1.) Check For and Clean Dirty Data

Film Table
Duplicate Data

The above query returned no duplicate values, however, I could have used the CREATE VIEW query down below to pull up a view that would return any existing duplicate values:

Rockbuster/postgres@PostgreSQL 15

Query   Query History

```
1   CREATE VIEW film_duplicate_records AS
2   SELECT  film_id,
3           title,
4           description,
5           release_year,
6           language_id,
7           rental_duration,
8           rental_rate,
9           length,
10          replacement_cost,
11          rating,
12          last_update,
13          special_features,
14          fulltext
15  FROM film
16  GROUP BY film_id,
17          title,
18          description,
19          release_year,
20          language_id,
21          rental_duration,
22          rental_rate,
23          length,
24          replacement_cost,
25          rating,
26          last_update,
27          special_features,
28          fulltext
```

Data Output   Messages   Notifications

CREATE VIEW

Query returned successfully in 78 msec.

Non-Uniform Data

Looking through the only category that could have non-uniform entries for the same value (i.e. activebool), we find that there are no such entries. In fact, there were the same number of distinct values found as there total rows, as demonstrated in the two queries below.

Missing Data

This query did not return any null values in any of the selected columns, meaning there are no missing values.

Customer Table

Duplicate Data

The above query returned no duplicate values, however, I could have used the CREATE VIEW query down below to pull up a view that would return any existing duplicate values:

Non-Uniform Data

Looking through the only category that could have non-uniform entries for the same value (i.e. activebool), we find that there are no such entries. In fact, there were the same number of distinct values found as there total rows, as demonstrated in the two queries below.

Missing Data

This query did not return any null values in any of the selected columns, meaning there are no missing values.

Dashboard   Properties   SQL   Statistics   Dependencies   Dependents   Processes   Rockbuster/postgres@PostgreSQL 15*

Rockbuster/postgres@PostgreSQL 15

Query   Query History

```
 1  SELECT  customer_id,
 2          store_id,
 3          first_name,
 4          last_name,
 5          email,
 6          address_id,
 7          activebool,
 8          create_date,
 9          last_update,
10          active
11  FROM customer
12  WHERE   customer_id IS NULL OR
13          store_id IS NULL OR
14          first_name IS NULL OR
15          last_name IS NULL OR
16          email IS NULL OR
17          address_id IS NULL OR
18          activebool IS NULL OR
19          create_date IS NULL OR
20          last_update IS NULL OR
21          active IS NULL
22
```

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

2.) <u>Summarize Your Data</u>

Film Table


```sql
1  SELECT
2         MIN(film_id),
3         MAX(film_id),
4         AVG(film_id),
5
6  MODE () WITHIN GROUP (ORDER BY title)
7         AS title_mode,
8
9  MODE () WITHIN GROUP (ORDER BY title)
10        AS title_mode,
11
12        MIN(release_year),
13        MAX(release_year),
14        AVG(release_year),
15
16 MODE () WITHIN GROUP (ORDER BY language_id)
17        AS language_mode,
18
19        MIN(rental_duration),
20        MAX(rental_duration),
21        AVG(rental_duration),
22
23        MIN(rental_rate),
24        MAX(rental_rate),
25        AVG(rental_rate),
26
27        MIN(length),
28        MAX(length),
29        AVG(length),
30
31        MIN(replacement_cost),
32        MAX(replacement_cost),
33        AVG(replacement_cost),
34
35 MODE () WITHIN GROUP (ORDER BY rating)
36        AS rating_mode,
37
38        MIN(last_update),
39        MAX(last_update),
40
41 MODE () WITHIN GROUP (ORDER BY special_features)
42        AS special_features_mode,
43
44 MODE () WITHIN GROUP (ORDER BY fulltext)
45        AS fulltext_mode
46 FROM film;
```

min integer	max integer	avg numeric	title_mode character varying	title_mode character varying	min integer	max integer	avg numeric	language_mode smallint	min smallint	max smallint	avg numeric	min numeric	max numeric	avg numeric
1	1000	500.5	Academy Dinosaur	Academy Dinosaur	2006	2006	2006	1	3	7	4.985	0.99	4.99	2.98

min smallint	max smallint	avg numeric	min numeric	max numeric	avg numeric	rating_mode mpaa_rating	min timestamp without time zone	max timestamp without time zone	special_features_mode text[]	fulltext_mode tsvector
46	185	115.272	9.99	29.99	19.984	PG-13	2013-05-26 14:50:58.951	2013-05-26 14:50:58.951	{Trailers,Commentaries,"Behind the Scenes"}	'baloon':19 'confront':14 'documentari':5 'feminist':8,11,16 'mile':2 'must':13 'spi':1 'thri...

Postgre would not allow me to run the AVG() function on the the last_update field as it is a date function.

Customer Table

```sql
1  SELECT
2          MIN(customer_id),
3          MAX(customer_id),
4          AVG(customer_id),
5
6          MIN(store_id),
7          MAX(store_id),
8          AVG(store_id),
9
10 MODE () WITHIN GROUP (ORDER BY first_name)
11          AS mode_first_name,
12
13 MODE () WITHIN GROUP (ORDER BY last_name)
14          AS mode_last_name,
15
16 MODE () WITHIN GROUP (ORDER BY email)
17          AS mode_email,
18
19          MIN(address_id),
20          MAX(address_id),
21          AVG(address_id),
22
23 MODE () WITHIN GROUP (ORDER BY activebool)
24          AS mode_active,
25
26          MIN(create_date),
27          MAX(create_date),
28
29          MIN(last_update),
30          MAX(last_update)
31
32 FROM customer;
```

Data Output   Messages   Notifications

min integer	max integer	avg numeric	min smallint	max smallint	avg numeric	mode_first_name character varying	mode_last_name character varying	mode_email character varying	min smallint	max smallint
1	1	599	300	1	2	1.4557595993322203	Jamie	Abney	aaron.selby@sakilacustomer.org	5	605

avg numeric	mode_active boolean	min date	max date	min timestamp without time zone	max timestamp without time zone
304.7245409015025	true	2006-02-14	2006-02-14	2013-05-26 14:49:45.738	2013-05-26 14:49:45.738

Postgre would not allow me to run the AVG() function on the create_date and last_update fields as they are in date formats.

3.) Reflect

When adjusting for the learning curve as it relates to SQL functions, I found SQL to be a quicker method for profiling the data of these two, respective tables because I was able to write queries for both in the same window (i.e. the Postgre Query writing area) and I was able to create these queries without creating Pivot Tables, which would have required me to go into each file, separately and create unique Pivot Tables.