

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:

Check for and clean dirty data:

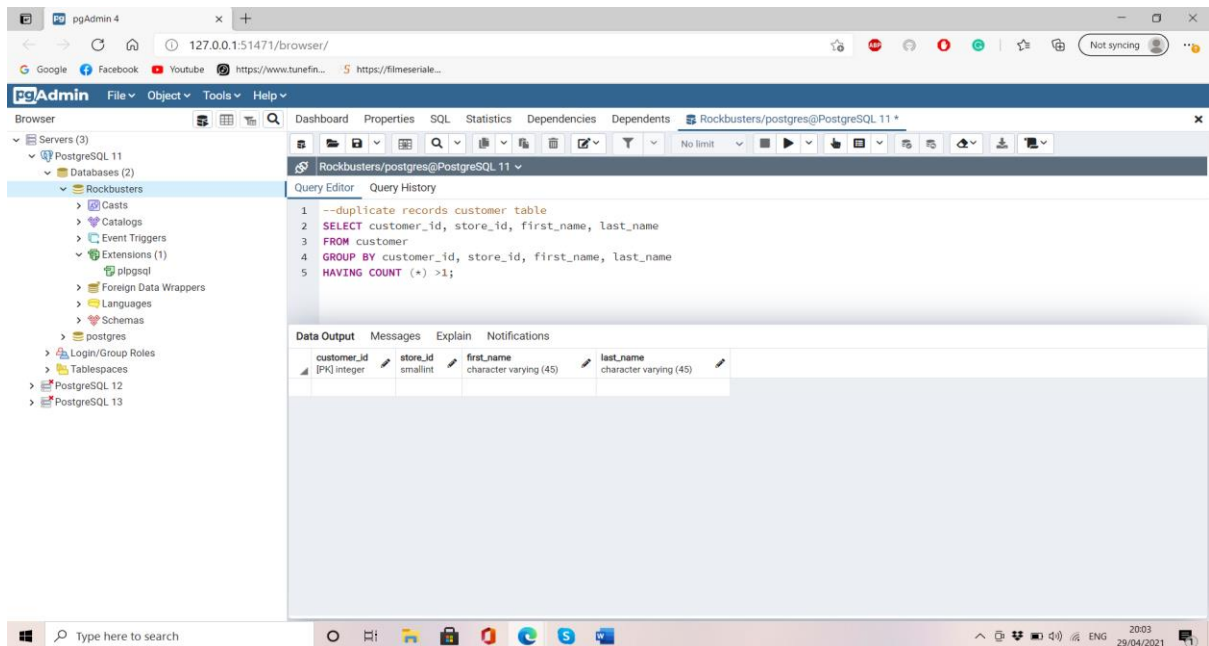
- Find out if the film table and the customer table contain any dirty data, specifically non-uniform or duplicate data, or missing values.
- Create a new “Answers 3.6” document and copy-paste your queries into it. Next to each query write 2 to 3 sentences explaining how you would clean the data (even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

Duplicate Data

The screenshot shows the pgAdmin 4 web interface in a browser window. The left sidebar displays the database structure, with 'Rockbusters' selected under 'PostgreSQL 11'. The main panel shows the 'Query Editor' with the following SQL query:

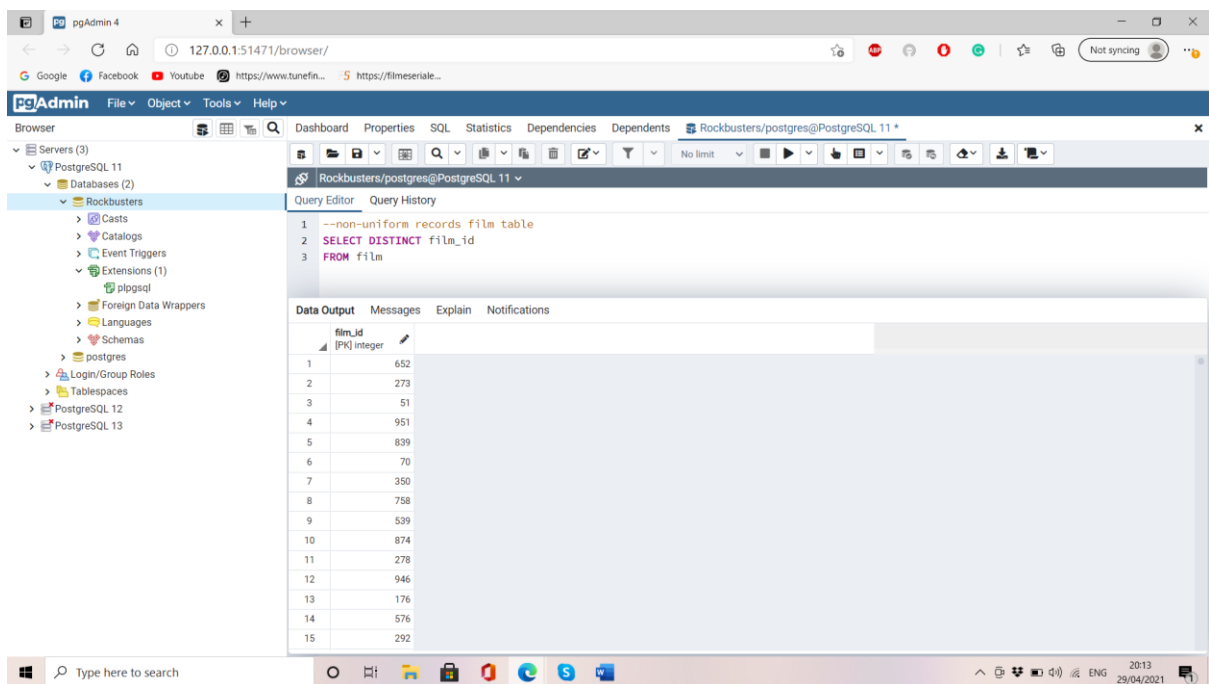
```
1 --duplicate records
2 SELECT film_id, title, description
3 FROM film
4 GROUP BY film_id, title, description
5 HAVING COUNT (*) > 1;
```

Below the query editor, the 'Data Output' tab is active, showing a table with the following headers: 'film_id [PK] integer', 'title character varying (255)', and 'description text'. The table is currently empty. At the bottom right, a green status bar indicates: 'Successfully run. Total query runtime: 163 msec. 0 rows affected.'



- This is the query that I had run, and it did not give any duplicate information.
- This means that the data is clean.
- I have selected the first few columns from each table to check for duplicates. There are none.

Non - uniform Values



- Scanning through the distinct values for film_id nothing looks non-uniform.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including 'Servers (3)', 'PostgreSQL 11', 'Databases (2)', and 'Rockbusters'. The main window shows the 'Query Editor' for 'Rockbusters/postgres@PostgreSQL 11 *'. The query is:

```
1 --non-uniform check on film table
2 SELECT title, release_year, language_id, rental_duration
3 FROM film
4 GROUP BY title, release_year, language_id, rental_duration;
```

The 'Data Output' tab shows the results of the query, which are 15 rows of film data:

	title	release_year	language_id	rental_duration
1	Jet Neighbors	2006	1	7
2	Perfect Groove	2006	1	7
3	Confidential Interview	2006	1	6
4	Devil Desire	2006	1	6
5	Empire Malkovich	2006	1	7
6	Roof Champion	2006	1	7
7	Manchurian Curtain	2006	1	5
8	Bunch Minds	2006	1	4
9	Women Dorado	2006	1	4
10	Rainbow Shock	2006	1	3
11	Million Ace	2006	1	4
12	Massage Image	2006	1	4
13	Fever Empire	2006	1	5
14	Behavior Runaway	2006	1	3
15	Gun Bonnie	2006	1	7

- Visual check shows no obvious inconsistencies.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including 'Servers (3)', 'PostgreSQL 11', 'Databases (2)', and 'Rockbusters'. The main window shows the 'Query Editor' for 'Rockbusters/postgres@PostgreSQL 11 *'. The query is:

```
1 --non-uniform records customer table
2 SELECT DISTINCT store_id
3 FROM customer
```

The 'Data Output' tab shows the results of the query, which are 2 rows of store_id values:

store_id
1
2

- Only two values for store_id – not non-uniform.

The screenshot shows the pgAdmin 4 web interface in a browser. The left sidebar displays a tree view of the database structure, including Servers, PostgreSQL 11, Databases, and the 'Rockbusters' database. The main panel is divided into a 'Query Editor' and a 'Data Output' section. The Query Editor contains a SQL query that checks for non-uniform values in the 'customer' table. The Data Output section displays the results of the query as a table with 15 rows.

Query Editor:

```
1 --non-uniform check on customer table
2 SELECT customer_id, store_id, first_name, last_name
3 FROM customer
4 GROUP BY customer_id, store_id, first_name, last_name;
```

Data Output:

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)
1	184	1	Vivian	Ruiz
2	87	1	Wanda	Patterson
3	477	1	Dan	Paine
4	273	2	Priscilla	Lowe
5	550	2	Guy	Brownlee
6	394	2	Chris	Brothers
7	51	1	Alice	Stewart
8	272	1	Kay	Caldwell
9	70	2	Christina	Ramirez
10	190	2	Yolanda	Weaver
11	350	1	Juan	Fraley
12	539	1	Mathew	Bolin
13	554	1	Dwayne	Olvera
14	278	2	Billie	Horton
15	424	2	Kyle	Spurlock

- Visual check shows no obvious inconsistencies.
- If I needed to correct non-uniform values, I would use the UPDATE command. For example, if store_id had been inputted incorrectly for some records (instead of 1 – 'one' or 'uno' had been entered) I would use the following command: UPDATE customer SET store_id = 1 WHERE store_id IN ('one', 'uno').

Missing Values

The first screenshot shows a SQL query in the Query Editor of pgAdmin 4. The query is designed to check for non-uniform records in the 'film' table by counting distinct values for 'film_id', 'description', and 'release_year'.

```
1 --non-uniform records film table
2 SELECT DISTINCT COUNT(film_id), COUNT(description), COUNT(release_year)
3 FROM film
```

The Data Output tab shows the results of the query:

	count bigint	count bigint	count bigint
1	1000	1000	1000

A green message at the bottom indicates: "Successfully run. Total query runtime: 125 msec. 1 rows affected."

The second screenshot shows a similar query for the 'customer' table, checking for non-uniform records by counting distinct values for 'customer_id', 'first_name', and 'last_name'.

```
1 --non-uniform records customer table
2 SELECT DISTINCT COUNT(customer_id), COUNT(first_name), COUNT(last_name)
3 FROM customer
```

The Data Output tab shows the results:

	count bigint	count bigint	count bigint
1	599	599	599

- To find non-uniform data it's important to check some random values and look for potential anomalies or inconsistencies.
- To fill in missing data following options should be considered: - Ignore columns with a high percentage of missing values - Impute the missing values with estimates.

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

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including 'Rockbusters' and its tables. The main window shows a SQL query in the 'Query Editor' and its results in the 'Data Output' tab.

Query Editor:

```
--summary of film table
1 SELECT MIN(release_year) AS min_release_year,
2 MAX(release_year) AS max_release_year,
3 mode() WITHIN GROUP (ORDER BY language_id) AS mode_language_id,
4 MIN(rental_duration) AS min_rental_duration,
5 MAX(rental_duration) AS max_rental_duration,
6 CAST(AVG(rental_duration) AS DECIMAL(3,2)) AS avg_rental_duration,
7 MIN(rental_rate) AS min_rent,
8 MAX(rental_rate) AS max_rent,
9 CAST(AVG(rental_rate) AS DECIMAL(3,2)) AS avg_rent,
10 MIN(length) AS min_length,
11 MAX(length) AS max_length,
12 CAST(AVG(length) AS DECIMAL(3)) AS avg_length,
13 MIN(replacement_cost) AS min_replacement,
14 MAX(replacement_cost) AS max_replacement,
15 CAST(AVG(replacement_cost) AS DECIMAL(3)) AS avg_replacement,
16 mode() WITHIN GROUP (ORDER BY rating) AS mode_rating
17 FROM film;
```

Data Output:

min_release_year	max_release_year	mode_language_id	min_rental_duration	max_rental_duration	avg_rental_duration	min_rent	max_rent	avg_rent
integer	integer	smallint	smallint	smallint	numeric (3,2)	numeric	numeric	numeric
1	2006	2006	1	3	7	4.99	0.99	4.99

Customer table

A screenshot of the pgAdmin 4 web interface. The browser window shows the URL 127.0.0.1:51471/browser/. The left sidebar shows the database structure: Servers (3) > PostgreSQL 11 > Databases (2) > Rockbusters. The main panel shows the 'Query Editor' for 'Rockbusters/postgres@PostgreSQL 11'. The query is:

```
1 SELECT mode()WITHIN GROUP(ORDER BY customer_id)
2 AS modal_value
3 FROM customer;
```

 The 'Data Output' tab shows the result:

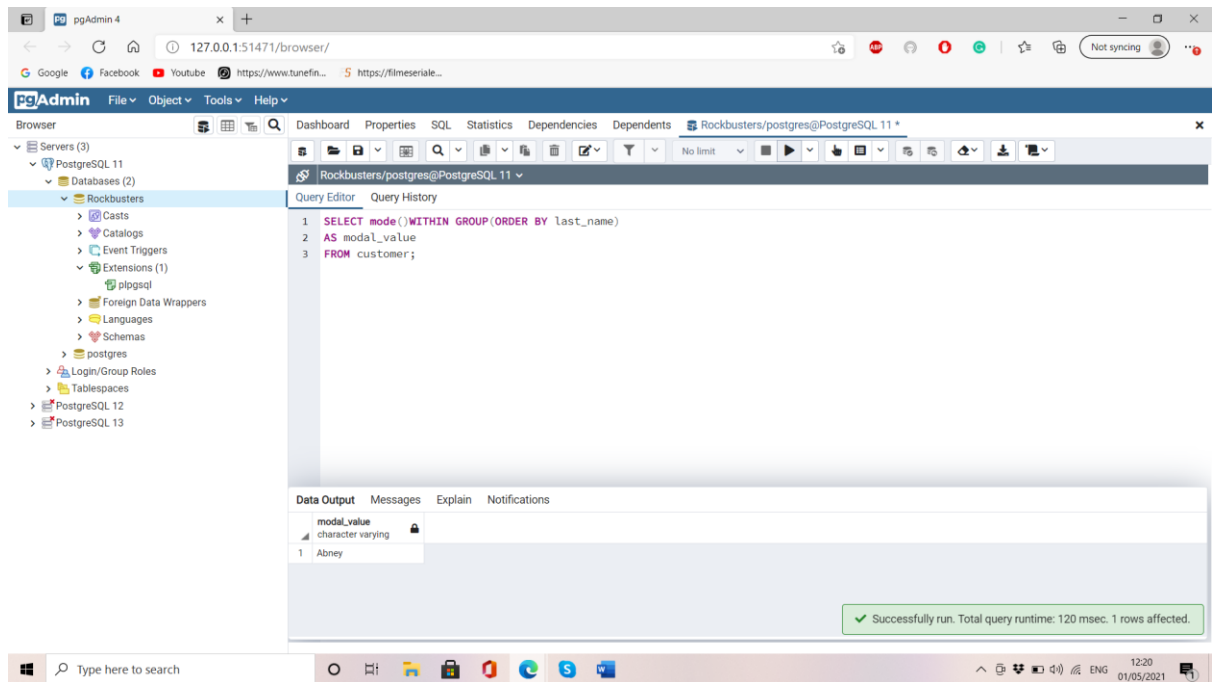
modal_value
integer
1

A screenshot of the pgAdmin 4 web interface. The browser window shows the URL 127.0.0.1:51471/browser/. The left sidebar shows the database structure: Servers (3) > PostgreSQL 11 > Databases (2) > Rockbusters. The main panel shows the 'Query Editor' for 'Rockbusters/postgres@PostgreSQL 11'. The query is:

```
1 SELECT mode()WITHIN GROUP(ORDER BY first_name)
2 AS modal_value
3 FROM customer;
```

 The 'Data Output' tab shows the result:

modal_value
character varying
Jamie



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.
-
- I think that the initial filtering and summarizing of data is easier in excel using pivot tables and filters however this is only the case if you are working with small data sets.
 - SQL is amazingly fast once you know the correct queries to use and is remains very fast when working with large datasets which is not the case with excel.
 - If I were only working with relatively small datasets, and no other user needed access to the data – I would use excel. Otherwise, SQL is the right choice.