDER BY
    rental_count DESC,
    total_sales DESC
LIMIT 5;

--2F******************************************************************************
FROM CUSTOMER C
INNER JOIN RENTAL R
```

```
USING (CUSTOMER_ID)

INNER JOIN INVENTORY I

USING (INVENTORY_ID)

INNER JOIN FILM F

USING (FILM_ID)

WHERE EXTRACT(DAY FROM (R.RETURN_DATE - R.RENTAL_DATE)) <=
F.RENTAL_DURATION

LIMIT 10



--2J*********************************************

CREATE VIEW top_5_genres_by_avg_revenue AS

SELECT

    category.name AS genre,

    AVG(payment.amount) AS avg_revenue

FROM

    category

LEFT JOIN

    film_category ON category.category_id = film_category.category_id

LEFT JOIN

    film ON film_category.film_id = film.film_id

LEFT JOIN

    inventory ON film.film_id = inventory.film_id

LEFT JOIN

    rental ON inventory.inventory_id = rental.inventory_id
```

```
LEFT JOIN
    payment ON rental.rental_id = payment.rental_id
GROUP BY
    category.name
ORDER BY
    avg_revenue DESC
LIMIT 5;


SELECT *
FROM TOP_5_GENRES_BY_AVG_REVENUE
```