ANSWERS 3.6

1. CHECK FOR AND CLEAN DIRTY DATA

Below are SQL queries to check for dirty data in the film and customer tables, along with explanations on how the data would be cleaned.

FILM TABLE DUPLICATE CHECK

```
SELECT film_id, 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;
```

This query identifies duplicate rows in the 'film' table. If duplicates are found, I would keep one occurrence and delete the others to ensure data integrity.

CUSTOMER TABLE DUPLICATE CHECK

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

This query identifies duplicate rows in the 'customer' table. If duplicates are found, I would review the data and remove redundant rows, keeping the most recent or accurate entries.

FILM TABLE MISSING DATA CHECK

```
SELECT *
FROM film
```

WHERE title IS NULL OR description IS NULL OR release_year IS NULL OR language_id IS NULL;

This query checks for missing values in critical columns of the 'film' table. If missing values are found, I would try to retrieve the correct data or replace it with appropriate defaults (e.g., 'Unknown' for text fields or 0 for numerical fields).

CUSTOMER TABLE MISSING DATA CHECK

*SELECT **
FROM customer
WHERE first_name IS NULL OR last_name IS NULL OR email IS NULL OR active IS NULL;

This query checks for missing values in critical columns of the 'customer' table. If missing values are found, I would reach out to retrieve the missing data from external sources or decide whether the incomplete rows should be removed.

2. SUMMARIZE YOUR DATA

Below are SQL queries used to calculate descriptive statistics for both the film and customer tables.

FILM TABLE DESCRIPTIVE STATISTICS

SELECT
COUNT(*) AS count_rows,
MIN(release_year) AS min_release_year,
MAX(release_year) AS max_release_year,
ROUND(AVG(release_year)) AS avg_release_year,
COUNT(release_year) AS count_release_year_values,
MIN(rental_duration) AS min_rental_duration,
MAX(rental_duration) AS max_rental_duration,
ROUND(AVG(rental_duration)) AS avg_rental_duration,
COUNT(rental_duration) AS count_rental_duration_values,
MIN(length) AS min_length,
MAX(length) AS max_length,
ROUND(AVG(length)) AS avg_length,
COUNT(length) AS count_length_values,
MIN(replacement_cost) AS min_replacement_cost,
MAX(replacement_cost) AS max_replacement_cost,

```
    ROUND(AVG(replacement_cost)) AS avg_replacement_cost,
    COUNT(replacement_cost) AS count_replacement_cost_values,
    MIN(rental_rate) AS min_rent,
    MAX(rental_rate) AS max_rent,
    ROUND(AVG(rental_rate)) AS avg_rent,
    COUNT(rental_rate) AS count_rent_values,
    MODE() WITHIN GROUP (ORDER BY rating) AS modal_rating_value,
    MODE() WITHIN GROUP (ORDER BY special_features) AS
modal_special_features_value,
    MODE() WITHIN GROUP (ORDER BY fulltext) AS modal_fulltext_value
FROM film;
```

This query calculates various descriptive statistics for the 'film' table, including:
- Numerical columns: Minimum, maximum, average, and the count of non-NULL values for columns such as `release_year`, `rental_duration`, `length`, `replacement_cost`, and `rental_rate`.
- Categorical columns: Mode (most common value) for `rating`, `special_features`, and `fulltext`.
These results provide insights into the distribution of film data, such as the typical rental duration, replacement cost, and the most common film ratings or features.

CUSTOMER TABLE DESCRIPTIVE STATISTICS

```
SELECT
    COUNT(*) AS count_rows,
    MODE() WITHIN GROUP (ORDER BY store_id) AS modal_store_id_value,
    MODE() WITHIN GROUP (ORDER BY first_name) AS modal_first_name_value,
    MODE() WITHIN GROUP (ORDER BY last_name) AS modal_last_name_value,
    MODE() WITHIN GROUP (ORDER BY email) AS modal_email_value,
    MODE() WITHIN GROUP (ORDER BY activebool) AS modal_activebool_value,
    MODE() WITHIN GROUP (ORDER BY create_date) AS modal_create_date_value,
    MODE() WITHIN GROUP (ORDER BY last_update) AS modal_last_update_value
FROM customer;
```

This query calculates descriptive statistics for the 'customer' table, including:
- Total rows: The total number of records in the `customer` table.
- Mode values: The most frequently occurring values for columns like `store_id`, `first_name`, `last_name`, `email`, `activebool`, `create_date`, and `last_update`.
These statistics provide an overview of customer data, such as the most common store ID or frequent customer names and whether the majority of customers are active.

3. REFLECT ON YOUR WORK

Based on my previous experience, both Excel and SQL are valuable tools for data profiling, but they serve different purposes. Excel is more intuitive and user-friendly, especially for smaller datasets or visualizing trends and patterns quickly through features like pivot tables. However, it becomes cumbersome for large datasets or complex operations.

SQL, on the other hand, is far more efficient for handling large-scale data and complex queries. It allows for precise and automated operations like filtering, grouping, and joining multiple tables with ease. While SQL has a steeper learning curve, its scalability and consistency make it the better choice for large datasets or data stored in relational databases.

In conclusion, for quick, small-scale profiling tasks, Excel might be more practical, but for robust, large-scale analysis, SQL is the better tool.