In this task, you'll calculate some descriptive statistics using the MIN, MAX, AVG, COUNT, SUM, and MODE() aggregates discussed in this Exercise, and you'll reflect on what you learned about data profiling back in Exercise 1.5: Data Profiling & Integrity.

Directions

Rockbuster's database engineers have loaded some new data into the database, and your manager has asked you to clean and profile it. Follow the instructions below to complete their request:

1. ***Check for and clean dirty data:*** *Find out if the film table and the customer table contain any dirty data, specifically non-uniform or duplicate data, or missing values. Create a new "Answers 3.6" document and copy-paste your queries into it. Next to each query write 2 to 3 sentences explaining how you would clean the data (even if the data is not dirty).*

Film table

Checking duplicate data

```
SELECT  film_id,
        title,
        release_year,
        language_id,
        rental_duration,
        COUNT(*)
FROM film
GROUP BY
        film_id,
        title,
        release_year,
        language_id,
        rental_duration
HAVING COUNT(*) > 1;
```

Data Output | Messages | Notifications

film_id [PK] integer	title character varying (255)	release_year integer	language_id smallint	rental_duration smallint	count bigint

No duplicates found. For cleaning I could either create a virtual table "VIEW" with unique records or delete the duplicate records. As creating a "View" is safest and preferred way I would use query to return unique records with GROUP BY or DISTINCT.

Checking missing values

Whether values are missing WHERE and OR commands can be combined.

```
SELECT * FROM film
WHERE film_id IS NULL
    OR title IS NULL
    OR description IS NULL
    OR release_year IS NULL
    OR release_year IS NULL
    OR language_id IS NULL
    OR rental_duration IS NULL
    OR rental_rate IS NULL
    OR length IS NULL
    OR replacement_cost IS NULL
    OR rating IS NULL
    OR last_update IS NULL
    OR special_features IS NULL
    OR fulltext IS NULL;
```

Data Output | Messages | Notifications

film_id [PK] integer	title character varying (255)	description text	release_year integer	language_id smallint	rental_duration smallint	rental_rate numeric (4,2)	length smallint	replacement_cost numeric (5,2)

Or, checking for NULLs for some random important variables can be done, e.g., as shown below.

```
SELECT
        SUM(CASE WHEN rental_duration IS NULL THEN 1 ELSE 0 END) AS rental_duration_missing,
        SUM(CASE WHEN rental_rate IS NULL THEN 1 ELSE 0 END) AS rental_rate_missing,
        SUM(CASE WHEN replacement_cost IS NULL THEN 1 ELSE 0 END) AS replacement_cost_missing,
        SUM(CASE WHEN rating IS NULL THEN 1 ELSE 0 END) AS rating_missing
FROM film;
```

	rental_duration_missing bigint	rental_rate_missing bigint	replacement_cost_missing bigint	rating_missing bigint
1	0	0	0	0

In this case no NULLs are found and we assume no value is missing. If any NULL is found, then further investigations can be done, e.g., identifying the rows with missing values to determine if missing value is less or more than 5%. If it is less than 5%, imputing averages can be done, else the column should be omitted from analysis and reporting for the output transparency.

Checking Non-Uniform data

To check the Non-Uniform data I would use GROUP BY and DISTINCT query functions to find inconsistency randomly for any column, such as checking 'rating' or release year column as example.

```
SELECT DISTINCT rating
FROM film
ORDER BY rating
```

	rating mpaa_rating
1	G
2	PG
3	PG-13
4	R
5	NC-17

```
-- For example checking release year
SELECT release_year
FROM film
GROUP BY release_year
```

	release_year integer
1	2006

No non-uniform data is found. If non-uniform data exist in the table, values in the table can be updated using UPDATE command.

Customer table

Checking duplicate data

```
SELECT  customer_id,
        store_id,
        first_name,
        last_name,
        email,
        COUNT(*)
FROM customer
GROUP BY
        customer_id,
        store_id,
        first_name,
        last_name,
        email
HAVING COUNT(*) > 1;
```

Data Output | Messages | Notifications

customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	count bigint

If duplicates are found, similar ways of cleaning can be approached.

Checking missing values

```
SELECT * FROM customer
WHERE customer_id IS NULL
    OR store_id IS NULL
    OR first_name IS NULL
    OR last_name IS NULL
    OR email IS NULL
    OR address_id IS NULL
    OR activebool IS NULL
    OR create_date IS NULL
    OR last_update IS NULL
    OR active IS NULL;
```

Data Output | Messages | Notifications

customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	create_date date	last_update timestamp without time zone	active integer

Similar approach as above for film table.

Checking Non-Uniform Data

```
SELECT DISTINCT store_id
FROM customer;
```

Data Output | Messages | Notifications

Showing rows: 1 to 2 Page No: 1

	store_id smallint
1	1
2	2

```
SELECT create_date
FROM customer
WHERE create_date < '1900-01-01'
   OR create_date > CURRENT_DATE;
```

Data Output | Messages | Notifications

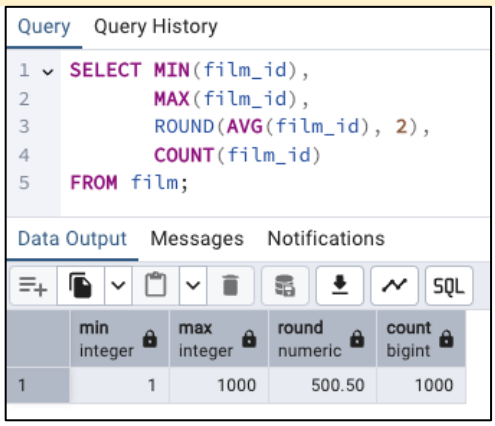
create_date date

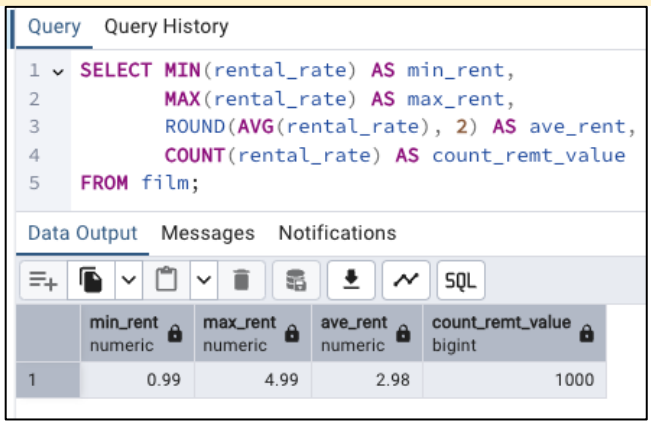
In the second query last_update has been been checked whether the date lies within the usual range as example. No non-uniforma data is found. In case of non-uniform data I would have use UPDATE command to correct the data to make it consistent.

2. ***Summarize your data:*** *Use SQL to calculate descriptive statistics for both the film table and the customer table. For numerical columns, this means finding the minimum, maximum, and average values. For non-numerical columns, calculate the mode value. Copy-paste your SQL queries and their outputs into your answers document.*

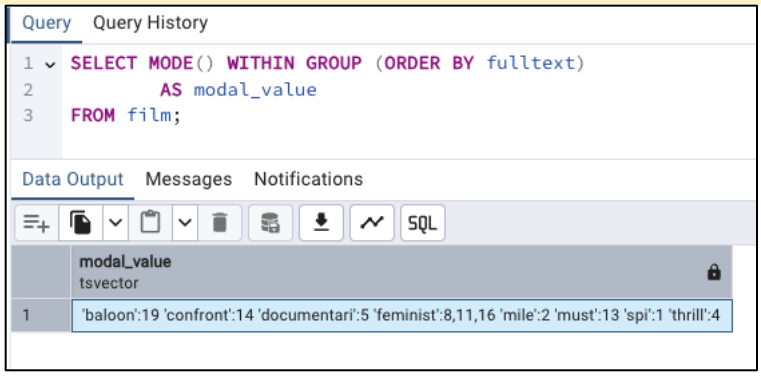
Film Table:

For numerical columns





For non-numerical columns



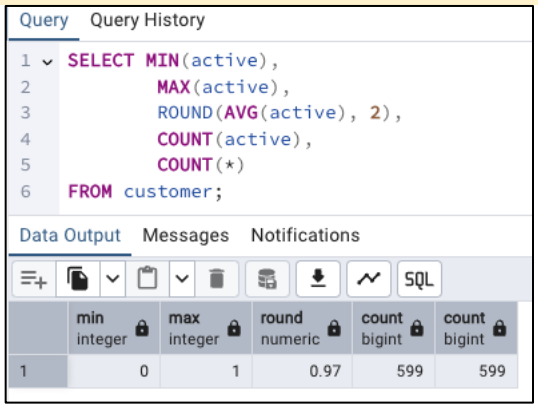
Summary:

Columns	MIN	MAX	AVG	COUNT	COUNT ROWS	MODE
film_id	1	1000	500.50	1000	1000	
title				1000	1000	Academy Dinosaur
release_year	2006	2006	2006	1000	1000	
language_id	1	1	1	1000	1000	
rental_duration	3	7	4.99	1000	1000	
rental_rate	0.99	4.99	2.98	1000	1000	
length	46	185	115.27	1000	1000	
replacement_cost	9.99	29.99	19.98	1000	1000	
rating				1000	1000	PG-13

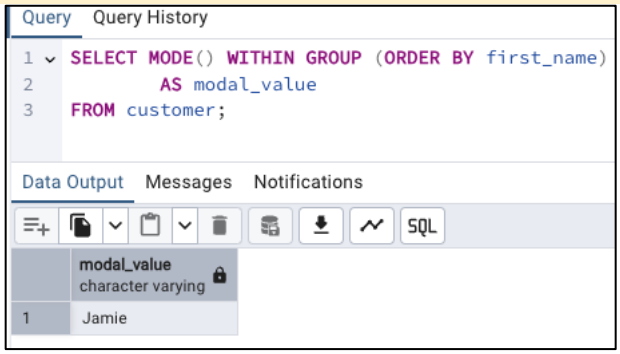
last_update				1000	1000	2013-05-26 14:50:58.951
special_features				1000	1000	{Trailers,Commentaries, ""Behind the Scenes""}
full_text				1000	1000	'baloon':19 'confront':14 'documentari':5 'feminist':8,11,16 'mile':2 'must':13 'spi':1 'thrill':4

Customer Table:

For numerical columns



For non-numerical columns



Summary:

Columns	MIN	MAX	AVG	COUNT	COUNT ROWS	MODE
customer_id	1	599	300.00	599	599	
store_id	1	2	1.46	599	599	
first_name				599	599	Jamie
last_name				599	599	Abney
email				599	599	aaron.selby@sakilacustomer.org
address_id	5	605	304.72	599	599	
activebool				599	599	true
create_date				599	599	2006-02-14
last_update				599	599	2013-05-26 14:49:45.738
active	0	1	0.97	599	599	

3. ***Reflect on your work:*** *Back in Achievement 1 you learned about data profiling in Excel. Based on your previous experience, which tool (Excel or SQL) do you think is more effective for data profiling, and why? Consider their respective functions, ease of use, and speed. Write a short paragraph in the running document that you have started.*

   As it is known that SQL is more effective for larger data set. Various functions such as MIN, MAX, AVG, COUNT etc. can be executed at once in SQL, which makes SQL faster than Excel. Furthermore, SQL provide powerful querying capabilities to aggregate, filter, and identify data quality issue quickly. Excel is user-friendly and great for quick, visual exploration of smaller datasets, offering built-in functions and filters that are easy to use without deep technical knowledge.

4. *Save your "Answers 3.6" document as a PDF and upload it here for your tutor to review.*