SUBMITTED BY:
Grace Skelley on 11.4.2022

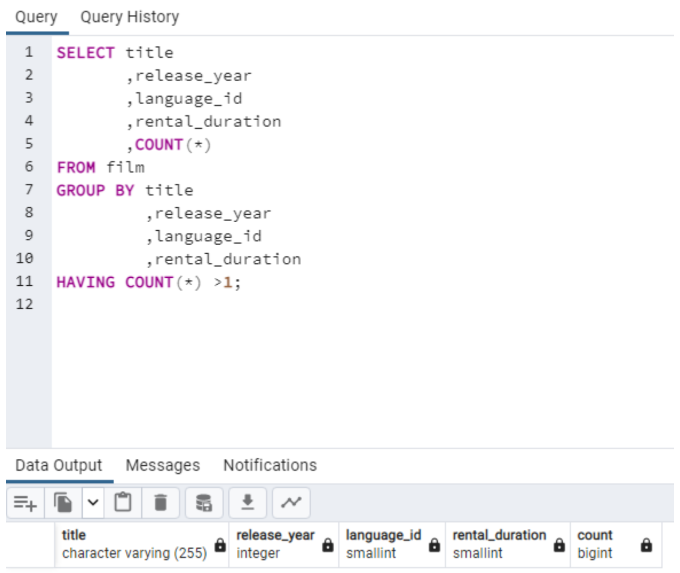
1. **Check for and clean dirty data:** Find out if the film table and the customer table contain any dirty data, specifically non-uniform or duplicate data, or missing values. Create a new "Answers 3.6" document and copy-paste your queries into it. Next to each query write 2 to 3 sentences explaining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

FILM TABLE | **CUSTOMER TABLE**

FINDING DUPLICATE

FILM TABLE

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


Query Query History

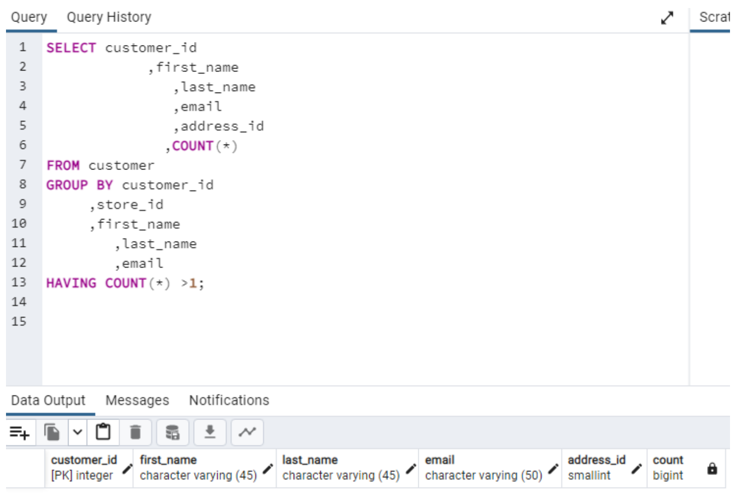
```sql
SELECT title
       ,release_year
       ,language_id
       ,rental_duration
       ,COUNT(*)
FROM film
GROUP BY title
         ,release_year
         ,language_id
         ,rental_duration
HAVING COUNT(*) >1;
```

Data Output Messages Notifications

title	release_year	language_id	rental_duration	count
character varying (255)	integer	smallint	smallint	bigint

CUSTOMER TABLE

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


Query Query History Scrat

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

Data Output Messages Notifications

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

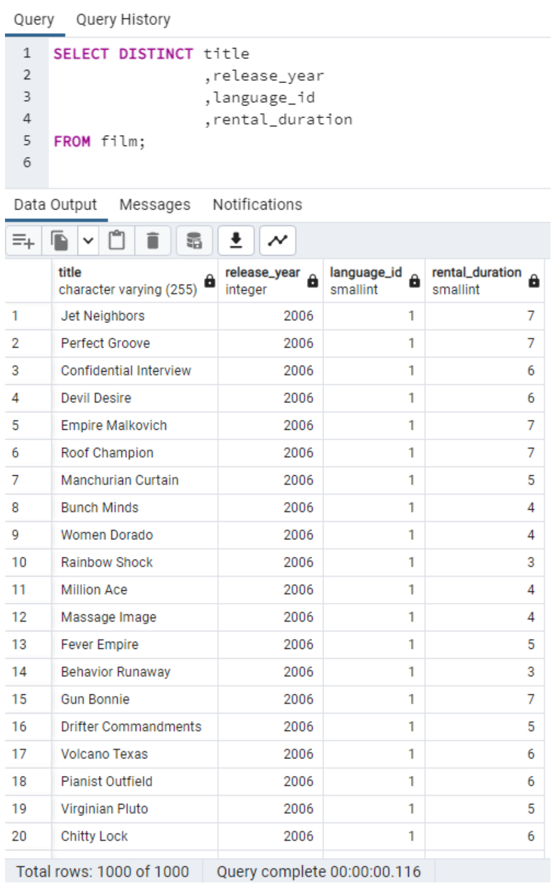
Scanning relevant records with the use of an aggregate function - `GROUP BY`, tallying all rows - `COUNT(*)` filtering by means of `HAVING COUNT (*) > 1`, the query return list has detected no duplicate records. Otherwise, identical values should be deleted from the database to maintain data consistency.

FILM TABLE | **CUSTOMER TABLE**

NON-UNIFORM

FILM TABLE

```
SELECT DISTINCT title
               ,release_year
               ,language_id
               ,rental_duration
FROM film;
```



Query | Query History

```
SELECT DISTINCT title
                ,release_year
                ,language_id
                ,rental_duration
FROM film;
```

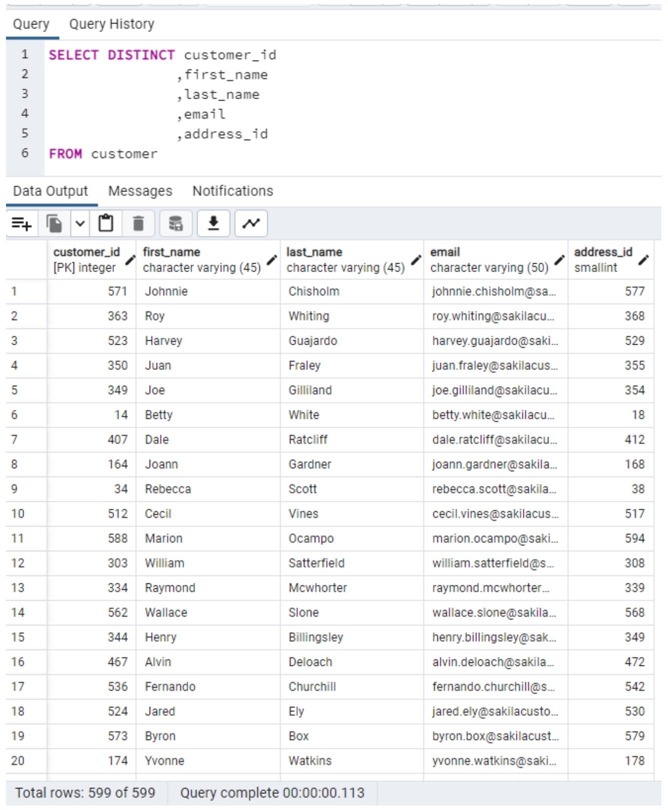
Data Output | Messages | Notifications

	title character varying (255)	release_year integer	language_id smallint	rental_duration smallint
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
16	Drifter Commandments	2006	1	5
17	Volcano Texas	2006	1	6
18	Pianist Outfield	2006	1	6
19	Virginian Pluto	2006	1	5
20	Chitty Lock	2006	1	6

Total rows: 1000 of 1000 Query complete 00:00:00.116

CUSTOMER TABLE

```
SELECT DISTINCT customer_id
               ,first_name
               ,last_name
               ,email
               ,address_id
FROM customer
```



Query | Query History

```
SELECT DISTINCT customer_id
                ,first_name
                ,last_name
                ,email
                ,address_id
FROM customer
```

Data Output | Messages | Notifications

	customer_id [PK] integer	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint
1	571	Johnnie	Chisholm	johnnie.chisholm@sa...	577
2	363	Roy	Whiting	roy.whiting@sakilacu...	368
3	523	Harvey	Guajardo	harvey.guajardo@saki...	529
4	350	Juan	Fraley	juan.fraley@sakilacus...	355
5	349	Joe	Gilliland	joe.gilliland@sakilacu...	354
6	14	Betty	White	betty.white@sakilacu...	18
7	407	Dale	Ratcliff	dale.ratcliff@sakilacu...	412
8	164	Joann	Gardner	joann.gardner@sakila...	168
9	34	Rebecca	Scott	rebecca.scott@sakila...	38
10	512	Cecil	Vines	cecil.vines@sakilacus...	517
11	588	Marion	Ocampo	marion.ocampo@saki...	594
12	303	William	Satterfield	william.satterfield@s...	308
13	334	Raymond	Mcwhorter	raymond.mcwhorter...	339
14	562	Wallace	Slone	wallace.slone@sakila...	568
15	344	Henry	Billingsley	henry.billingsley@sak...	349
16	467	Alvin	Deloach	alvin.deloach@sakila...	472
17	536	Fernando	Churchill	fernando.churchill@s...	542
18	524	Jared	Ely	jared.ely@sakilacusto...	530
19	573	Byron	Box	byron.box@sakilacust...	579
20	174	Yvonne	Watkins	yvonne.watkins@saki...	178

Total rows: 599 of 599 Query complete 00:00:00.113

The data output for `DISTINCT` syntax provides an overview of records that may exhibit random values. Data that show inconsistent structure must be fixed through `UPDATE` syntax to meet standard format.

FILM TABLE	CUSTOMER TABLE
MISSING VALUES	

If the majority of the records contain a missing value, null or automatically prefilled a default value instead, there are two ways to manage this matter:

- **IGNORE** - columns with a high percentage of missing values can be excluded from the query search

```
SELECT col1
       ,col2
       ,col4... --col3 ignored in select because it has a lot of missing values
FROM tablename
```

- **IMPUTE –** values can be statistically manipulated by calculating and filling in estimate values.

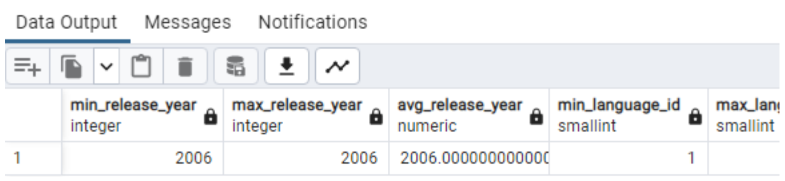
```
UPDATE tablename
SET = AVG(col1)
WHERE col1 IS NULL
```

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

NUMERICAL COLUMNS

```
SELECT MIN(release_year) AS min_release_year
       ,MAX(release_year) AS max_release_year
       ,AVG(release_year) AS avg_release_year

       ,MIN(language_id) AS min_language_id
       ,MAX(language_id) AS max_language_id
       ,AVG(language_id) AS avg_language_id

       ,MIN(rental_duration) AS min_rental_duration
       ,MAX(rental_duration) AS max_rental_duration
       ,AVG(rental_duration) AS avg_rental_duration

       ,MIN(rental_rate) AS min_rental_rate
       ,MAX(rental_rate) AS max_rental_rate
       ,AVG(rental_rate) AS avg_rental_rate

       ,MIN(length) AS min_length
       ,MAX(length) AS max_length
       ,AVG(length) AS avg_length

       ,MIN(replacement_cost) AS min_replecament_cost
       ,MAX(replacement_cost) AS max_replecament_cost
       ,AVG(replacement_cost) AS avg_replecament_cost

FROM film;
```

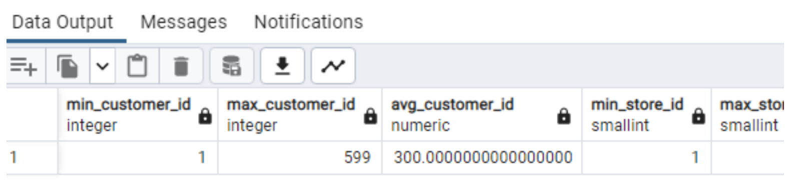


```
   min_release_year  max_release_year  avg_release_year    min_language_id  max_lan
   integer           integer           numeric             smallint         smallint
1              2006              2006  2006.00000000000                  1
```

CUSTOMER TABLE

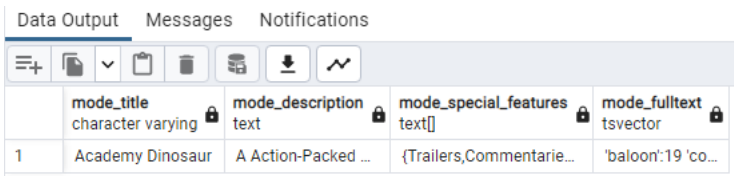
NUMERICAL COLUMNS

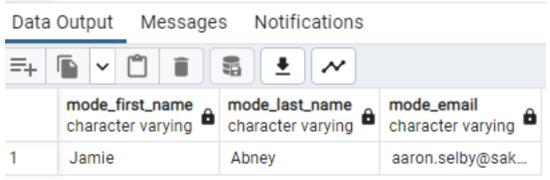
```
SELECT MIN(customer_id) AS min_customer_id
       ,MAX(customer_id) AS max_customer_id
       ,AVG(customer_id) AS avg_customer_id

       ,MIN(store_id) AS min_store_id
       ,MAX(store_id) AS max_store_id
       ,AVG(store_id) AS avg_store_id

       ,MIN(address_id) AS min_address_id
       ,MAX(address_id) AS max_address_id
       ,AVG(address_id) AS avg_address_id

       ,MIN(active) AS min_active
       ,MAX(active) AS max_active
       ,AVG(active) AS avg_active

FROM customer;
```



```
   min_customer_id  max_customer_id  avg_customer_id        min_store_id  max_sto
   integer          integer          numeric                smallint      smallint
1                1              599  300.0000000000000000              1
```

NON-NUMERICAL COLUMNS	

FILM TABLE	
	```
	SELECT MODE() WITHIN GROUP (ORDER BY title) AS mode_title
	       ,MODE() WITHIN GROUP (ORDER BY description) AS mode_description
	       ,MODE() WITHIN GROUP (ORDER BY special_features) AS mode_special_features
	       ,MODE() WITHIN GROUP (ORDER BY fulltext) AS mode_fulltext

	FROM film;
	```
	 

CUSTOMER TABLE	
	```
	SELECT MODE() WITHIN GROUP (ORDER BY first_name) AS mode_first_name
	       ,MODE() WITHIN GROUP (ORDER BY last_name) AS mode_last_name
	       ,MODE() WITHIN GROUP (ORDER BY email) AS mode_email

	FROM customer;
	```
	 

	mode_title character varying	mode_description text	mode_special_features text[]	mode_fulltext tsvector
1	Academy Dinosaur	A Action-Packed ...	{Trailers,Commentarie...	'baloon':19 'co...

	mode_first_name character varying	mode_last_name character varying	mode_email character varying
1	Jamie	Abney	aaron.selby@sak...

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.

   The functions, ease of use and speed are subjective to the database dimensions however both have advantages and flaws. Considering the Rockbuster database carries a large volume of data entries in this case, SQL would be the best platform for processing analytical insights. The table below identifies each platform's "PROs" and "CONS".

	FUNCTIONS	**EASE OF USE**	**SPEED**
EXCEL	Easy to execute but can cause a lot of human error as it involves keen attention to detail.	Requires multiple steps to achieve the desired output.	Processing time must be accounted for, to appropriately handle data points and develop some insights for analysis.
SQL	Functions allow minimal errors which can be helpful in most cases.	Must be familiar with formulating commands and scripts to efficiently run the query accurately.	Displays output almost immediately which saves time with high-quality results.