Answers 3.6

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).

```
--looking for non-uniform values
SELECT DISTINCT rating
FROM film
GROUP BY rating
```

- Values could be updated to make them uniform. Columns could also be assigned a data type and other constraints to ensure consistency when new records are being added.

```
--looking for duplicate values
SELECT title,
       release_year,
       language_id,
       rental_duration,
       COUNT(*)
FROM film
GROUP BY title,
         release_year,
         language_id,
         rental_duration
HAVING COUNT(*) >1;
--
SELECT first_name,
       last_name,
       address_id,
       email,
       active,
       COUNT(*)
FROM customer
GROUP BY first_name,
       last_name,
       address_id,
       email,
       active
HAVING COUNT(*) >1;
```

- Duplicate values can either be hidden using the view option or deleted. Alternatively, the query can be limited to distinct values if neither of the other 2 options are available.

```
--looking for missing values
SELECT
COUNT(title) AS count_title,
COUNT(rental_duration) AS count_rental_duration,
COUNT(rental_rate) AS count_rental_rate,
COUNT(length) AS count_length,
COUNT(replacement_cost) AS count_replacement_cost,
    COUNT(*) AS count_rows
FROM film;
--
SELECT
COUNT(email) AS count_email,
COUNT(last_name) AS count_last_name,
COUNT(first_name) AS count_first_name,
COUNT(customer_id) AS count_customer_id,
COUNT(store_id) AS count_store_id,
```

```
      COUNT(*) AS count_rows
FROM customer;
```

- Missing numerical values could be added using the column average or the entire column could be omitted from the query.

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.

```
--descriptive statistics for numerical columns for film table
SELECT MIN(rental_rate) AS min_renatl_rate,
MAX(rental_rate) AS max_rental_rate,
AVG(rental_rate) AS avg_renatal_rate,
MIN(rental_duration) AS min_rental_duration,
MAX(rental_duration) AS max_rental_duration,
AVG(rental_duration) AS avg_rental_duration,
MIN(film_id) AS min_film,
MAX(film_id) AS max_film,
AVG(film_id) AS avg_film,
MIN(language_id) AS min_language,
MAX(language_id) AS max_language,
AVG(language_id) AS avg_language,
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
FROM film
```

min_renatl_rate	max_rental_rate	avg_renatal_rate	min_rental_duration	max_rental_duration	avg_rental_duration
0.99	4.99	2.98	3	7	4.985

min_film	max_film	avg_film	min_language	max_language	avg_language
1	1000	500.5	1	1	1

min_length	max_length	avg_length	min_replacement_cost	max_replacement_cost	avg_replacement_cost
46	185	115.272	9.99	29.99	19.984

```
--descriptive statistics for numerical columns for customer table
SELECT MIN(active) AS min_active,
MAX(active) AS max_active,
AVG(active) AS avg_active,
MIN(address_id) AS min_address,
MAX(address_id) AS max_address,
AVG(address_id) AS avg_address,
MIN(customer_id) AS min_customer,
MAX(customer_id) AS max_customer,
AVG(customer_id) AS avg_customer,
MIN(store_id) AS min_store,
MAX(store_id) AS max_store,
AVG(store_id) AS avg_store
FROM customer;
```

min_active	max_active	avg_active	min_address	max_address	avg_address
0	1	0.974958	5	605	304.7245

min_customer	max_customer	avg_customer	min_store	max_store	avg_store

min_customer	max_customer	avg_customer	min_store	max_store	avg_store
1	599	300	1	2	1.45576

```
--mode value for non-numerical columns for film table
SELECT mode() WITHIN GROUP (ORDER BY rating)
       AS rating_value,
     mode() WITHIN GROUP (ORDER BY special_features)
     AS Feature_value,
     mode() WITHIN GROUP (ORDER BY release_year)
     AS year_value,
     mode() WITHIN GROUP (ORDER BY title)
     AS title_value
FROM film;
```

rating_value	feature_value	year_value	title_value
PG-13	{Trailers,Commentaries,"Behind the Scenes"}	2006	Academy Dinosaur

```
--mode value for non-numerical columns for customer table
SELECT mode() WITHIN GROUP (ORDER BY first_name)
       AS first_name_value,
     mode() WITHIN GROUP (ORDER BY last_name)
     AS last_name_value,
     mode() WITHIN GROUP (ORDER BY email)
     AS email_value
FROM customer;
```

first_name_value	last_name_value	email_value
Jamie	Abney	aaron.selby@sakilacustomer.org

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.
    - SQL seems much better suited for data profiling, even when working with a smaller data set in Excel it would take longer to compile these as there are multiple steps involved in the process. With SQL once the code has been written it can be applied time and again without much effort.