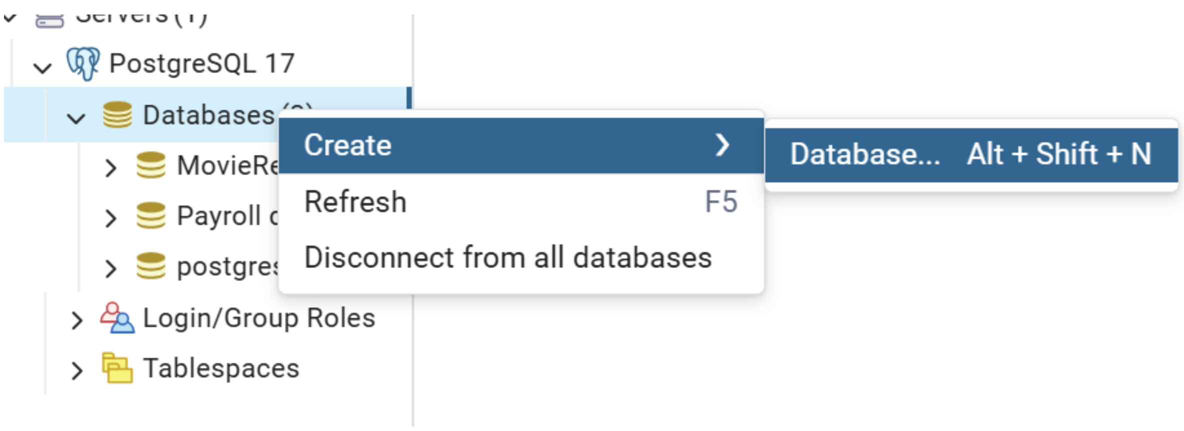
Task 4

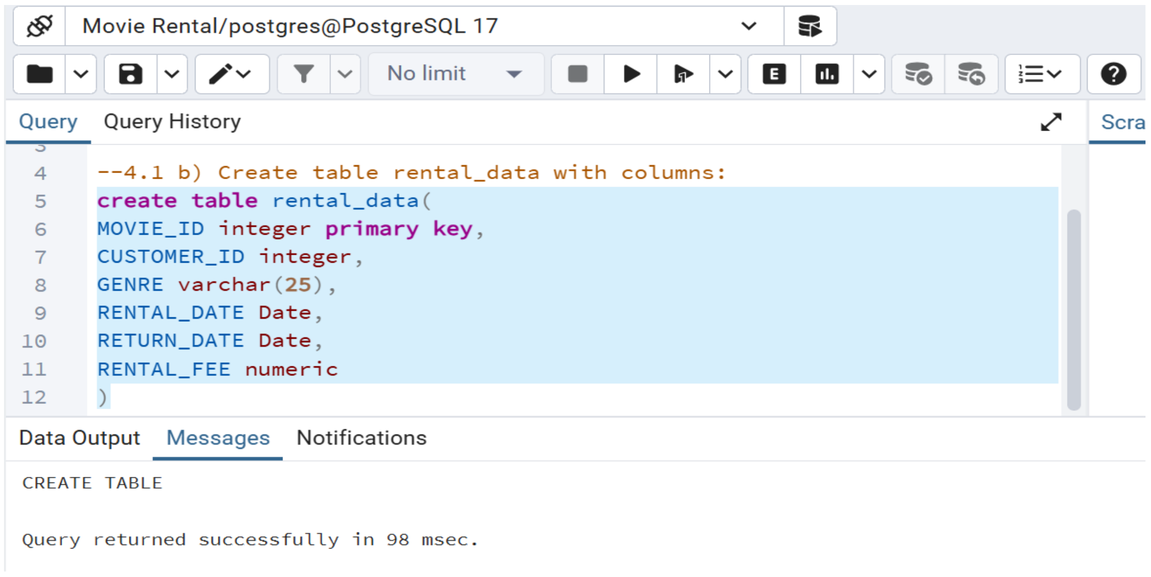
Project: Movie Rental Analysis System (using Redshift or PostgreSQL)

Objective: Perform advanced analysis on movie rental data using OLAP operations.

4.1 a) Database Creation: Create a database named MovieRental.



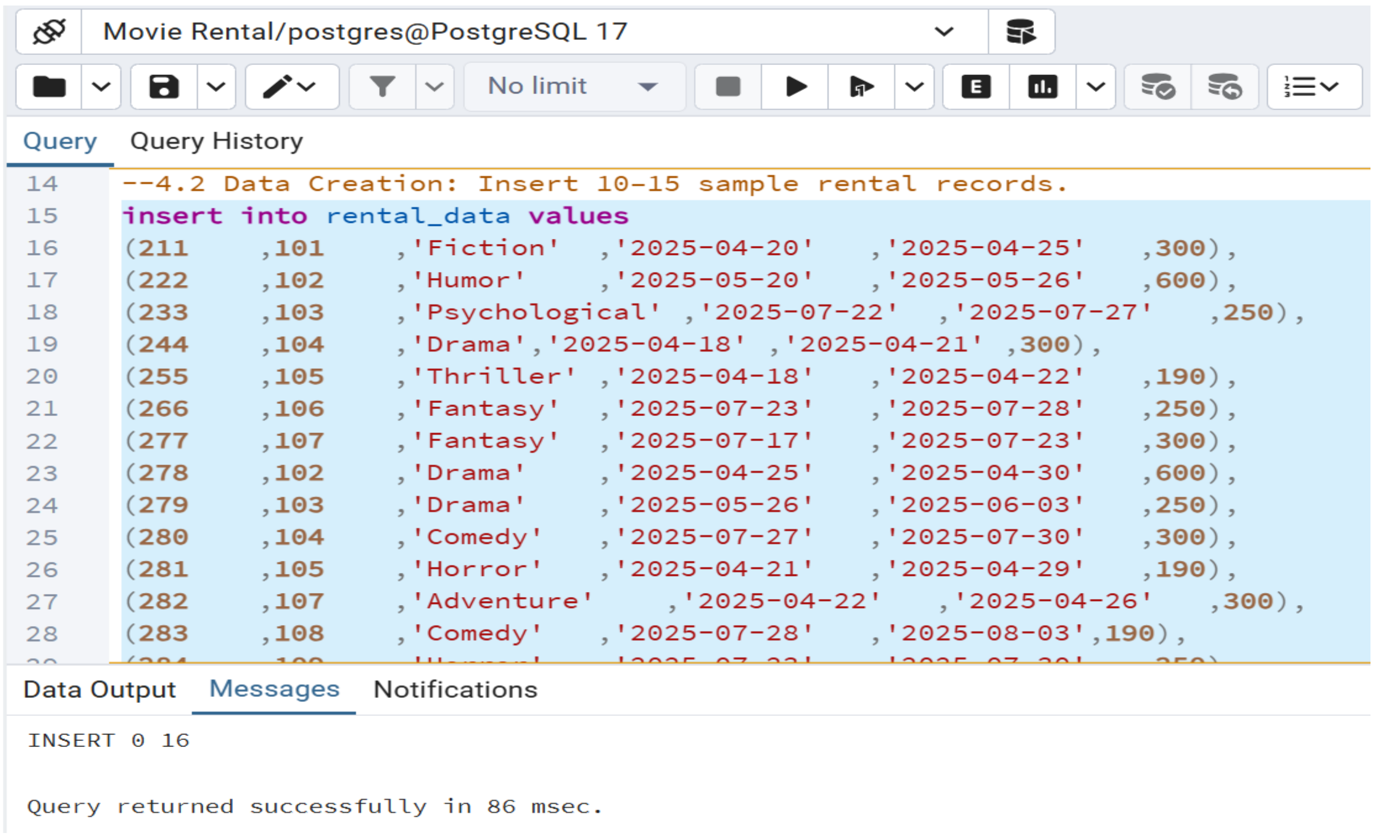
4.1 b) Create table rental_data with columns:
MOVIE_ID (integer), CUSTOMER_ID (integer), GENRE (varchar), RENTAL_DATE (date), RETURN_DATE (date), RENTAL_FEE (numeric).



```
--4.1 b) Create table rental_data with columns:
create table rental_data(
MOVIE_ID integer primary key,
CUSTOMER_ID integer,
GENRE varchar(25),
RENTAL_DATE Date,
RETURN_DATE Date,
RENTAL_FEE numeric
)
```

```
CREATE TABLE

Query returned successfully in 98 msec.
```

4.2 Data Creation: Insert 10–15 sample rental records.



Movie Rental/postgres@PostgreSQL 17

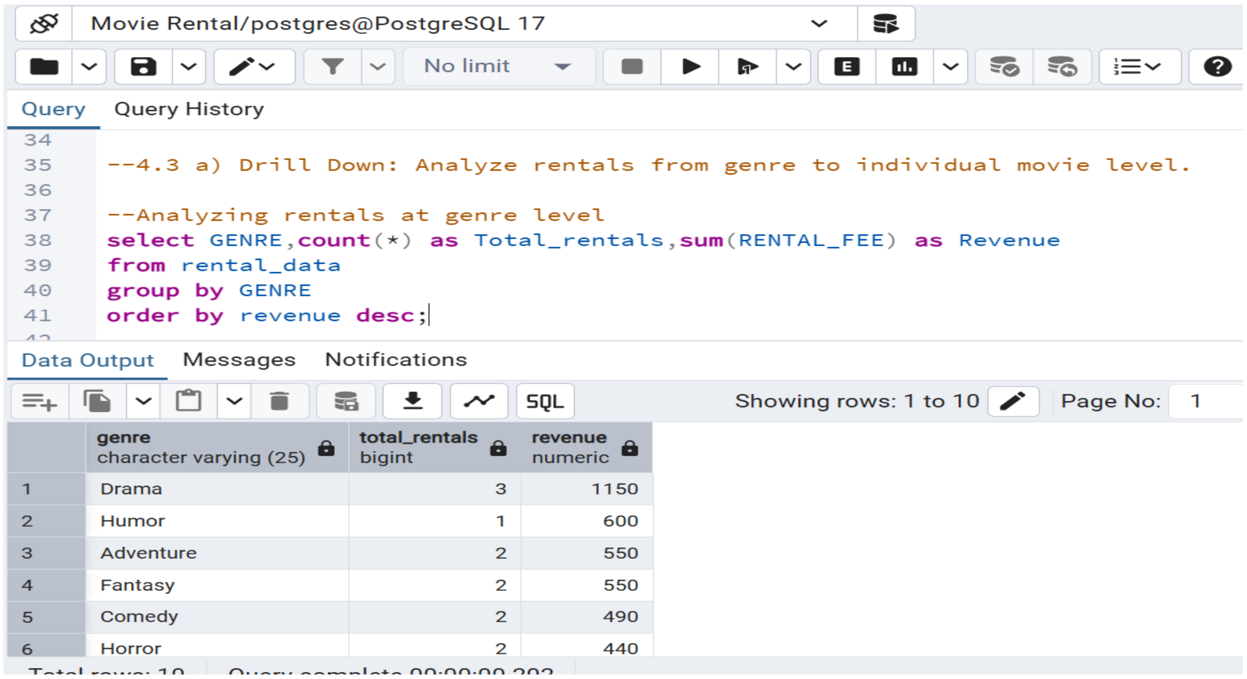
```
--4.2 Data Creation: Insert 10-15 sample rental records.
insert into rental_data values
(211    ,101    ,'Fiction'  ,'2025-04-20'   ,'2025-04-25'   ,300),
(222    ,102    ,'Humor'    ,'2025-05-20'   ,'2025-05-26'   ,600),
(233    ,103    ,'Psychological' ,'2025-07-22'  ,'2025-07-27'   ,250),
(244    ,104    ,'Drama','2025-04-18' ,'2025-04-21' ,300),
(255    ,105    ,'Thriller' ,'2025-04-18'   ,'2025-04-22'   ,190),
(266    ,106    ,'Fantasy'  ,'2025-07-23'   ,'2025-07-28'   ,250),
(277    ,107    ,'Fantasy'  ,'2025-07-17'   ,'2025-07-23'   ,300),
(278    ,102    ,'Drama'    ,'2025-04-25'   ,'2025-04-30'   ,600),
(279    ,103    ,'Drama'    ,'2025-05-26'   ,'2025-06-03'   ,250),
(280    ,104    ,'Comedy'   ,'2025-07-27'   ,'2025-07-30'   ,300),
(281    ,105    ,'Horror'   ,'2025-04-21'   ,'2025-04-29'   ,190),
(282    ,107    ,'Adventure'    ,'2025-04-22'   ,'2025-04-26'   ,300),
(283    ,108    ,'Comedy'   ,'2025-07-28'   ,'2025-08-03',190),
```

Data Output Messages Notifications

```
INSERT 0 16

Query returned successfully in 86 msec.
```

4.3 OLAP Operations:

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



Movie Rental/postgres@PostgreSQL 17

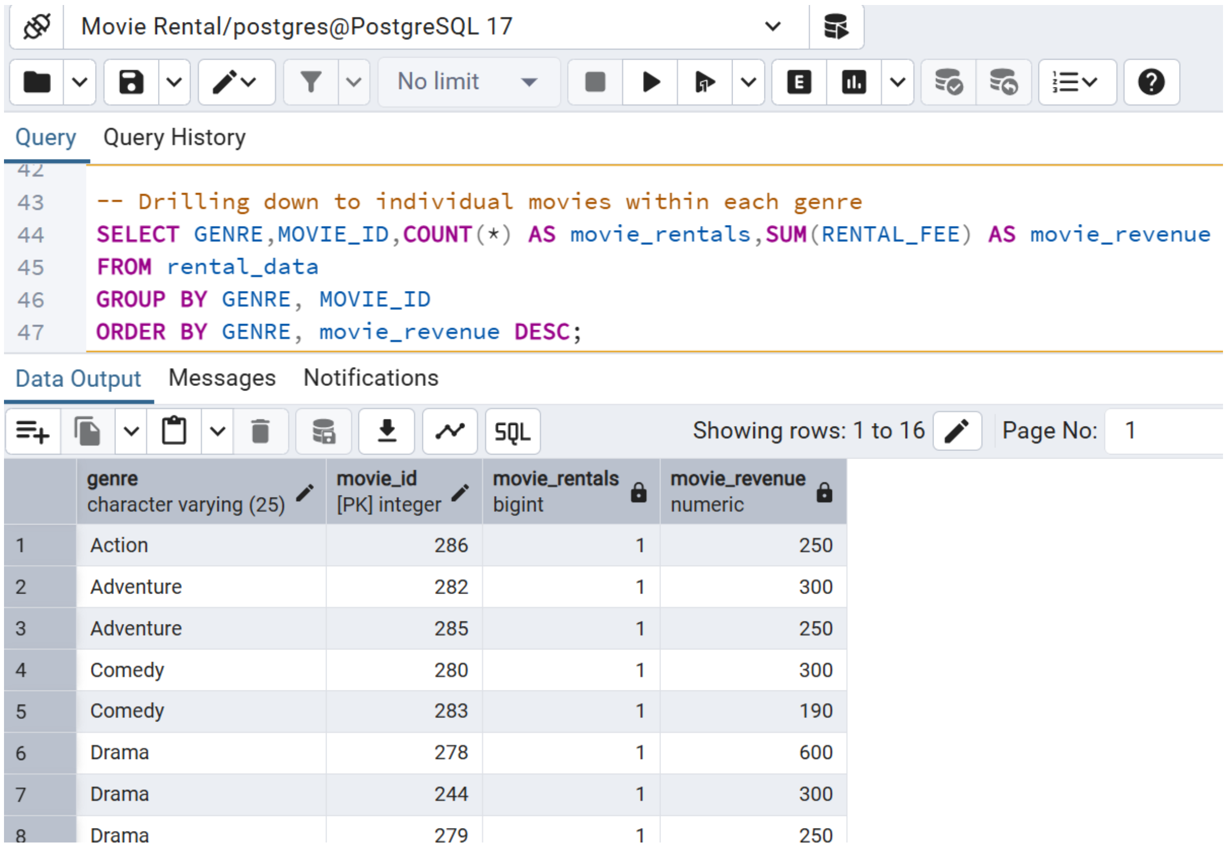
```
--4.3 a) Drill Down: Analyze rentals from genre to individual movie level.

--Analyzing rentals at genre level
select GENRE,count(*) as Total_rentals,sum(RENTAL_FEE) as Revenue
from rental_data
group by GENRE
order by revenue desc;
```

Data Output Messages Notifications

Showing rows: 1 to 10 Page No: 1

	genre character varying (25)	total_rentals bigint	revenue numeric
1	Drama	3	1150
2	Humor	1	600
3	Adventure	2	550
4	Fantasy	2	550
5	Comedy	2	490
6	Horror	2	440

-- Drilling down to individual movies within each genre



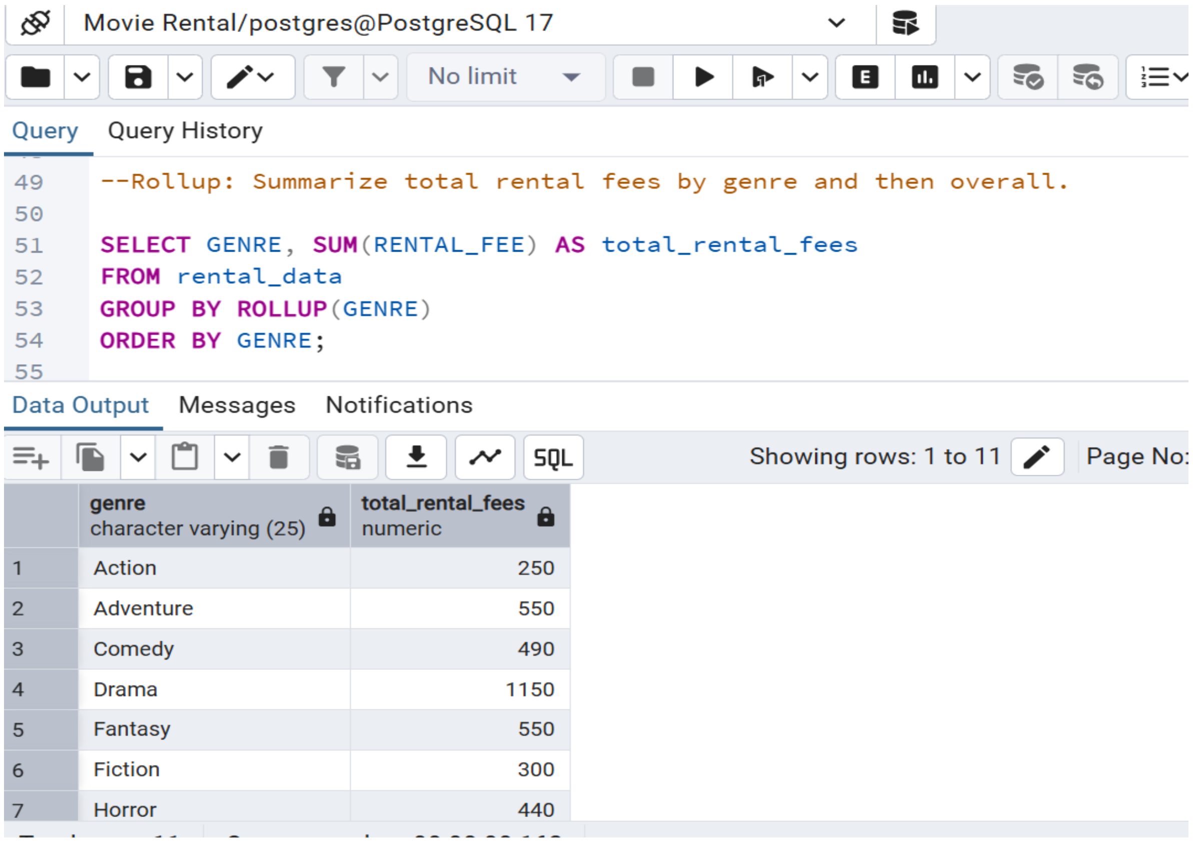
Movie Rental/postgres@PostgreSQL 17

```sql
-- Drilling down to individual movies within each genre
SELECT GENRE,MOVIE_ID,COUNT(*) AS movie_rentals,SUM(RENTAL_FEE) AS movie_revenue
FROM rental_data
GROUP BY GENRE, MOVIE_ID
ORDER BY GENRE, movie_revenue DESC;
```

Showing rows: 1 to 16 Page No: 1

	genre character varying (25)	movie_id [PK] integer	movie_rentals bigint	movie_revenue numeric
1	Action	286	1	250
2	Adventure	282	1	300
3	Adventure	285	1	250
4	Comedy	280	1	300
5	Comedy	283	1	190
6	Drama	278	1	600
7	Drama	244	1	300
8	Drama	279	1	250

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



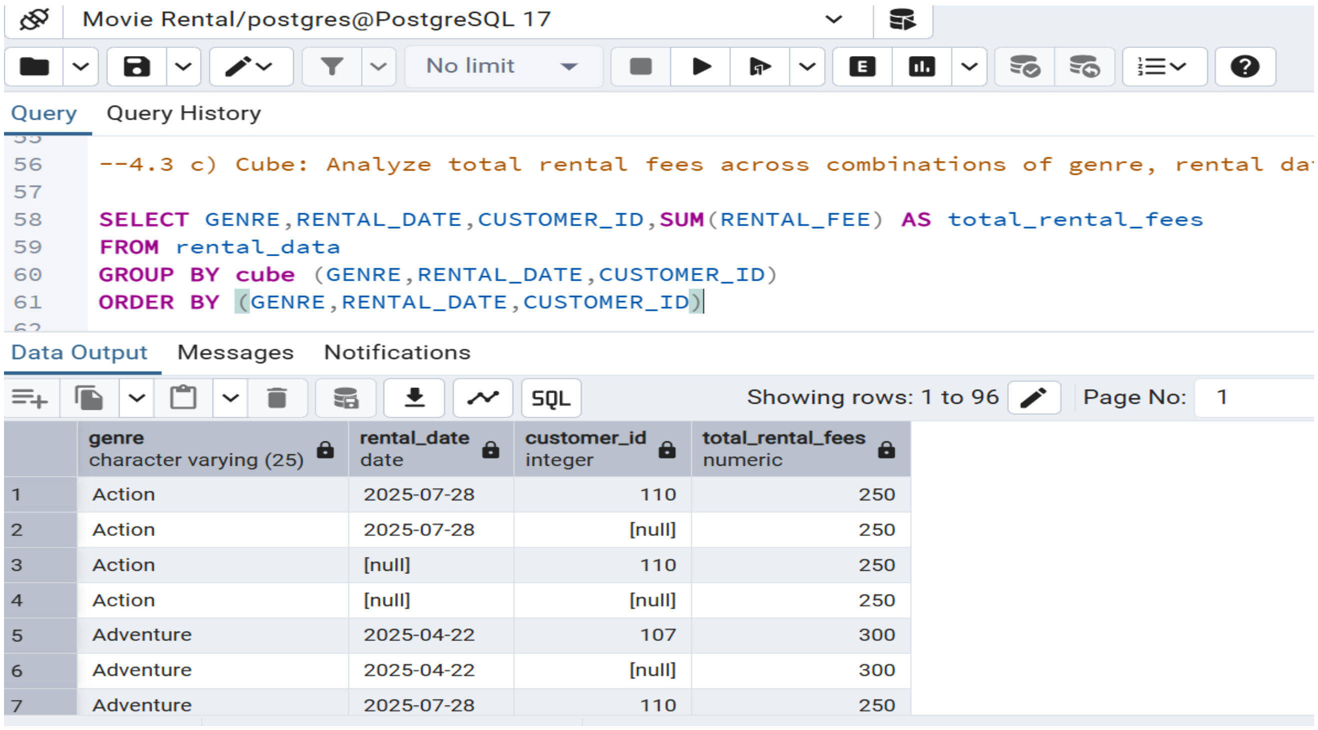
Movie Rental/postgres@PostgreSQL 17

```sql
--Rollup: Summarize total rental fees by genre and then overall.

SELECT GENRE, SUM(RENTAL_FEE) AS total_rental_fees
FROM rental_data
GROUP BY ROLLUP(GENRE)
ORDER BY GENRE;
```

Showing rows: 1 to 11 Page No:

	genre character varying (25)	total_rental_fees numeric
1	Action	250
2	Adventure	550
3	Comedy	490
4	Drama	1150
5	Fantasy	550
6	Fiction	300
7	Horror	440

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



Movie Rental/postgres@PostgreSQL 17

Query | Query History

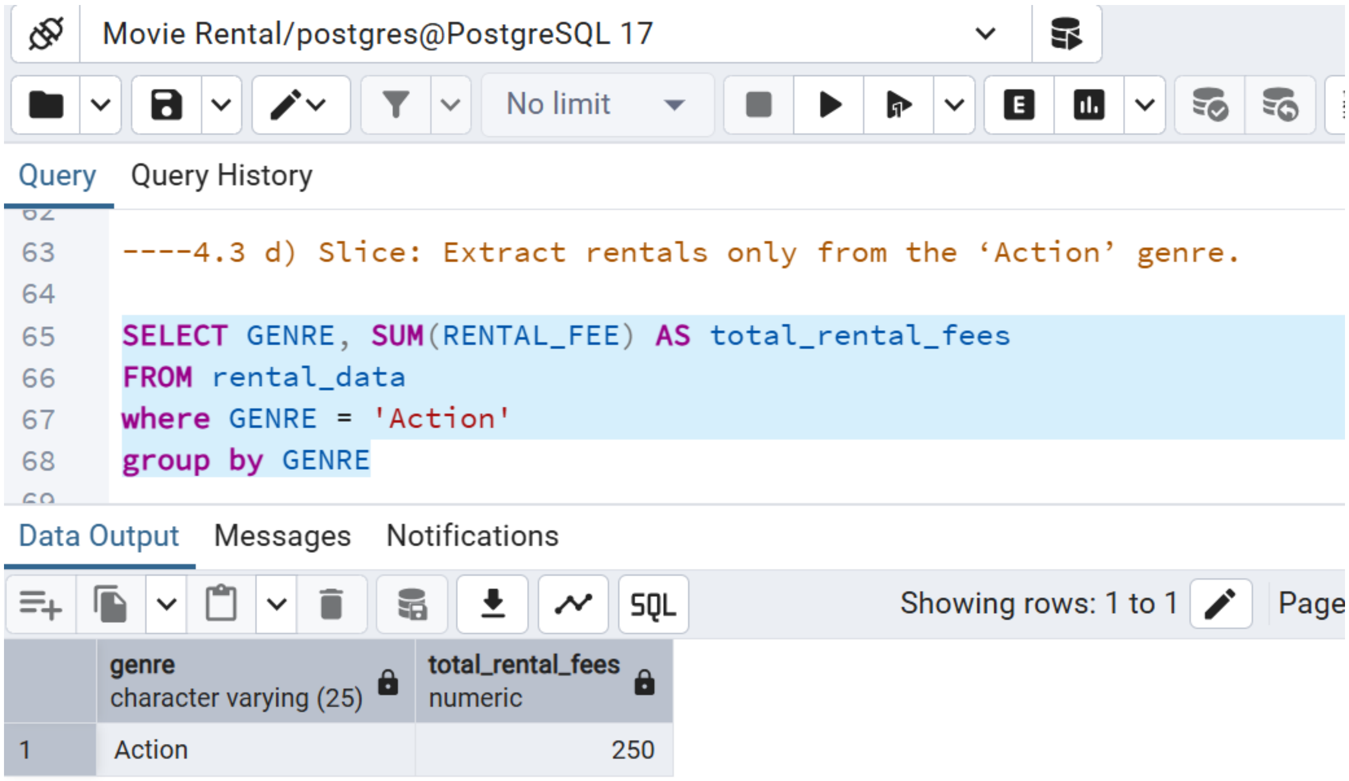
```sql
--4.3 c) Cube: Analyze total rental fees across combinations of genre, rental da

SELECT GENRE,RENTAL_DATE,CUSTOMER_ID,SUM(RENTAL_FEE) AS total_rental_fees
FROM rental_data
GROUP BY cube (GENRE,RENTAL_DATE,CUSTOMER_ID)
ORDER BY (GENRE,RENTAL_DATE,CUSTOMER_ID)
```

Data Output | Messages | Notifications

Showing rows: 1 to 96 Page No: 1

	genre character varying (25)	rental_date date	customer_id integer	total_rental_fees numeric
1	Action	2025-07-28	110	250
2	Action	2025-07-28	[null]	250
3	Action	[null]	110	250
4	Action	[null]	[null]	250
5	Adventure	2025-04-22	107	300
6	Adventure	2025-04-22	[null]	300
7	Adventure	2025-07-28	110	250

4.3 d) Slice: Extract rentals only from the ‘Action’ genre.



Movie Rental/postgres@PostgreSQL 17

Query | Query History

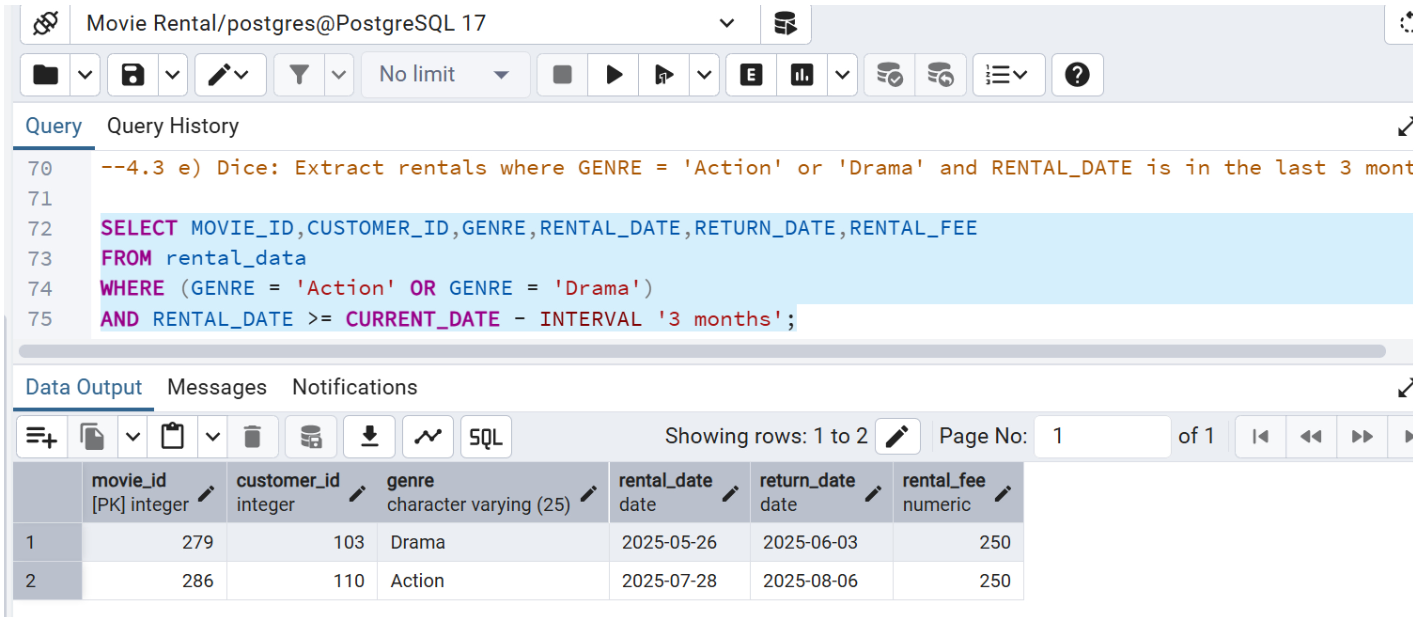
```sql
----4.3 d) Slice: Extract rentals only from the ‘Action’ genre.

SELECT GENRE, SUM(RENTAL_FEE) AS total_rental_fees
FROM rental_data
where GENRE = 'Action'
group by GENRE
```

Data Output | Messages | Notifications

Showing rows: 1 to 1 Page

	genre character varying (25)	total_rental_fees numeric
1	Action	250

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



```
--4.3 e) Dice: Extract rentals where GENRE = 'Action' or 'Drama' and RENTAL_DATE is in the last 3 mont

SELECT MOVIE_ID,CUSTOMER_ID,GENRE,RENTAL_DATE,RETURN_DATE,RENTAL_FEE
FROM rental_data
WHERE (GENRE = 'Action' OR GENRE = 'Drama')
AND RENTAL_DATE >= CURRENT_DATE - INTERVAL '3 months';
```

	movie_id [PK] integer	customer_id integer	genre character varying (25)	rental_date date	return_date date	rental_fee numeric
1	279	103	Drama	2025-05-26	2025-06-03	250
2	286	110	Action	2025-07-28	2025-08-06	250