

Lakshmipriya Anil (TLPoo52)

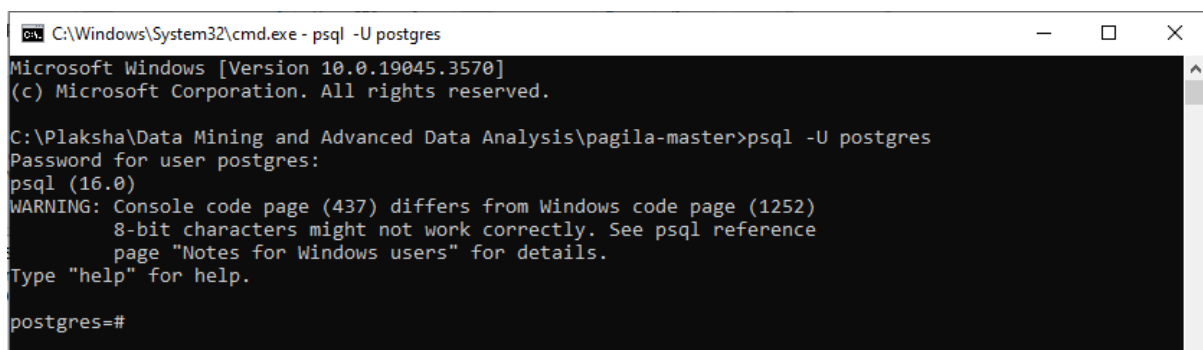
## DATA MINING AND ADVANCED DATA ANALYSIS

### Assignment 1

#### **DATABASE SETUP:**

1] Run psql as user postgres

```
psql -U postgres
```

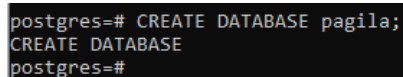
A screenshot of a Windows command prompt window. The title bar shows 'C:\Windows\System32\cmd.exe - psql -U postgres'. The window content shows the following text: 'Microsoft Windows [Version 10.0.19045.3570] (c) Microsoft Corporation. All rights reserved. C:\Plaksha\Data Mining and Advanced Data Analysis\pagila-master>psql -U postgres Password for user postgres: psql (16.0) WARNING: Console code page (437) differs from Windows code page (1252) 8-bit characters might not work correctly. See psql reference page "Notes for Windows users" for details. Type "help" for help. postgres=#'. The password field is masked with asterisks.

```
C:\Windows\System32\cmd.exe - psql -U postgres
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Plaksha\Data Mining and Advanced Data Analysis\pagila-master>psql -U postgres
Password for user postgres:
psql (16.0)
WARNING: Console code page (437) differs from Windows code page (1252)
8-bit characters might not work correctly. See psql reference
page "Notes for Windows users" for details.
Type "help" for help.
postgres=#
```

2] Create a new database called pagila

```
CREATE DATABASE pagila;
```

A screenshot of a psql command prompt. The text shows: 'postgres=# CREATE DATABASE pagila; CREATE DATABASE postgres=#'.

```
postgres=# CREATE DATABASE pagila;
CREATE DATABASE
postgres=#
```

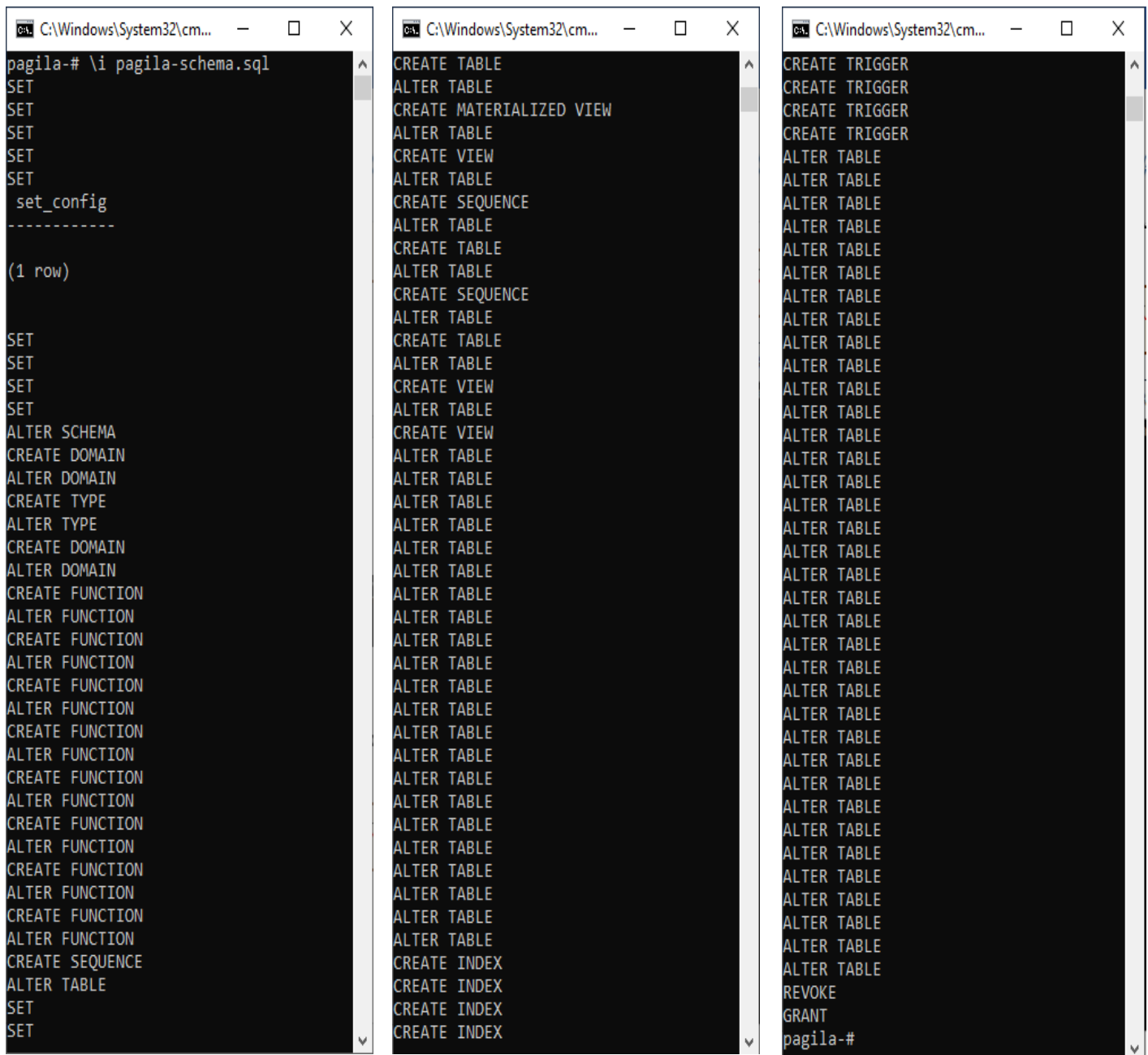
3] Connect to database pagila

```
\c pagila
```

```
postgres=# \c pagila
You are now connected to database "pagila" as user "postgres".
pagila=#
```

#### 4] Load the database schema

```
\i pagila-schema.sql
```



The image displays three terminal windows side-by-side, each showing the execution of the `\i pagila-schema.sql` script in a PostgreSQL database. The windows are titled `C:\Windows\System32\cm...`.

The first window shows the initial execution of the script, starting with `pagila-# \i pagila-schema.sql`. It displays a series of `SET` commands, followed by a `set_config` command, and then a `(1 row)` message. The script continues with a long list of `CREATE` and `ALTER` statements for various database objects.

The second window shows the continuation of the script, displaying a long list of `CREATE` and `ALTER` statements for various database objects, including tables, sequences, views, and functions.

The third window shows the final part of the script, displaying a long list of `ALTER` statements for various database objects, including tables, sequences, views, and functions, followed by `REVOKE` and `GRANT` statements. The prompt `pagila-#` is visible at the bottom of the window.

## 5] Import the data

```
\i pagila-data.sql
```

```
C:\Windows\System32\cm... - □ X
pagila-# \i pagila-data.sql
SET
SET
SET
SET
SET
  set_config
-----
(1 row)

SET
SET
SET
SET
COPY 200
COPY 109
COPY 600
COPY 603
COPY 16
COPY 2
COPY 599
COPY 6
COPY 1000
COPY 5462
COPY 1000
COPY 4581
COPY 2
COPY 16044
COPY 723
COPY 2401
COPY 2713
COPY 2547
COPY 2677
COPY 2654
COPY 2334
  setval
-----
    200
(1 row)

C:\Windows\System32\cm... - □ X
  setval
-----
    605
(1 row)

  setval
-----
    16
(1 row)

  setval
-----
    600
(1 row)

  setval
-----
    109
(1 row)

  setval
-----
    599
(1 row)

  setval
-----
    1000
(1 row)

  setval
-----
    4581
(1 row)

C:\Windows\System32\cm... - □ X
  setval
-----
    6
(1 row)

  setval
-----
  32098
(1 row)

  setval
-----
  16049
(1 row)

  setval
-----
    2
(1 row)

  setval
-----
    2
(1 row)

pagila-#
```

6] To see the imported tables

```
\dt
```

```
C:\Windows\System32\cmd.exe - psql -U postgres
pagila-# \c
You are now connected to database "pagila" as user "postgres".
pagila-# \dt
```

Schema	Name	Type	Owner
public	actor	table	postgres
public	address	table	postgres
public	category	table	postgres
public	city	table	postgres
public	country	table	postgres
public	customer	table	postgres
public	film	table	postgres
public	film_actor	table	postgres
public	film_category	table	postgres
public	inventory	table	postgres
public	language	table	postgres
public	payment	partitioned table	postgres
public	payment_p2022_01	table	postgres
public	payment_p2022_02	table	postgres
public	payment_p2022_03	table	postgres
public	payment_p2022_04	table	postgres
public	payment_p2022_05	table	postgres
public	payment_p2022_06	table	postgres
public	payment_p2022_07	table	postgres
public	rental	table	postgres
public	staff	table	postgres
public	store	table	postgres

```
(22 rows)
```

## **IMPLEMENTATION:**

- 1) The film table stores information about movies. By appropriately querying the film table, find the top 10 most popular films of all time, based on the number of rentals. Your output should have 2 columns containing the film name and rental duration respectively.

Solution (Using film table only):

```
SELECT title, rental_duration FROM film
ORDER BY rental_rate DESC LIMIT 10;
```

```
C:\Windows\System32\cmd.exe - psql -U postgres
public | film | table | postgres
(1 row)

pagila=# \d film
               Table "public.film"
   Column   |      Type      | Collation | Nullable |      Default
-----+-----+-----+-----+-----
 film_id    | integer         |           | not null | nextval('film_film_id_seq'::regclass)
  title     | text            |           | not null |
description | text            |           |          |
release_year | integer         |           |          |
 language_id | integer         |           | not null |
original_language_id | integer         |           |          |
rental_duration | smallint        |           | not null | 3
rental_rate  | numeric(4,2)    |           | not null | 4.99
length       | smallint        |           |          |
replacement_cost | numeric(5,2)    |           | not null | 19.99
rating       | mpaa_rating     |           |          | 'G'::mpaa_rating
last_update  | timestamp with time zone |           | not null | now()
special_features | text[]          |           |          |
fulltext     | tsvector        |           | not null |

Indexes:
  "film_pkey" PRIMARY KEY, btree (film_id)
  "film_fulltext_idx" gist (fulltext)
  "idx_fk_language_id" btree (language_id)
  "idx_fk_original_language_id" btree (original_language_id)
  "idx_title" btree (title)
Foreign-key constraints:
  "film_language_id_fkey" FOREIGN KEY (language_id) REFERENCES language(language_id) ON UPDATE CASCADE ON DELETE RESTRICT
  "film_original_language_id_fkey" FOREIGN KEY (original_language_id) REFERENCES language(language_id) ON UPDATE CASCADE ON DELETE RESTRICT
Referenced by:
  TABLE "film_actor" CONSTRAINT "film_actor_film_id_fkey" FOREIGN KEY (film_id) REFERENCES film(film_id) ON UPDATE CASCADE ON DELETE RESTRICT
  TABLE "film_category" CONSTRAINT "film_category_film_id_fkey" FOREIGN KEY (film_id) REFERENCES film(film_id) ON UPDATE CASCADE ON DELETE RESTRICT
  TABLE "inventory" CONSTRAINT "inventory_film_id_fkey" FOREIGN KEY (film_id) REFERENCES film(film_id) ON UPDATE CASCADE ON DELETE RESTRICT
Triggers:
  film_fulltext_trigger BEFORE INSERT OR UPDATE ON film FOR EACH ROW EXECUTE FUNCTION tsvector_update_trigger('fulltext', 'pg_catalog.english', 'title', 'description')
  last_updated BEFORE UPDATE ON film FOR EACH ROW EXECUTE FUNCTION last_updated()

pagila=#
```

```
C:\Windows\System32\cmd.exe - psql -U postgres
pagila=# SELECT title, rental_duration FROM film ORDER BY rental_rate DESC LIMIT 10;
   title      | rental_duration
-----+-----
AMERICAN CIRCUS |             3
APACHE DIVINE  |             5
ALI FOREVER    |             4
AMELIE HELLFIGHTERS |             4
ANTHEM LUKE    |             5
AIRPLANE SIERRA |             6
AIRPORT POLLOCK |             6
ACE GOLDFINGER |             3
ALADDIN CALENDAR |             6
APOCALYPSE FLAMINGOS |             6
(10 rows)
```

Alternate solution (Using the tables film, inventory and rental):

```
SELECT f.title, f.rental_duration
FROM film AS f
JOIN inventory AS i ON f.film_id = i.film_id
JOIN rental AS r ON i.inventory_id = r.inventory_id
GROUP BY f.title, f.rental_duration
ORDER BY COUNT(*) DESC
LIMIT 10;
```

```

pagila=# SELECT f.title, f.rental_duration
pagila=# FROM film AS f
pagila=# JOIN inventory AS i ON f.film_id = i.film_id
pagila=# JOIN rental AS r ON i.inventory_id = r.inventory_id
pagila=# GROUP BY f.title, f.rental_duration
pagila=# ORDER BY COUNT(*) DESC
pagila=# LIMIT 10;

```

title	rental_duration
BUCKET BROTHERHOOD	7
ROCKETEER MOTHER	3
SCALAWAG DUCK	6
FORWARD TEMPLE	6
GRIT CLOCKWORK	3
JUGGLER HARDLY	4
RIDGEMONT SUBMARINE	3
ROBBERS JOON	7
RUSH GOODFELLAS	3
NETWORK PEAK	5

(10 rows)

- 2) Which are the top 3 most popular films among customers who have also rented the film TEQUILA PAST? Your output should have 4 columns- film id, film name, rental count (in descending order) and rating.

Solution:

```

WITH FilInvRentalJoin AS (
    SELECT f.film_id, f.title, f.rating,
           r.rental_id, r.customer_id
    FROM film AS f
         JOIN inventory AS i ON f.film_id = i.film_id
         JOIN rental AS r ON i.inventory_id = r.inventory_id
)
SELECT
    j.film_id,
    j.title AS film_name,
    COUNT(j.rental_id) AS rental_count,
    j.rating
FROM FilInvRentalJoin j
WHERE j.customer_id IN (SELECT DISTINCT customer_id
                        FROM FilInvRentalJoin
                        WHERE title = 'TEQUILA PAST')
    AND j.title <> 'TEQUILA PAST'
GROUP BY j.film_id, j.title, j.rating
ORDER BY rental_count DESC
LIMIT 3;

```

```
C:\Windows\System32\cmd.exe - psql -U postgres
pagila=# SELECT * FROM rental LIMIT 5;
 rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update
-----+-----+-----+-----+-----+-----+-----
        2 | 2022-05-25 03:24:33+05:30 |        1525 |         459 | 2022-05-29 00:10:33+05:30 |         1 | 2022-02-16 08:00:53+05:30
        3 | 2022-05-25 03:33:39+05:30 |        1711 |         408 | 2022-06-02 02:42:39+05:30 |         1 | 2022-02-16 08:00:53+05:30
        4 | 2022-05-25 03:34:41+05:30 |       2452 |         333 | 2022-06-03 06:13:41+05:30 |         2 | 2022-02-16 08:00:53+05:30
        5 | 2022-05-25 03:35:21+05:30 |       2079 |         222 | 2022-06-02 09:03:21+05:30 |         1 | 2022-02-16 08:00:53+05:30
        6 | 2022-05-25 03:38:07+05:30 |       2792 |         549 | 2022-05-27 06:02:07+05:30 |         1 | 2022-02-16 08:00:53+05:30
(5 rows)

pagila=# SELECT * FROM inventory LIMIT 5;
 inventory_id | film_id | store_id | last_update
-----+-----+-----+-----
          1 |      1 |         1 | 2022-02-15 15:39:17+05:30
          2 |      1 |         1 | 2022-02-15 15:39:17+05:30
          3 |      1 |         1 | 2022-02-15 15:39:17+05:30
          4 |      1 |         1 | 2022-02-15 15:39:17+05:30
          5 |      1 |         2 | 2022-02-15 15:39:17+05:30
(5 rows)

pagila=# SELECT * FROM customer LIMIT 5;
 customer_id | store_id | first_name | last_name | email | address_id | activebool | create_date | last_update | active
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
          1 |         1 | MARY | SMITH | MARY.SMITH@sakilacustomer.org |         5 | t | 2022-02-14 | 2022-02-15 15:27:20+05:30 | 1
          2 |         1 | PATRICIA | JOHNSON | PATRICIA.JOHNSON@sakilacustomer.org |         6 | t | 2022-02-14 | 2022-02-15 15:27:20+05:30 | 1
          3 |         1 | LINDA | WILLIAMS | LINDA.WILLIAMS@sakilacustomer.org |         7 | t | 2022-02-14 | 2022-02-15 15:27:20+05:30 | 1
          4 |         2 | BARBARA | JONES | BARBARA.JONES@sakilacustomer.org |         8 | t | 2022-02-14 | 2022-02-15 15:27:20+05:30 | 1
          5 |         1 | ELIZABETH | BROWN | ELIZABETH.BROWN@sakilacustomer.org |         9 | t | 2022-02-14 | 2022-02-15 15:27:20+05:30 | 1
(5 rows)

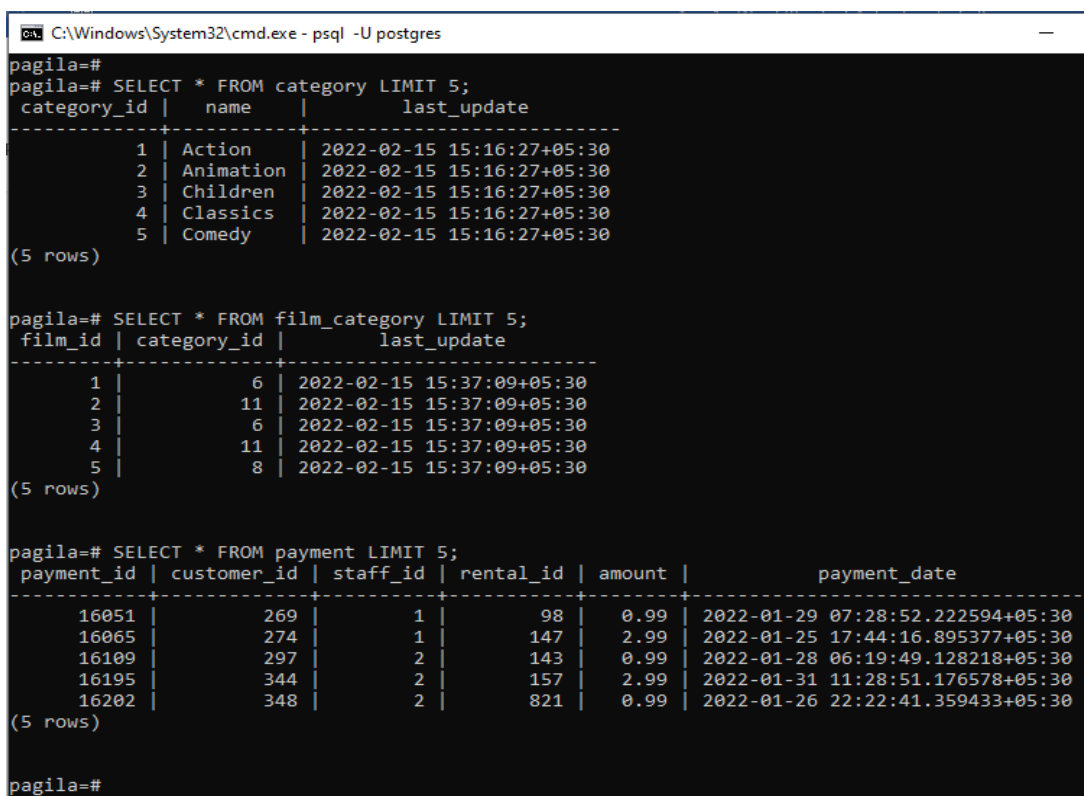
pagila=#
```

```
pagila=# WITH FilInvRentalJoin AS (
pagila(#   SELECT f.film_id, f.title, f.rating,
pagila(#   r.rental_id, r.customer_id
pagila(#   FROM film AS f
pagila(# JOIN inventory AS i ON f.film_id = i.film_id
pagila(# JOIN rental AS r ON i.inventory_id = r.inventory_id
pagila(# )
pagila=# SELECT
pagila(#   j.film_id,
pagila(#   j.title AS film_name,
pagila(#   COUNT(j.rental_id) AS rental_count,
pagila(#   j.rating
pagila(# FROM FilInvRentalJoin j
pagila(# WHERE j.customer_id IN (SELECT DISTINCT customer_id
pagila(#   FROM FilInvRentalJoin
pagila(#   WHERE title = 'TEQUILA PAST')
pagila(# AND j.title <> 'TEQUILA PAST'
pagila(# GROUP BY j.film_id, j.title, j.rating
pagila(# ORDER BY rental_count DESC
pagila=# LIMIT 3;
 film_id | film_name | rental_count | rating
-----+-----+-----+-----
      381 | GRINCH MESSAGE |          3 | R
       55 | BARBARELLA STREETCAR |          3 | G
      588 | MODEL FISH |          2 | NC-17
(3 rows)
```

- 3) Calculate the total revenue generated by each film category (ex: Action, Drama, Sports, etc) and list the category name along with their total revenue, as well as the average revenue per film in each category. Your output shall thus have 3 columns.

### Solution:

```
WITH FilInvRentalJoin AS (  
    SELECT f.film_id, r.rental_id  
    FROM rental AS r  
        JOIN inventory AS i ON r.inventory_id = i.inventory_id  
        JOIN film AS f ON i.film_id = f.film_id  
    )  
SELECT  
    c.name AS category_name,  
    SUM(p.amount) AS total_revenue,  
    AVG(p.amount) AS average_revenue_per_film  
FROM  
    payment AS p  
JOIN  
    FilInvRentalJoin AS j ON p.rental_id = j.rental_id  
JOIN  
    film_category AS fc ON j.film_id = fc.film_id  
JOIN  
    category AS c ON fc.category_id = c.category_id  
GROUP BY  
    c.name  
ORDER BY  
    total_revenue DESC;
```



The screenshot shows a PostgreSQL command prompt window with the following content:

```
C:\Windows\System32\cmd.exe - psql -U postgres  
pagila=#  
pagila=# SELECT * FROM category LIMIT 5;  
category_id | name      | last_update  
-----  
1 | Action    | 2022-02-15 15:16:27+05:30  
2 | Animation | 2022-02-15 15:16:27+05:30  
3 | Children  | 2022-02-15 15:16:27+05:30  
4 | Classics  | 2022-02-15 15:16:27+05:30  
5 | Comedy    | 2022-02-15 15:16:27+05:30  
(5 rows)  
  
pagila=# SELECT * FROM film_category LIMIT 5;  
film_id | category_id | last_update  
-----  
1 | 6 | 2022-02-15 15:37:09+05:30  
2 | 11 | 2022-02-15 15:37:09+05:30  
3 | 6 | 2022-02-15 15:37:09+05:30  
4 | 11 | 2022-02-15 15:37:09+05:30  
5 | 8 | 2022-02-15 15:37:09+05:30  
(5 rows)  
  
pagila=# SELECT * FROM payment LIMIT 5;  
payment_id | customer_id | staff_id | rental_id | amount | payment_date  
-----  
16051 | 269 | 1 | 98 | 0.99 | 2022-01-29 07:28:52.222594+05:30  
16065 | 274 | 1 | 147 | 2.99 | 2022-01-25 17:44:16.895377+05:30  
16109 | 297 | 2 | 143 | 0.99 | 2022-01-28 06:19:49.128218+05:30  
16195 | 344 | 2 | 157 | 2.99 | 2022-01-31 11:28:51.176578+05:30  
16202 | 348 | 2 | 821 | 0.99 | 2022-01-26 22:22:41.359433+05:30  
(5 rows)  
  
pagila=#
```



```
C:\Windows\System32\cmd.exe - psql -U postgres

pagila=#
pagila=#
pagila=# WITH FilInvRentalJoin AS (
pagila(# SELECT f.film_id, r.rental_id
pagila(# FROM rental AS r
pagila(# JOIN inventory AS i ON r.inventory_id = i.inventory_id
pagila(# JOIN film AS f ON i.film_id = f.film_id
pagila(# )
pagila=# SELECT
pagila(# c.name AS category_name,
pagila(# SUM(p.amount) AS total_revenue,
pagila(# AVG(p.amount) AS average_revenue_per_film
pagila=# FROM
pagila(# payment AS p
pagila=# JOIN
pagila(# FilInvRentalJoin AS j ON p.rental_id = j.rental_id
pagila=# JOIN
pagila(# film_category AS fc ON j.film_id = fc.film_id
pagila=# JOIN
pagila(# category AS c ON fc.category_id = c.category_id
pagila=# GROUP BY
pagila(# c.name
pagila=# ORDER BY
pagila(# total_revenue DESC;
category_name | total_revenue | average_revenue_per_film
-----+-----+-----
Sports        | 5314.21       | 4.5073876166242578
Sci-Fi        | 4756.98       | 4.3205994550408719
Animation     | 4656.30       | 3.9933962264150943
Drama         | 4587.39       | 4.3277264150943396
Comedy        | 4383.58       | 4.6584272051009564
Action        | 4375.85       | 3.9351169064748201
New           | 4361.57       | 4.6154179894179894
Games         | 4281.33       | 4.4182972136222910
Foreign       | 4270.67       | 4.1342400774443369
Family        | 4226.07       | 3.8559032846715328
Documentary   | 4217.52       | 4.0166857142857143
Horror        | 3722.54       | 4.4001654846335697
Children      | 3655.55       | 3.8683068783068783
Classics      | 3639.59       | 3.8760276890308839
Travel        | 3549.64       | 4.2409080047789725
Music         | 3417.72       | 4.1177349397590361
(16 rows)
```

- 4) Use a nested select query to find the first name and the last name of customers who paid more than 10 dollars in any transaction. You have to look at the payment table and then match the customer id with that in the customer table. How many such customers are there?

Solution:

```
SELECT first_name, last_name
FROM customer
WHERE customer_id IN (
    SELECT DISTINCT customer_id
    FROM payment
    WHERE amount > 10);
```

```
C:\Windows\System32\cmd.exe - psql -U postgres

pagila=# SELECT first_name, last_name
pagila=# FROM customer
pagila=# WHERE customer_id IN (
pagila(#       SELECT DISTINCT customer_id
pagila(#       FROM payment
pagila(#       WHERE amount > 10);
 first_name | last_name
-----+-----
 PATRICIA   | JOHNSON
 LINDA      | WILLIAMS
 NANCY      | THOMAS
 KAREN      | JACKSON
 MICHELLE   | CLARK
 ANGELA     | HERNANDEZ
 ANNA       | HILL
 JANET      | PHILLIPS
 JOYCE      | EDWARDS
 DIANE      | COLLINS
 TERESA     | ROGERS
 IRENE      | PRICE
 LORI       | WOOD
 JACQUELINE | LONG
 WANDA      | PATTERSON
 RITA       | GRAHAM
 GRACE      | ELLIS
 VICTORIA   | GIBSON
 ELAINE     | STEVENS
 CARRIE     | PORTER
 ANITA      | MORALES
 EVA        | RAMOS
 ELEANOR    | HUNT
 VALERIE    | BLACK
 CATHY      | SPENCER
 REGINA     | BERRY
 MARION     | SNYDER
 BRITTANY   | RILEY
 VANESSA    | SIMS
 ALMA       | AUSTIN
 JEANNE     | LAWSON
 TARA       | RYAN
 ROSEMARY   | SCHMIDT
 TERRI      | VASQUEZ
 WILMA      | RICHARDS
```

- To find the count of such customers

```
pagila=# SELECT COUNT(*)
pagila=# FROM customer
pagila=# WHERE customer_id IN (
pagila(#       SELECT DISTINCT customer_id
pagila(#       FROM payment
pagila(#       WHERE amount > 10);
 count
-----
    107
(1 row)
```

- 5) Determine the top 10 customers who have rented the most films. Your output should include their total rental count and the total amount they've spent on rentals.

Solution:

```
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    COUNT(r.rental_id) AS total_rental_count,
    SUM(p.amount) AS total_amount_spent
FROM
    customer AS c
JOIN
    rental AS r ON c.customer_id = r.customer_id
JOIN
    payment AS p ON r.rental_id = p.rental_id
GROUP BY
    c.customer_id
ORDER BY
    total_rental_count DESC
LIMIT 10;
```

```
C:\Windows\System32\cmd.exe - psql -U postgres
pagila=#
pagila=#
pagila=# SELECT
pagila=#     c.customer_id,
pagila=#     c.first_name,
pagila=#     c.last_name,
pagila=#     COUNT(r.rental_id) AS total_rental_count,
pagila=#     SUM(p.amount) AS total_amount_spent
pagila=# FROM
pagila=#     customer AS c
pagila=# JOIN
pagila=#     rental AS r ON c.customer_id = r.customer_id
pagila=# JOIN
pagila=#     payment AS p ON r.rental_id = p.rental_id
pagila=# GROUP BY
pagila=#     c.customer_id
pagila=# ORDER BY
pagila=#     total_rental_count DESC
pagila=# LIMIT 10;
 customer_id | first_name | last_name | total_rental_count | total_amount_spent
-----+-----+-----+-----+-----
          148 | ELEANOR   | HUNT      |          46         |         216.54
          526 | KARL      | SEAL      |          45         |         221.55
          144 | CLARA     | SHAW      |          42         |         195.58
          236 | MARCIA    | DEAN      |          42         |         175.58
           75 | TAMMY     | SANDERS   |          41         |         155.59
          469 | WESLEY    | BULL      |          40         |         177.60
          197 | SUE       | PETERS    |          40         |         154.60
          468 | TIM       | CARY      |          39         |         175.61
          137 | RHONDA    | KENNEDY   |          39         |         194.61
          178 | MARION    | SNYDER    |          39         |         194.61
(10 rows)

pagila=#
```

- 6) Now, request a solution from a large language model (LLM) like ChatGPT or BARD for each of these problems. After obtaining the solutions, compare them with your solutions. (Advice: LLMs do not always generate an appropriate code required. Therefore, it is recommended to solve the questions in the order provided, first by yourself, and then prompt the LLM)

### Solution (1):

**OBSERVATION** - ChatGPT computes the 10 most popular films by using `rental_duration` which is not an appropriate attribute for the task.



The film table stores information about movies. By appropriately querying the film table, find the top 10 most popular films of all time, based on the number of rentals. Your output should have 2 columns containing the film name and rental duration respectively.



To find the top 10 most popular films of all time based on the number of rentals, you can query the "film" table and order the films by the number of rentals in descending order. Here's the SQL query to achieve that:

```
sql
SELECT title, rental_duration
FROM film
ORDER BY rental_duration DESC
LIMIT 10;
```

Copy code

This query retrieves the film titles and rental durations from the "film" table, orders the results by rental duration in descending order (from the longest to the shortest), and limits the output to the top 10 films.

Regenerate

```
pagila=# SELECT title, rental_duration
pagila=# FROM film
pagila=# ORDER BY rental_duration DESC
pagila=# LIMIT 10;
   title          | rental_duration
-----+-----
BOILED DARES     |              7
BORN SPINAL      |              7
BIKINI BORROWERS |              7
BLANKET BEVERLY  |              7
ADAPTATION HOLES |              7
BOONDOCK BALLROOM |             7
ANONYMOUS HUMAN  |              7
ARGONAUTS TOWN   |              7
BLACKOUT PRIVATE |              7
BORROWERS BEDAZZLED |             7
(10 rows)
```

## Solution (2):

**OBSERVATION** - ChatGPT solution also includes TEQUILA PAST among the top three most popular films.



Which are the top 3 most popular films among customers who have also rented the film TEQUILA PAST? Your output should have 4 columns- film id, film name, rental count (in descending order) and rating.



To find the top 3 most popular films among customers who have also rented the film "TEQUILA PAST," you can use a SQL query that first identifies customers who have rented "TEQUILA PAST," and then finds the most popular films among those customers. Here's the SQL query to achieve this:



sql

Copy code

```
WITH TequilaCustomers AS (  
    SELECT DISTINCT r.customer_id  
    FROM rental AS r  
    JOIN inventory AS i ON r.inventory_id = i.inventory_id  
    JOIN film AS f ON i.film_id = f.film_id  
    WHERE f.title = 'TEQUILA PAST'  
)  
SELECT  
    f.film_id,  
    f.title AS film_name,  
    COUNT(r.rental_id) AS rental_count,  
    f.rating  
FROM rental AS r  
JOIN inventory AS i ON r.inventory_id = i.inventory_id  
JOIN film AS f ON i.film_id = f.film_id  
WHERE r.customer_id IN (SELECT customer_id FROM TequilaCustomers)  
GROUP BY f.film_id, f.title, f.rating  
ORDER BY rental_count DESC  
LIMIT 3;
```

Reger

```

pagila=# WITH TequilaCustomers AS (
pagila(#   SELECT DISTINCT r.customer_id
pagila(#   FROM rental AS r
pagila(#   JOIN inventory AS i ON r.inventory_id = i.inventory_id
pagila(#   JOIN film AS f ON i.film_id = f.film_id
pagila(#   WHERE f.title = 'TEQUILA PAST'
pagila(# )
pagila=# SELECT
pagila(#   f.film_id,
pagila(#   f.title AS film_name,
pagila(#   COUNT(r.rental_id) AS rental_count,
pagila(#   f.rating
pagila(# FROM rental AS r
pagila(# JOIN inventory AS i ON r.inventory_id = i.inventory_id
pagila(# JOIN film AS f ON i.film_id = f.film_id
pagila(# WHERE r.customer_id IN (SELECT customer_id FROM TequilaCustomers)
pagila(# GROUP BY f.film_id, f.title, f.rating
pagila(# ORDER BY rental_count DESC
pagila=# LIMIT 3;
 film_id |      film_name      | rental_count | rating
-----+-----+-----+-----
      883 | TEQUILA PAST        |           6 | PG
      381 | GRINCH MASSAGE      |           3 | R
       55 | BARBARELLA STREETCAR |           3 | G
(3 rows)

```

### Solution (3):

**OBSERVATION** - ChatGPT solution is correct.

**DIFFERENCE FROM MY SOLUTION** - ChatGPT performs five consecutive joins sequentially. I had performed the joins on tables rental, inventory and film separately before for better readability.




Calculate the total revenue generated by each film category  
(ex: Action, Drama, Sports, etc) and list the category name along with  
their total revenue, as well as the average revenue per film in each category.  
Your output shall thus have 3 columns.



To calculate the total revenue generated by each film category and list the category name  
along with their total revenue and average revenue per film in each category, you can use the  
following SQL query:



sql

 Copy code

```
SELECT
    c.name AS category_name,
    SUM(p.amount) AS total_revenue,
    AVG(p.amount) AS average_revenue_per_film
FROM
    payment AS p
JOIN
    rental AS r ON p.rental_id = r.rental_id
JOIN
    inventory AS i ON r.inventory_id = i.inventory_id
JOIN
    film AS f ON i.film_id = f.film_id
JOIN
    film_category AS fc ON f.film_id = fc.film_id
JOIN
    category AS c ON fc.category_id = c.category_id
GROUP BY
    c.name
ORDER BY
    total_revenue DESC;
```

 Regen

```

pagila=# SELECT
pagila-#      c.name AS category_name,
pagila-#      SUM(p.amount) AS total_revenue,
pagila-#      AVG(p.amount) AS average_revenue_per_film
pagila-# FROM
pagila-#      payment AS p
pagila-# JOIN
pagila-#      rental AS r ON p.rental_id = r.rental_id
pagila-# JOIN
pagila-#      inventory AS i ON r.inventory_id = i.inventory_id
pagila-# JOIN
pagila-#      film AS f ON i.film_id = f.film_id
pagila-# JOIN
pagila-#      film_category AS fc ON f.film_id = fc.film_id
pagila-# JOIN
pagila-#      category AS c ON fc.category_id = c.category_id
pagila-# GROUP BY
pagila-#      c.name
pagila-# ORDER BY
pagila-#      total_revenue DESC;

```

category_name	total_revenue	average_revenue_per_film
Sports	5314.21	4.5073876166242578
Sci-Fi	4756.98	4.3205994550408719
Animation	4656.30	3.9933962264150943
Drama	4587.39	4.3277264150943396
Comedy	4383.58	4.6584272051009564
Action	4375.85	3.9351169064748201
New	4361.57	4.6154179894179894
Games	4281.33	4.4182972136222910
Foreign	4270.67	4.1342400774443369
Family	4226.07	3.8559032846715328
Documentary	4217.52	4.0166857142857143
Horror	3722.54	4.4001654846335697
Children	3655.55	3.8683068783068783
Classics	3639.59	3.8760276890308839
Travel	3549.64	4.2409080047789725
Music	3417.72	4.1177349397590361

(16 rows)

## Solution (4):

**OBSERVATION** - ChatGPT solution is correct.

**DIFFERENCE FROM MY SOLUTION** - ChatGPT renames the tables using AS keyword even if the solution can work without it.



Use a nested select query to find the first name and the last name of customers who paid more than 10 dollars in any transaction. You have to look at the payment table and then match the customer id with that in the customer table. How many such customers are there?



To find the first names and last names of customers who paid more than \$10 in any transaction and to count the number of such customers, you can use a nested subquery to first identify the customer IDs with payments exceeding \$10, and then join that information with the customer table. Here's the SQL query to achieve this:





```
sql Copy code

SELECT
    c.first_name,
    c.last_name
FROM
    customer AS c
WHERE
    c.customer_id IN (
        SELECT DISTINCT p.customer_id
        FROM payment AS p
        WHERE p.amount > 10
    );
```

```
C:\Windows\System32\cmd.exe - psql -U postgres

pagila=#
pagila=# SELECT
pagila=#     c.first_name,
pagila=#     c.last_name
pagila=# FROM
pagila=#     customer AS c
pagila=# WHERE
pagila=#     c.customer_id IN (
pagila=#         SELECT DISTINCT p.customer_id
pagila=#         FROM payment AS p
pagila=#         WHERE p.amount > 10
pagila=#     );
 first_name | last_name
-----+-----
PATRICIA   | JOHNSON
LINDA      | WILLIAMS
NANCY      | THOMAS
KAREN      | JACKSON
MICHELLE   | CLARK
ANGELA     | HERNANDEZ
ANNA       | HILL
JANET      | PHILLIPS
JOYCE      | EDWARDS
DIANE      | COLLINS
TERESA     | ROGERS
IRENE      | PRICE
LORI       | WOOD
JACQUELINE | LONG
WANDA      | PATTERSON
RITA       | GRAHAM
GRACE      | ELLIS
VICTORIA   | GIBSON
ELAINE     | STEVENS
CARRIE     | PORTER
ANITA      | MORALES
EVA        | RAMOS
ELEANOR    | HUNT
VALERIE    | BLACK
CATHY      | SPENCER
REGINA     | BERRY
MARION     | SNYDER
BRITTANY   | RILEY
VANESSA    | SIMS
```

To count the number of such customers, you can use the following query:

```
sql Copy code

SELECT COUNT(*)
FROM customer AS c
WHERE c.customer_id IN (
    SELECT DISTINCT p.customer_id
    FROM payment AS p
    WHERE p.amount > 10
);
```

This query counts the number of rows in the result set, which corresponds to the number of customers who paid more than \$10 in any transaction.

```
pagila=# SELECT COUNT(*)
pagila=# FROM customer AS c
pagila=# WHERE c.customer_id IN (
pagila(#      SELECT DISTINCT p.customer_id
pagila(#      FROM payment AS p
pagila(#      WHERE p.amount > 10
pagila(# );
count
-----
    107
(1 row)
```

---

### Solution (5):

**OBSERVATION** - ChatGPT solution is correct.




Determine the top 10 customers who have rented the most films. Your output should include their total rental count and the total amount they've spent on rentals.



To determine the top 10 customers who have rented the most films and find their total rental count as well as the total amount they've spent on rentals, you can use the following SQL query: Copy Like Dislike

sql

 Copy code

```
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    COUNT(r.rental_id) AS rental_count,
    SUM(p.amount) AS total_rental_amount
FROM
    customer AS c
JOIN
    rental AS r ON c.customer_id = r.customer_id
JOIN
    payment AS p ON r.rental_id = p.rental_id
GROUP BY
    c.customer_id, c.first_name, c.last_name
ORDER BY
    rental_count DESC
LIMIT 10;
```

C:\Windows\System32\cmd.exe - psql -U postgres

```
pagila=#
pagila=#
pagila=# SELECT
pagila=#     c.customer_id,
pagila=#     c.first_name,
pagila=#     c.last_name,
pagila=#     COUNT(r.rental_id) AS rental_count,
pagila=#     SUM(p.amount) AS total_rental_amount
pagila=# FROM
pagila=#     customer AS c
pagila=# JOIN
pagila=#     rental AS r ON c.customer_id = r.customer_id
pagila=# JOIN
pagila=#     payment AS p ON r.rental_id = p.rental_id
pagila=# GROUP BY
pagila=#     c.customer_id, c.first_name, c.last_name
pagila=# ORDER BY
pagila=#     rental_count DESC
pagila=# LIMIT 10;
 customer_id | first_name | last_name | rental_count | total_rental_amount
-----
          148 | ELEANOR   | HUNT      |          46 |          216.54
          526 | KARL      | SEAL      |          45 |          221.55
          144 | CLARA     | SHAW      |          42 |          195.58
          236 | MARCIA    | DEAN      |          42 |          175.58
           75 | TAMMY     | SANDERS   |          41 |          155.59
          469 | WESLEY    | BULL      |          40 |          177.60
          197 | SUE       | PETERS    |          40 |          154.60
          468 | TIM       | CARY      |          39 |          175.61
          137 | RHONDA    | KENNEDY   |          39 |          194.61
          178 | MARION    | SNYDER    |          39 |          194.61
(10 rows)

pagila=#
```