VIDEO RENTAL STORE DATABASE ANALYSIS

Chisom Ibeneme Juan

INTRODUCTION

The DVD rental industry has witnessed significant transformation over the past few decades, driven by advancements in technology and shifting consumer preferences. This project focuses on analyzing the database of a video rental store to gain insights into the store's operations and customer behaviour. PostgreSQL was utilized for data extraction, transformation and analysis.

OBJECTIVES

- Analyze the database schema and understand the relationships between different tables.
- Retrieve relevant data using SQL queries to answer specific business questions.
- Perform data manipulation and transformation to prepare the data for analysis.
- Generate insights and recommendations based on the rental patterns, customer behavior, and inventory management to drive decision making.

DATA SET DESCRIPTION

- **actor: Table to store information about actors, with actor_id as the primary key.**
- **address: Table to store address information, with address_id as the primary key and a foreign key city_id referencing the city table.**
- **category: Table to store film categories, with category_id as the primary key.**
- **city: Table to store city information, with city_id as the primary key and a foreign key country_id referencing the country table.**
- **country: Table to store country information, with country_id as the primary key.**
- **customer: Table to store customer information, with customer_id as the primary key and a foreign key address_id referencing the address table.**
- **film: Table to store film information, with film_id as the primary key and a foreign key language_id referencing the language table.**

DATA SET DESCRIPTION

- **film_category: Table to store the relationship between films and categories, with a composite primary key (film_id, category_id) and foreign keys to the film and category tables.**
- **film_rating: Likely a table to store film ratings, but the columns are not defined in this script.**
- **inventory: Table to store inventory information, with inventory_id as the primary key and a foreign key film_id referencing the film table.**
- **language: Table to store language information, with language_id as the primary key.**
- **payment: Table to store payment information, with payment_id as the primary key and foreign keys to the customer, rental, and staff tables.**
- **rental: Table to store rental information, with rental_id as the primary key and foreign keys to the customer, inventory, and staff tables.**
- **staff: Table to store staff information, with staff_id as the primary key and a foreign key address_id referencing the address table. The staff table also has a self-referencing foreign key manager_staff_id that references the staff table.**

APPROACH

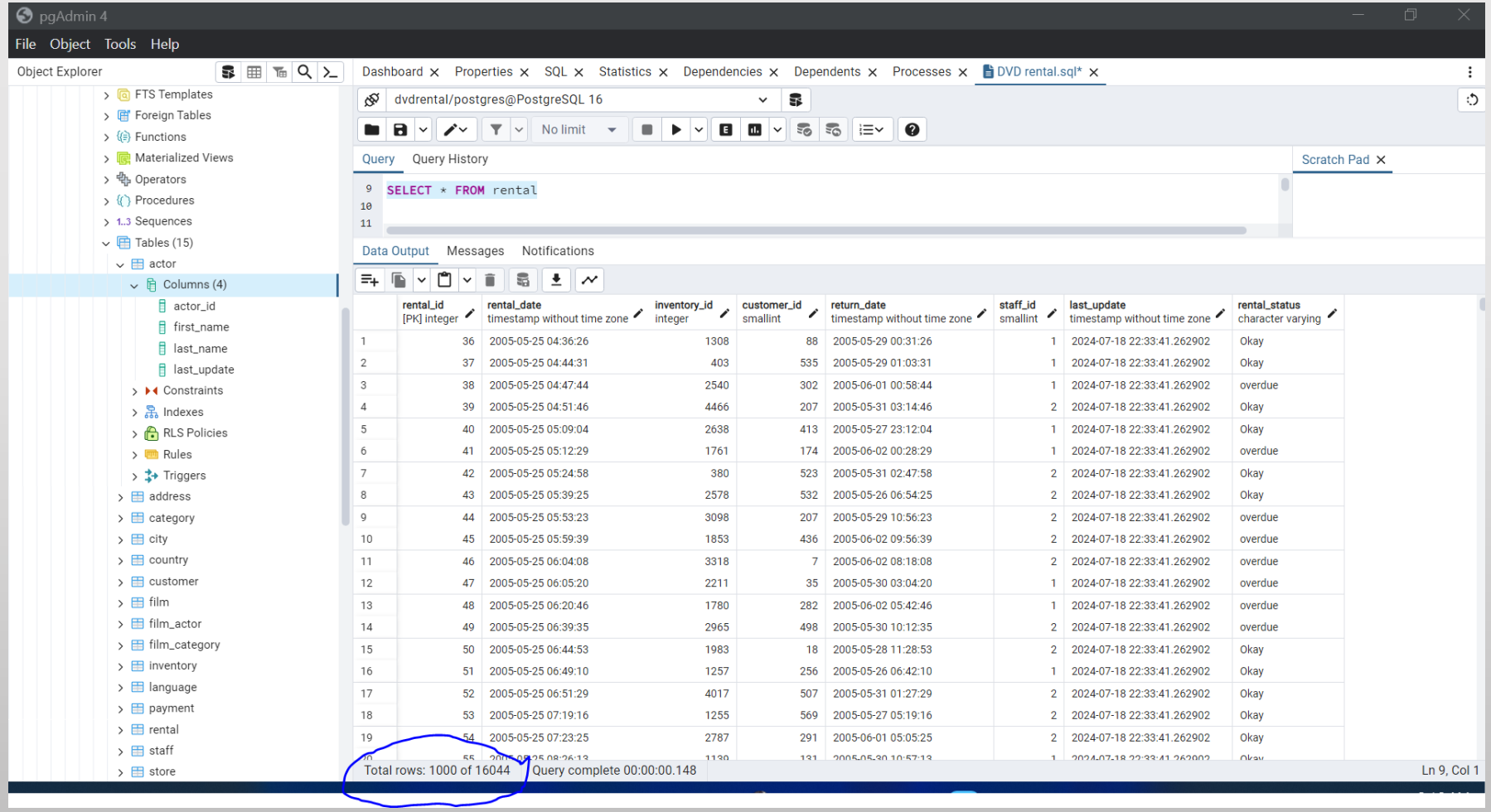
The analysis will involve the following steps:

- ❑ **Data Source Identification**
- ❑ **Creation of Database (dvdrental)**
- ❑ **Import Data into pgAdmin**
- ❑ **Data cleaning (Checked for missing values, duplicates, data type and outliers)**
- ❑ **Data Manipulation and Transformation (Converted data types as necessary (e.g., dates,**

numerical values).

- ❑ **Data Analysis**
- ❑ **Interpretation: Drawing meaningful insights from the data analysis.**

EXAMINING THE DATASET IN PGADMIN

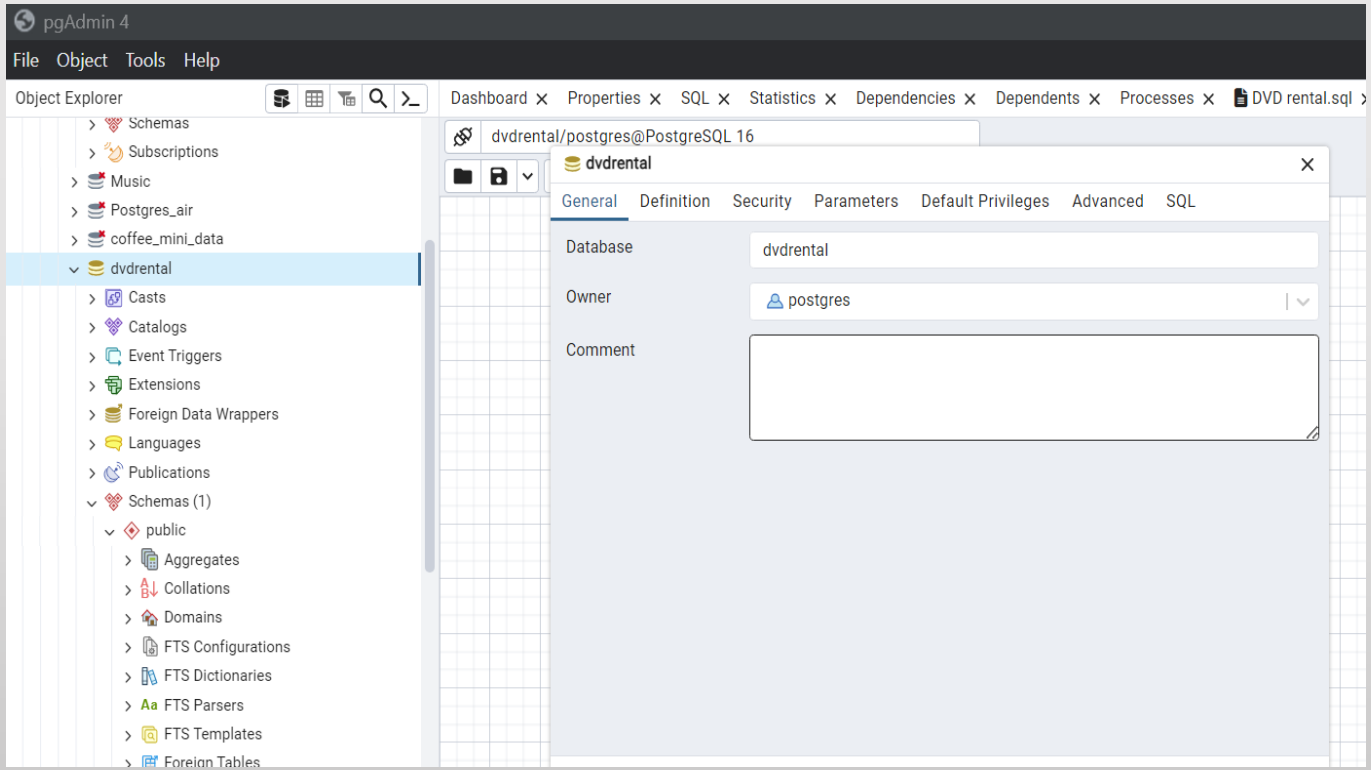


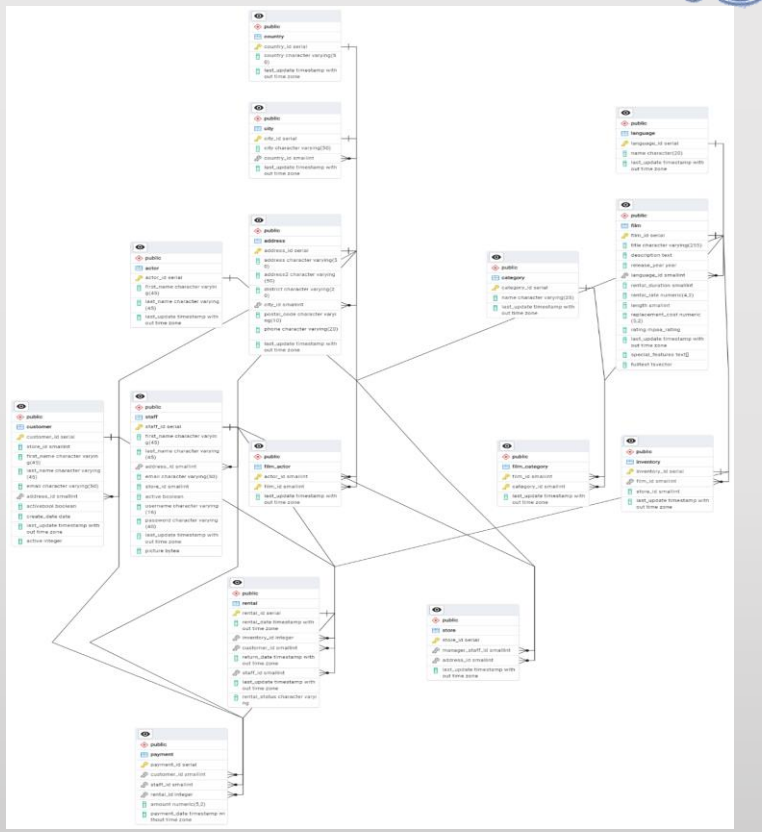
Knowing the total rows in a dataset is important for several reasons:

- ✓ **Data Integrity**
- ✓ **Quality Assurance**
- ✓ **Performance Tuning**
- ✓ **Data Analysis**
- ✓ **Auditing and Compliance**

DATABASE & ERD

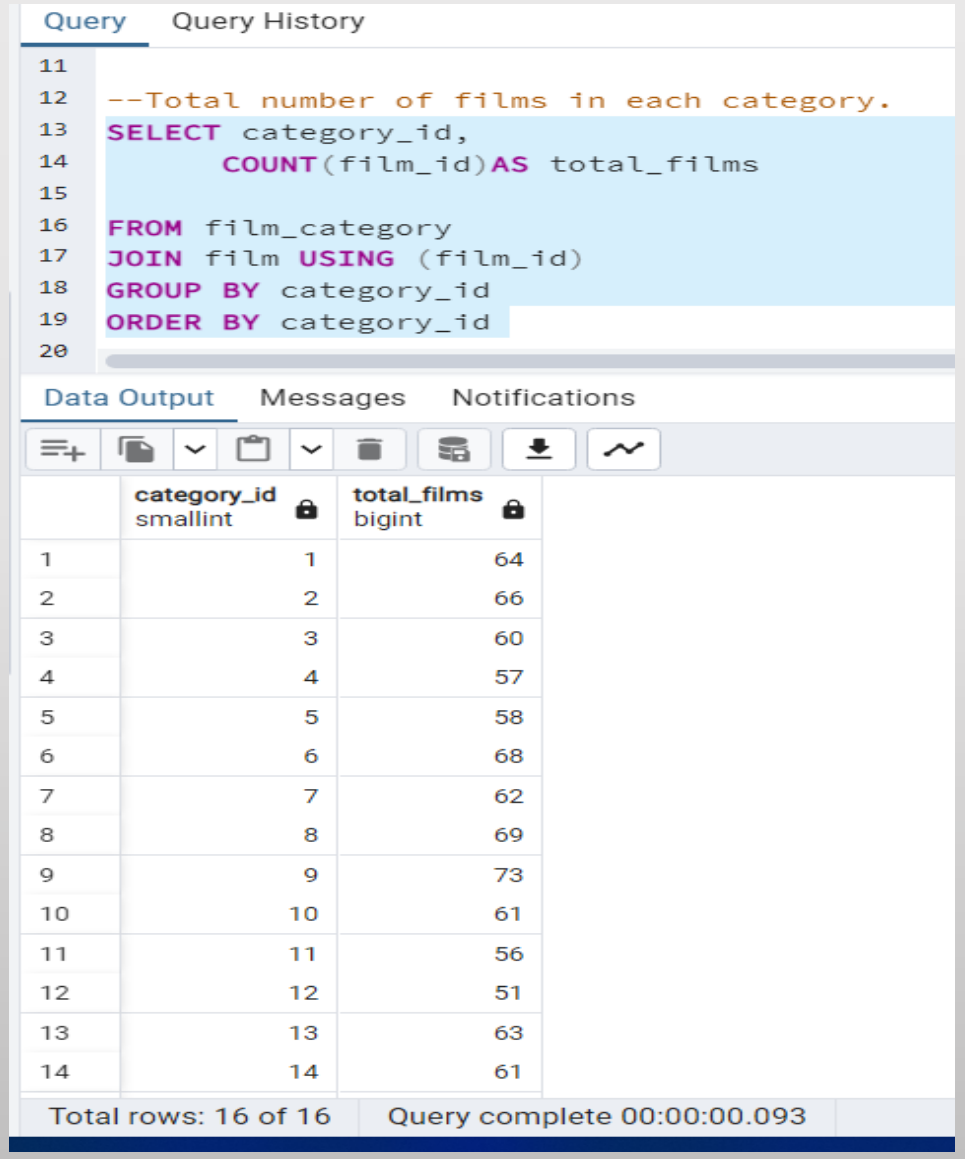
DVDRENTAL DATABASE





Entity relationship diagram that shows how the tables are connected to the primary keys and secondary keys

DATA RETRIVAL AND ANALYSIS



Total number of films in each category.

TOP 5 CUSTOMERS WHO HAVE RENTED THE MOST FILMS.

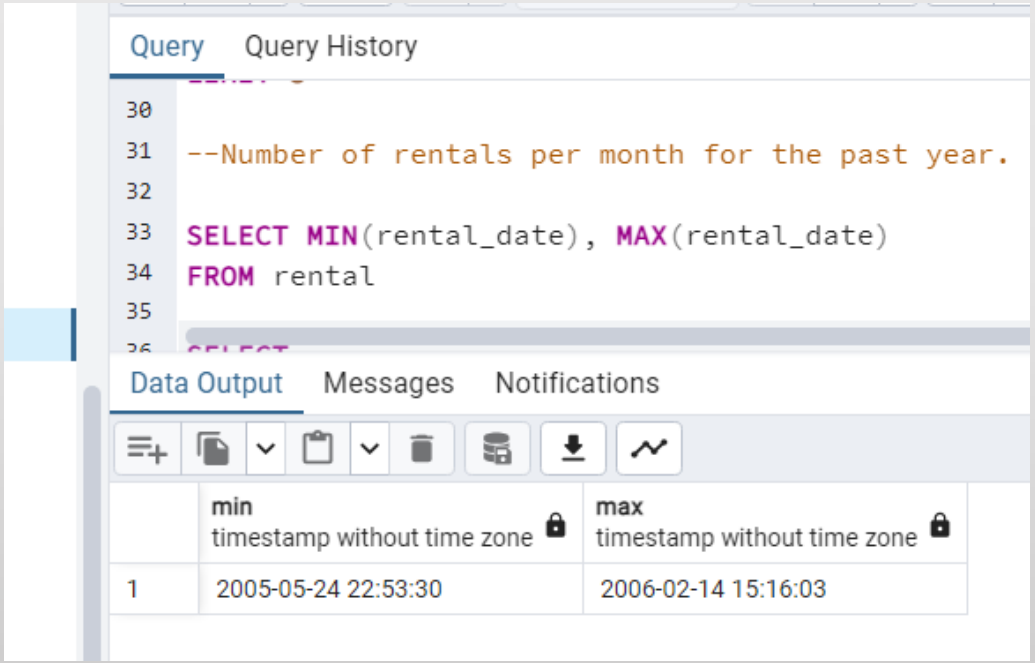
```
20
21  --Top 5 customers who have rented the most films.
22  SELECT customer_id,
23         CONCAT (first_name, ' ', last_name) AS customers,
24         COUNT(rental_id) AS total_rental
25  FROM customer
26  JOIN rental USING (customer_id)
27  GROUP BY customer_id
28  ORDER BY total_rental DESC
29  LIMIT 5
30
```

Data Output | Messages | Notifications

	customer_id [PK] integer	customers text	total_rental bigint
1	148	Eleanor Hunt	46
2	526	Karl Seal	45
3	236	Marcia Dean	42
4	144	Clara Shaw	42
5	75	Tammy Sanders	41

- ✓ **This provides insight on who the top customers are and will drive decision making for the company.**

THE MIN AND MAX DATE.



- ✓ **Ensuring that the min date and max date are within expected ranges helps maintain data integrity.**
- ✓ **It prevents incorrect or invalid dates from being entered into the dataset, which can affect the accuracy of analyses and reports.**
- ✓ **It helps identify inconsistencies such as end dates that precede start dates, which can indicate data entry errors or anomalies.**

NUMBER OF RENTALS PER MONTH FOT THE PAST YEAR.

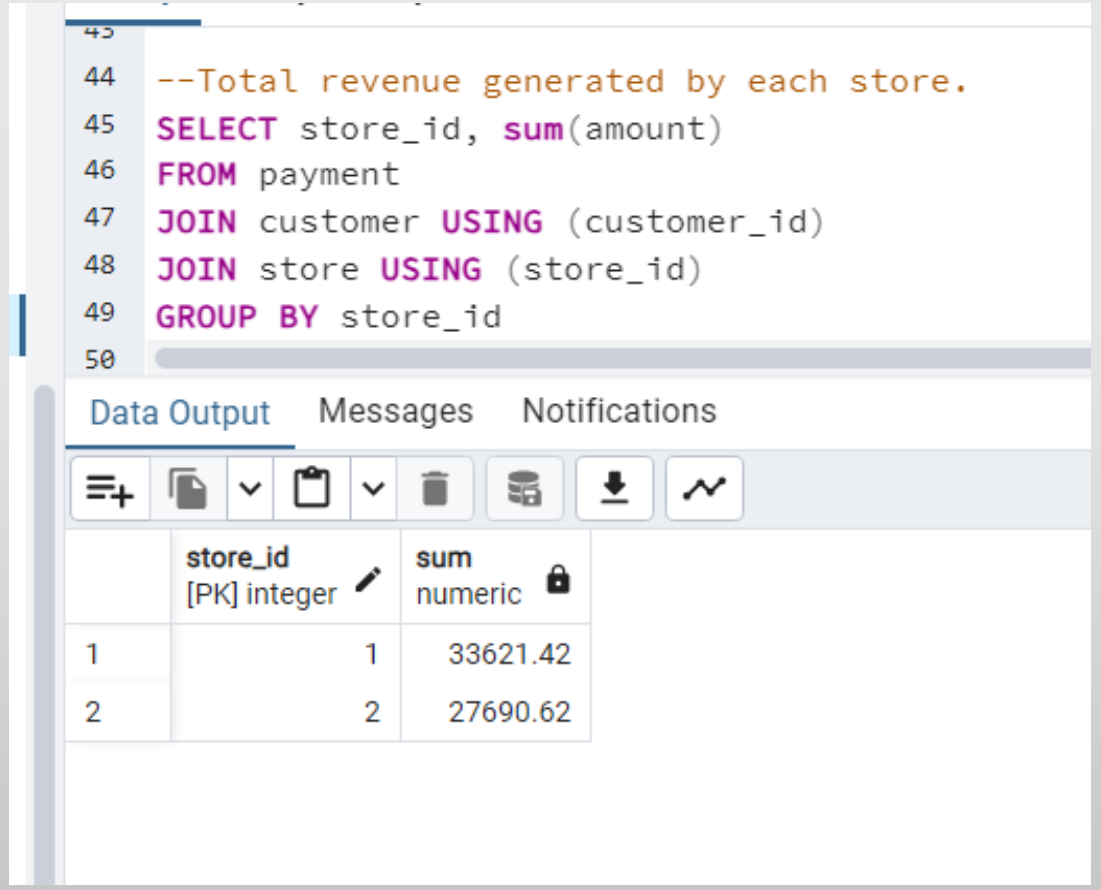
```sql
--Number of rentals per month for the past year.

SELECT MIN(rental_date), MAX(rental_date)
FROM rental

SELECT
    EXTRACT (MONTH FROM rental_date) rental_month,
    COUNT(*) rental_no
FROM rental
WHERE EXTRACT(YEAR FROM rental_date) = 2005
GROUP BY EXTRACT (MONTH FROM rental_date)
ORDER BY rental_month;
```

Data Output | Messages | Notifications

	rental_month numeric	rental_no bigint
1	5	1156
2	6	2311
3	7	6709
4	8	5686

- ✓ **Calculating number of rental per month helps to identify trend which provides insight for decision making**

TOTAL REVENUE GENERATED BY EACH STORE.



```
44  --Total revenue generated by each store.
45  SELECT store_id, sum(amount)
46  FROM payment
47  JOIN customer USING (customer_id)
48  JOIN store USING (store_id)
49  GROUP BY store_id
50
```

Data Output | Messages | Notifications

	store_id [PK] integer	sum numeric
1	1	33621.42
2	2	27690.62

- ✓ **Knowing store that generate more revenue is a very important insight in a video rental store. This will help drive decision making.**

AVERAGE RENTAL DURATION FOR EACH FILM.

Query | Query History

```
56  --Calculate the average rental duration for each film
57
58  SELECT fc.film_id,
59         AVG(AGE(return_date, rental_date)) AS avg_duration
60  FROM rental r
61  JOIN inventory i USING (inventory_id)
62  JOIN film_category fc USING (film_id)
63  GROUP BY fc.film_id
64  ORDER BY fc.film_id;
65
66
```

Data Output | Messages | Notifications

	film_id smallint	avg_duration interval
1	1	4 days 23:55:51.818182
2	2	5 days 15:25:30
3	3	2 days 35:10:30
4	4	4 days 18:12:00
5	5	6 days 26:33:32.727272
6	6	5 days 21:03:42.857143
7	7	3 days 37:58:04
8	8	4 days 25:21:10
9	9	5 days 13:46:00
10	10	4 days 19:10:54.782609
11	11	4 days 24:21:50
12	12	4 days 31:03:43.846154
13	13	4 days 15:04:30
14	15	4 days 20:38:57.272727

Total rows: 958 of 958 | Query complete 00:00:00.109

- ✓ **Calculating average rental duration helps to understand the trend. It provides insight as to whether a film rental duration should be increased or not.**

FILMS THAT HAVE NOT BEEN RENTED IN THELAST 90 DAYS.

Query | Query History

```
77  --Identify films that have not been rented in the last 90 days.
78
79  WITH recent_rentals AS (
80      SELECT DISTINCT r.inventory_id
81      FROM rental r
82      WHERE r.rental_date >= TIMESTAMP '2006-02-14 15:16:03' - INTERVAL '90 days'
83  )
84  SELECT f.film_id, f.title
85  FROM film f
86  WHERE f.film_id NOT IN (
87      SELECT i.film_id
88      FROM inventory i
89      WHERE i.inventory_id IN (SELECT inventory_id FROM recent_rentals)
90  )
91  ORDER BY f.film_id
92
```

Data Output | Messages | Notifications

	film_id [PK] integer	title character varying (255)
1	1	Academy Dinosaur
2	3	Adaptation Holes
3	6	Agent Truman
4	7	Airplane Sierra
5	8	Airport Pollock
6	9	Alabama Devil
7	10	Aladdin Calendar
8	11	Alamo Videotape
9	12	Alaska Phantom

Total rows: 832 of 832 | Query complete 00:00:00.225

- ✓ **Knowing films that have not been rented for sometime provides insight and helps drive business decision.**

DATA MANIPULATION ON RENTAL TABLE.

Step 1

Query | Query History

```
--Update the rental table to mark rentals that are overdue (return_date is null and rental_duration exceeded)
SELECT f.film_id, i.inventory_id, rental_duration,
      (return_date - rental_date) AS duration
FROM rental r
JOIN inventory i Using (inventory_id)
JOIN film f Using (film_id)
GROUP BY f.film_id, i.inventory_id,r.return_date, r.rental_date

SELECT * FROM rental

SELECT f.film_id, i.inventory_id,rental_duration,
       (return_date - rental_date) AS duration,
       CASE
         WHEN (return_date - rental_date) > (rental_duration * interval '1 day') THEN 'overdue'
         WHEN (return_date - rental_date) <= (rental_duration * interval '1 day') THEN 'Okay'
         ELSE 'NULL'
       END AS status
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
GROUP BY f.film_id, i.inventory_id, r.return_date, r.rental_date;
```

Data Output | Messages | Notifications

	rental_id [PK] integer	rental_date timestamp without time zone	inventory_id integer	customer_id smallint	return_date timestamp without time zone	staff_id smallint	last_update timestamp without time zone
1	36	2005-05-25 04:36:26	1308	88	2005-05-29 00:31:26	1	2024-07-18 22:33:41.262902
2	37	2005-05-25 04:44:31	403	535	2005-05-29 01:03:31	1	2024-07-18 22:33:41.262902
3	38	2005-05-25 04:47:44	2540	302	2005-06-01 00:58:44	1	2024-07-18 22:33:41.262902
4	39	2005-05-25 04:51:46	4466	207	2005-05-31 03:14:46	2	2024-07-18 22:33:41.262902

Total rows: 1000 of 16044 | Query complete 00:00:00.312

Query | Query History

```
--Update the rental table to mark rentals that are overdue
SELECT f.film_id, i.inventory_id, rental_duration,
      (return_date - rental_date) AS duration
FROM rental r
JOIN inventory i Using (inventory_id)
JOIN film f Using (film_id)
GROUP BY f.film_id, i.inventory_id,r.return_date, r.rental_
```

Data Output | Messages | Notifications

	film_id integer	inventory_id integer	rental_duration smallint	duration interval
1	556	2537	6	20:52:00
2	846	3866	3	6 days 04:26:00
3	727	3301	6	19:13:00
4	50	223	3	6 days 23:37:00
5	683	3108	5	3 days 04:01:00
6	216	970	3	8 days 19:57:00
7	654	2985	4	5 days 20:36:00
8	624	2848	3	5 days 18:49:00
9	962	4400	7	8 days 02:55:00
10	4	19	5	1 day 21:53:00
11	11	55	6	6 days 18:31:00
12	346	1575	4	7 days 20:50:00
13	850	3892	7	1 day 20:17:00
14	732	3329	5	8 days 04:05:00
15	847	3874	7	7 days 21:00:00

Total rows: 1000 of 16044 | Query complete 00:00:00.126

Rental table overview. And query to see the exceeded duration

UPDATE RENTAL TABLE TO MARK RENTALS THAT ARE OVERDUE.

Step 2

Query | Query History

```
116 --To now update the rental table, we first created the table
117 ALTER TABLE rental
118 ADD COLUMN rental_status VARCHAR
119
```

This query was used to create a new table and then populated it

Query | Query History

```
122 --Then to update
123 WITH overdue AS (
124     SELECT
125         r.rental_id,
126         CASE
127             WHEN (r.return_date - r.rental_date) > (f.rental_duration * interval '1 day') THEN 'overdue'
128             WHEN (r.return_date - r.rental_date) <= (f.rental_duration * interval '1 day') THEN 'Okay'
129             ELSE 'NULL'
130         END AS rental_status
131     FROM rental r
132     JOIN inventory i ON r.inventory_id = i.inventory_id
133     JOIN film f ON i.film_id = f.film_id
134 )
135 UPDATE rental
136 SET rental_status = (SELECT o.rental_status FROM overdue o WHERE rental.rental_id = o.rental_id)
137 WHERE rental.rental_id IN (SELECT o.rental_id FROM overdue o);
138
```

Data Output | Messages | Notifications

```
UPDATE 16044

Query returned successfully in 31 secs 356 msec.
```

Total rows: 1000 of 16044 | Query complete 00:00:31.356

UPDATE RENTAL TABLE TO MARK RENTALS THAT ARE OVERDUE.

Step 3

Query | Query History | Scratch Pa

```
SELECT * FROM rental
```

Data Output | Messages | Notifications

	rental_id [PK] integer	rental_date timestamp without time zone	inventory_id integer	customer_id smallint	return_date timestamp without time zone	staff_id smallint	last_update timestamp without time zone	rental_status character varying
1	70	2005-05-25 10:15:23	2168	73	2005-05-27 05:56:23	2	2024-07-25 15:22:48.212316	Okay
2	71	2005-05-25 10:26:39	2408	100	2005-05-28 04:59:39	1	2024-07-25 15:22:48.212316	Okay
3	72	2005-05-25 10:52:13	2260	48	2005-05-28 05:52:13	2	2024-07-25 15:22:48.212316	Okay
4	73	2005-05-25 11:00:07	517	391	2005-06-01 13:56:07	2	2024-07-25 15:22:48.212316	overdue
5	74	2005-05-25 11:09:48	1744	265	2005-05-26 12:23:48	2	2024-07-25 15:22:48.212316	Okay
6	75	2005-05-25 11:13:34	3393	510	2005-06-03 12:58:34	1	2024-07-25 15:22:48.212316	overdue
7	76	2005-05-25 11:30:37	3021	1	2005-06-03 12:00:37	2	2024-07-25 15:22:48.212316	overdue
8	77	2005-05-25 11:31:59	1303	451	2005-05-26 16:53:59	2	2024-07-25 15:22:48.212316	Okay
9	78	2005-05-25 11:35:18	4067	135	2005-05-31 12:48:18	2	2024-07-25 15:22:48.212316	overdue
10	79	2005-05-25 12:11:07	3299	245	2005-06-03 10:54:07	2	2024-07-25 15:22:48.212316	overdue
11	80	2005-05-25 12:12:07	2478	314	2005-05-31 17:46:07	2	2024-07-25 15:22:48.212316	overdue
12	81	2005-05-25 12:15:19	2610	286	2005-06-02 14:08:19	2	2024-07-25 15:22:48.212316	overdue
13	82	2005-05-25 12:17:46	1388	427	2005-06-01 10:48:46	1	2024-07-25 15:22:48.212316	overdue
14	83	2005-05-25 12:30:15	466	131	2005-05-27 15:40:15	1	2024-07-25 15:22:48.212316	Okay
15	84	2005-05-25 12:36:30	1829	492	2005-05-29 18:33:30	1	2024-07-25 15:22:48.212316	Okay
16	85	2005-05-25 13:05:34	470	414	2005-05-29 16:53:34	1	2024-07-25 15:22:48.212316	Okay
17	86	2005-05-25 13:36:12	2275	266	2005-05-30 14:53:12	1	2024-07-25 15:22:48.212316	overdue
18	87	2005-05-25 13:52:43	1586	331	2005-05-29 11:12:43	2	2024-07-25 15:22:48.212316	Okay
19	88	2005-05-25 14:13:54	2221	53	2005-05-29 09:32:54	2	2024-07-25 15:22:48.212316	overdue
20	89	2005-05-25 14:28:29	2181	499	2005-05-29 14:33:29	1	2024-07-25 15:22:48.212316	Okay

Total rows: 1000 of 16044 | Query complete 00:00:00.179

```
SELECT * FROM rental
WHERE return_date IS NULL
```

Data Output | Messages | Notifications

	rental_id [PK] integer	rental_date timestamp without time zone	inventory_id integer	customer_id smallint	return_date timestamp without time zone	staff_id smallint	last_update timestamp without time zone	rental_status character varying
1	12101	2006-02-14 15:16:03	1556	479	[null]	1	2024-07-25 15:22:48.212316	NULL
2	11563	2006-02-14 15:16:03	1545	83	[null]	1	2024-07-25 15:22:48.212316	NULL
3	14060	2006-02-14 15:16:03	3100	175	[null]	2	2024-07-25 15:22:48.212316	NULL
4	11577	2006-02-14 15:16:03	4106	219	[null]	2	2024-07-25 15:22:48.212316	NULL
5	11593	2006-02-14 15:16:03	817	99	[null]	1	2024-07-25 15:22:48.212316	NULL
6	11646	2006-02-14 15:16:03	478	11	[null]	2	2024-07-25 15:22:48.212316	NULL
7	11652	2006-02-14 15:16:03	1622	597	[null]	2	2024-07-25 15:22:48.212316	NULL
8	13719	2006-02-14 15:16:03	3547	208	[null]	1	2024-07-25 15:22:48.212316	NULL
9	11672	2006-02-14 15:16:03	3947	521	[null]	2	2024-07-25 15:22:48.212316	NULL
10	11676	2006-02-14 15:16:03	4496	216	[null]	2	2024-07-25 15:22:48.212316	NULL
11	11709	2006-02-14 15:16:03	1720	330	[null]	1	2024-07-25 15:22:48.212316	NULL
12	11739	2006-02-14 15:16:03	4568	373	[null]	2	2024-07-25 15:22:48.212316	NULL
13	11754	2006-02-14 15:16:03	3747	163	[null]	2	2024-07-25 15:22:48.212316	NULL
14	11757	2006-02-14 15:16:03	1295	550	[null]	2	2024-07-25 15:22:48.212316	NULL
15	11782	2006-02-14 15:16:03	4098	354	[null]	1	2024-07-25 15:22:48.212316	NULL
16	11847	2006-02-14 15:16:03	1784	337	[null]	1	2024-07-25 15:22:48.212316	NULL
17	11848	2006-02-14 15:16:03	3680	152	[null]	1	2024-07-25 15:22:48.212316	NULL
18	11866	2006-02-14 15:16:03	2758	200	[null]	2	2024-07-25 15:22:48.212316	NULL

Total rows: 183 of 183 | Query complete 00:00:00.178

Updated Table Overview

ANALYSIS, VISUALIZATION AND INSIGHT.

Top 10 rented films

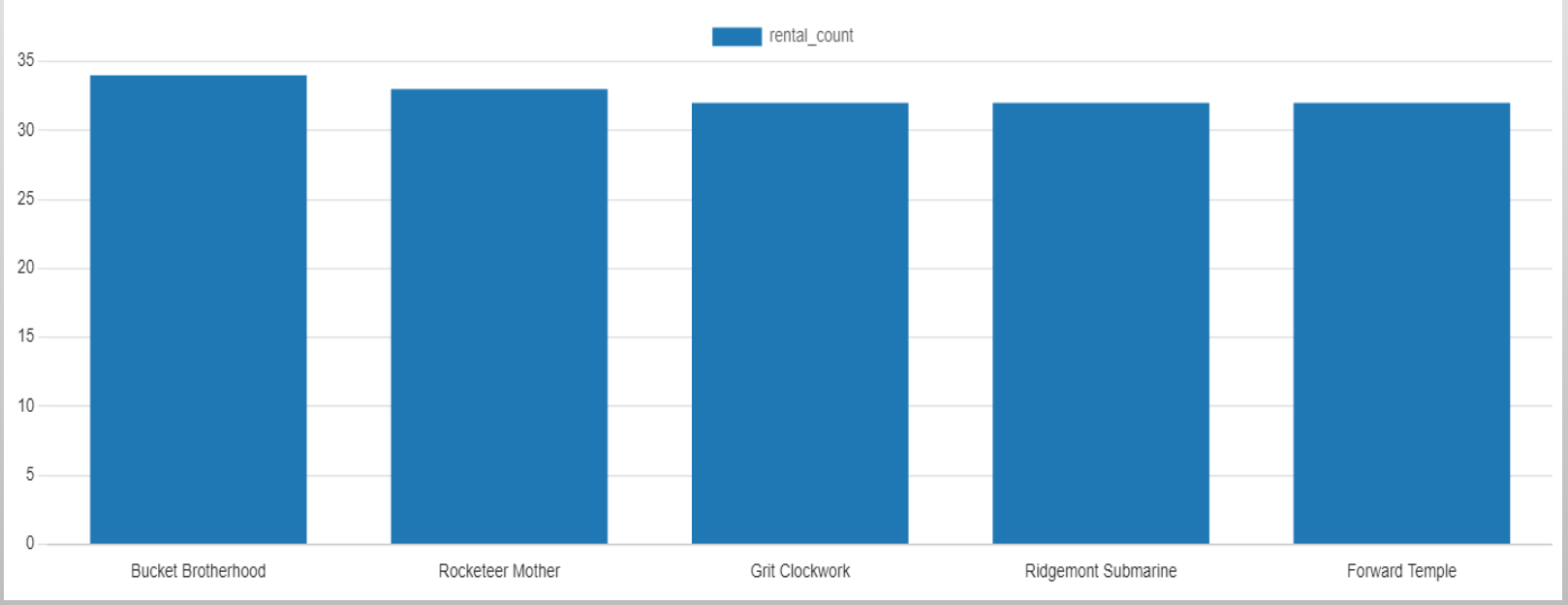
Query | Query History

```
146 --Top 10 popular films
147 SELECT
148     f.film_id,
149     f.title,
150     COUNT(r.rental_id) AS rental_count
151 FROM
152     rental r
153 JOIN
154     inventory i ON r.inventory_id = i.inventory_id
155 JOIN
156     film f ON i.film_id = f.film_id
157 GROUP BY
158     f.film_id, f.title
159 ORDER BY
160     rental_count DESC
161 LIMIT 10;
162
```

Data Output | Messages | Notifications

	film_id [PK] integer	title character varying (255)	rental_count bigint
2	738	Rocketeer Mother	33
3	730	Ridgemont Submarine	32
4	767	Scalawag Duck	32
5	382	Grit Clockwork	32
6	331	Forward Temple	32
7	489	Juggler Hardly	32
8	31	Apache Divine	31
9	735	Robbers Joon	31
10	418	Hobbit Alien	31

Total rows: 10 of 10 | Query complete 00:00:00.179



- ✓ I generated all my visuals from PgAdmin. The alternative will be to import the data into excel or Power Bi and visualize with them

- ✓ **Bucket Brotherhood** holds the highest rental count followed by **Rocketeer Mother.** From this bar chart, it is concluded that our rental data demonstrates a varied and vibrant interest in different types of movies. The high rental counts for these specific movies indicate trends that could guide future film acquisitions and marketing strategies ensuring continuous meeting the diverse taste of customers in order to increase revenue.

ANALYSIS AND INSIGHT.

Top 10 rented categories

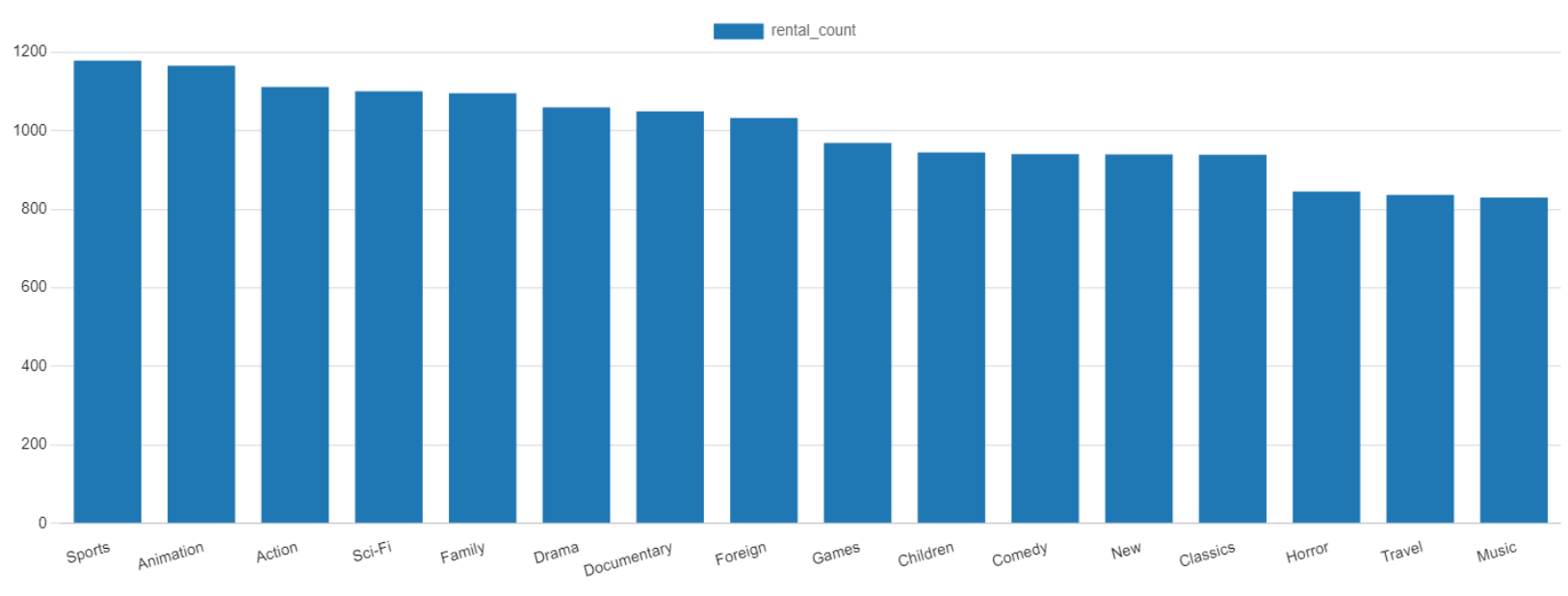
Query | Query History

```
164 --Top 10 Categories
165
166 SELECT
167     c.category_id,
168     c.name AS category_name,
169     COUNT(r.rental_id) AS rental_count
170 FROM
171     rental r
172 JOIN
173     inventory i ON r.inventory_id = i.inventory_id
174 JOIN
175     film_category fc ON i.film_id = fc.film_id
176 JOIN category c ON fc.category_id = c.category_id
177 GROUP BY
178     c.category_id, c.name
179 ORDER BY
180     rental_count DESC
181 LIMIT 10;
```

Data Output | Messages | Notifications

	category_id [PK] integer	category_name character varying (25)	rental_count bigint
1	15	Sports	1179
2	2	Animation	1166
3	1	Action	1112
4	14	Sci-Fi	1101
5	8	Family	1096
6	7	Drama	1060
7	6	Documentary	1050
8	9	Foreign	1033
9	10	Games	969
10	3	Children	945

Total rows: 10 of 10 | Query complete 00:00:00.219



- ✓ **Sports films have the highest rental count while Music films have the least rental**
- ✓ **This significantly shows customers preference and shows insight that could guide future film acquisitions and marketing strategies ensuring continuous meeting the diverse taste of customers in order to increase revenue.**

ANALYSIS AND INSIGHT.

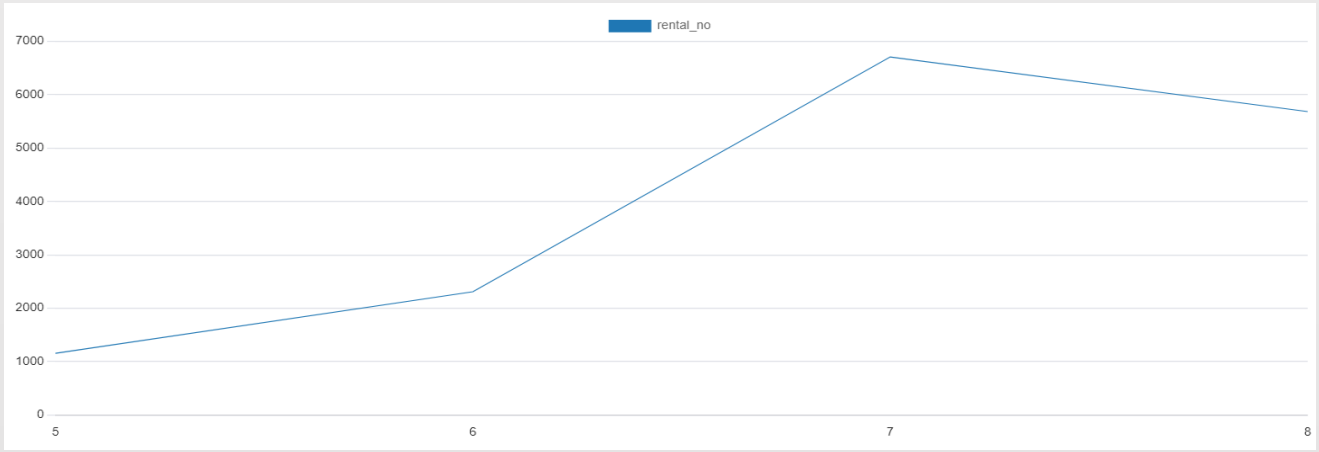
Rental trend by Month

Query | Query History

```
183 --Rental trend by month and year
184 --By month
185 SELECT
186     EXTRACT (MONTH FROM rental_date) rental_month,
187     COUNT(*) rental_no
188 FROM rental
189 WHERE EXTRACT(YEAR FROM rental_date) = 2005
190 GROUP BY EXTRACT (MONTH FROM rental_date)
191 ORDER BY rental_month;
192
```

Data Output | Messages | Notifications

	rental_month numeric	rental_no bigint
1	5	1156
2	6	2311
3	7	6709
4	8	5686



✓ **2005 has more rental records. Digging deep I observed that 2006 only rented movies in February. Month of July 2005, has highest rental number followed by August while the least is in may. This shows that people rent more movies during summer and holiday season**

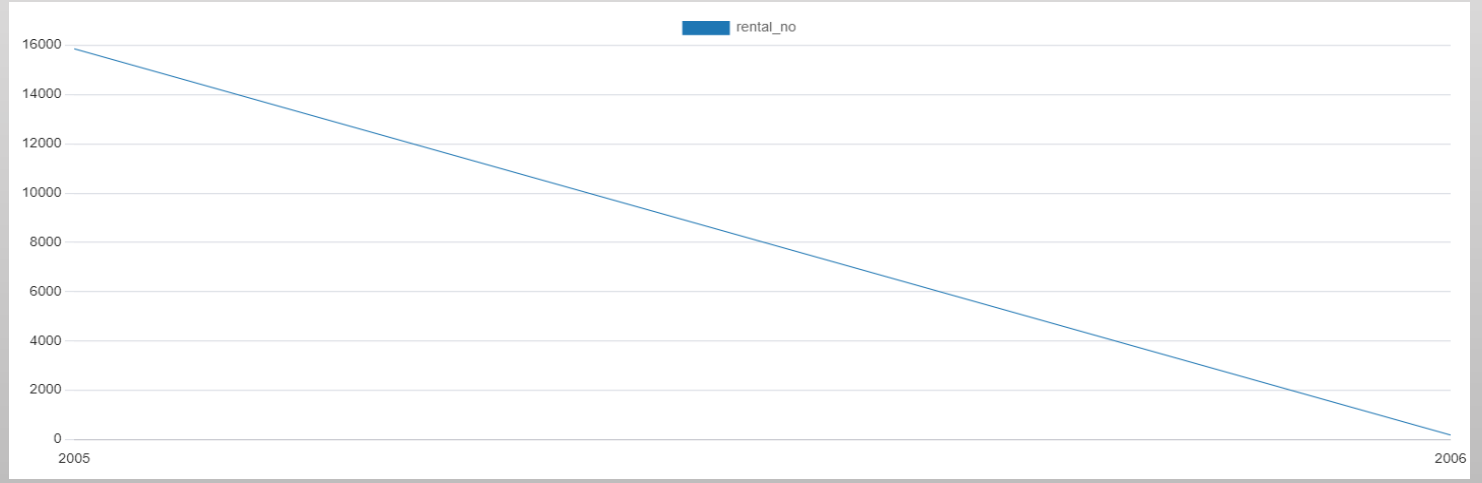
Rental trend by Year

Query | Query History

```
193 --Rental trend by year
194 SELECT
195     EXTRACT (YEAR FROM rental_date) rental_year,
196     COUNT(*) rental_no
197 FROM rental
198 WHERE EXTRACT(YEAR FROM rental_date) IN (2005, 2006)
199 GROUP BY EXTRACT (YEAR FROM rental_date)
200 ORDER BY rental_year;
201
202 --Monthly and yearly trend
```

Data Output | Messages | Notifications

	rental_year numeric	rental_no bigint
1	2005	15862
2	2006	182



ANALYSIS AND INSIGHT.

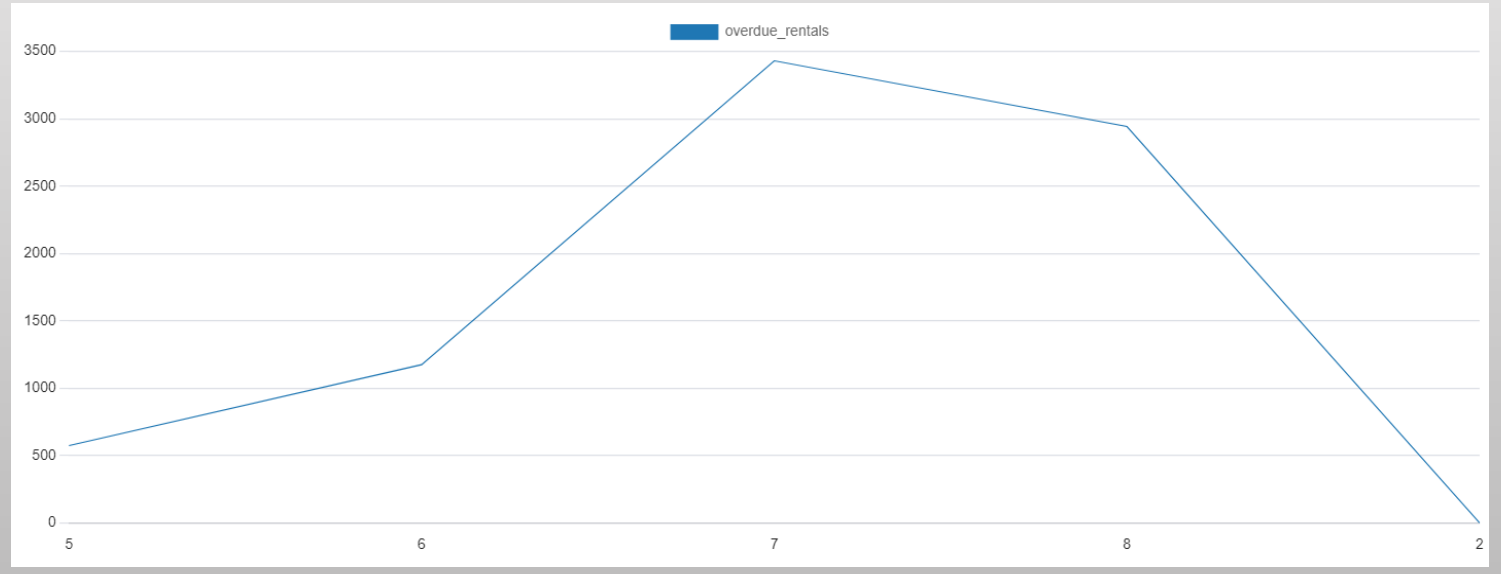
Overdue and late return pattern

Query | Query History

```
212 --Identify any patterns in late returns and overdue rentals.
213 --Overdue rentals by Month and Year
214 SELECT
215     EXTRACT(YEAR FROM rental_date) AS rental_year,
216     EXTRACT(MONTH FROM rental_date) AS rental_month,
217     COUNT(*) AS total_rentals,
218     SUM(CASE WHEN (return_date - rental_date) > (rental_duration * INTERVAL '1 day') THEN 1 ELSE 0 END) AS overdue_renta
219 FROM rental r
220 JOIN inventory i ON r.inventory_id = i.inventory_id
221 JOIN film f ON i.film_id = f.film_id
222 GROUP BY EXTRACT(YEAR FROM rental_date), EXTRACT(MONTH FROM rental_date)
223 ORDER BY rental_year, rental_month;
224
225 --Customers who frequently return rentals late
```

Data Output | Messages | Notifications

	rental_year numeric	rental_month numeric	total_rentals bigint	overdue_rentals bigint
1	2005	5	1156	574
2	2005	6	2311	1175
3	2005	7	6709	3430
4	2005	8	5686	2942
5	2006	2	182	0



- ✓ **Month of July 2005, has highest late return pattern followed by August while the least is in may which could be as a result of high rentals in those months**

ANALYSIS AND INSIGHT.

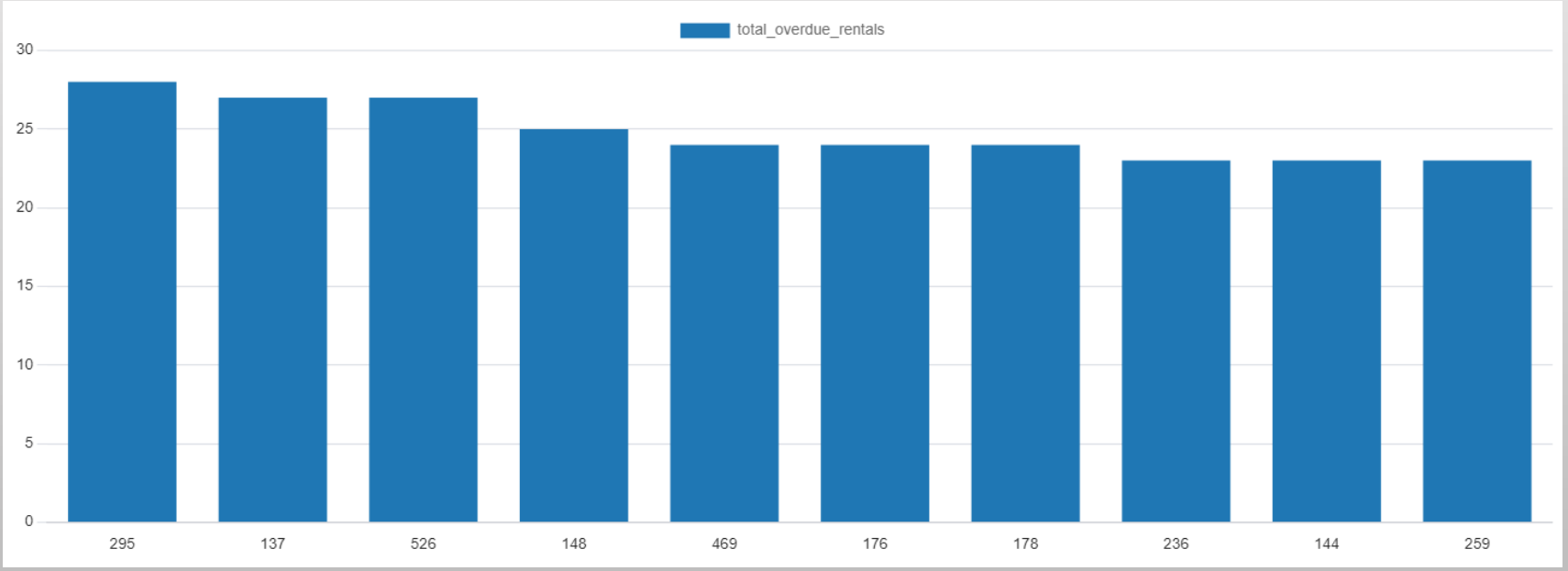
Top 10 customer that always return films late with their ID

Query | Query History

```
225 --Customers who frequently return rentals late
226 SELECT
227     customer_id,
228     COUNT(*) AS total_overdue_rentals
229 FROM rental
230 WHERE rental_status = 'overdue'
231 GROUP BY customer_id, rental_status
232 ORDER BY total_overdue_rentals DESC
233 LIMIT 10
234
```

Data Output | Messages | Notifications

	customer_id smallint	total_overdue_rentals bigint
1	295	28
2	137	27
3	526	27
4	148	25
5	469	24
6	176	24
7	178	24
8	236	23
9	144	23
10	259	23



- ✓ **Customer ID 295 is the cutomer with highest over due rental of 28. This insight is necessary such that company can take decision on how to handle the customers.**

INSIGHT AND RECOMMENDATION.

- From the analysis, Bucket Brotherhood holds the highest rental count followed by Rocketeer Mother. The high rental counts for these specific movies indicate trends that could guide future film acquisitions and marketing strategies ensuring continuous meeting the diverse taste of customers in order to increase revenue. More of these type of should be stored.

- Sports films have the highest rental count while Music films have the least rental
  This significantly shows customers prefer Sports films and more of sports films should be stored.

- 2005 has more rental records and the 7th month July, has highest rental number followed by August while the least is in may. This shows that people rent more movies during summer and holiday season

- Store ID 1 generated more revenue of 33621.42 compared to Store ID 2 that generated 27690.62

- Month of July 2005, has highest late return pattern followed by August while the least is in may which could be as a result of high rentals in those months.

- Customer ID 295 is the cutomer with highest over due rental of 28. This insight is necessary such that company can take decision on how to handle the customers.

- In summary, company can have a policy that penalize customers that return late, reward customers that return early and also reward top rental customers. This will encourage customers hence leads to more rentals which inturn generate more revenue.

- Also, films not rented in the last 90 days should be archived to create space for more films that will generate revenue