

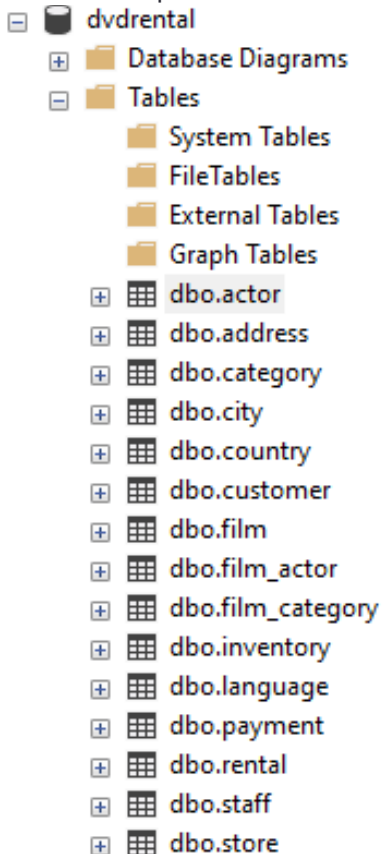
Análisis SQL: Base de Datos de Alquiler de Películas

Estructura de la Base de datos.

Primero realizamos un analisis exploratorio de la base de datos, esto es importante para entender la estructura de la base de datos, identificar patrones y detectar posibles problemas.

Conociendo la estructura de la Base de datos.

El Objetivo es comprender cómo están organizadas las tablas y sus relaciones, anexo una pantalla de las tablas que visualmente podemos encontrar en esta BD



Ahora inspeccionaremos el contenido de cada una de estas tablas para obtener información básica, esto nos permitirá obtener información básica como las columnas y los tipos de datos de cada tabla, a continuación, anexo las pantallas de estas consultas. En algunos casos realizo una consulta del contenido de las columnas (solo cuando son varias columnas, ya que por el espacio no tengo visibilidad completa de la tabla)

SQLQuery1.sql* X

1 SELECT TOP 10 * FROM dbo.actor;

91 %

Results Messages

	actor_id	first_name	last_name	last_update
1	1	Penelope	Guiness	2013-05-26 14:47:57.62
2	2	Nick	Wahlberg	2013-05-26 14:47:57.62
3	3	Ed	Chase	2013-05-26 14:47:57.62
4	4	Jennifer	Davis	2013-05-26 14:47:57.62
5	5	Johnny	Lollobrigida	2013-05-26 14:47:57.62
6	6	Bette	Nicholson	2013-05-26 14:47:57.62
7	7	Grace	Mostel	2013-05-26 14:47:57.62
8	8	Matthew	Johansson	2013-05-26 14:47:57.62
9	9	Joe	Swank	2013-05-26 14:47:57.62
10	10	Christian	Gable	2013-05-26 14:47:57.62

SQLQuery1.sql* X

```

1 SELECT TOP 10 * FROM dbo.actor;
2 SELECT TOP 10 * FROM dbo.address;

```

91 %

Results Messages

	address_id	address	address2	district	city_id	postal_code	phone	last_update
1	1	47 MySakila Drive		Alberta	300	''''		2006-02-15 09:45:30
2	2	28 MySQL Boulevard		QLD	576	''''		2006-02-15 09:45:30
3	3	23 Workhaven Lane		Alberta	300	''''	14033335568	2006-02-15 09:45:30
4	4	1411 Lillydale Drive		QLD	576	''''	6172235589	2006-02-15 09:45:30
5	5	1913 Hanoi Way	''''	Nagasaki	463	35200	28303384290	2006-02-15 09:45:30
6	6	1121 Loja Avenue	''''	California	449	17886	838635286649	2006-02-15 09:45:30
7	7	692 Joliet Street	''''	Attika	38	83579	448477190408	2006-02-15 09:45:30
8	8	1566 Inegl Manor	''''	Mandalay	349	53561	705814003527	2006-02-15 09:45:30
9	9	53 Idfu Parkway	''''	Nantou	361	42399	10655648674	2006-02-15 09:45:30
10	10	1795 Santiago de Compostela Way	''''	Texas	295	18743	860452626434	2006-02-15 09:45:30

SQLQuery1.sql* X

```

1 SELECT TOP 10 * FROM dbo.actor;
2 SELECT TOP 10 * FROM dbo.address;
3 SELECT TOP 10 * FROM dbo.category;

```

91 %

Results Messages

	category_id	name	last_update
1	1	Action	2006-02-15 09:46:27
2	2	Animation	2006-02-15 09:46:27
3	3	Children	2006-02-15 09:46:27
4	4	Classics	2006-02-15 09:46:27
5	5	Comedy	2006-02-15 09:46:27
6	6	Documentary	2006-02-15 09:46:27
7	7	Drama	2006-02-15 09:46:27
8	8	Family	2006-02-15 09:46:27
9	9	Foreign	2006-02-15 09:46:27
10	10	Games	2006-02-15 09:46:27

Execute (F5)

```

1 SELECT TOP 10 * FROM dbo.actor;
2 SELECT TOP 10 * FROM dbo.address;
3 SELECT TOP 10 * FROM dbo.category;
4 SELECT TOP 10 * FROM dbo.city;

```

91 %

Results Messages

	city_id	city	country_id	last_update
1	1	A Corua (La Corua)	87	2006-02-15 09:45:25
2	2	Abha	82	2006-02-15 09:45:25
3	3	Abu Dhabi	101	2006-02-15 09:45:25
4	4	Acua	60	2006-02-15 09:45:25
5	5	Adana	97	2006-02-15 09:45:25
6	6	Addis Abeba	31	2006-02-15 09:45:25
7	7	Aden	107	2006-02-15 09:45:25
8	8	Adoni	44	2006-02-15 09:45:25
9	9	Ahmadnagar	44	2006-02-15 09:45:25
10	10	Akishima	50	2006-02-15 09:45:25

SQLQuery1.sql* X

```

2 SELECT TOP 10 * FROM dbo.address;
3 SELECT TOP 10 * FROM dbo.category;
4 SELECT TOP 10 * FROM dbo.city;
5 SELECT TOP 10 * FROM dbo.country;

```

91 %

Results Messages Client Statistics

	country_id	country	last_update
1	1	Afghanistan	15/02/2006 09:44
2	2	Algeria	15/02/2006 09:44
3	3	American Samoa	15/02/2006 09:44
4	4	Angola	15/02/2006 09:44
5	5	Anguilla	15/02/2006 09:44
6	6	Argentina	15/02/2006 09:44
7	7	Armenia	15/02/2006 09:44
8	8	Australia	15/02/2006 09:44
9	9	Austria	15/02/2006 09:44
10	10	Azerbaijan	15/02/2006 09:44

SQLQuery1.sql* X

```

3 SELECT TOP 10 * FROM dbo.category;
4 SELECT TOP 10 * FROM dbo.city;
5 SELECT TOP 10 * FROM dbo.country;
6 SELECT TOP 10 * FROM dbo.customer;

```

91 %

Results Messages

	customer_id	store_id	first_name	last_name	email	address_id	activebool	create_date	last_update	active
1	524	1	Jared	Ely	jared.ely@sakilacustomer.org	530	t	2006-02-14	2013-05-26 14:49:45.738	1
2	1	1	Mary	Smith	mary.smith@sakilacustomer.org	5	t	2006-02-14	2013-05-26 14:49:45.738	1
3	2	1	Patricia	Johnson	patricia.johnson@sakilacustomer.org	6	t	2006-02-14	2013-05-26 14:49:45.738	1
4	3	1	Linda	Williams	linda.williams@sakilacustomer.org	7	t	2006-02-14	2013-05-26 14:49:45.738	1
5	4	2	Barbara	Jones	barbara.jones@sakilacustomer.org	8	t	2006-02-14	2013-05-26 14:49:45.738	1
6	5	1	Elizabeth	Brown	elizabeth.brown@sakilacustomer.org	9	t	2006-02-14	2013-05-26 14:49:45.738	1
7	6	2	Jennifer	Davis	jennifer.davis@sakilacustomer.org	10	t	2006-02-14	2013-05-26 14:49:45.738	1
8	7	1	Maria	Miller	maria.miller@sakilacustomer.org	11	t	2006-02-14	2013-05-26 14:49:45.738	1
9	8	2	Susan	Wilson	susan.wilson@sakilacustomer.org	12	t	2006-02-14	2013-05-26 14:49:45.738	1
10	9	2	Margaret	Moore	margaret.moore@sakilacustomer.org	13	t	2006-02-14	2013-05-26 14:49:45.738	1

SQLQuery1.sql* X

```

7 SELECT TOP 10 * FROM dbo.film;
8 SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
9 FROM INFORMATION_SCHEMA.COLUMNS
10 WHERE TABLE_NAME = 'film' AND TABLE_SCHEMA = 'dbo';

```

91 %

Results Messages

	film_id	title	description	releas...	la...	re...	rent...	length	replac...	rating	last_update	special_features	fulltext
1	133	Chamber Italian	A Fateful Reflection of a Moose And a Husband who	2006	1	7	4.99	117	14.99	NC-17	2013-05-...	{Trailers}	'chamber':1 'fate':4 husba
2	384	Grosse Wonderful	A Epic Drama of a Cat And a Explorer who must Rede	2006	1	5	4.99	49	19.99	R	2013-05-...	"{"Behind the Sce...	'australia':18 'cat':8 'drama
3	8	Airport Pollock	A Epic Tale of a Moose And a Girl who must Confron	2006	1	6	4.99	54	15.99	R	2013-05-...	{Trailers}	'airport':1 'ancient':18 'con
4	98	Bright Encounters	A Fateful Yam of a Lumberjack And a Feminist who	2006	1	4	4.99	73	12.99	PG-13	2013-05-...	{Trailers}	'boat':20 'bright':1 'conque
5	1	Academy Dinosaur	A Epic Drama of a Feminist And a Mad Scientist who	2006	1	6	0.99	86	20.99	PG	2013-05-...	"{"Deleted Scenes"	"{"Behind the Scenes"}";,z
6	2	Ace Goldfinger	A Astounding Epistle of a Database Administrator A	2006	1	3	4.99	48	12.99	G	2013-05-...	{Trailers	"{"Deleted Scenes"}";,ace
7	3	Adantation Holes	A Astounding Reflection of a Lumberjack And a Car	2006	1	7	2.99	50	18.99	NC-17	2013-05-...	{Trailers	"{"Deleted Scenes"}";,ar

	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
1	film_id	varchar	50
2	title	varchar	50
3	description	varchar	50
4	release_year	varchar	50
5	language_id	varchar	50
6	rental_duration	varchar	50
7	rental_rate	varchar	50
8	length	varchar	50
9	replacement_cost	varchar	50
10	rating	varchar	50
11	last_update	varchar	50
12	special_features	varchar	50
13	fulltext	varchar	50

Query executed successfully.

DESKTOP-ELTKT8 (16.0 RTM) | DESKTOP-ELTKT8\karin ... | dvdrental | 00:00:00 | 10 rows

SQLQuery1.sql* X

```

10 WHERE TABLE_NAME = 'film' AND TABLE_SCHEMA = 'dbo';
11 SELECT TOP 10 * FROM dbo.film_actor;

```

91 %

Results Messages

	actor_id	film_id	last_update
1	1	1	2006-02-15 10:05:03
2	1	23	2006-02-15 10:05:03
3	1	25	2006-02-15 10:05:03
4	1	106	2006-02-15 10:05:03
5	1	140	2006-02-15 10:05:03
6	1	166	2006-02-15 10:05:03
7	1	277	2006-02-15 10:05:03
8	1	361	2006-02-15 10:05:03
9	1	438	2006-02-15 10:05:03
10	1	499	2006-02-15 10:05:03

SQLQuery1.sql* X

```

11 SELECT TOP 10 * FROM dbo.film_actor;
12 SELECT TOP 10 * FROM dbo.film_category;

```

91 %

Results Messages

	film_id	category_id	last_update
1	1	6	2006-02-15 10:07:09
2	2	11	2006-02-15 10:07:09
3	3	6	2006-02-15 10:07:09
4	4	11	2006-02-15 10:07:09
5	5	8	2006-02-15 10:07:09
6	6	9	2006-02-15 10:07:09
7	7	5	2006-02-15 10:07:09
8	8	11	2006-02-15 10:07:09
9	9	11	2006-02-15 10:07:09
10	10	15	2006-02-15 10:07:09

SQLQuery1.sql* X

```

12 SELECT TOP 10 * FROM dbo.film_category;
13 SELECT TOP 10 * FROM dbo.inventory;

```

91 %

Results Messages

	inventory_id	film_id	store_id	last_update
1	1	1	1	2006-02-15 10:09:17
2	2	1	1	2006-02-15 10:09:17
3	3	1	1	2006-02-15 10:09:17
4	4	1	1	2006-02-15 10:09:17
5	5	1	2	2006-02-15 10:09:17
6	6	1	2	2006-02-15 10:09:17
7	7	1	2	2006-02-15 10:09:17
8	8	1	2	2006-02-15 10:09:17
9	9	2	2	2006-02-15 10:09:17
10	10	2	2	2006-02-15 10:09:17

SQLQuery1.sql* X

```

13 SELECT TOP 10 * FROM dbo.inventory;
14 SELECT TOP 10 * FROM dbo.language;

```

91 %

Results Messages

	language_id	name	last_update
1	1	English	2006-02-15 10:02:19
2	2	Italian	2006-02-15 10:02:19
3	3	Japanese	2006-02-15 10:02:19
4	4	Mandarin	2006-02-15 10:02:19
5	5	French	2006-02-15 10:02:19
6	6	German	2006-02-15 10:02:19

SQLQuery1.sql* X

```

14 SELECT TOP 10 * FROM dbo.language;
15 SELECT TOP 10 * FROM dbo.payment;

```

91 %

Results Messages

	payment_id	customer_id	staff_id	rental_id	amount	payment_date
1	17503	341	2	1520	7.99	2007-02-15 22:25:46.996577
2	17504	341	1	1778	1.99	2007-02-16 17:23:14.996577
3	17505	341	1	1849	7.99	2007-02-16 22:41:45.996577
4	17506	341	2	2829	2.99	2007-02-19 19:39:56.996577
5	17507	341	2	3130	7.99	2007-02-20 17:31:48.996577
6	17508	341	1	3382	5.99	2007-02-21 12:33:49.996577
7	17509	342	2	2190	5.99	2007-02-17 23:58:17.996577
8	17510	342	1	2914	5.99	2007-02-20 02:11:44.996577
9	17511	342	1	3081	2.99	2007-02-20 13:57:39.996577
10	17512	343	2	1547	4.99	2007-02-16 00:10:50.996577

SQLQuery1.sql* X

```

15 SELECT TOP 10 * FROM dbo.payment;
16 SELECT TOP 10 * FROM dbo.rental;

```

91 %

Results Messages

	rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update
1	10773	2005-08-01 20:53:45	3854	181	2005-08-07 00:16:45	1	2006-02-16 02:30:53
2	10774	2005-08-01 20:54:33	4231	407	2005-08-08 20:59:33	2	2006-02-16 02:30:53
3	10775	2005-08-01 20:59:52	4190	263	2005-08-04 19:31:52	2	2006-02-16 02:30:53
4	10776	2005-08-01 20:59:58	1598	565	2005-08-10 20:33:58	2	2006-02-16 02:30:53
5	10777	2005-08-01 21:03:50	3487	493	2005-08-06 19:29:50	1	2006-02-16 02:30:53
6	10778	2005-08-01 21:11:39	1939	220	2005-08-02 22:59:39	2	2006-02-16 02:30:53
7	10779	2005-08-01 21:11:54	2092	578	2005-08-09 21:00:54	2	2006-02-16 02:30:53
8	10780	2005-08-01 21:14:24	1450	51	2005-08-07 16:32:24	2	2006-02-16 02:30:53
9	10781	2005-08-01 21:22:41	1321	259	2005-08-06 01:02:41	1	2006-02-16 02:30:53
10	10782	2005-08-01 21:23:25	1507	577	2005-08-03 20:15:25	2	2006-02-16 02:30:53

```

16 SELECT TOP 10 * FROM dbo.rental;
17 SELECT TOP 10 * FROM dbo.staff;
18 SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
19 FROM INFORMATION_SCHEMA.COLUMNS
20 WHERE TABLE_NAME = 'staff'
21 AND TABLE_SCHEMA = 'dbo';

```

91 %

Results Messages

	staff_id	first_name	last_name	address_id	email	store_id	active	username	password	last_update	picture
1	1	Mike	Hillyer	3	Mike.Hillyer@sakilasta...	1	t	Mike	8cb2237d0679ca88db6464eac60da96345513964	2006-05-16 16:13:11.79328	\x89504e470d0a5a0a
2	2	Jon	Stephens	4	Jon.Stephens@sakilas...	2	t	Jon	8cb2237d0679ca88db6464eac60da96345513964	2006-05-16 16:13:11.79328	

	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
1	staff_id	varchar	50
2	first_name	varchar	50
3	last_name	varchar	50
4	address_id	varchar	50
5	email	varchar	50
6	store_id	varchar	50
7	active	varchar	50
8	username	varchar	50
9	password	varchar	50
10	last_update	varchar	50
11	picture	varchar	50

Query executed successfully.

DESKTOP-ELTKTT8 (16.0 RTM) | DESKTOP-ELTKTT8\karin ... | dvdrental | 00:00:00 | 2 rows

```

21 AND TABLE_SCHEMA = 'dbo';
22 SELECT TOP 10 * FROM dbo.store;

```

91 %

Results Messages

	store_id	manager_staff_id	address_id	last_update
1	1	1	1	2006-02-15 09:57:12
2	2	2	2	2006-02-15 09:57:12

Con estas impresiones de pantalla podemos ver que nuestra Base de Datos "dvdrental" gestiona un sistema de alquiler de películas. Incluye tablas clave como: *dbo.actor*, *dbo.film*, *dbo.customer*, *dbo.store* y *dbo.rental*, por la información que contienen podemos decir que son el centro del negocio, también podemos ver que columnas como títulos de películas, nombres de actores, datos de clientes y registros de alquileres; además también existen tablas de apoyo como: *dbo.category*, *dbo.language*, *dbo.inventory* y *dbo.payment*, que detallan categorías, idiomas, inventario disponible y transacciones.

Comprender esta estructura es esencial para formular consultas y extraer información de valor para el negocio, es por ello que se anexan para dar un contexto inicial.

Explorar datos nulos

Una vez que conocemos la estructura, es importante evaluar que no tengamos datos nulos, ya que en este documento nos enfocaremos a consultas y lo hacemos por tablas iniciando con **dbo.actor** y **continuando con las siguientes Tablas** adjunta de las capturas provenientes de esta exploración.

```
25 -- Verificar si hay valores NULOS en 'dbo.actor'
26 SELECT
27     SUM(CASE WHEN actor_id IS NULL THEN 1 ELSE 0 END) AS Nulos_actor_id,
28     SUM(CASE WHEN first_name IS NULL THEN 1 ELSE 0 END) AS Nulos_first_name,
29     SUM(CASE WHEN last_name IS NULL THEN 1 ELSE 0 END) AS Nulos_last_name,
30     SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS Nulos_last_update
31 FROM dbo.actor;
```

Results			
Messages			
Nulos_actor_id	Nulos_first_name	Nulos_last_name	Nulos_last_update
0	0	0	0

```
34 -- Verificar si hay valores NULOS en 'dbo.address'
35 SELECT COUNT(*) AS TotalFilas,
36     SUM(CASE WHEN address_id IS NULL THEN 1 ELSE 0 END) AS Nulos_address_id,
37     SUM(CASE WHEN address IS NULL THEN 1 ELSE 0 END) AS Nulos_address,
38     SUM(CASE WHEN address2 IS NULL THEN 1 ELSE 0 END) AS Nulos_address2,
39     SUM(CASE WHEN district IS NULL THEN 1 ELSE 0 END) AS Nulos_district,
40     SUM(CASE WHEN city_id IS NULL THEN 1 ELSE 0 END) AS Nulos_city_id,
41     SUM(CASE WHEN postal_code IS NULL THEN 1 ELSE 0 END) AS Nulos_postal_code,
42     SUM(CASE WHEN phone IS NULL THEN 1 ELSE 0 END) AS Nulos_phone,
43     SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS Nulos_last_update
44 FROM dbo.address;
```

Results								
Messages								
TotalFilas	Nulos_address_id	Nulos_address	Nulos_address2	Nulos_district	Nulos_city_id	Nulos_postal_code	Nulos_phone	Nulos_last_update
603	0	0	0	0	0	0	0	0

```
45
46 -- Verificar si hay valores NULOS en 'dbo.category'
47 SELECT
48     COUNT(*) AS total_rows,
49     SUM(CASE WHEN category_id IS NULL THEN 1 ELSE 0 END) AS category_id_null_count,
50     SUM(CASE WHEN name IS NULL THEN 1 ELSE 0 END) AS name_null_count,
51     SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS last_update_null_count
52 FROM dbo.category;
```

Results			
Messages			
total_rows	category_id_null_count	name_null_count	last_update_null_count
16	0	0	0

```
54 -- Verificar si hay valores NULOS en 'dbo.city'
55 SELECT
56     COUNT(*) AS total_rows,
57     SUM(CASE WHEN city_id IS NULL THEN 1 ELSE 0 END) AS city_id_null_count,
58     SUM(CASE WHEN city IS NULL THEN 1 ELSE 0 END) AS city_null_count,
59     SUM(CASE WHEN country_id IS NULL THEN 1 ELSE 0 END) AS country_id_null_count,
60     SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS last_update_null_count
61 FROM dbo.city;
```

Results				
Messages				
total_rows	city_id_null_count	city_null_count	country_id_null_count	last_update_null_count
600	0	0	0	0


```

101
102 -- Verificar si hay valores NULOS en 'dbo.film_actor'
103 SELECT COUNT(*) AS total_rows,
104        SUM(CASE WHEN actor_id IS NULL THEN 1 ELSE 0 END) AS actor_id_null_count,
105        SUM(CASE WHEN film_id IS NULL THEN 1 ELSE 0 END) AS film_id_null_count,
106        SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS last_update_null_count
107 FROM dbo.film_actor;

```

91 %

Results Messages

	total_rows	actor_id_null_count	film_id_null_count	last_update_null_count
1	5462	0	0	0

```

108
109 -- Verificar si hay valores NULOS en 'dbo.film_category'
110 SELECT COUNT(*) AS total_rows,
111        SUM(CASE WHEN film_id IS NULL THEN 1 ELSE 0 END) AS film_id_null_count,
112        SUM(CASE WHEN category_id IS NULL THEN 1 ELSE 0 END) AS category_id_null_count,
113        SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS last_update_null_count
114 FROM dbo.film_category;

```

.%

Results Messages

	total_rows	film_id_null_count	category_id_null_count	last_update_null_count
1	1000	0	0	0

```

115
116 -- Verificar si hay valores NULOS en 'dbo.inventory'
117 SELECT COUNT(*) AS total_rows,
118        SUM(CASE WHEN inventory_id IS NULL THEN 1 ELSE 0 END) AS inventory_id_null_count,
119        SUM(CASE WHEN film_id IS NULL THEN 1 ELSE 0 END) AS film_id_null_count,
120        SUM(CASE WHEN store_id IS NULL THEN 1 ELSE 0 END) AS store_id_null_count,
121        SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS last_update_null_count
122 FROM dbo.inventory;
123

```

91 %

Results Messages

	total_rows	inventory_id_null_count	film_id_null_count	store_id_null_count	last_update_null_count
1	4581	0	0	0	0

```

124
125 -- Verificar si hay valores NULOS en 'dbo.language'
126 SELECT COUNT(*) AS total_rows,
127        SUM(CASE WHEN language_id IS NULL THEN 1 ELSE 0 END) AS language_id_null_count,
128        SUM(CASE WHEN name IS NULL THEN 1 ELSE 0 END) AS name_null_count,
129        SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS last_update_null_count
129 FROM dbo.language;

```

91 %

Results Messages

	total_rows	language_id_null_count	name_null_count	last_update_null_count
1	6	0	0	0


```

130
131 -- Verificar si hay valores NULOS en 'dbo.payment'
132 SELECT COUNT(*) AS total_rows,
133     SUM(CASE WHEN payment_id IS NULL THEN 1 ELSE 0 END) AS payment_id_null_count,
134     SUM(CASE WHEN customer_id IS NULL THEN 1 ELSE 0 END) AS customer_id_null_count,
135     SUM(CASE WHEN staff_id IS NULL THEN 1 ELSE 0 END) AS staff_id_null_count,
136     SUM(CASE WHEN rental_id IS NULL THEN 1 ELSE 0 END) AS rental_id_null_count,
137     SUM(CASE WHEN amount IS NULL THEN 1 ELSE 0 END) AS amount_null_count,
138     SUM(CASE WHEN payment_date IS NULL THEN 1 ELSE 0 END) AS payment_date_null_count
139 FROM dbo.payment;

```

91 %

Results Messages

	total_rows	payment_id_null_count	customer_id_null_count	staff_id_null_count	rental_id_null_count	amount_null_count	payment_date_null_count
1	14596	0	0	0	0	0	0

```

140
141 -- Verificar si hay valores NULOS en 'dbo.rental'
142 SELECT COUNT(*) AS total_rows,
143     SUM(CASE WHEN rental_id IS NULL THEN 1 ELSE 0 END) AS rental_id_null_count,
144     SUM(CASE WHEN rental_date IS NULL THEN 1 ELSE 0 END) AS rental_date_null_count,
145     SUM(CASE WHEN inventory_id IS NULL THEN 1 ELSE 0 END) AS inventory_id_null_count,
146     SUM(CASE WHEN customer_id IS NULL THEN 1 ELSE 0 END) AS customer_id_null_count,
147     SUM(CASE WHEN return_date IS NULL THEN 1 ELSE 0 END) AS return_date_null_count,
148     SUM(CASE WHEN staff_id IS NULL THEN 1 ELSE 0 END) AS staff_id_null_count,
149     SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS last_update_null_count
150 FROM dbo.rental;
151

```

91 %

Results Messages

	total_rows	rental_id_null_count	rental_date_null_count	inventory_id_null_count	customer_id_null_count	return_date_null_count	staff_id_null_count	last_update_null_count
1	16044	0	0	0	0	0	0	0

```

152 -- Verificar si hay valores NULOS en 'dbo.staff'
153 SELECT COUNT(*) AS total_rows,
154     SUM(CASE WHEN staff_id IS NULL THEN 1 ELSE 0 END) AS staff_id_null_count,
155     SUM(CASE WHEN first_name IS NULL THEN 1 ELSE 0 END) AS first_name_null_count,
156     SUM(CASE WHEN last_name IS NULL THEN 1 ELSE 0 END) AS last_name_null_count,
157     SUM(CASE WHEN address_id IS NULL THEN 1 ELSE 0 END) AS address_id_null_count,
158     SUM(CASE WHEN email IS NULL THEN 1 ELSE 0 END) AS email_null_count,
159     SUM(CASE WHEN store_id IS NULL THEN 1 ELSE 0 END) AS store_id_null_count,
160     SUM(CASE WHEN active IS NULL THEN 1 ELSE 0 END) AS active_null_count,
161     SUM(CASE WHEN username IS NULL THEN 1 ELSE 0 END) AS username_null_count,
162     SUM(CASE WHEN password IS NULL THEN 1 ELSE 0 END) AS password_null_count,
163     SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS last_update_null_count,
164     SUM(CASE WHEN picture IS NULL THEN 1 ELSE 0 END) AS picture_null_count
165 FROM dbo.staff;

```

91 %

Results Messages

	total_rows	staff_id...	first_...	last_...	addres...	email...	store_j...	active_...	usem...	pass...	last_u...	picture_null_count
1	2	0	0	0	0	0	0	0	0	0	0	0

```

166
167 -- Verificar si hay valores NULOS en 'dbo.store'
168 SELECT COUNT(*) AS total_rows,
169     SUM(CASE WHEN store_id IS NULL THEN 1 ELSE 0 END) AS store_id_null_count,
170     SUM(CASE WHEN manager_staff_id IS NULL THEN 1 ELSE 0 END) AS manager_staff_id_null_count,
171     SUM(CASE WHEN address_id IS NULL THEN 1 ELSE 0 END) AS address_id_null_count,
172     SUM(CASE WHEN last_update IS NULL THEN 1 ELSE 0 END) AS last_update_null_count
173 FROM dbo.store;

```

%

Results Messages

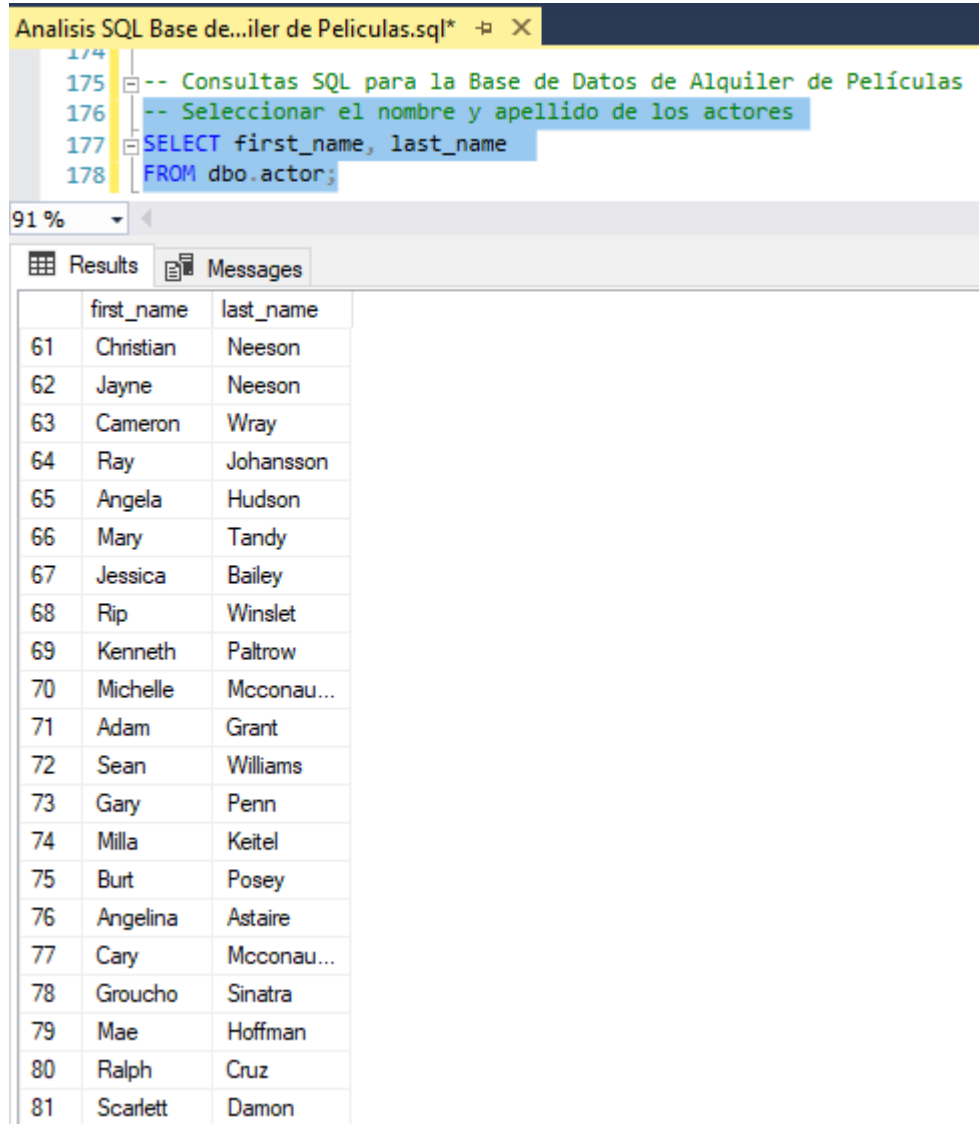
	total_rows	store_id_null_count	manager_staff_id_null_count	address_id_null_count	last_update_null_count
1	2	0	0	0	0

Las imágenes anteriores nos indican el número de registros individuales por tablas, en todas las tablas podemos ver que tenemos cero valores nulos en todas las columnas. Esto demuestra una excelente calidad y completitud de datos, sugiriendo la aplicación de fuertes restricciones de integridad. No se requiere manejo de datos faltantes para estas columnas, gracias a la impecable integridad de datos, las consultas basadas en estas tablas serán altamente fiables.

Consultas

Se realizan 15 consultas de la base de datos de alquiler de películas, llamada "dvdrental" las cuales se exponen a continuación.

1. Vamos a seleccionar el nombre y apellido de los actores



The screenshot shows a SQL query window titled "Analisis SQL Base de...iler de Peliculas.sql". The query is as follows:

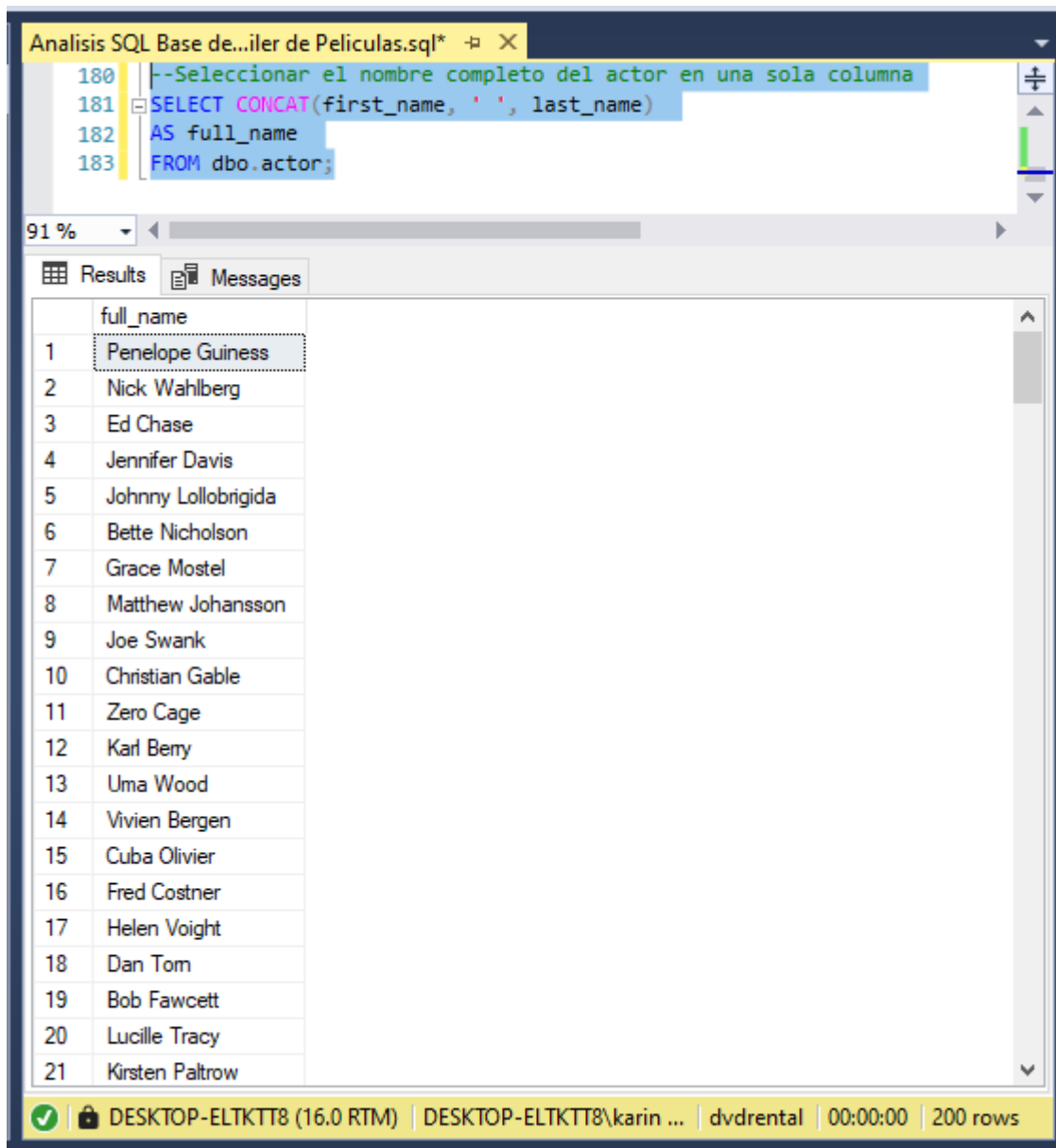
```
-- Consultas SQL para la Base de Datos de Alquiler de Películas
-- Seleccionar el nombre y apellido de los actores
SELECT first_name, last_name
FROM dbo.actor;
```

Below the query window, the "Results" tab is active, displaying a table with 2 columns: "first_name" and "last_name". The table contains 21 rows of actor names, numbered 61 to 81 in the first column.

	first_name	last_name
61	Christian	Neeson
62	Jayne	Neeson
63	Cameron	Wray
64	Ray	Johansson
65	Angela	Hudson
66	Mary	Tandy
67	Jessica	Bailey
68	Rip	Winslet
69	Kenneth	Paltrow
70	Michelle	Mcconau...
71	Adam	Grant
72	Sean	Williams
73	Gary	Penn
74	Milla	Keitel
75	Burt	Posey
76	Angelina	Astaire
77	Cary	Mcconau...
78	Groucho	Sinatra
79	Mae	Hoffman
80	Ralph	Cruz
81	Scarlett	Damon

Esta consulta simplemente recupera la lista de nombres y apellidos de todos los actores en la base de datos, y nos permite ver quiénes son los actores registrados en el sistema.

2. Vamos a seleccionar el nombre completo del actor en una sola columna



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results grid. The query editor contains the following SQL code:

```
--Seleccionar el nombre completo del actor en una sola columna
SELECT CONCAT(first_name, ' ', last_name)
AS full_name
FROM dbo.actor;
```

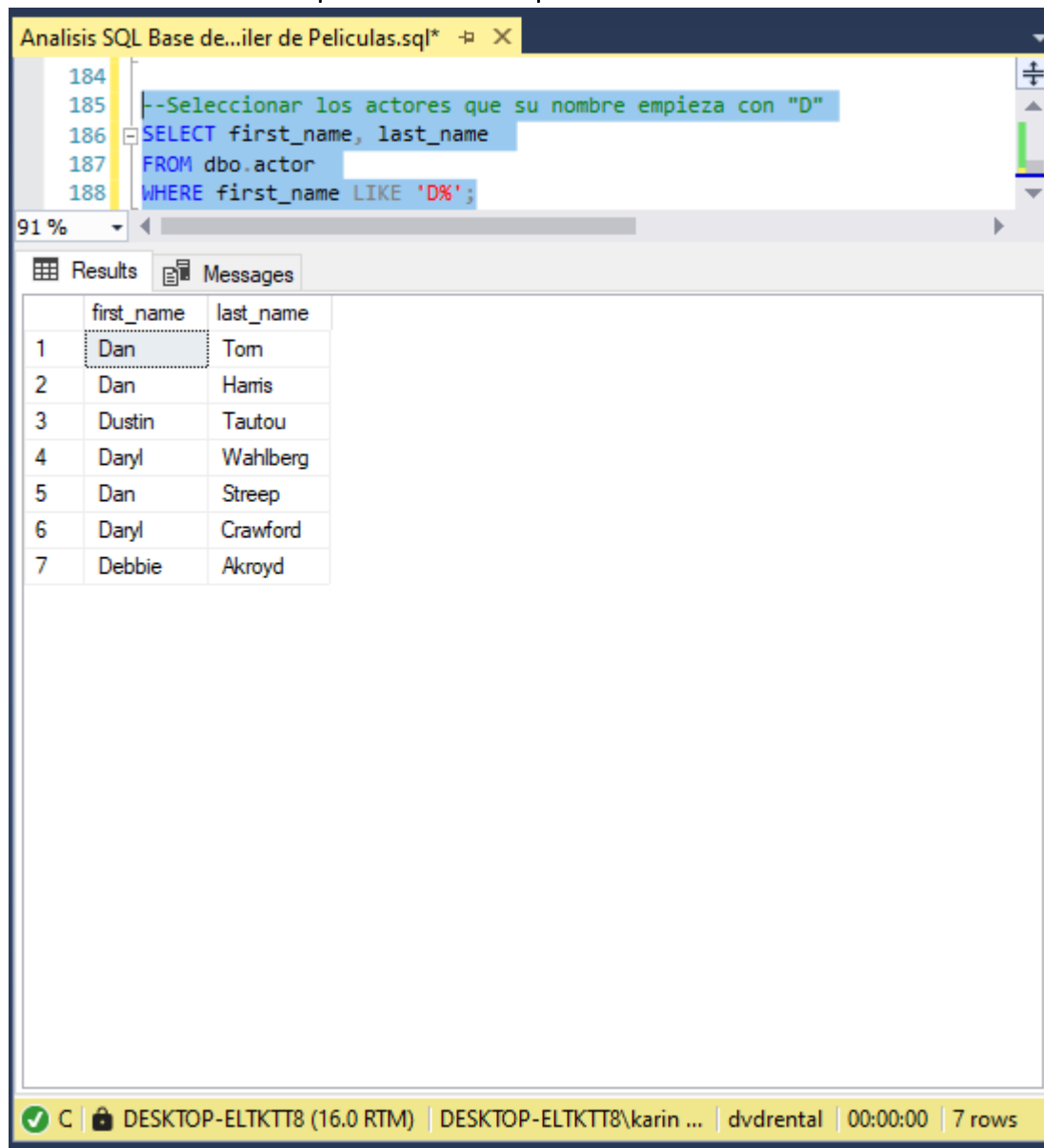
The results grid displays the output of the query, showing a single column named 'full_name' with 21 rows of actor names. The first row is highlighted.

	full_name
1	Penelope Guinness
2	Nick Wahlberg
3	Ed Chase
4	Jennifer Davis
5	Johnny Lollobrigida
6	Bette Nicholson
7	Grace Mostel
8	Matthew Johansson
9	Joe Swank
10	Christian Gable
11	Zero Cage
12	Karl Berry
13	Uma Wood
14	Vivien Bergen
15	Cuba Olivier
16	Fred Costner
17	Helen Voight
18	Dan Tom
19	Bob Fawcett
20	Lucille Tracy
21	Kirsten Paltrow

The status bar at the bottom indicates the connection is successful, showing the server name 'DESKTOP-ELTKT8 (16.0 RTM)', the user 'DESKTOP-ELTKT8\karin ...', the database 'dvdrental', the execution time '00:00:00', and the number of rows '200 rows'.

Se combinan el nombre y el apellido de los actores en una única columna llamada "full_name". Esto nos facilita la visualización de los nombres completos de los actores, lo que puede ser útil para listados o informes.

3. Selecciona los actores que su nombre empieza con "D"



The screenshot shows a SQL Server Enterprise Manager window titled "Analisis SQL Base de...iler de Peliculas.sql*". The query editor displays the following SQL code:

```
--Seleccionar los actores que su nombre empieza con "D"
SELECT first_name, last_name
FROM dbo.actor
WHERE first_name LIKE 'D%';
```

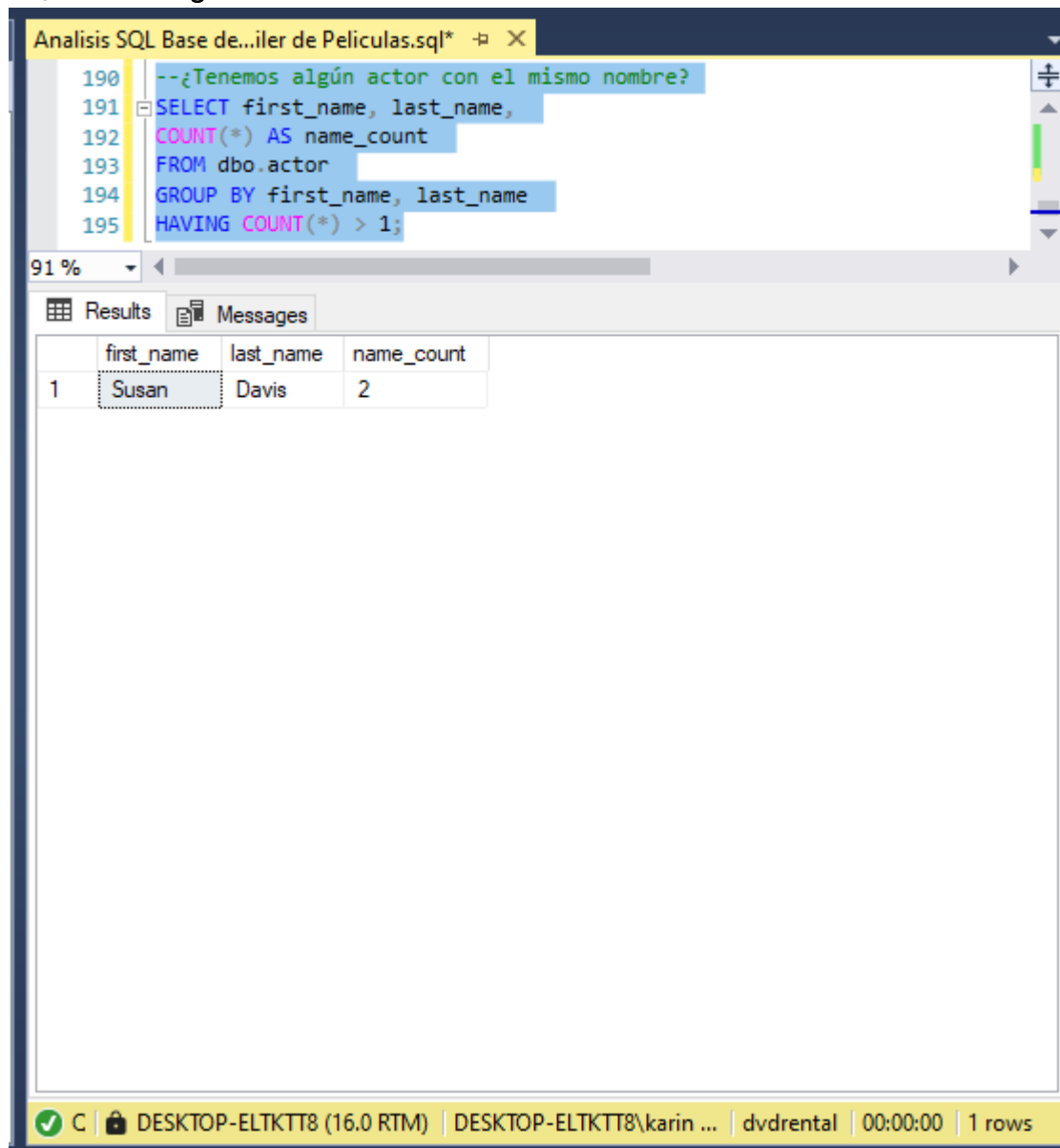
The query results are displayed in a table with two columns: first_name and last_name. The results are as follows:

	first_name	last_name
1	Dan	Tom
2	Dan	Harris
3	Dustin	Tautou
4	Daryl	Wahlberg
5	Dan	Streep
6	Daryl	Crawford
7	Debbie	Akroyd

The status bar at the bottom indicates the connection is successful (green checkmark), the server is DESKTOP-ELTKTT8 (16.0 RTM), the user is DESKTOP-ELTKTT8\karin ..., the database is dvdrental, the execution time is 00:00:00, and 7 rows were returned.

Esta consulta filtra los actores cuyo nombre comienza con la letra "D", y nos permite encontrar rápidamente actores con un patrón específico en sus nombres, útil para búsquedas o segmentación.

4. ¿Tenemos algún actor con el mismo nombre?



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains the following SQL code:

```
--¿Tenemos algún actor con el mismo nombre?  
SELECT first_name, last_name,  
COUNT(*) AS name_count  
FROM dbo.actor  
GROUP BY first_name, last_name  
HAVING COUNT(*) > 1;
```

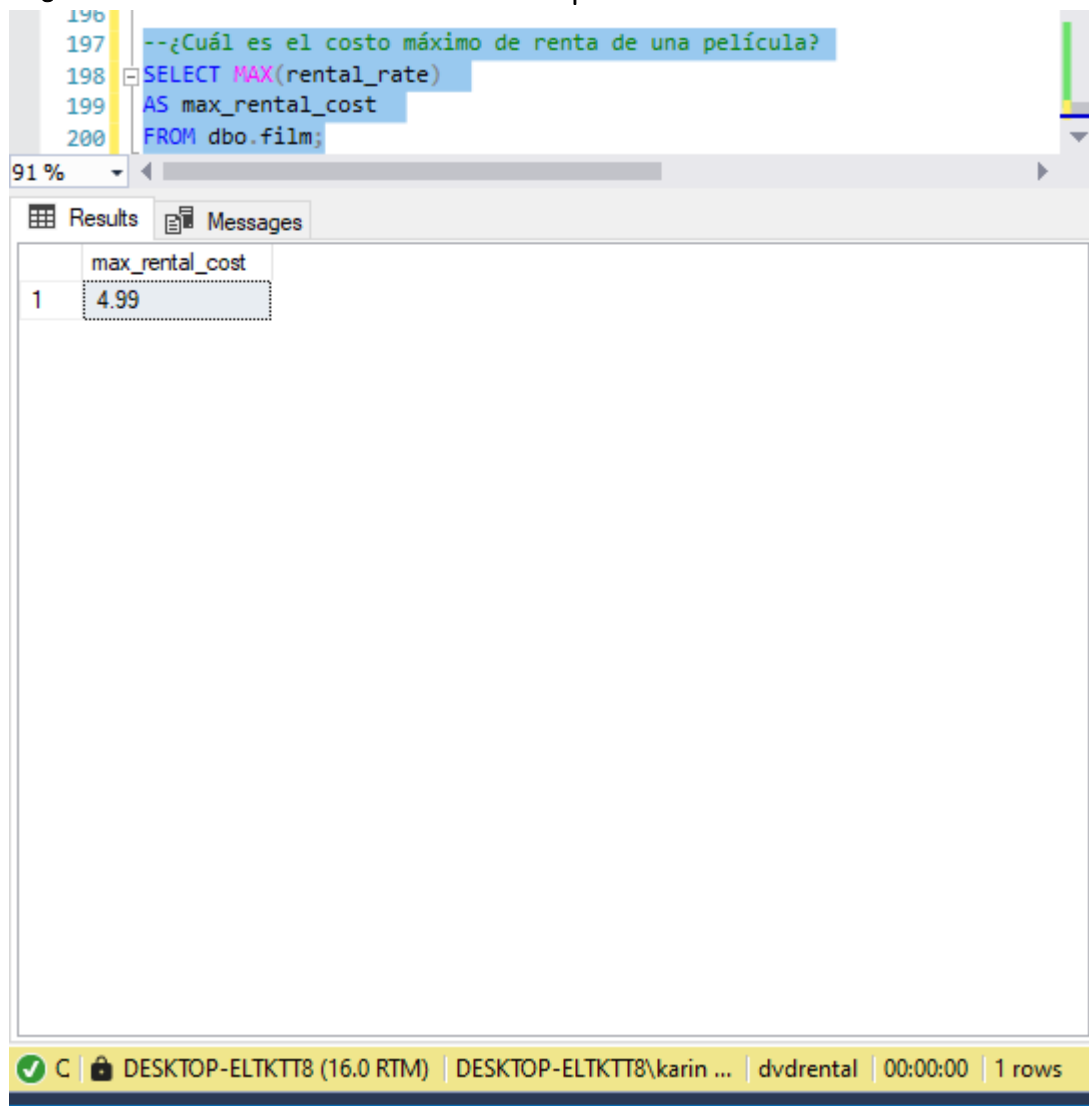
The results pane shows a single row of data:

	first_name	last_name	name_count
1	Susan	Davis	2

The status bar at the bottom indicates the connection is successful, the server is DESKTOP-ELTKT8 (16.0 RTM), the user is DESKTOP-ELTKT8\karin ..., the database is dvdrental, the execution time is 00:00:00, and there is 1 row returned.

Esta consulta identifica si hay actores con el mismo nombre y apellido, lo cual nos revela si existen duplicados en los nombres de los actores, esto es importante para la integridad de los datos. En este caso, "Susan Davis" aparece 2 veces.

5. ¿Cuál es el costo máximo de renta de una película?



The screenshot shows a SQL Server Enterprise Manager window. The top pane displays a query in the SQL Server Enterprise Manager query editor:

```
196  
197 --¿Cuál es el costo máximo de renta de una película?  
198 SELECT MAX(rental_rate)  
199 AS max_rental_cost  
200 FROM dbo.film;
```

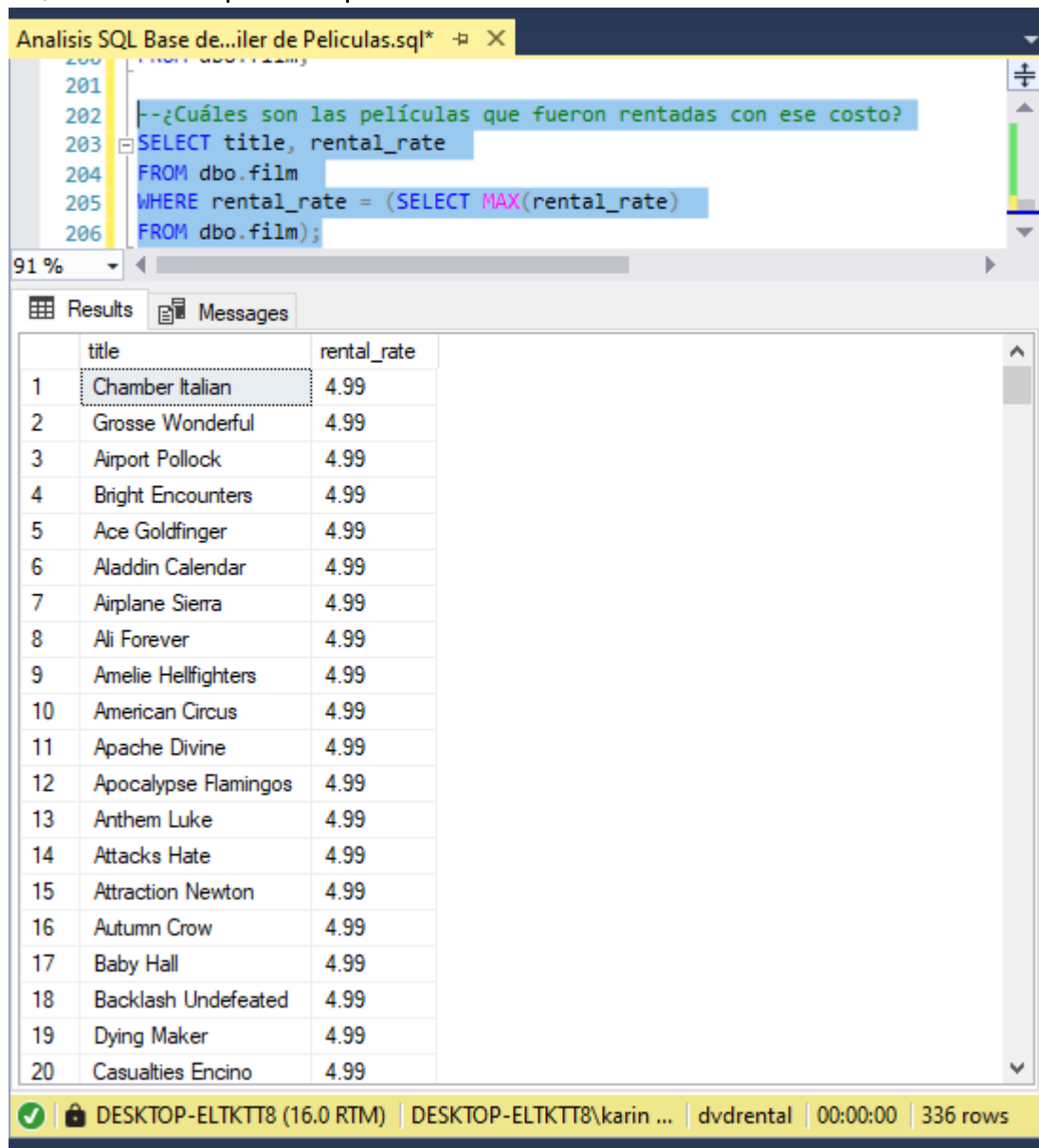
The bottom pane shows the results of the query. The 'Results' tab is active, displaying a table with one row and one column:

	max_rental_cost
1	4.99

The status bar at the bottom indicates: 91 % zoom, Results tab, Messages tab, 1 row(s), 00:00:00 execution time, and 1 row(s) returned.

Determina el precio más alto de alquiler de una película en el catálogo, el costo máximo de renta es 4.99. Esto nos da una idea del rango de precios.

6. ¿Cuáles son las películas que fueron rentadas con ese costo?



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains the following SQL code:

```
--¿Cuáles son las películas que fueron rentadas con ese costo?
SELECT title, rental_rate
FROM dbo.film
WHERE rental_rate = (SELECT MAX(rental_rate)
FROM dbo.film);
```

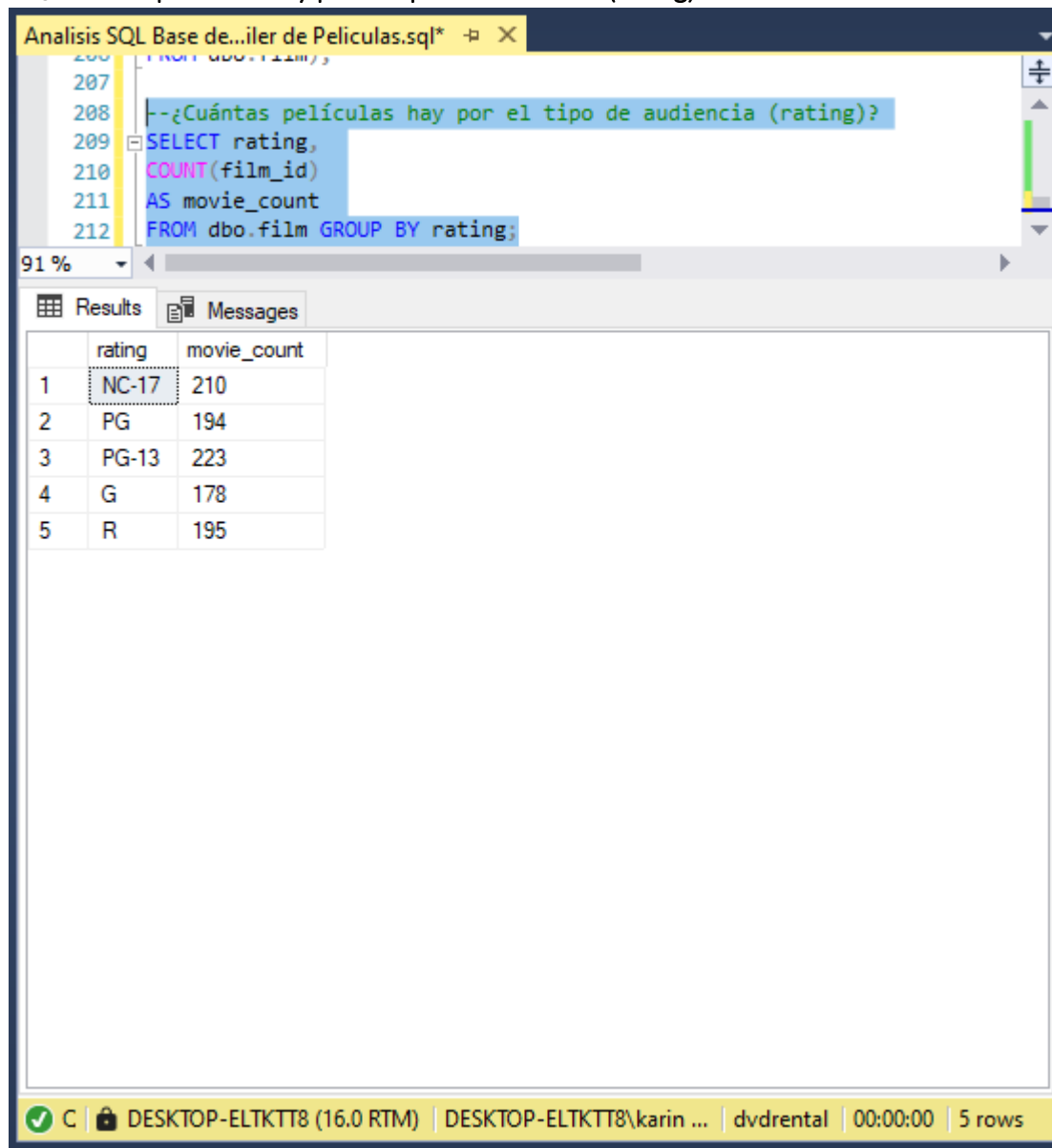
The results pane displays a table with two columns: 'title' and 'rental_rate'. The table contains 20 rows of data, all with a rental rate of 4.99. The first row is highlighted.

	title	rental_rate
1	Chamber Italian	4.99
2	Grosse Wonderful	4.99
3	Airport Pollock	4.99
4	Bright Encounters	4.99
5	Ace Goldfinger	4.99
6	Aladdin Calendar	4.99
7	Airplane Sierra	4.99
8	Ali Forever	4.99
9	Amelie Hellfighters	4.99
10	American Circus	4.99
11	Apache Divine	4.99
12	Apocalypse Flamingos	4.99
13	Anthem Luke	4.99
14	Attacks Hate	4.99
15	Attraction Newton	4.99
16	Autumn Crow	4.99
17	Baby Hall	4.99
18	Backlash Undeclared	4.99
19	Dying Maker	4.99
20	Casualties Encino	4.99

The status bar at the bottom indicates the connection is successful (green checkmark), the server is DESKTOP-ELTKTT8 (16.0 RTM), the user is DESKTOP-ELTKTT8\karin ..., the database is dvdrental, the query took 00:00:00, and there are 336 rows in the table.

Muestra los títulos de todas las películas que tienen el costo máximo de alquiler encontrado en la consulta anterior y nos permite identificar qué películas son las más "caras" de alquilar, lo que podría ser interesante para estrategias de precios o promociones.

7. ¿Cuántas películas hay por el tipo de audiencia (rating)?



The screenshot shows a SQL Server Enterprise Manager window with a query executed in the 'Analisis SQL Base de...iler de Peliculas.sql*' query editor. The query is as follows:

```
--¿Cuántas películas hay por el tipo de audiencia (rating)?  
SELECT rating,  
COUNT(film_id)  
AS movie_count  
FROM dbo.film GROUP BY rating;
```

The results are displayed in the 'Results' tab, showing a table with 5 rows and 2 columns: 'rating' and 'movie_count'.

	rating	movie_count
1	NC-17	210
2	PG	194
3	PG-13	223
4	G	178
5	R	195

The status bar at the bottom indicates the connection is successful (green checkmark), the server is 'DESKTOP-ELTKTT8 (16.0 RTM)', the user is 'DESKTOP-ELTKTT8\karin ...', the database is 'dvdrental', the execution time is '00:00:00', and there are '5 rows'.

Agrupar las películas por su clasificación de audiencia (rating) y cuenta cuántas películas hay en cada categoría, nos da un desglose de la distribución de películas según su clasificación, como "NC-17", "PG", "PG-13", "G" y "R", lo que es útil para entender el catálogo y las preferencias de audiencia.

8. Selecciona las películas que no tienen un rating R o NC-17

Analisis SQL Base de...iler de Peliculas.sql* X

```
212 FROM dbo.film GROUP BY rating;
213
214 --Selecciona las películas que no tienen un rating R o NC-17
215 SELECT title, rating
216 FROM dbo.film
217 WHERE rating NOT IN ('R', 'NC-17');
```

91 %

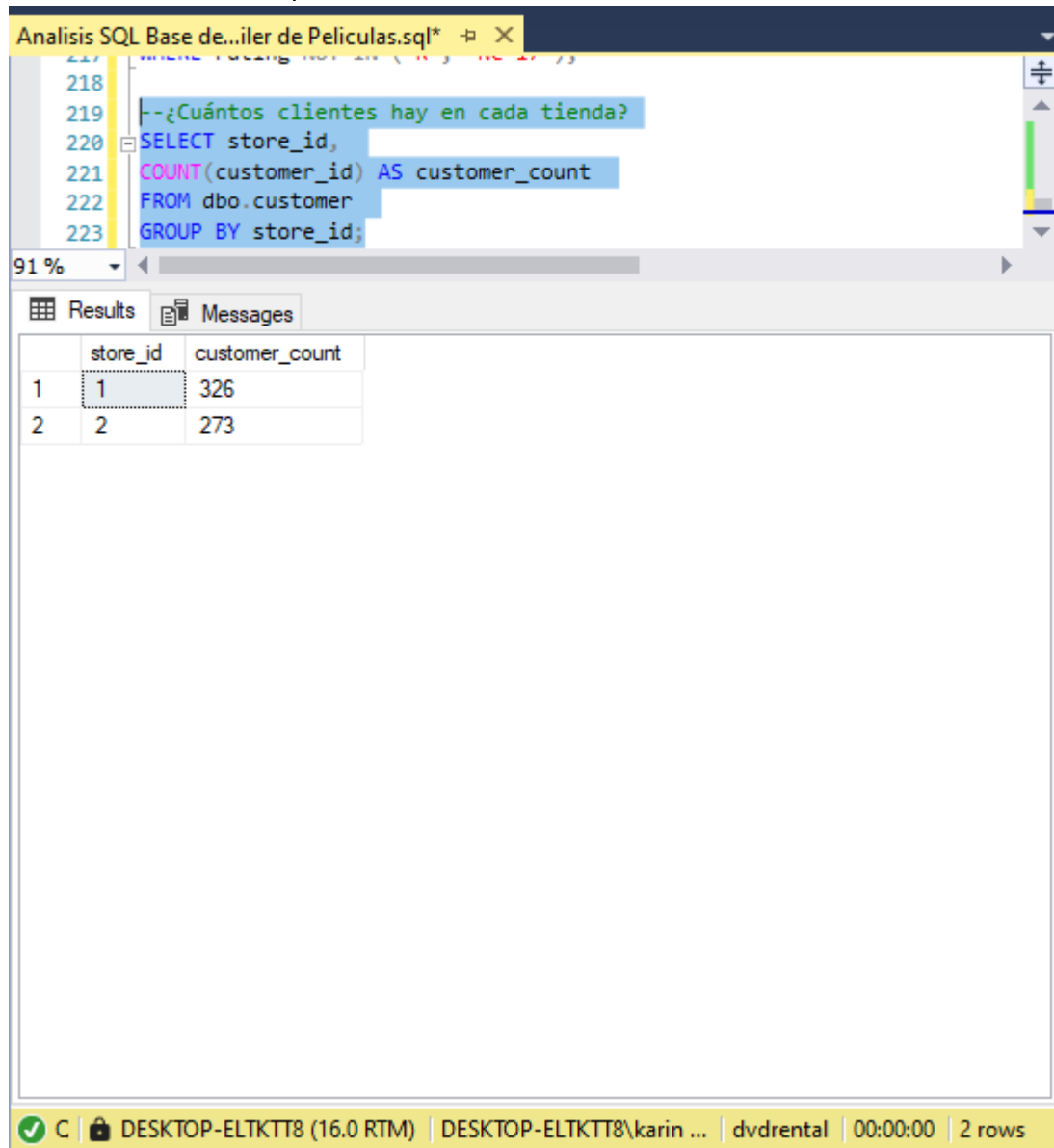
Results Messages

	title	rating
1	Bright Encounters	PG-13
2	Academy Dinosaur	PG
3	Ace Goldfinger	G
4	Affair Prejudice	G
5	African Egg	G
6	Agent Truman	PG
7	Airplane Sierra	PG-13
8	Alabama Devil	PG-13
9	Alamo Videotape	G
10	Alaska Phantom	PG
11	Ali Forever	PG
12	Alter Victory	PG-13
13	Amadeus Holy	PG
14	Amistad Midsummer	G
15	Angels Life	G
16	Annie Identity	G
17	Anthem Luke	PG-13
18	Apollo Teen	PG-13
19	Arachnophobia Rollercoaster	PG-13
20	Argonauts Town	PG-13

DESKTOP-ELTKTT8 (16.0 RTM) | DESKTOP-ELTKTT8\karin ... | dvdrental | 00:00:00 | 595 rows

Filtra las películas para mostrar solo aquellas que no están clasificadas como "R" o "NC-17", esto podría ser útil para identificar películas aptas para audiencias más amplias, excluyendo contenido restringido.

9. ¿Cuántos clientes hay en cada tienda?



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains the following SQL code:

```
--¿Cuántos clientes hay en cada tienda?  
SELECT store_id,  
COUNT(customer_id) AS customer_count  
FROM dbo.customer  
GROUP BY store_id;
```

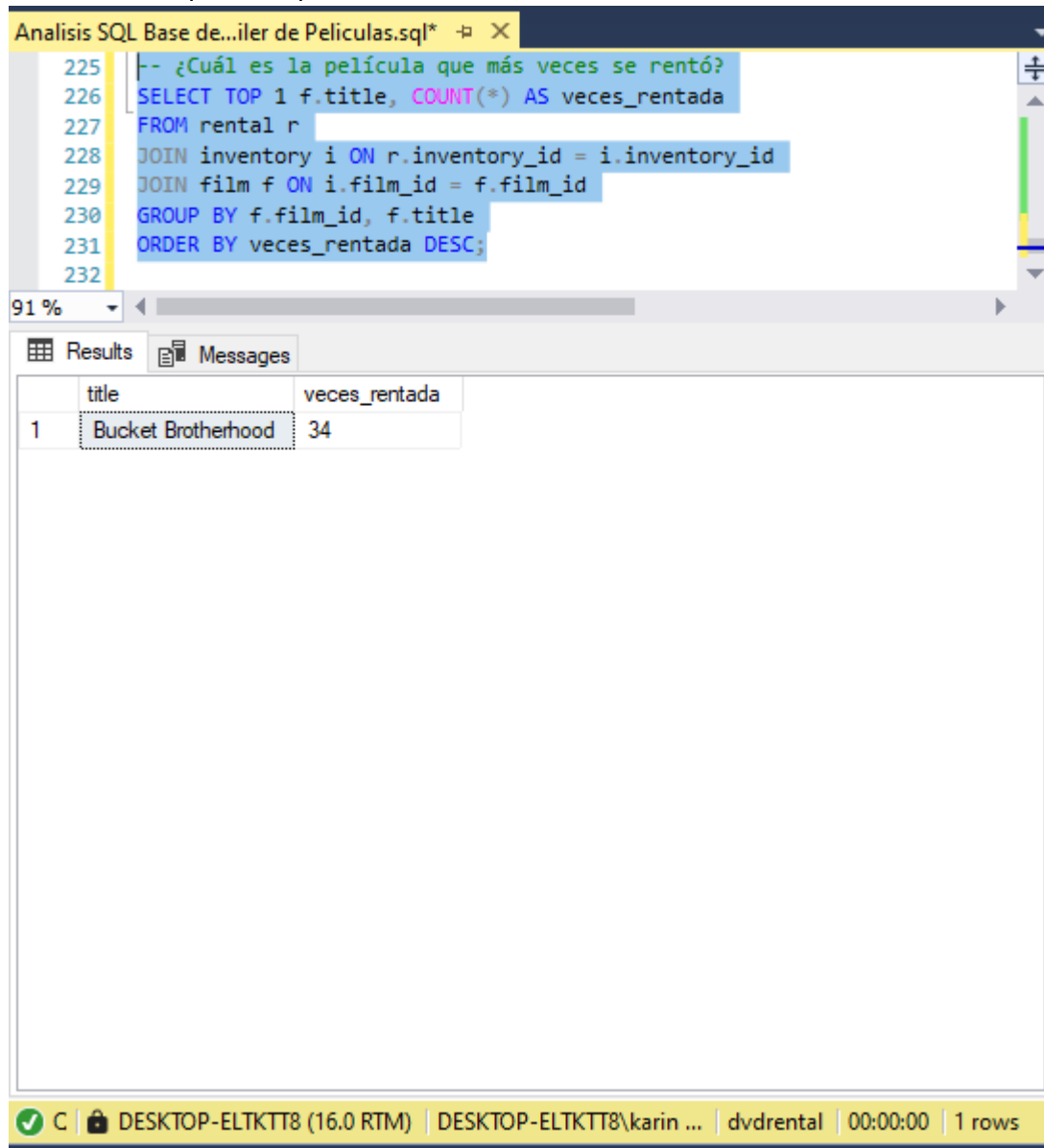
The results pane displays the following data:

	store_id	customer_count
1	1	326
2	2	273

The status bar at the bottom indicates that the query was executed successfully, showing 2 rows and a duration of 00:00:00.

Cuenta la cantidad de clientes asociados a cada tienda y nos indica la distribución de clientes entre las dos tiendas existentes: la tienda 1 tiene 326 clientes y la tienda 2 tiene 273 clientes, es relevante para la gestión de recursos o marketing por tienda.

10. ¿Cuál es la película que más veces se rentó?



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window titled 'Analisis SQL Base de...iler de Peliculas.sql'. The query is as follows:

```
225 -- ¿Cuál es la película que más veces se rentó?
226 SELECT TOP 1 f.title, COUNT(*) AS veces_rentada
227 FROM rental r
228 JOIN inventory i ON r.inventory_id = i.inventory_id
229 JOIN film f ON i.film_id = f.film_id
230 GROUP BY f.film_id, f.title
231 ORDER BY veces_rentada DESC;
232
```

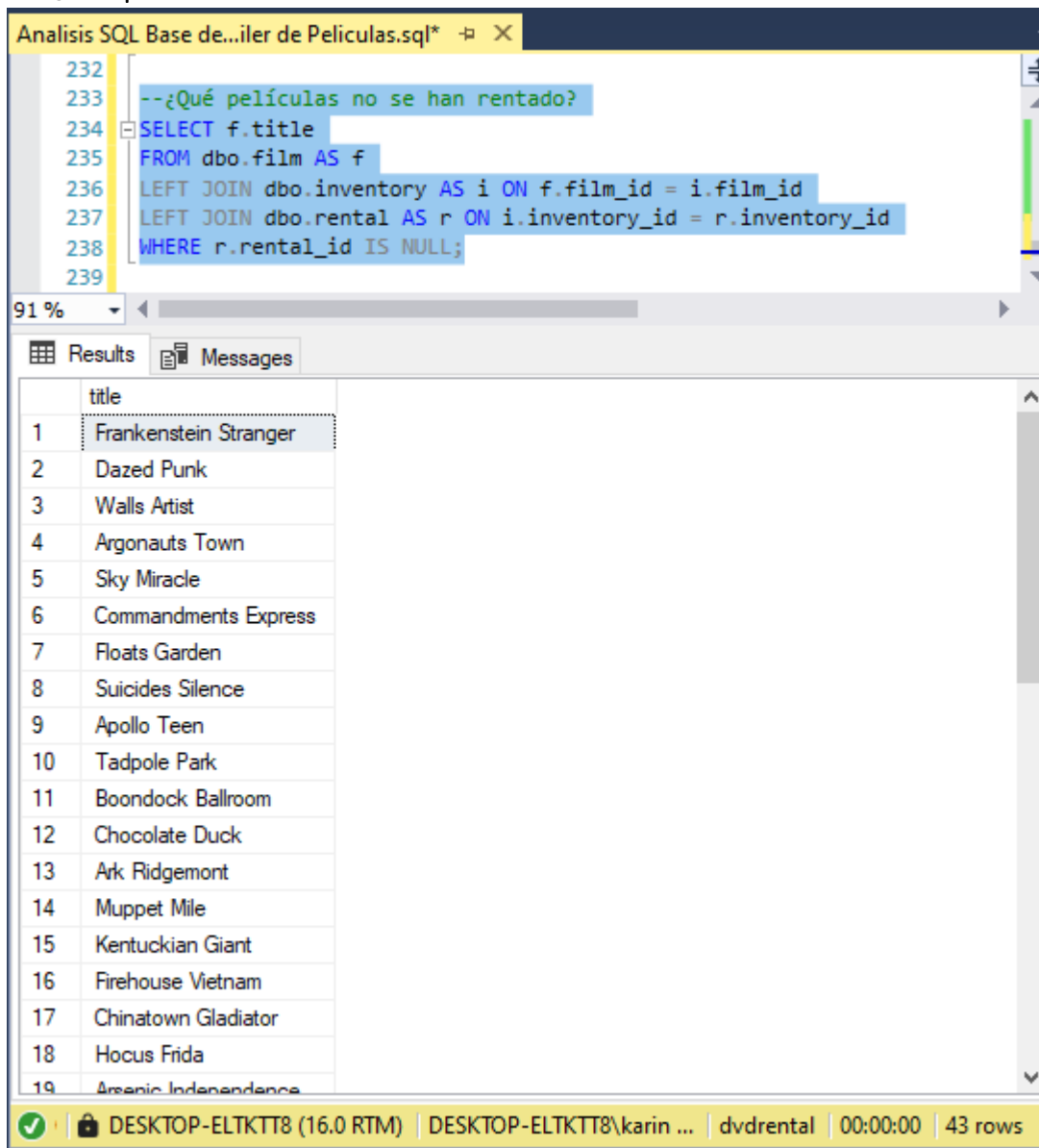
The bottom pane shows the 'Results' tab with a single row of data:

	title	veces_rentada
1	Bucket Brotherhood	34

The status bar at the bottom indicates the connection is to 'DESKTOP-ELTKTT8 (16.0 RTM)' and the query returned 1 row.

Identifica la película que ha sido alquilada la mayor cantidad de veces, que es "Bucket Brotherhood", con 34 alquileres. Esta información es clave para conocer la popularidad de las películas.

11. ¿Qué películas no se han rentado?



The screenshot shows a SQL query window with the following text:

```
--¿Qué películas no se han rentado?  
SELECT f.title  
FROM dbo.film AS f  
LEFT JOIN dbo.inventory AS i ON f.film_id = i.film_id  
LEFT JOIN dbo.rental AS r ON i.inventory_id = r.inventory_id  
WHERE r.rental_id IS NULL;
```

Below the query window, the 'Results' tab is active, displaying a table with 19 rows of film titles. The status bar at the bottom indicates 43 rows in total.

	title
1	Frankenstein Stranger
2	Dazed Punk
3	Walls Artist
4	Argonauts Town
5	Sky Miracle
6	Commandments Express
7	Floats Garden
8	Suicides Silence
9	Apollo Teen
10	Tadpole Park
11	Boondock Ballroom
12	Chocolate Duck
13	Ark Ridgemont
14	Muppet Mile
15	Kentuckian Giant
16	Firehouse Vietnam
17	Chinatown Gladiator
18	Hocus Frida
19	America Independence

DESKTOP-ELTKT8 (16.0 RTM) | DESKTOP-ELTKT8\karin ... | dvdrental | 00:00:00 | 43 rows

Muestra las películas que nunca han sido alquiladas, y vemos que hay 43 películas que no se han rentado, lo cual es importante para identificar contenido poco popular o que requiere promoción.

12. ¿Qué clientes no han rentado ninguna película?

The screenshot shows the SQL Developer interface with a query window titled "Análisis SQL Base de...iler de Peliculas.sql". The query is as follows:

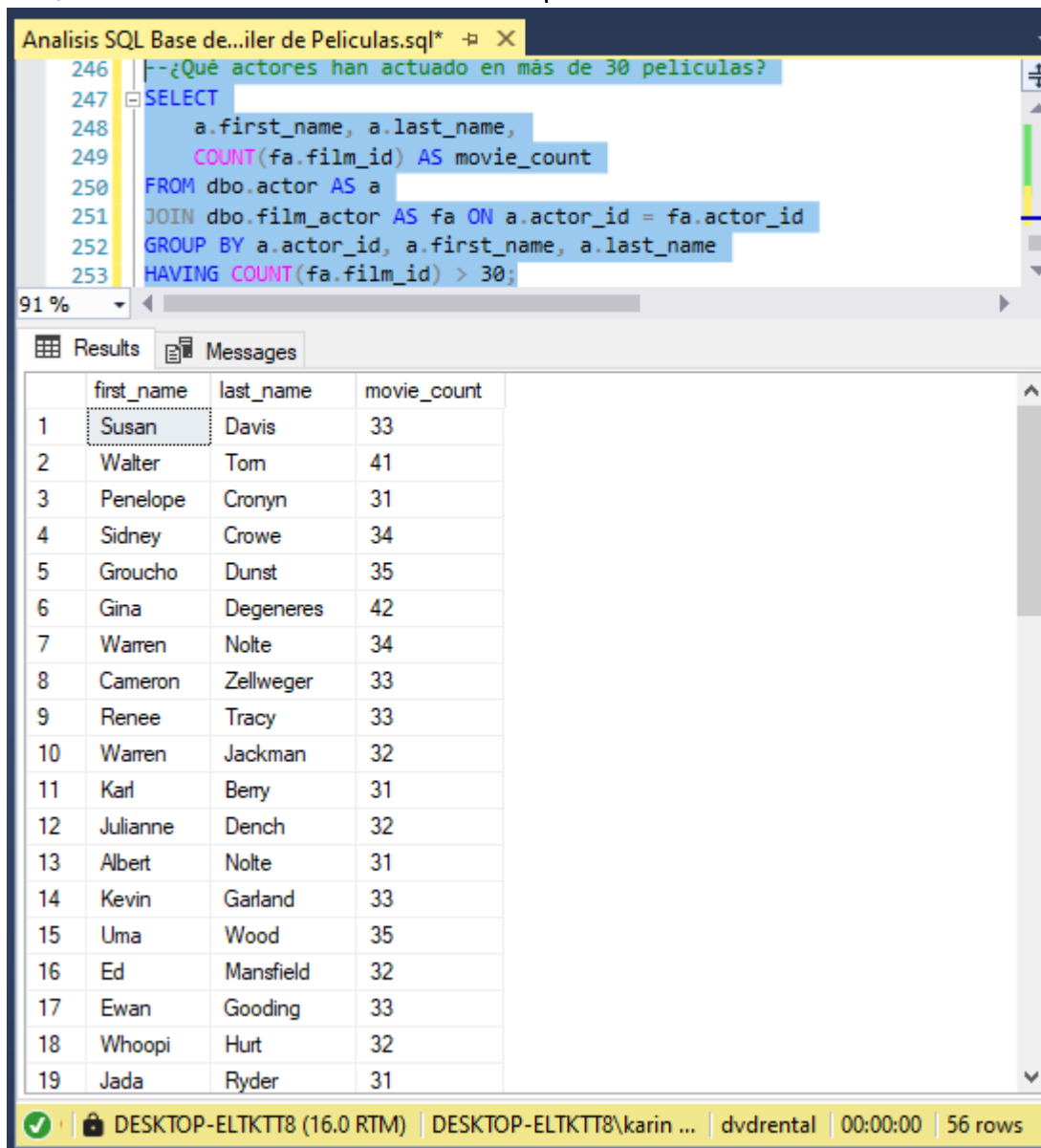
```
240 --¿Qué clientes no han rentado ninguna película?
241 SELECT c.customer_id, c.first_name, c.last_name
242 FROM customer c
243 LEFT JOIN rental r ON c.customer_id = r.customer_id
244 WHERE r.rental_id IS NULL;
245
```

The query is executed, and the Results tab is active. The result set is empty, showing only the column headers: customer_id, first_name, and last_name. The status bar at the bottom indicates "0 rows".

customer_id	first_name	last_name
-------------	------------	-----------

Busca clientes que no tienen ningún registro de alquiler, pero no se encontraron clientes que no hayan rentado ninguna película.

13. ¿Qué actores han actuado en más de 30 películas?



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window titled 'Analisis SQL Base de...iler de Peliculas.sql*'. The query is as follows:

```
246 --¿Qué actores han actuado en más de 30 películas?
247 SELECT
248     a.first_name, a.last_name,
249     COUNT(fa.film_id) AS movie_count
250 FROM dbo.actor AS a
251 JOIN dbo.film_actor AS fa ON a.actor_id = fa.actor_id
252 GROUP BY a.actor_id, a.first_name, a.last_name
253 HAVING COUNT(fa.film_id) > 30;
```

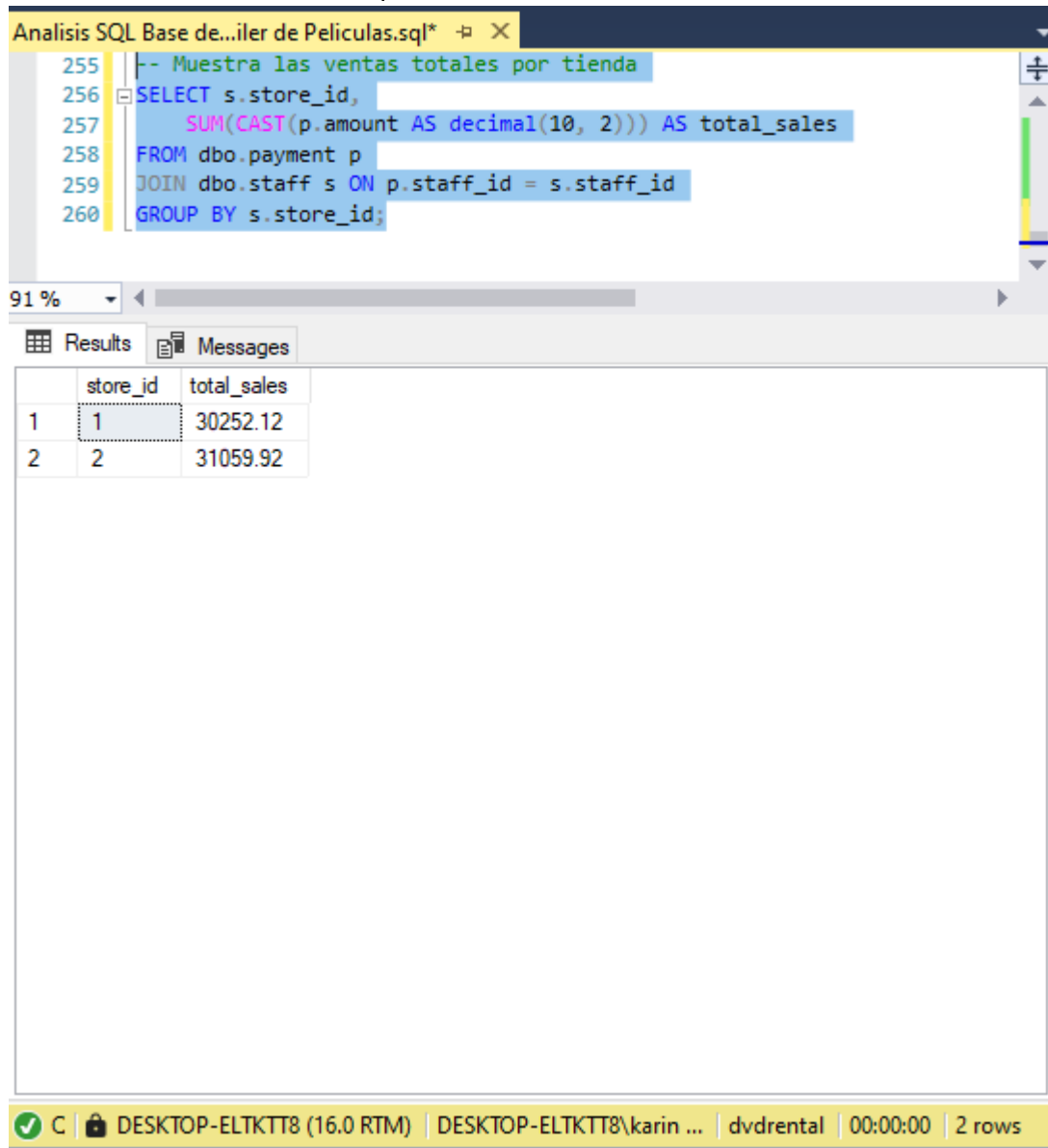
The bottom pane shows the 'Results' tab with a table containing 19 rows of data. The table has four columns: 'first_name', 'last_name', and 'movie_count'. The first row is highlighted.

	first_name	last_name	movie_count
1	Susan	Davis	33
2	Walter	Tom	41
3	Penelope	Cronyn	31
4	Sidney	Crowe	34
5	Groucho	Dunst	35
6	Gina	Degeneres	42
7	Warren	Nolte	34
8	Cameron	Zellweger	33
9	Renee	Tracy	33
10	Warren	Jackman	32
11	Karl	Berry	31
12	Julianne	Dench	32
13	Albert	Nolte	31
14	Kevin	Garland	33
15	Uma	Wood	35
16	Ed	Mansfield	32
17	Ewan	Gooding	33
18	Whoopi	Hurt	32
19	Jada	Ryder	31

The status bar at the bottom indicates that the query executed successfully, returning 56 rows.

Lista los actores que han participado en más de 30 películas, y los resultados arrojan a 18 actores que han actuado en más de 30 películas, resaltando a los actores más prolíficos en el catálogo.

14. Muestra las ventas totales por tienda



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains the following SQL code:

```
-- Muestra las ventas totales por tienda
SELECT s.store_id,
       SUM(CAST(p.amount AS decimal(10, 2))) AS total_sales
FROM   dbo.payment p
JOIN   dbo.staff s ON p.staff_id = s.staff_id
GROUP BY s.store_id;
```

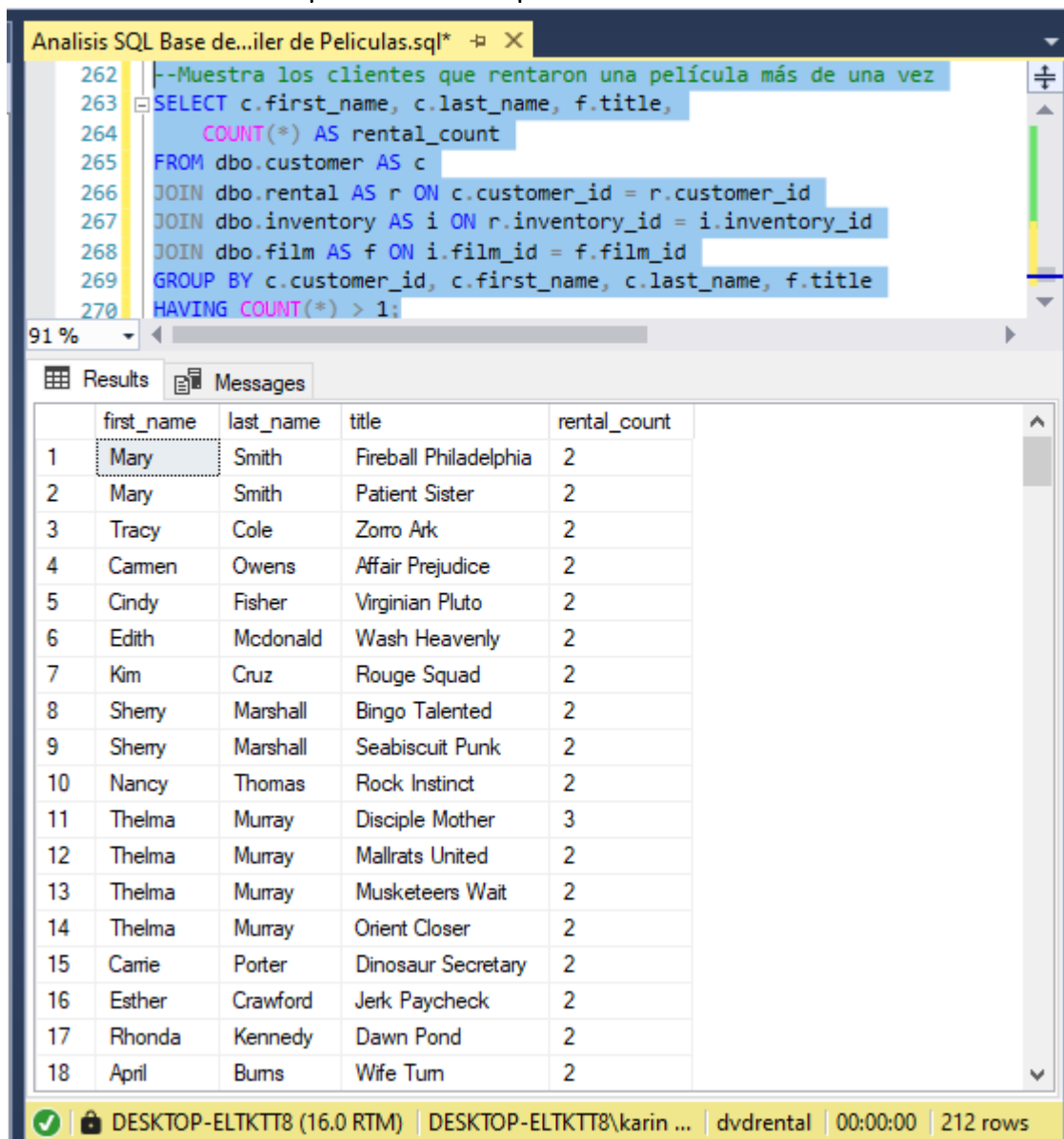
The results pane displays the following data:

	store_id	total_sales
1	1	30252.12
2	2	31059.92

The status bar at the bottom indicates that the query executed successfully, returning 2 rows in 00:00:00 seconds.

Calcula la suma total de los pagos recibidos por cada tienda, vemos que la tienda 1 ha generado 30252.12 en ventas, y la tienda 2 ha generado 31059.92, nos proporciona una visión clara del rendimiento financiero de cada sucursal.

15. Muestra los clientes que rentaron una película más de una vez



The screenshot shows a SQL query window titled "Analisis SQL Base de...iler de Peliculas.sql". The query is as follows:

```
262 --Muestra los clientes que rentaron una película más de una vez
263 SELECT c.first_name, c.last_name, f.title,
264        COUNT(*) AS rental_count
265 FROM dbo.customer AS c
266 JOIN dbo.rental AS r ON c.customer_id = r.customer_id
267 JOIN dbo.inventory AS i ON r.inventory_id = i.inventory_id
268 JOIN dbo.film AS f ON i.film_id = f.film_id
269 GROUP BY c.customer_id, c.first_name, c.last_name, f.title
270 HAVING COUNT(*) > 1;
```

The query results are displayed in a table with the following columns: first_name, last_name, title, and rental_count. The results show 18 rows of data, where each row represents a customer who has rented a specific movie more than once. The rental_count column indicates the number of times each customer has rented the movie.

	first_name	last_name	title	rental_count
1	Mary	Smith	Fireball Philadelphia	2
2	Mary	Smith	Patient Sister	2
3	Tracy	Cole	Zorro Ark	2
4	Camen	Owens	Affair Prejudice	2
5	Cindy	Fisher	Virginian Pluto	2
6	Edith	Mcdonald	Wash Heavenly	2
7	Kim	Cruz	Rouge Squad	2
8	Sherry	Marshall	Bingo Talented	2
9	Sherry	Marshall	Seabiscuit Punk	2
10	Nancy	Thomas	Rock Instinct	2
11	Thelma	Murray	Disciple Mother	3
12	Thelma	Murray	Mallrats United	2
13	Thelma	Murray	Musketeers Wait	2
14	Thelma	Murray	Orient Closer	2
15	Carie	Porter	Dinosaur Secretary	2
16	Esther	Crawford	Jerk Paycheck	2
17	Rhonda	Kennedy	Dawn Pond	2
18	April	Burns	Wife Tum	2

The status bar at the bottom indicates that the query was executed successfully, showing 212 rows in total.

Identifica a los clientes que han alquilado la misma película en múltiples ocasiones, como resultado se encontraron clientes que han rentado la misma película más de una vez. Por ejemplo, esto podría indicar lealtad hacia ciertas películas o simplemente la necesidad de re-alquilar.

Conclusión General:

El análisis de estas 15 consultas SQL proporciona una visión completa y valiosa sobre la operación de la base de datos de alquiler de películas "dvdrental". Las consultas no solo confirman la alta calidad y completitud de los datos (sin valores nulos), sino que también ofrecen información clave para la toma de decisiones comerciales.

Podemos identificar tendencias en la popularidad de las películas y actores, evaluar el rendimiento de las tiendas, detectar películas que no generan ingresos y entender los patrones de alquiler de los clientes. En resumen, este análisis es fundamental para optimizar la gestión del inventario, mejorar las estrategias de marketing y, en última instancia, potenciar el negocio de alquiler de películas.