

DVD Rental SQL Project

By: Matthew Cooper

Introduction:

For this project, I am using the DVD Rental database that is often used for learning SQL. The goal for this project is to develop a few questions to answer using SQL queries.

Database details (and the database itself) can be found:

<https://www.postgresqltutorial.com/postgresql-getting-started/postgresql-sample-database/>

The website itself does not mention if the data is cleaned and prepared for analysis. I decided that it would be best to double check the data and look for potential errors. I don't think there will be any, but it doesn't hurt to check.

The Project:

Due to the number of tables and columns in this database, I will just describe the error-checking results.

Using very basic queries, I displayed first 10 rows of data for each table to check the data type for each column and to check for missing data.

#####

The Queries used:

Type: `select * from table_name LIMIT 5;`

NULL: `select * from table_name WHERE NOT (table_name IS NOT NULL) LIMIT 5;`

#####

The Table Results

actor: no problems

address: has nulls in 'address2', 'postal_code', and 'phone'

 'postal_code' has type character

 'phone' has type character

 'Phone' type can be ignored

 'Postal' type can be ignored

category: no problems

city: no problems

country: no problems
customer: no problems
film: no problems
film_actor: no problems
film_category: no problems
inventory: no problems
language: no problems
payment: no problems
rental: has nulls in 'return_date'
staff: has null in 'picture'
 'picture' can be ignored
store: no problems

A few issues were found:

The 'address' table contained nulls in three (3) columns. Two (2) of these columns of type 'character' when they could be numerical or integer.

The 'rental' table contained nulls in one (1) column I may want to add a new column of type bool to determine whether rental was returned or not.

The 'staff' table contained a null in one (1) column, but it is a non-issue and can be ignored because the column is not useful.

Using the following query, I wanted to look at if the postal_code would be useful at all. After running the query, the results showed that the 'postal_code' column really wouldn't provide any information that I couldn't get from other address columns.

```
select postal_code,  
       count(postal_code)  
from address  
group by postal_code  
order by count(postal_code) DESC;
```

Looking at the other errors found, I determined that there was no need to fix anything as the affected columns were not going to be used in any analysis that I may perform.

Going through the data as I did, gave me the chance to look at the data and start to get ideas on how to analyze it. I want to create a few questions to use queries to try to answer.

The Questions
#####

Q-1: What is the most rented film genre by store?

```
select
    count(rental.rental_id) as num_rented,
    /*count(rental.staff_id),
    count(rental.inventory_id),
    count(inventory.film_id),
    count(category.category_id),*/
    category.name,
    store.store_id
from rental
Inner Join staff
    on rental.staff_id = staff.staff_id
Inner Join inventory
    on rental.inventory_id = inventory.inventory_id
Inner join film_category
    on film_category.film_id = inventory.film_id
Inner join category
    on film_category.category_id = category.category_id
Inner join store
    on staff.store_id = store.store_id
group by category.name, store.store_id
order by num_rented DESC, store.store_id
LIMIT 5;
```

	num_rented bigint	name character varying (25)	store_id integer
1	614	Sports	2
2	584	Animation	2
3	582	Animation	1
4	565	Family	1
5	565	Sports	1






Q-2: Average rental price by genre by store?

```
select
    ROUND(AVG(payment.amount), 2) as Avg_payment,
    category.name as Genre,
    store.store_id as Store
from rental
Inner Join inventory
    on rental.inventory_id = inventory.inventory_id
Inner join film_category
    on film_category.film_id = inventory.film_id
Inner join category
    on film_category.category_id = category.category_id
Inner join payment
    on rental.customer_id = payment.customer_id
Inner Join staff
    on rental.staff_id = staff.staff_id
Inner join store
    on staff.store_id = store.store_id
group by category.name, store.store_id
order by store.store_id ASC, Avg_payment DESC;
```

	avg_payment numeric	genre character varying (25)	store integer		avg_payment numeric	genre character varying (25)	store integer
1	4.26	Horror	1	17	4.25	Comedy	2
2	4.26	Sports	1	18	4.25	Games	2
3	4.24	Games	1	19	4.23	Family	2
4	4.24	Animation	1	20	4.23	Horror	2
5	4.22	Travel	1	21	4.21	Action	2
6	4.22	Comedy	1	22	4.21	Sci-Fi	2
7	4.22	Classics	1	23	4.21	Travel	2
8	4.21	Family	1	24	4.21	Sports	2
9	4.21	Foreign	1	25	4.21	New	2
10	4.21	New	1	26	4.21	Music	2
11	4.21	Action	1	27	4.20	Foreign	2
12	4.19	Music	1	28	4.18	Classics	2
13	4.19	Sci-Fi	1	29	4.17	Children	2
14	4.18	Children	1	30	4.17	Animation	2
15	4.17	Documentary	1	31	4.17	Drama	2
16	4.17	Drama	1	32	4.16	Documentary	2



Q-3: What are the top ten customers with the most rentals?

```
select
    count(rental.rental_id) as amt_rented,
    customer.first_name,
    customer.last_name,
    customer.email
from rental
Inner Join customer
    on rental.customer_id = customer.customer_id
group by customer.first_name, customer.last_name, customer.email
order by amt_rented DESC
Limit 10;
```

	 amt_rented bigint 	first_name character varying (45) 	last_name character varying (45) 	email character varying (50) 
1	46	Eleanor	Hunt	eleanor.hunt@sakilacustomer.org
2	45	Karl	Seal	karl.seal@sakilacustomer.org
3	42	Marcia	Dean	marcia.dean@sakilacustomer.org
4	42	Clara	Shaw	clara.shaw@sakilacustomer.org
5	41	Tammy	Sanders	tammy.sanders@sakilacustomer.org
6	40	Wesley	Bull	wesley.bull@sakilacustomer.org
7	40	Sue	Peters	sue.peters@sakilacustomer.org
8	39	Marion	Snyder	marion.snyder@sakilacustomer.org
9	39	Rhonda	Kennedy	rhonda.kennedy@sakilacustomer.org
10	39	Tim	Cary	tim.cary@sakilacustomer.org

Q-4: Total rental payments by store (also include total number of rentals per store)

```
select
    ROUND(SUM(payment.amount), 2) as total_payment,
    count(rental.rental_id) as num_rental,
    category.name as Genre,
    store.store_id as Store
from rental
Inner Join inventory
    on rental.inventory_id = inventory.inventory_id
Inner join film_category
    on film_category.film_id = inventory.film_id
Inner join category
    on film_category.category_id = category.category_id
Inner join payment
    on rental.customer_id = payment.customer_id
Inner Join staff
    on rental.staff_id = staff.staff_id
Inner join store
    on staff.store_id = store.store_id
group by category.name, store.store_id
order by store.store_id ASC, total_payment DESC;
```

	 total_payment numeric 🔒	num_rental bigint 🔒	genre character varying (25) 🔒	store integer 🔒		 total_payment numeric 🔒	num_rental bigint 🔒	genre character varying (25) 🔒	store integer 🔒
1	63181.04	14896	Animation	1	17	64666.37	15363	Sports	2
2	60880.99	14301	Sports	1	18	61790.91	14809	Animation	2
3	60870.38	14462	Family	1	19	59063.79	14021	Action	2
4	59498.77	14123	Action	1	20	58216.40	13860	Foreign	2
5	58426.41	13959	Sci-Fi	1	21	57701.88	13712	Sci-Fi	2
6	55768.42	13358	Drama	1	22	57352.49	13551	Family	2
7	55082.81	13219	Documentary	1	23	55939.61	13439	Documentary	2
8	53337.18	12582	Games	1	24	55305.34	13266	Drama	2
9	52715.91	12509	New	1	25	51201.46	12054	Games	2
10	51997.46	12354	Foreign	1	26	49956.52	11748	Comedy	2
11	51025.93	12207	Children	1	27	48997.85	11715	Classics	2
12	50716.75	12025	Classics	1	28	48660.39	11661	Children	2
13	50190.16	11884	Comedy	1	29	47813.37	11363	New	2
14	45532.32	10868	Music	1	30	46696.06	11094	Travel	2
15	44995.26	10574	Horror	1	31	44449.81	10519	Horror	2
16	41861.74	9926	Travel	1	32	41653.07	9893	Music	2

Q-5: Average rental time period by genre? (in days)

```
select
    category.name as genre,
    ROUND( CAST(float8 (AVG(EXTRACT(DAY FROM rental.return_date -
rental.rental_date)))) as numeric), 1) as avg_rent_time
    from rental
Inner Join inventory
    on rental.inventory_id = inventory.inventory_id
Inner join film_category
    on film_category.film_id = inventory.film_id
Inner join category
    on film_category.category_id = category.category_id
group by genre
Order by avg_rent_time DESC;
```

	genre character varying (25)	avg_rent_time numeric
1	Sports	4.7
2	Games	4.7
3	Comedy	4.7
4	Music	4.6
5	Documentary	4.6
6	Horror	4.6
7	Sci-Fi	4.6
8	Family	4.5
9	New	4.5
10	Action	4.5
11	Foreign	4.5
12	Classics	4.4
13	Drama	4.4
14	Animation	4.4
15	Children	4.4
16	Travel	4.3

._*_*_*_*_-END OF PROJECT-*_*_*_*_*_-