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 (Identifying Duplicates)

SELECT title, description, release_year, language_id, rental_duration, rental_rate, length,
replacement_cost, rating, last_update, special_features, COUNT (*)
FROM film
GROUP BY film_id, title, description, release_year, language_id, rental_duration, rental_rate, length,
replacement_cost, rating, last_update, special_features
HAVING COUNT (*) >1

Query Query History Scratch Pad

```
SELECT title, description, release_year, language_id, rental_duration, rental_rate, length,
replacement_cost, rating, last_update, special_features, COUNT (*)
FROM film
GROUP BY film_id, title, description, release_year, language_id, rental_duration, rental_rate, length,
replacement_cost, rating, last_update, special_features
HAVING COUNT (*) >1
```

Data Output Messages Notifications

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)	rating mpaa_rating	last_update timestamp without time zone	special_features text[]	count bigint

Customer Table (Identifying Duplicates)

SELECT store_id, first_name, last_name, email, address_id, activebool, active,
COUNT(*)
FROM customer
GROUP BY customer_id, store_id, first_name, last_name, email, address_id, activebool, active
HAVING COUNT (*) >1

Query Query History

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

Data Output Messages Notifications

store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	active integer	count bigint

The film table and customer table thankfully do not have any duplicates, but if there were any, I would first identify and confirm it is indeed a duplicate (or dirty in general), then utilize the ALTER function in SQL to adjust it to conform with the existing items, or I could DELETE the whole record altogether – this is assuming I have editing permissions as an analyst. If not, I would flag the duplicate / dirty data for IT or the data engineers to fix for me. In the meantime, I could use the VIEW feature to adjust what I need to adjust to perform my analysis.

Film Table (Non-Uniform Data)

SELECT DISTINCT title, description, release_year, language_id, rental_duration, rental_rate, length, replacement_cost, rating, last_update, special_features
FROM film;

Query | Query History | Scratch Pad

```
SELECT DISTINCT title, description, release_year, language_id, rental_duration, rental_rate,
length, replacement_cost, rating, last_update, special_features
FROM film;
```

Data Output | Messages | Notifications

	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)
1	Rocketeer Mother	A Awe-Inspiring Character Study of a Robot And a Sumo Wrestler who must Discover a Womanizer in A Shark Tank	2006	1	3	0.99	178	27.
2	Holiday Games	A Insightful Reflection of a Waitress And a Madman who must Pursue a Boy in Ancient Japan	2006	1	7	4.99	78	10.
3	Daddy Pittsburgh	A Epic Story of a A Shark And a Student who must Confront a Explorer in The Gulf of Mexico	2006	1	5	4.99	161	26.
4	Mannequin Worst	A Astounding Saga of a Mad Cow And a Pastry Chef who must Discover a Husband in Ancient India	2006	1	3	2.99	71	18.
5	Everyone Craft	A Fateful Display of a Waitress And a Dentist who must Reach a Butler in Nigeria	2006	1	4	0.99	163	29.
6	Gump Date	A Intrepid Yarn of a Explorer And a Student who must Kill a Husband in An Abandoned Mine Shaft	2006	1	3	4.99	53	12.
7	Stranger Strangers	A Awe-Inspiring Yarn of a Womanizer And a Explorer who must Fight a Woman in The First Manned Space Station	2006	1	3	4.99	139	12.
8	Fever Empire	A Insightful Panorama of a Cat And a Boat who must Defeat a Boat in The Gulf of Mexico	2006	1	5	4.99	158	20.
9	Excitement Eve	A Brilliant Documentary of a Monkey And a Car who must Conquer a Crocodile in A Shark Tank	2006	1	3	0.99	51	20.
10	Wild Apollo	A Beautiful Story of a Monkey And a Sumo Wrestler who must Conquer a A Shark in A MySQL Convention	2006	1	4	0.99	181	24.
11	Island Exorcist	A Fanciful Panorama of a Technical Writer And a Boy who must Find a Dentist in An Abandoned Fun House	2006	1	7	2.99	84	23.
12	Kramer Chocolate	A Amazing Yarn of a Robot And a Pastry Chef who must Redeem a Mad Scientist in The Outback	2006	1	3	2.99	171	24.
13	Troopers Metal	A Fanciful Drama of a Monkey And a Feminist who must Sink a Man in Berlin	2006	1	3	0.99	115	20.
14	Orange Grapes	A Astounding Documentary of a Butler And a Womanizer who must Face a Dog in A U-Boat	2006	1	4	0.99	76	21.

SELECT DISTINT rating
FROM film
GROUP BY rating

Query | Query History

```
SELECT DISTINCT rating
FROM film
GROUP by rating
```

Data Output | Messages | Notificatio

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

Customer Table (Non-Uniform Data)

SELECT DISTINCT store_id, first_name, last_name, email, address_id, activebool, active
FROM customer;

Query Query History

```
SELECT DISTINCT store_id, first_name, last_name, email, address_id, activebool, active
FROM customer;
```

Data Output Messages Notifications

	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	active integer
1	1	Leslie	Gordon	leslie.gordon@sakilacustomer.org	147	true	1
2	1	Florence	Woods	florence.woods@sakilacustomer.org	111	true	1
3	1	Martin	Bales	martin.bales@sakilacustomer.org	388	true	1
4	1	Wendy	Harrison	wendy.harrison@sakilacustomer.org	119	true	1
5	1	Rafael	Abney	rafael.abney@sakilacustomer.org	510	true	1
6	1	Gabriel	Harder	gabriel.harder@sakilacustomer.org	524	true	1
7	2	Joel	Francisco	joel.francisco@sakilacustomer.org	436	true	1
8	1	Gene	Sanborn	gene.sanborn@sakilacustomer.org	503	true	1
9	1	Tanya	Gilbert	tanya.gilbert@sakilacustomer.org	241	true	1
10	2	Jeanne	Lawson	jeanne.lawson@sakilacustomer.org	204	true	1
11	2	Fernando	Churchill	fernando.churchill@sakilacustomer.org	542	true	1
12	2	Jesus	Mccartney	jesus.mccartney@sakilacustomer.org	432	true	1
13	1	Diane	Collins	diane.collins@sakilacustomer.org	54	true	1
14	2	Warren	Sherrod	warren.sherrod@sakilacustomer.org	467	true	1
15	1	Laura	Rodriguez	laura.rodriguez@sakilacustomer.org	26	true	1
16	1	Wallace	Slone	wallace.slone@sakilacustomer.org	568	true	1

SELECT DISTINCT activebool
FROM customer
GROUP BY activebool

Query Query History

```
SELECT DISTINCT activebool
FROM customer
GROUP BY activebool
```

Data Output Messages Notifications

	activebool boolean
1	true

Using SELECT DISTINT instead of SELECT would provide us with all unique data points, so we can quickly identify any records that aren't uniformed with the others. I first ran it for all of the columns in the respective tables, then picked a column I thought may have non-uniformed records to test them out. Again, thankfully there aren't any non-uniformed records, but if there were, once confirmed, I can use the ALTER or UPDATE function to fix them assuming I have editing permissions. If not, I would flag them and report them to IT and/or the data engineer(s).

Film Table (Missing Values)

```
SELECT title, description, language_id, release_year
FROM film
WHERE title
IS NULL;
```

Customer Table (Missing Values)

```
SELECT first_name, last_name, email
FROM customer
WHERE email
IS NULL;
```

I ran this portion multiple times changing the portion after WHERE to different columns to see if any showed up. Thankfully, like before, there aren't any missing values. If there were, it'd be harder to fix than other types of dirty data since we can't be completely sure what was supposed to go in there. In this very particular example, we may be able to deduce someone's missing first or last name from their email if they happen to use their first and last name in their email address; however, not every column is going to be as easy to deduce from. As analysts, we could remove the records with missing data from our analysis or count them as "N/A" and follow-up with the respective teams to try to get the missing data filled in.

Query Query History

```
1 SELECT title, description, language_id, release_year
2 FROM film
3 WHERE title
4 IS NULL;
5
```

Data Output Messages Notifications

title character varying (255)	description text	language_id smallint	release_year integer

Query Query History

```
1 SELECT first_name, last_name, email
2 FROM customer
3 WHERE email
4 IS NULL;
5
```

Data Output Messages Notifications

first_name character varying (45)	last_name character varying (45)	email character varying (50)

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

```
SELECT
MIN (film_id) AS min_film_id,
MAX (film_id) AS max_film_id,
AVG (film_id) AS avg_film_id,
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_replacement_cost,
MAX (replacement_cost) AS max_replacement_cost,
AVG (replacement_cost) AS avg_replacement_cost,
MODE () WITHIN GROUP (ORDER BY title) AS mode_title,
MODE () WITHIN GROUP (ORDER BY description) AS mode_description,
MODE () WITHIN GROUP (ORDER BY rating) AS mode_rating,
MODE () WITHIN GROUP (ORDER BY special_features) AS mode_special_features,
MODE () WITHIN GROUP (ORDER BY fulltext) AS mode_fulltext
FROM film;
```

Query Query History

```
SELECT
MIN (film_id) AS min_film_id,
MAX (film_id) AS max_film_id,
AVG (film_id) AS avg_film_id,
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,
```

Scratch Pad

Data Output Messages Notifications

	min_film_id integer	max_film_id integer	avg_film_id numeric	min_release_year integer	max_release_year integer	avg_release_year numeric	min_language_id smallint	max_language_id smallint	avg_language_id numeric	min_rental_duration smallint	max_rental_duration smallint	avg_rental_dur numeric
1	1	1000	500.5000000000000000	2006	2006	2006.0000000000000000	1	1	1.00000000000000000000	3	7	4.985000000

Customer Table

```
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,
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,
MODE () WITHIN GROUP (ORDER BY active) AS mode_active,
MODE () WITHIN GROUP (ORDER BY activebool) AS mode_activebool
FROM customer;
```

Query Query History Scratch Pad

```sql
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,
MODE () WITHIN GROUP (ORDER BY first_name) AS mode_first_name,
MODE () WITHIN GROUP (ORDER BY last_name) AS mode_last_name,
```

Data Output Messages Notifications

	min_customer_id integer	max_customer_id integer	avg_customer_id numeric	min_store_id smallint	max_store_id smallint	avg_store_id numeric	min_address_id smallint	max_address_id smallint	avg_address_id numeric	min_active integer	max_active integer	avg_active numeric	mode_fir characte
1	1	599	300.0000000000000000	1	2	1.4557595993322204	5	605	304.7245409015025042	0	1	0.97495826377295492487	Jamie

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 I'm still more comfortable using Excel, it was easier for me to perform the data profiling using it versus SQL. It took me a bit to manually type out the SQL to accomplish what I needed, but once I had it laid out, I know I have it ready for the next time. Unless I'm using Excel VBA, SQL would be the quicker option for repetition versus manually doing it over and over in Excel. Once the SQL is written out, pressing the play/run button would be the quicker option versus writing out the respective formulas in Excel and flash-filling the rest of the column. Currently, I'm still on the fence between the two options. I can see that SQL would be the quicker option, especially with bigger datasets with a lot more columns, but as I'm still learning it, it's taking me much longer to type everything out, have it in the correct order, and becoming fluent in it as I am in Excel.