ROCKBUSTER STEALTH LLC

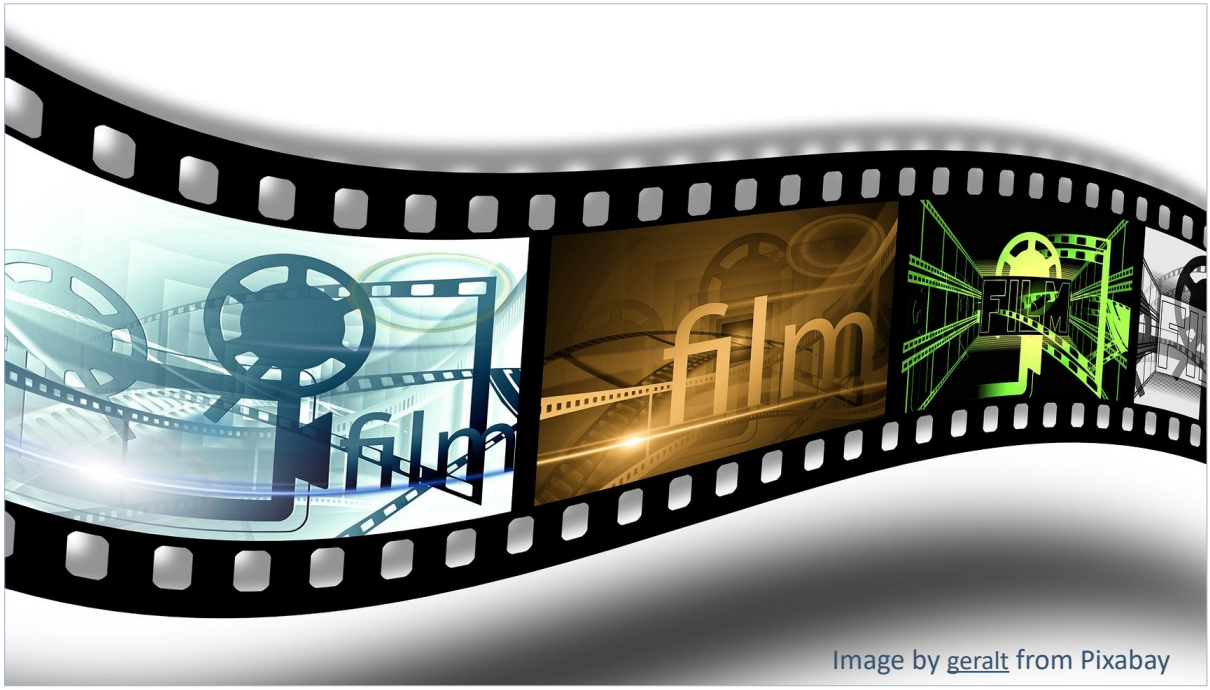


Image by geralt from Pixabay

Online Video Rental Launch Strategy: Data Dictionary and SQL Queries

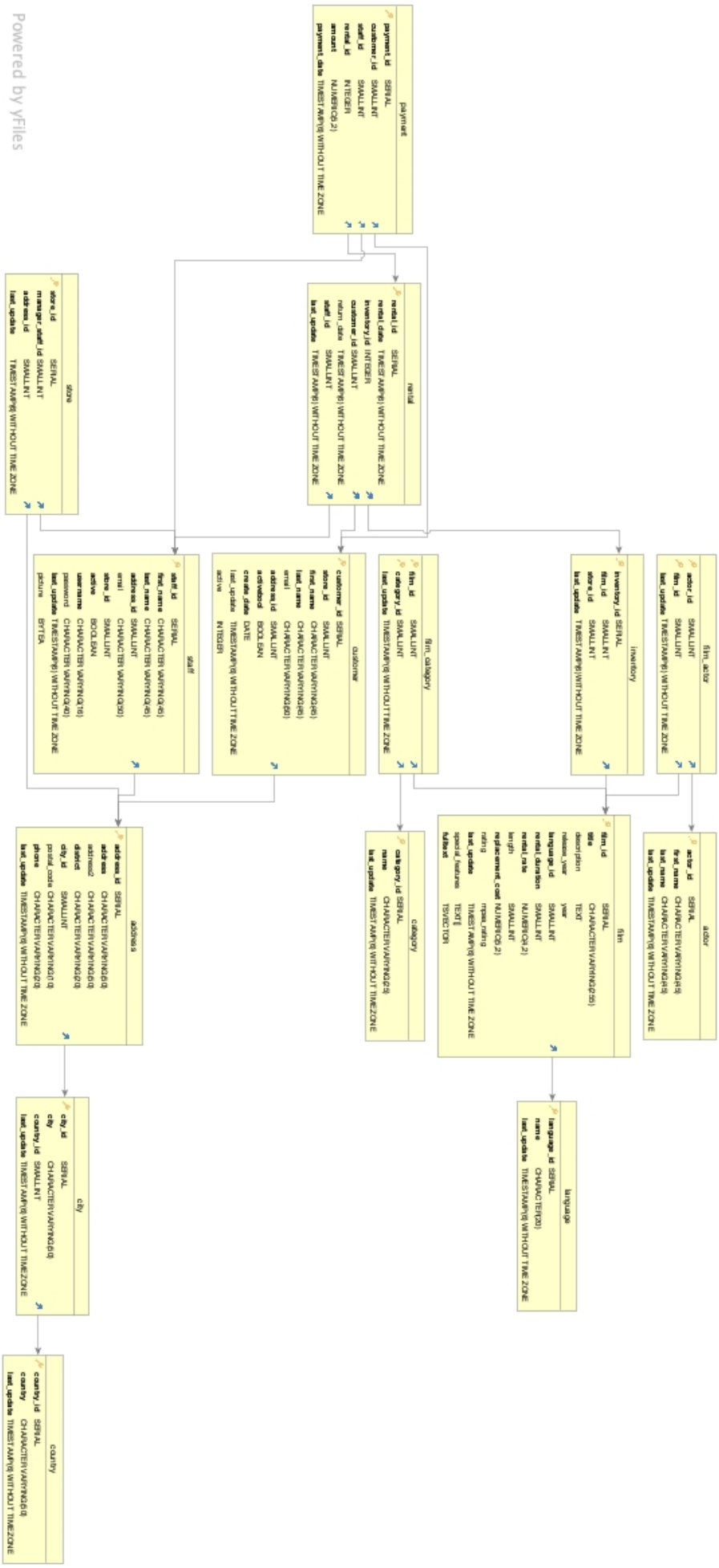
Elsa Ekevall

20 May 2022

Table of Contents

Version	Author	Date	Changes
V 0.1	Elsa Ekevall	14 June 2022	First draft
V 1.0	Elsa Ekevall	19 June 2022	Version 1.0 approved by Project Manager

Rockbuster Entity Relationship Diagram



1. Rockbuster

The Rockbuster database supports standard online payment transaction processing for the companies' video rental business.

The database was downloaded on the 5th of April 2022 and is archived here: Rockbuster " actor.csv" file

2. Rockbuster Entity Relationship Diagram

Customers, staff and stores have common tables for address, city and country. The payment table that holds ID for payment, customer, staff and rental, plus amount and payment date I can be linked to all the tables.

3. Legend


Primary Key


Primary Key Relation

The following tables describe the fact and dimension tables that are in the Rockbuster database. The 'Links To' column indicates which table the foreign key links to and the 'Links From' column indicates the table where the primary key is listed as a foreign key.

4. Data Tables

4.1 Fact Tables

4.1.1 payment

Payment information for the rental transactions

Key	Columns	Data type	Description	Links To	Links From
	payment_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (payment) is inserted.		
	customer_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (customer) is inserted into the customer table.	Customer	
	staff_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (staff member) is inserted into the staff table.	Staff	
	rental_id	INTEGER	The unique integer that is automatically assigned by the database server when a new record (customer rental) is inserted into the rental table.	rental	
	amount	NUMERIC(5,2)	The payment amount stored as a decimal number with a maximum (precision) of 5 digits to the left and (scale) 2 digits to the right.		
	payment_date	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time the payment was made without the time zone data.		

4.1.2 rental

Rental information for the rental transactions

Key	Columns	Data type	Description	Links To	Links From
	rental_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (customer rental) is inserted.		payment
	rental_date	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time the rental was made without the time zone data. [If the database server time zone changes the database will not update automatically.]		
	inventory_id	INTEGER	The unique integer that is automatically assigned by the database server when a new record (rental item) is inserted into the inventory table.	Inventory	
	customer_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new row/record (customer) is inserted into the customer table.	Customer	
	return_date	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time the rental was returned without the time zone data.		
	staff_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (staff member) is inserted into the staff table.	Staff	
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (customer rental) was last updated without the time zone data.		

4.2 Dimension Tables

4.2.1 store

Store details

Key	Columns	Data type	Description	Links To	Links From
	store_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (store) is inserted.		
	manager_staff_id	SMALLINT	The unique integer that is automatically assigned by the database server to this managerial staff member in the staff table.	staff	
	address_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (address) is inserted into the address table.	address	
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (store_id) was last updated without the time zone data.		

4.2.2 film_actor

Table linking actor details and film details

Key	Columns	Data type	Description	Links To	Links From
	actor_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (actor) is inserted into the actor table.	actor	
	film_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (film) is inserted into the film table.	film	
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (film_actor_id) was last updated without the time zone data.		

4.2.3 inventory

Inventory details

Key	Columns	Data type	Description	Links To	Links From
	inventory_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (rental item) is inserted.		rental
	film_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (film) is inserted into the film table.	film	
	store_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (store) is inserted into the store table.		
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (inventory_id) was last updated without the time zone data.		

4.2.4 film_category

Table linking film details and film category (genre)

Key	Columns	Data type	Description	Links To	Links From
	film_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (film) is inserted into the film table.	film	
	category_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (category) is inserted into the category table.	category	
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (film_category_id) was last updated without the time zone data.		

4.2.5 customer

Customer details

Key	Columns	Data type	Description	Links To	Links From
	customer_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (customer) is inserted.		payment rental
	store_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (store) is inserted into the store table.		
	first_name	CHARACTER VARYING(45)	The customer's first name stored in a variable length string of up to 45 characters.		
	last_name	CHARACTER VARYING(45)	The customer's last name stored in a variable length string of up to 45 characters.		
	email	CHARACTER VARYING(50)	The customer's email address stored in a variable length string of up to 50 characters.		
	address_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (address) is inserted into the address table.	address	
	activebool	BOOLEAN	True if the record is currently actively being used.		
	create_date	DATE	The date and time the rental was returned without the time zone data.		
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (customer_id) was last updated without the time zone data.		
	active	INTEGER	The status of the customer; where active = 1 or inactive = 0.		

4.2.6 staff

Staff details

Key	Columns	Data type	Description	Links To	Links From
	staff_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (staff member) is inserted.		payment rental
	first_name	CHARACTER VARYING(45)	The customer's first name stored in a variable length string of up to 45 characters.		
	last_name	CHARACTER VARYING(45)	The customer's last name stored in a variable length string of up to 45 characters.		
	address_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (address) is inserted into the address table.	address	
	email	CHARACTER VARYING(50)	The staff member's email address stored in a variable length string of up to 50 characters.		
	store_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (store) is inserted into the store table.		
	active	BOOLEAN	The status of the staff member; where active = True and inactive = False.		
	username	CHARACTER VARYING(16)	The staff member's username stored in a variable length string of up to 16 characters.		
	password	CHARACTER VARYING(40)	The staff member's password stored in a variable length string of up to 40 characters.		
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (staff_id) was last updated without the time zone data.		
	picture	BYTEA	A picture of the staff member stored as a variable-length binary string.		

4.2.7 actor

Details of the actors in the films

Key	Columns	Data type	Description	Links To	Links From
	actor_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (actor) is inserted.		film_actor
	first_name	CHARACTER VARYING(45)	The actor's first name stored in a variable length string of up to 45 characters.		
	last_name	CHARACTER VARYING(45)	The actor's last name stored in a variable length string of up to 45 characters.		
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (actor_id) was last updated without the time zone data.		

4.2.8 film

Details of the films

Key	Columns	Data type	Description	Links To	Links From
	film_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (film) is inserted.		film_actor inventory film_category
	language_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (language) is inserted into the language table.	language	
	title	CHARACTER VARYING(255)	The title of the film stored in a variable length string of up to 225 characters.		
	description	TEXT	The description of the film stored in a variable length character string.		
	release_year	year	The year the film was released stored as year (custom data type).		
	rental_duration	SMALLINT	The duration the film can be rented for stored as an integer.		
	rental_rate	NUMERIC(4,2)	The charge for renting the film stored as a decimal number with a maximum (precision) of 4 digits to the left and (scale) 2 digits to the right.		
	length	SMALLINT	The duration of the film (running time) stored as an integer.		
	replacement_cost	NUMERIC(5,2)	The cost of replacing the film stored as a decimal number with a maximum (precision) of 5 digits to the left and (scale) 2 digits to the right.		
	rating	mpaa_rating	The Motion Picture Association film rating that assess suitability for certain audiences (custom data type).		
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (film_id) was last updated without the time zone data.		
	special_features	TEXT[]	A description of any special features e.g. director's cuts, behind the scenes that the film has stored in a variable length character string.		
	fulltext	TSVECTOR	The full text search document that allows full-text queries on the character based columns in the film table.		

4.2.9 category

Film categories (genres)

Key	Columns	Data type	Description	Links To	Links From
	category_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (category) is inserted.		film_category
	name	CHARACTER VARYING(25)	The name of the film category for example action, comedy, drama, romance stored in a variable length character string.		
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (category_id) was last updated without the time zone data.		

4.2.10 address

Address information for the customers, staff and stores

Key	Columns	Data type	Description	Links To	Links From
	address_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (address) is inserted.		customer staff store
	address	CHARACTER VARYING(50)	The first line of the entity's address stored in a variable length character string.		
	address2	CHARACTER VARYING(50)	The second line of the entity's address stored in a variable length character string.		
	district	CHARACTER VARYING(20)	The name of the district where the entity resides stored in a variable length character string.		
	city_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (city) is inserted into the city table.	city	
	postal_code	CHARACTER VARYING(10)	The postal code where the entity resides stored in a variable length character string.		
	phone	CHARACTER VARYING(20)	The phone number of the entity stored in a variable length character string.		
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (address_id) was last updated without the time zone data.		

4.2.11 language

Film languages

Key	Columns	Data type	Description	Links To	Links From
	language_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (language) is inserted.		film
	name	CHARACTER(20)	The name of the main language in the film stored in a character string.		
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (language_id) was last updated without the time zone data.		

4.2.12 city

City information for the customers, staff and stores

Key	Columns	Data type	Description	Links To	Links From
	city_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (city) is inserted.		address
	city	CHARACTER VARYING(50)	The name of the city where the entity resides stored in a variable length character string.		
	country_id	SMALLINT	The unique integer that is automatically assigned by the database server when a new record (country) is inserted into the country table.	country	
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (city_id) was last updated without the time zone data.		

4.2.13 country

Country information for the customers, staff and stores

Key	Columns	Data type	Description	Links To	Links From
	country_id	SERIAL	A unique identifying sequential integer that is automatically assigned by the database server when a new row/record (country) is inserted.		city
	country	CHARACTER VARYING(50)	The name of the country where the entity resides stored in a variable length character string.		
	last_update	TIMESTAMP(6) WITHOUT TIME ZONE	The date and time that the record (country_id) was last updated without the time zone data.		

5. Descriptive Summary of Rockbuster Film and Customer Tables

Film Table

Not included release_year which has only one value 2006 or language_id which also has only one value 1. Film_id, title, description and full text should all be unique therefore not included. [I have run the analysis separately including the count as a visual check to see that the values = rows i.e. no data is missing. Also checked with IS NULL command.]

```
—To find the min, max, avg and count of rental_rate
SELECT MIN(rental_rate) AS min_rent,
        MAX(rental_rate) AS max_rent,
        AVG(rental_rate) AS avg_rent,
        COUNT(rental_rate) AS count_rent_values,
        COUNT(*) AS count_rows
FROM film
```

Data Output | Explain | Messages | Notifications

	min_rent numeric	max_rent numeric	avg_rent numeric	count_rent_values bigint	count_rows bigint
1	0.99	4.99	2.9800000000000000	1000	1000

```
--To find the min, max, avg and count of replacement_cost
SELECT MIN(replacement_cost) AS min_replacement_cost,
        MAX(replacement_cost AS max_replacement_cost,
        AVG(replacement_cost) AS avg_replacement_cost,
        COUNT(replacement_cost) AS count_replacement_cost_values,
        COUNT(*) AS count_rows
FROM film
```

Data Output | Explain | Messages | Notifications

	min_replacement_cost numeric	max_replacement_cost numeric	avg_replacement_cost numeric	count_replacement_cost_values bigint	count_rows bigint
1	9.99	29.99	19.9840000000000000	1000	1000

```
--To find the min, max, avg and count of rental_duration
SELECT MIN(rental_duration) AS min_duration,
        MAX(rental_duration) AS max_duration,
        AVG(rental_duration) AS avg_duration,
        COUNT(rental_duration) AS count_duration_values,
    COUNT(*) AS count_rows
FROM film
```

Data Output | Explain | Messages | Notifications

	min_duration smallint	max_duration smallint	avg_duration numeric	count_duration_values bigint	count_rows bigint
1	3	7	4.9850000000000000	1000	1000

```
--To find the min, max, avg and count of film_length
SELECT MIN(length) AS min_film_length,
    MAX(length) AS max_film_length,
    AVG(length) AS avg_film_length,
    COUNT(length) AS count_film_length_values,
    COUNT(*) AS count_rows
FROM film
```

Data Output | Explain | Messages | Notifications

	min_film_length smallint	max_film_length smallint	avg_film_length numeric	count_film_length_values bigint	count_rows bigint
1	46	185	115.2720000000000000	1000	1000

```
--To find the min and max of last_update
SELECT MIN(last_update) AS min_last_update,
    MAX(last_update) AS max_last_update,
    COUNT(last_update) AS count_last_update_values,
    COUNT(*) AS count_rows
FROM film
```

Data Output | Explain | Messages | Notifications

	min_last_update timestamp without time zone	max_last_update timestamp without time zone	count_last_update_values bigint	count_rows bigint
1	2013-05-26 14:50:58.951	2013-05-26 14:50:58.951	1000	1000

```
--To find the mode of rating
SELECT mode() WITHIN GROUP (ORDER BY rating)
        AS rating_modal_value,
         COUNT(rating) AS count_rating_values,
         COUNT(*) AS count_rows
FROM film
```

Data Output | Explain | Messages | Notifications

	rating_modal_value mpaa_rating	count_ratinge_values bigint	count_rows bigint
1	PG-13	1000	1000

```
--To find the mode of special_features
SELECT mode() WITHIN GROUP (ORDER BY special_features)
         AS special_features_modal_value,
         COUNT(special_features) AS count_special_features_values,
         COUNT(*) AS count_rows
FROM film
```

Data Output | Explain | Messages | Notifications

	special_features_modal_value text[]	count_special_features_values bigint	count_rows bigint
1	{Trailers,Commentaries,"Behind the Scenes"}	1000	1000

Customer Table

Not included activebool which is always true. Customer_id, store_id, email and address_id should all be unique therefore not included.

```
--To find the min and max of last_update
SELECT MIN(last_update) AS min_last_update,
      MAX(last_update) AS max_last_update,
      COUNT(last_update) AS
count_last_update_values,
      COUNT(*) AS count_rows
FROM customer
```

Data Output | Explain | Messages | Notifications

	min_last_update timestamp without time zone	max_last_update timestamp without time zone	count_last_update_values bigint	count_rows bigint
1	2013-05-26 14:49:45.738	2013-05-26 14:49:45.738	599	599

```
--To find the min and max of create_date
SELECT MIN(create_date) AS min_create_date,
      MAX(create_date) AS max_create_date,
      COUNT(create_date) AS count_create_date_values,
      COUNT(*) AS count_rows
FROM customer
```

Data Output | Explain | Messages | Notifications

	min_create_date date	max_create_date date	count_create_date_values bigint	count_rows bigint
1	2006-02-14	2006-02-14	599	599

```
--To find the mode of store_id [not very informative when only two
values]
SELECT mode() WITHIN GROUP (ORDER BY store_id)
         AS store_id_modal_value,
         COUNT(store_id) AS count_store_id_values,
         COUNT(*) AS count_rows
FROM customer
```

Data Output | Explain | Messages | Notifications

	store_id_modal_value smallint	count_store_id_values bigint	count_rows bigint
1	1	599	599

Further data summarisation/exploration:

```
--To explore the count/mode of first_name [using mode brings up Jamie]
SELECT first_name,
         COUNT(first_name) AS count_first_name,
FROM customer
GROUP BY first_name
ORDER BY count_first_name DESC
```

Data Output | Explain | Messages | Notifications

	first_name character varying (45)	count_first_name bigint
1	Marion	2
2	Jamie	2
3	Tracy	2
4	Kelly	2
5	Leslie	2
6	Willie	2
7	Jessie	2
8	Terry	2
9	Geraldine	1
10	Carolyn	1

```
--To explore the count/mode of last_name [using mode brings up Abney]
SELECT last_name,
        COUNT(last_name) AS count_last_name,
FROM customer
GROUP BY last_name
ORDER BY count_last_name DESC
```

Data Output | Explain | Messages | Notifications

	last_name character varying (45)	count_last_name bigint
1	Banda	1
2	Kruger	1
3	Mendoza	1
4	Bolin	1
5	Harris	1
6	Mcadams	1
7	Perkins	1
8	Wright	1
9	Hardison	1
10	Ray	1

```
--To explore store_id
SELECT store_id,
        COUNT(store_id) AS count_store_id,
FROM customer
GROUP BY store_id
ORDER BY count_store_id DESC
```

Data Output | Explain | Messages | Notifications

	store_id smallint	count_store_id bigint
1	1	326
2	2	273

```
--To explore active
SELECT active,
        COUNT(active) AS count_active
FROM customer
GROUP BY active
ORDER BY count_active DESC
```

Data Output | Explain | Messages | Notifications

	active integer	count_active bigint
1	1	584
2	0	15

6. The top 10 countries for Rockbuster in terms of customer numbers.

```
--Find the top 10 countries in terms of customer numbers
SELECT D.country,
        COUNT(A.customer_id) AS no_customers
FROM customer A
INNER JOIN address B ON A.address_id = B.address_id
INNER JOIN city C ON B.city_id = C.city_id
INNER JOIN country D ON C.country_id = D.country_id
GROUP BY D.country
ORDER BY no_customers DESC
LIMIT 10 -- only top 10
```

Data Output | Explain | Messages | Notifications

	country character varying (50)	no_customers bigint
1	India	60
2	China	53
3	United States	36
4	Japan	31
5	Mexico	30
6	Brazil	28
7	Russian Federation	28
8	Philippines	20
9	Turkey	15
10	Indonesia	14

7. The top 10 cities within the top 10 countries identified above.

```
--Find the top 10 cities within the top 10 countries in terms of customer numbers
SELECT C.city AS city_name,
        D.country,
        COUNT(A.customer_id) AS no_customers
FROM customer A
INNER JOIN address B ON A.address_id = B.address_id
INNER JOIN city C ON B.city_id = C.city_id
INNER JOIN country D ON C.country_id = D.country_id
WHERE D.country IN ('India',
                    'China',
                    'United States',
                    'Japan',
                    'Mexico',
                    'Brazil',
                    'Russian Federation',
                    'Philippines',
                    'Turkey',
                    'Indonesia')
GROUP BY C.city,
         D.country
ORDER BY no_customers DESC
LIMIT 10 -- only top 10
```

Data Output | Explain | Messages | Notifications

	city_name character varying (50)	country character varying (50)	no_customers bigint
1	Aurora	United States	2
2	Atlixco	Mexico	1
3	Xintai	China	1
4	Adoni	India	1
5	Dhule (Dhulia)	India	1
6	Kurashiki	Japan	1
7	Pingxiang	China	1
8	Sivas	Turkey	1
9	Celaya	Mexico	1
10	So Leopoldo	Brazil	1

Being in the top 10 doesn't mean much when most of the cities only have one customer and are therefore chosen based on the order they are in the scanning index. Another option is to find the top city in each country.

```
—Find the top city within each of the top 10 countries in terms of customer numbers
SELECT DISTINCT ON (D.country) D.country,
                                C.city AS city_name,
                                  COUNT(customer_id) AS
no_customers
FROM customer A
INNER JOIN address B ON A.address_id = B.address_id
INNER JOIN city C ON B.city_id = C.city_id
INNER JOIN country D ON C.country_id = D.country_id
WHERE D.country IN ('India',
                    'China',
                    'United States',
                    'Japan',
                    'Mexico',
                    'Brazil',
                    'Russian Federation',
                    'Philippines',
                    'Turkey',
                        'Indonesia')
GROUP BY C.city,
         D.country
ORDER BY D.country,
         no_customers DESC
```

Data Output Explain Messages Notifications

	country character varying (50)	city_name character varying (50)	no_customers bigint
1	Brazil	Anpolis	1
2	China	Xinxiang	1
3	India	Yamuna Nagar	1
4	Indonesia	Pemalang	1
5	Japan	Omiya	1
6	Mexico	Cuautla	1
7	Philippines	Lapu-Lapu	1
8	Russian Federation	Syktyvkar	1
9	Turkey	Kilis	1
10	United States	Aurora	2

8. The top 5 customers in the top 10 cities who have paid the highest total amounts to Rockbuster.

```
--Find the top 5 customers in the top 10 cities who have paid the highest total amounts
SELECT B.customer_id AS "Customer ID",
        B.first_name || ' ' || B.last_name AS "Customer First Name and Last Name",
        E.country AS "Country",
        D.city AS "City",
        SUM(A.amount) AS "Total Amount Paid"
FROM payment A
INNER JOIN customer B ON A.customer_id = B.customer_id
INNER JOIN address C ON B.address_id = C.address_id
INNER JOIN city D ON C.city_id = D.city_id
INNER JOIN country E ON D.country_id = E.country_id
WHERE E.country IN ('India',
                    'China',
                    'United States',
                    'Japan',
                    'Mexico',
                    'Brazil',
                    'Russian Federation',
                    'Philippines',
                    'Turkey',
                        'Indonesia')
AND D.city IN ('Aurora',
               'Atlixco',
               'Xintai',
               'Adoni',
               'Dhule (Dhulia)',
               'Kurashiki',
               'Pingxiang',
               'Sivas',
                 'Celaya',
                 'So Leopoldo')
GROUP BY B.customer_id,
         B.first_name,
         B.last_name,
         E.country,
         D.city
ORDER BY "Total Amount Paid" DESC
```

Data Output | Explain | Messages | Notifications

	Customer ID integer	Customer First Name and Last Name text	Country character varying (50)	City character varying (50)	Total Amount Paid numeric
1	84	Sara Perry	Mexico	Atlixco	128.70
2	518	Gabriel Harder	Turkey	Sivas	108.75
3	587	Sergio Stanfield	Mexico	Celaya	102.76
4	537	Clinton Buford	United States	Aurora	98.76
5	367	Adam Gooch	India	Adoni	97.80

LIMIT 5 -- only top 5

Find the top 5 customers in the top city in each country.

```
--Find the top 5 customers in the top city within each of the top 10 countries who have paid the highest total amounts
SELECT B.customer_id AS "Customer ID",
        B.first_name || ' ' || B.last_name AS "Customer First Name and Last Name",
        E.country AS "Country",
        D.city AS "City",
        C.district,
        SUM(A.amount) AS "Total Amount Paid"
FROM payment A
INNER JOIN customer B ON A.customer_id = B.customer_id
INNER JOIN address C ON B.address_id = C.address_id
INNER JOIN city D ON C.city_id = D.city_id
INNER JOIN country E ON D.country_id = E.country_id
WHERE E.country IN ('India',
                    'China',
                    'United States',
                    'Japan',
                    'Mexico',
                    'Brazil',
                    'Russian Federation',
                    'Philippines',
                    'Turkey',
                    'Indonesia')
AND D.city IN ('Aurora',
               'Anpolis',
               'Xinxiang',
               'Yamuna Nagar',
               'Pemalang',
               'Omiya',
               'Cuautla',
               'Lapu-Lapu',
               'Syktyvkar',
               'Kilis')
GROUP BY B.customer_id,
         B.first_name,
         B.last_name,
         E.country,
         D.city,
         C.district
ORDER BY "Total Amount Paid" DESC
LIMIT 5 -- only top 5
```

Data Output | Explain | Messages | Notifications

	Customer ID integer	Customer First Name and Last Name text	Country character varying (50)	City character varying (50)	district character varying (20)	Total Amount Paid numeric
1	404	Stanley Scroggins	Japan	Omiya	Saitama	133.71
2	244	Viola Hanson	Philippines	Lapu-Lapu	Central Visayas	122.70
3	116	Victoria Gibson	Indonesia	Pemalang	Central Java	106.74
4	537	Clinton Buford	United States	Aurora	Colorado	98.76
5	339	Walter Perryman	China	Xinxiang	Henan	95.76

9. Further PostgreSQL Queries

```
--To find the payment and rental transaction periods
SELECT MAX(payment_date) AS max_payment_date,
       MIN(payment_date) AS min_payment_date,
       MAX(rental_date) AS max_rental_date,
       MIN(rental_date) AS min_rental_date
FROM payment A
INNER JOIN rental B ON A.rental_id = B.rental_id

--To find the rental facts in figures (counts)
SELECT  COUNT(rental_id) AS transactions,
        COUNT(DISTINCT staff_id) AS employees,
        COUNT(DISTINCT customer_id) AS customers,
         SUM(amount) AS total_revenue
FROM payment

--To find the film facts in figures (counts)
SELECT  COUNT(DISTINCT title) AS film_titles,
        COUNT(DISTINCT rating) AS ratings,
        COUNT(DISTINCT language_id) AS languages
FROM film

--To find the inventory facts in figures (counts)
SELECT store_id,
       COUNT(DISTINCT inventory_id) AS videos,
       COUNT(DISTINCT category_id) AS genres,
       COUNT(DISTINCT A.film_id) AS inventory_title_count
FROM inventory A
INNER JOIN film_category B ON A.film_id = B.film_id
GROUP BY store_id
ORDER BY store_id DESC

--To find the number of languages
SELECT name AS language,
                COUNT(film_id) AS film_count
FROM language A
INNER JOIN film B ON A.language_id = B.language_id
GROUP BY language
ORDER BY COUNT(film_id) DESC

--To find the number of videos
SELECT COUNT(inventory_id) AS number_of_videos
FROM inventory

--To find the number of rentals
SELECT COUNT(rental_id) AS number_of_rentals
FROM rental
```

```
--To find the location, number of staff, staff name and revenue per store
SELECT  B.store_id,
        country,
        B.staff_id,
        B.first_name || ' ' || B.last_name AS staff_member,
        SUM(amount) AS store_revenue
FROM payment A
INNER JOIN staff B ON A.staff_id = B.staff_id
INNER JOIN address C ON B.address_id = C.address_id
INNER JOIN city D ON C.city_id = D.city_id
INNER JOIN country E ON D.country_id = E.country_id
GROUP BY country,
         B.store_id,
         B.staff_id

--To find the store rentals)
SELECT store_id,
       COUNT(rental_id) AS rentals
FROM inventory A
INNER JOIN rental B ON A.Inventory_id = B.inventory_id
GROUP BY store_id
ORDER BY store_id DESC

--To find the number of countries supplied, no of customers registered with each store and the registered
customer revenue
SELECT  store_id,
        COUNT(DISTINCT country) AS number_of_countries,
        COUNT(DISTINCT B.customer_id) AS number_of_customers,
        SUM(amount) AS registered_customer_revenue
FROM payment A
INNER JOIN customer B ON A.customer_id = B.customer_id
INNER JOIN address C ON B.address_id = C.address_id
INNER JOIN city D ON C.city_id = D.city_id
INNER JOIN country E ON D.country_id = E.country_id
GROUP BY store_id
ORDER BY  number_of_countries DESC

--To find the number per rental_rate
SELECT rental_rate,
       COUNT(film_id) AS film_count
FROM film
GROUP BY rental_rate
ORDER BY COUNT(film_id) DESC
```

```
—To find the rental information by genre
SELECT name AS film_genre,
            AVG(rental_rate) AS average_rental_rate,
            MIN(rental_duration) AS min_rental_duration,
            MAX(rental_duration) AS max_rental_duration
FROM film A
INNER JOIN film_category B ON A.film_id = B.film_id
INNER JOIN category C ON B.category_id = C.category_id
GROUP BY film_genre

-- To find the total revenue, number of film titles and number of films rented per genre
SELECT name AS film_genre,
          SUM(amount) AS total_revenue,
          COUNT(DISTINCT B.inventory_id) AS genre_video_count,
          COUNT(DISTINCT D.film_id) AS inventory_title_count
FROM payment A
INNER JOIN rental B ON A.rental_id = B.rental_id
INNER JOIN inventory C ON B.inventory_id = C.inventory_id
INNER JOIN film_category D ON C.film_id = D.film_id
INNER JOIN category E ON D.category_id = E.category_id
GROUP BY name
ORDER BY total_revenue DESC

--To find the number of film titles per rating
SELECT rating AS film_rating,
         COUNT.film_id) AS film_title_count
FROM film
GROUP BY film_rating
ORDER BY COUNT(film_id) DESC

--To find film titles not in the inventory
SELECT  A.film_id,
             title AS film_title,
                    rental_rate,
                    rating,
                    name AS film_genre
FROM    film A
INNER JOIN film_category B ON A.film_id = B.film_id
INNER JOIN category C ON B.category_id = C.category_id
WHERE   A.film_id
NOT IN (SELECT inventory.film_id FROM inventory)
```

```
--To find the total revenue and number of films rented by rating
SELECT rating AS film_rating,
        SUM(amount) AS total_revenue,
        COUNT(DISTINCT C.inventory_id) AS video_count,
        COUNT(DISTINCT C.film_id) AS inventory_title_count
FROM payment A
INNER JOIN rental B ON A.rental_id = B.rental_id
INNER JOIN inventory C ON B.inventory_id = C.inventory_id
INNER JOIN film D ON C.film_id = D.film_id
GROUP BY film_rating

--To find the number of customers, number of stores, number of staff and total revenue per country
WITH staff_details_cte (country, number_of_stores, number_of_staff, total_revenue) AS
(SELECT country.country,
            COUNT(DISTINCT store_id) AS store,
            COUNT(DISTINCT staff.staff_id) AS number_of_staff,
            SUM(amount) AS total_revenue
FROM staff
INNER JOIN payment ON staff.staff_id = payment.staff_id
INNER JOIN address ON staff.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
GROUP BY country.country,
        store_id
ORDER BY total_revenue DESC)
SELECT E.country,
            COUNT(DISTINCT B.customer_id) AS number_of_customers,
            COUNT(DISTINCT number_of_stores) AS number_of_stores,
            COUNT(DISTINCT number_of_staff) AS number_of_staff,
            SUM(amount) AS total_revenue
FROM payment A
INNER JOIN customer B ON A.customer_id = B.customer_id
INNER JOIN address C ON B.address_id = C.address_id
INNER JOIN city D ON C.city_id = D.city_id
FULL JOIN country E ON D.country_id = E.country_id
FULL JOIN staff_details_cte ON E.country = staff_details_cte.country
GROUP BY E.country
ORDER BY total_revenue DESC
```

```
--To find how many times films have been rented
SELECT COUNT(rental.inventory_id) AS no_times_rented,
        title AS film_title,
        rating AS film_rating,
        category.name AS film_genre
FROM rental
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
INNER JOIN film_category ON film.film_id = film_category.film_id
INNER JOIN category ON film_category.category_id = category.category_id
GROUP BY film_title,
          film_rating,
          category.name
ORDER BY no_times_rented DESC

--To find film titles not in the inventory
SELECT  film.film_id,
          title AS film_title
FROM    film
WHERE   film.film_id NOT IN (SELECT inventory.film_id FROM inventory)

--To find items never rented
SELECT inventory.inventory_id AS inventory_id,
                title AS film_title
FROM inventory
INNER JOIN film ON inventory.film_id = film.film_id
WHERE NOT EXISTS
                (SELECT rental.inventory_id
                 FROM rental
                WHERE inventory.inventory_id = rental.inventory_id)
GROUP BY inventory_id,
                film_title
```

```
-- To find the top 50 films by revenue
SELECT title AS film_title,
        name AS film_genre,
        rating AS film_rating,
        rental_rate,
        SUM(amount) AS total_revenue
FROM payment A
INNER JOIN rental B ON A.rental_id = B.rental_id
INNER JOIN inventory C ON B.inventory_id = C.inventory_id
INNER JOIN film D ON C.film_id = D.film_id
INNER JOIN film_category E ON D.film_id = E.film_id
INNER JOIN category F ON E.category_id = F.category_id
GROUP BY title,
            name,
            rating,
            rental_rate
ORDER BY total_revenue DESC
LIMIT 50

-- To find the bottom 50 films by revenue
SELECT title AS film_title,
        name AS film_genre,
        rating AS film_rating,
        rental_rate,
        SUM(amount) AS total_revenue
FROM payment A
INNER JOIN rental B ON A.rental_id = B.rental_id
INNER JOIN inventory C ON B.inventory_id = C.inventory_id
INNER JOIN film D ON C.film_id = D.film_id
INNER JOIN film_category E ON D.film_id = E.film_id
INNER JOIN category F ON E.category_id = F.category_id
GROUP BY title,
            name,
            rating,
            rental_rate
ORDER BY total_revenue ASC
LIMIT 50
```

```
-- To find the film details
SELECT title AS film_title,
        name AS film_genre,
        rating AS film_rating,
        length AS film_length,
        rental_rate,
        AVG(rental_duration) AS avg_rental_duration,
        SUM(amount) AS total_revenue
FROM payment A
INNER JOIN rental B ON A.rental_id = B.rental_id
INNER JOIN inventory C ON B.inventory_id = C.inventory_id
INNER JOIN film D ON C.film_id = D.film_id
INNER JOIN film_category E ON D.film_id = E.film_id
INNER JOIN category F ON E.category_id = F.category_id
GROUP BY title,
            name,
            rating,
            length,
            rental_rate
ORDER BY total_revenue DESC

--To find customer count and total payment received against each country
SELECT country,
    COUNT(DISTINCT A.customer_id) AS customer_count,
    COUNT(A.customer_id) AS transaction_count,
    SUM(amount) AS total_payment
FROM customer A
INNER JOIN address B ON A.address_id = B.address_id
INNER JOIN city C ON B.city_id = C.city_id
INNER JOIN country D ON C.country_ID = D.country_ID
INNER JOIN payment E ON a.customer_id = E.customer_id
GROUP BY country
```