

Task 4

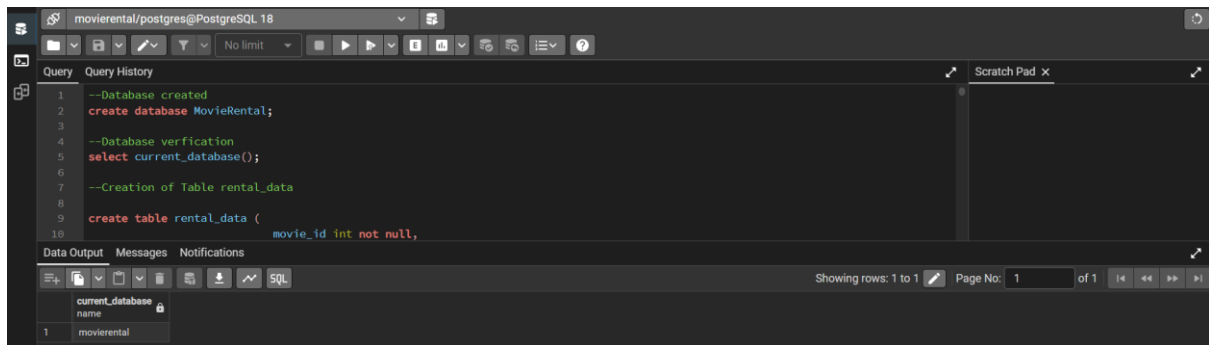
Project: Movie Rental Analysis System (using Redshift or PostgreSQL)

Objective:

Perform advanced analysis on movie rental data using OLAP operations.

GITHUB: https://github.com/xrahulcrx/Movie_Rental_Analysis_System

Database Creation:

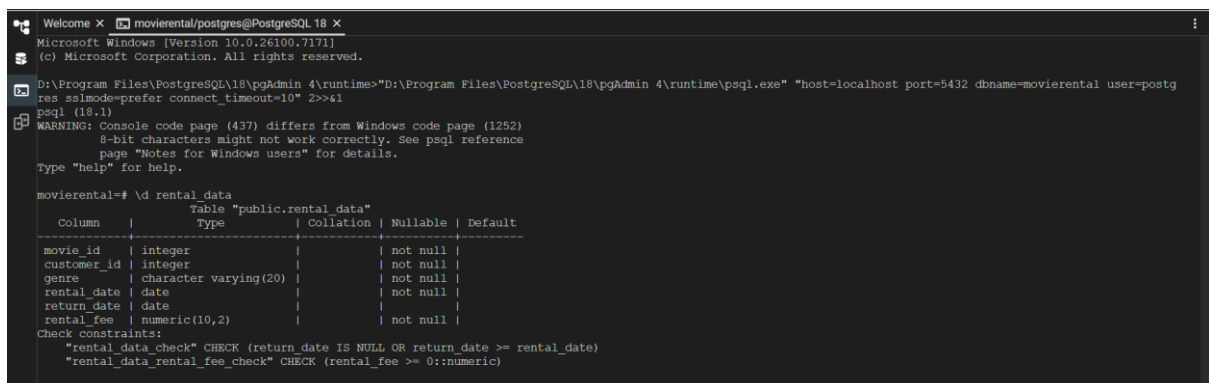


```
1 --Database created
2 create database MovieRental;
3
4 --Database verification
5 select current_database();
6
7 --Creation of Table rental_data
8
9 create table rental_data (
10     movie_id int not null,
```

Data Output

| current_database |
|------------------|
| name |
| 1 movierental |

Create table rental_data with columns:



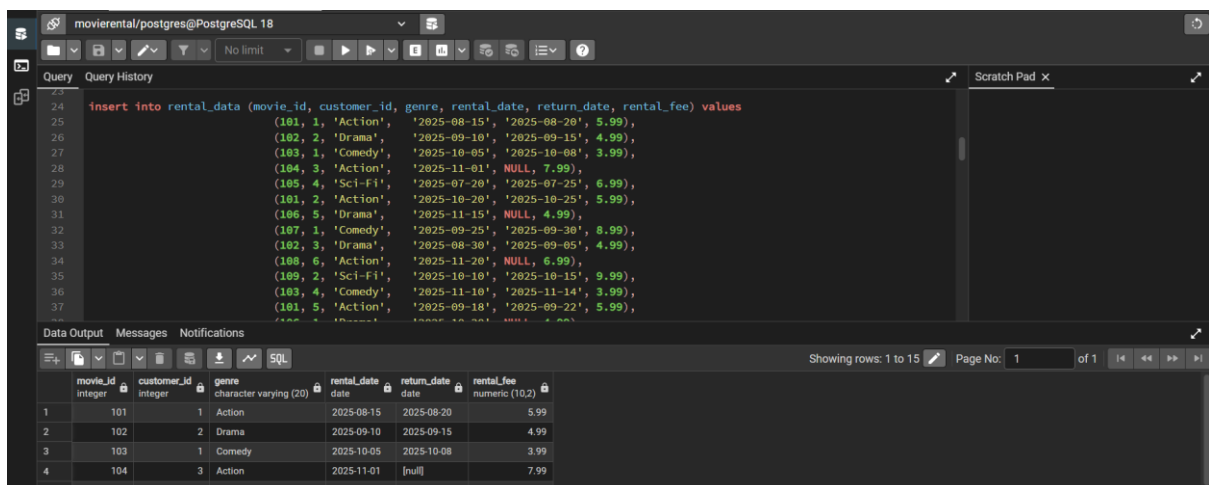
```
Welcome X movierental/postgres@PostgreSQL 18 X
Microsoft Windows [Version 10.0.26100.7171]
(c) Microsoft Corporation. All rights reserved.

D:\Program Files\PostgreSQL\18\pgAdmin 4\runtime>"D:\Program Files\PostgreSQL\18\pgAdmin 4\runtime\psql.exe" "host=localhost port=5432 dbname=movierental user=postgres sslmode=prefer connect_timeout=10" 2>>41
psql (18.1)
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.

movierental=# \d rental_data
              Table "public.rental_data"
   Column   |      Type      | Collation | Nullable | Default
-----|-----|-----|-----|-----
 movie_id   | integer         |           | not null |
customer_id | integer         |           | not null |
 genre      | character varying(20) |           | not null |
 rental_date | date            |           | not null |
 return_date | date            |           | not null |
 rental_fee  | numeric(10,2)    |           | not null |
Check constraints:
 "rental_data_check" CHECK (return_date IS NULL OR return_date >= rental_date)
 "rental_data_rental_fee_check" CHECK (rental_fee >= 0::numeric)
```

Data Creation:

Insert 10–15 sample rental records.



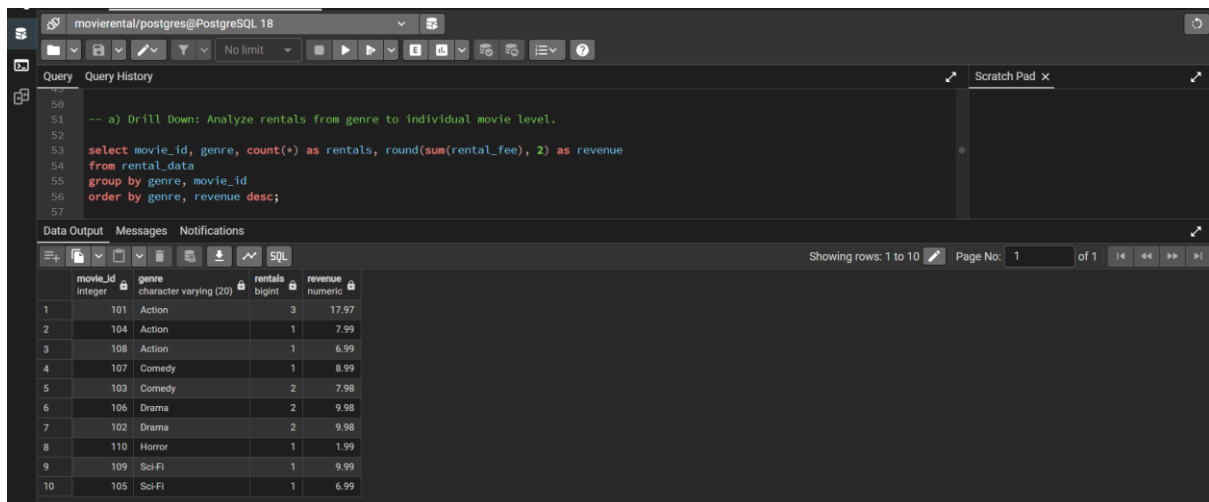
```
24 insert into rental_data (movie_id, customer_id, genre, rental_date, return_date, rental_fee) values
25     (181, 1, 'Action', '2025-08-15', '2025-08-20', 5.99),
26     (182, 2, 'Drama', '2025-09-10', '2025-09-15', 4.99),
27     (183, 1, 'Comedy', '2025-10-05', '2025-10-08', 3.99),
28     (184, 3, 'Action', '2025-11-01', NULL, 7.99),
29     (185, 4, 'Sci-Fi', '2025-07-20', '2025-07-25', 6.99),
30     (181, 2, 'Action', '2025-10-20', '2025-10-25', 5.99),
31     (186, 5, 'Drama', '2025-11-15', NULL, 4.99),
32     (187, 1, 'Comedy', '2025-09-25', '2025-09-30', 8.99),
33     (182, 3, 'Drama', '2025-08-30', '2025-09-05', 4.99),
34     (188, 6, 'Action', '2025-11-20', NULL, 6.99),
35     (189, 2, 'Sci-Fi', '2025-10-10', '2025-10-15', 9.99),
36     (183, 4, 'Comedy', '2025-11-10', '2025-11-14', 3.99),
37     (181, 5, 'Action', '2025-09-18', '2025-09-22', 5.99),
```

Data Output

| movie_id | customer_id | genre | rental_date | return_date | rental_fee |
|----------|-------------|------------------------|-------------|-------------|----------------|
| integer | integer | character varying (20) | date | date | numeric (10,2) |
| 1 | 101 | 1 Action | 2025-08-15 | 2025-08-20 | 5.99 |
| 2 | 102 | 2 Drama | 2025-09-10 | 2025-09-15 | 4.99 |
| 3 | 103 | 1 Comedy | 2025-10-05 | 2025-10-08 | 3.99 |
| 4 | 104 | 3 Action | 2025-11-01 | [null] | 7.99 |
| 5 | 105 | 4 Sci-Fi | 2025-07-20 | 2025-07-25 | 6.99 |

OLAP Operations:

a) Drill Down: Analyze rentals from genre to individual movie level.



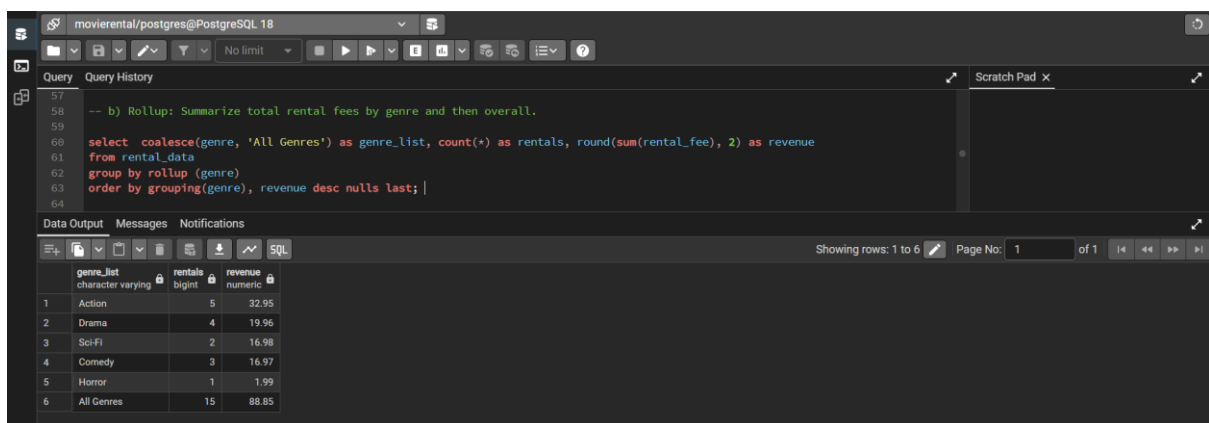
The screenshot shows a PostgreSQL query editor with the following SQL query:

```
-- a) Drill Down: Analyze rentals from genre to individual movie level.  
select movie_id, genre, count(*) as rentals, round(sum(rental_fee), 2) as revenue  
from rental_data  
group by genre, movie_id  
order by genre, revenue desc;
```

The query results are displayed in a table with the following data:

| | movie_id integer | genre character varying (20) | rentals bigint | revenue numeric |
|----|---------------------|---------------------------------|-------------------|--------------------|
| 1 | 101 | Action | 3 | 17.97 |
| 2 | 104 | Action | 1 | 7.99 |
| 3 | 108 | Action | 1 | 6.99 |
| 4 | 107 | Comedy | 1 | 8.99 |
| 5 | 103 | Comedy | 2 | 7.98 |
| 6 | 106 | Drama | 2 | 9.98 |
| 7 | 102 | Drama | 2 | 9.98 |
| 8 | 110 | Horror | 1 | 1.99 |
| 9 | 109 | Sci-Fi | 1 | 9.99 |
| 10 | 105 | Sci-Fi | 1 | 6.99 |

b) Rollup: Summarize total rental fees by genre and then overall.



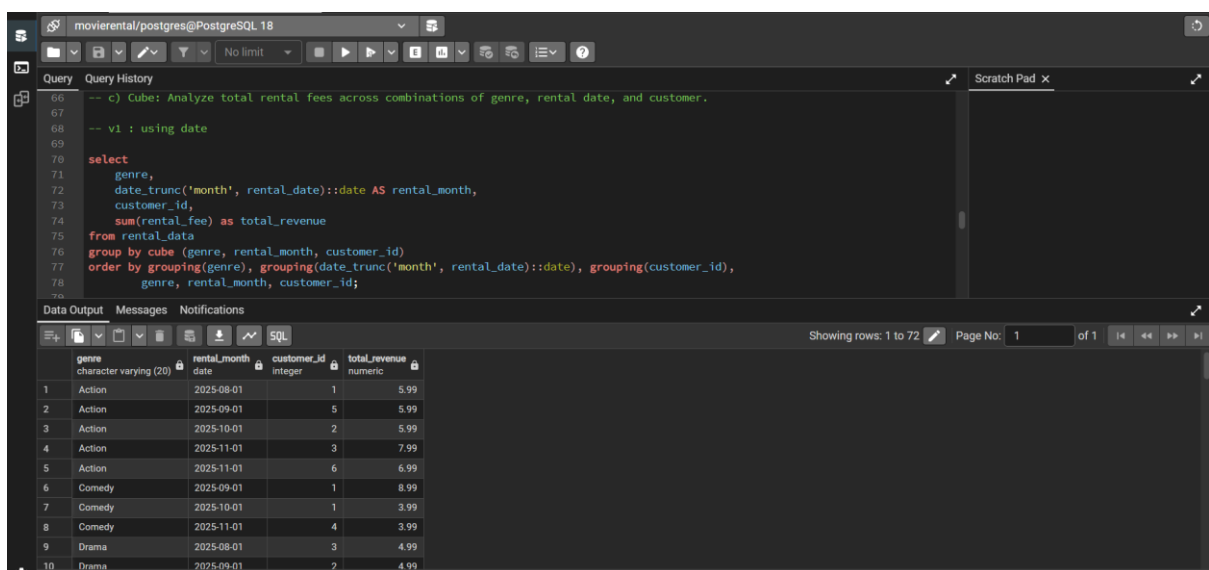
The screenshot shows a PostgreSQL query editor with the following SQL query:

```
-- b) Rollup: Summarize total rental fees by genre and then overall.  
select coalesce(genre, 'All Genres') as genre_list, count(*) as rentals, round(sum(rental_fee), 2) as revenue  
from rental_data  
group by rollup (genre)  
order by grouping(genre), revenue desc nulls last;
```

The query results are displayed in a table with the following data:

| | genre_list character varying | rentals bigint | revenue numeric |
|---|---------------------------------|-------------------|--------------------|
| 1 | Action | 5 | 32.95 |
| 2 | Drama | 4 | 19.96 |
| 3 | Sci-Fi | 2 | 16.98 |
| 4 | Comedy | 3 | 16.97 |
| 5 | Horror | 1 | 1.99 |
| 6 | All Genres | 15 | 88.85 |

c) Cube: Analyze total rental fees across combinations of genre, rental date, and customer.



The screenshot shows a PostgreSQL query editor with the following SQL query:

```
-- c) Cube: Analyze total rental fees across combinations of genre, rental date, and customer.  
-- v1 : using date  
select  
    genre,  
    date_trunc('month', rental_date)::date AS rental_month,  
    customer_id,  
    sum(rental_fee) as total_revenue  
from rental_data  
group by cube (genre, rental_month, customer_id)  
order by grouping(genre), grouping(date_trunc('month', rental_date)::date), grouping(customer_id),  
    genre, rental_month, customer_id;
```

The query results are displayed in a table with the following data:

| | genre character varying (20) | rental_month date | customer_id integer | total_revenue numeric |
|----|---------------------------------|----------------------|------------------------|--------------------------|
| 1 | Action | 2025-08-01 | 1 | 5.99 |
| 2 | Action | 2025-09-01 | 5 | 5.99 |
| 3 | Action | 2025-10-01 | 2 | 5.99 |
| 4 | Action | 2025-11-01 | 3 | 7.99 |
| 5 | Action | 2025-11-01 | 6 | 6.99 |
| 6 | Comedy | 2025-09-01 | 1 | 8.99 |
| 7 | Comedy | 2025-10-01 | 1 | 3.99 |
| 8 | Comedy | 2025-11-01 | 4 | 3.99 |
| 9 | Drama | 2025-08-01 | 3 | 4.99 |
| 10 | Drama | 2025-09-01 | 2 | 4.99 |

movierental/postgres@PostgreSQL 18

Query Query History

```
-- v2 using month year - optimised
select coalesce(genre, 'All Genre') as genre_list,
       coalesce(to_char(date_trunc('Month', rental_date), 'Mon YYYY'), 'All months') as month_year,
       coalesce(customer_id::text, 'All customer') as customer_id,
       round(sum(rental_fee),2) as revenue
from rental_data
group by cube(genre, date_trunc('Month', rental_date), customer_id)
order by grouping(genre), grouping(date_trunc('Month', rental_date)), grouping(customer_id),
       genre, date_trunc('Month', rental_date) nulls first, customer_id;
```

Data Output Messages Notifications

Showing rows: 1 to 72 Page No: 1 of 1

| | genre_list | month_year | customer_id | revenue |
|----|-------------------|------------|--------------|---------|
| | character varying | text | text | numeric |
| 61 | All Genre | Jul 2025 | All customer | 6.99 |
| 62 | All Genre | Aug 2025 | All customer | 10.98 |
| 63 | All Genre | Sep 2025 | All customer | 19.97 |
| 64 | All Genre | Oct 2025 | All customer | 24.96 |
| 65 | All Genre | Nov 2025 | All customer | 25.95 |
| 66 | All Genre | All months | 1 | 23.96 |
| 67 | All Genre | All months | 2 | 20.97 |
| 68 | All Genre | All months | 3 | 14.97 |
| 69 | All Genre | All months | 4 | 10.98 |
| 70 | All Genre | All months | 5 | 10.98 |
| 71 | All Genre | All months | 6 | 6.99 |
| 72 | All Genre | All months | All customer | 88.85 |

d) Slice: Extract rentals only from the 'Action' genre.

movierental/postgres@PostgreSQL 18

Query Query History

```
-- d) Slice: Extract rentals only from the 'Action' genre.
select customer_id, movie_id, rental_date, rental_fee, coalesce(return_date::text, 'Yet to return') as return_date
from rental_data
where genre = 'Action'
order by rental_date desc;
```

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1 of 1

| | customer_id | movie_id | rental_date | rental_fee | return_date |
|---|-------------|----------|----------------|----------------|---------------|
| | integer | integer | date | numeric (10,2) | text |
| 1 | | 6 | 108 2025-11-20 | 6.99 | Yet to return |
| 2 | | 3 | 104 2025-11-01 | 7.99 | Yet to return |
| 3 | | 2 | 101 2025-10-20 | 5.99 | 2025-10-25 |
| 4 | | 5 | 101 2025-09-18 | 5.99 | 2025-09-22 |
| 5 | | 1 | 101 2025-08-15 | 5.99 | 2025-08-20 |

e) Dice: Extract rentals where GENRE = 'Action' or 'Drama' and RENTAL_DATE is in the last 3 months.

movierental/postgres@PostgreSQL 18

Query Query History

```
-- e) Dice: Extract rentals where GENRE = 'Action' or 'Drama' and RENTAL_DATE is in the last 3 months.
select movie_id, customer_id, genre, rental_date, coalesce(return_date::text, 'Yet to return') as return_date, rental_fee
from rental_data
where genre in ('Action', 'Drama') and rental_date >= (current_date - interval '3 months')
order by genre, rental_date desc;
```

Data Output Messages Notifications

Showing rows: 1 to 7 Page No: 1 of 1

| | movie_id | customer_id | genre | rental_date | return_date | rental_fee |
|---|----------|-------------|------------------------|-------------|---------------|----------------|
| | integer | integer | character varying (20) | date | text | numeric (10,2) |
| 1 | 108 | 6 | Action | 2025-11-20 | Yet to return | 6.99 |
| 2 | 104 | 3 | Action | 2025-11-01 | Yet to return | 7.99 |
| 3 | 101 | 2 | Action | 2025-10-20 | 2025-10-25 | 5.99 |
| 4 | 101 | 5 | Action | 2025-09-18 | 2025-09-22 | 5.99 |
| 5 | 106 | 5 | Drama | 2025-11-15 | Yet to return | 4.99 |
| 6 | 106 | 1 | Drama | 2025-10-30 | Yet to return | 4.99 |
| 7 | 102 | 2 | Drama | 2025-09-10 | 2025-09-15 | 4.99 |