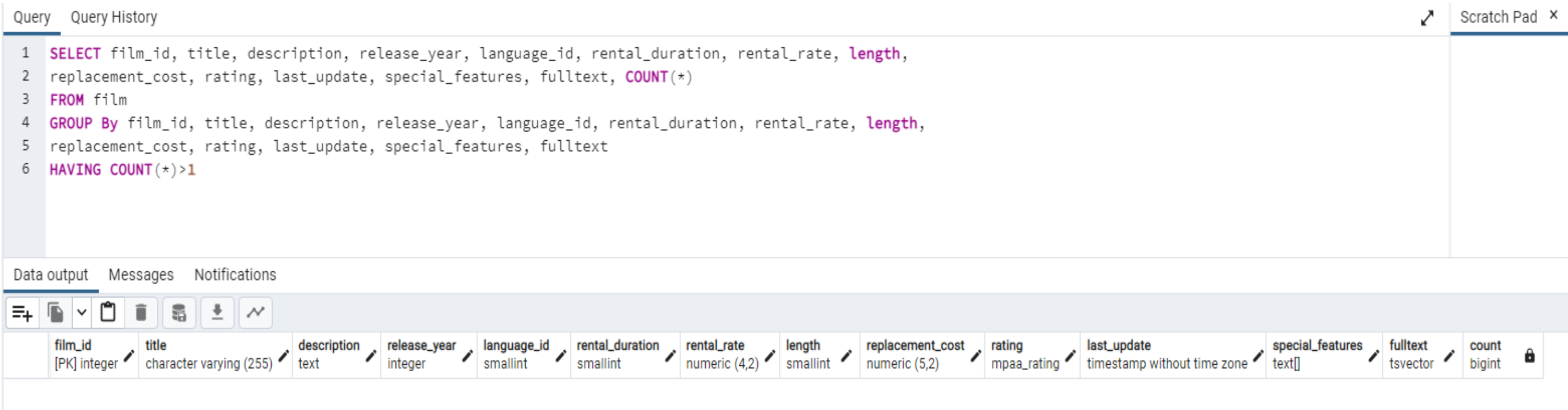
## 3.6 Summarizing & Cleaning Data Task

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).

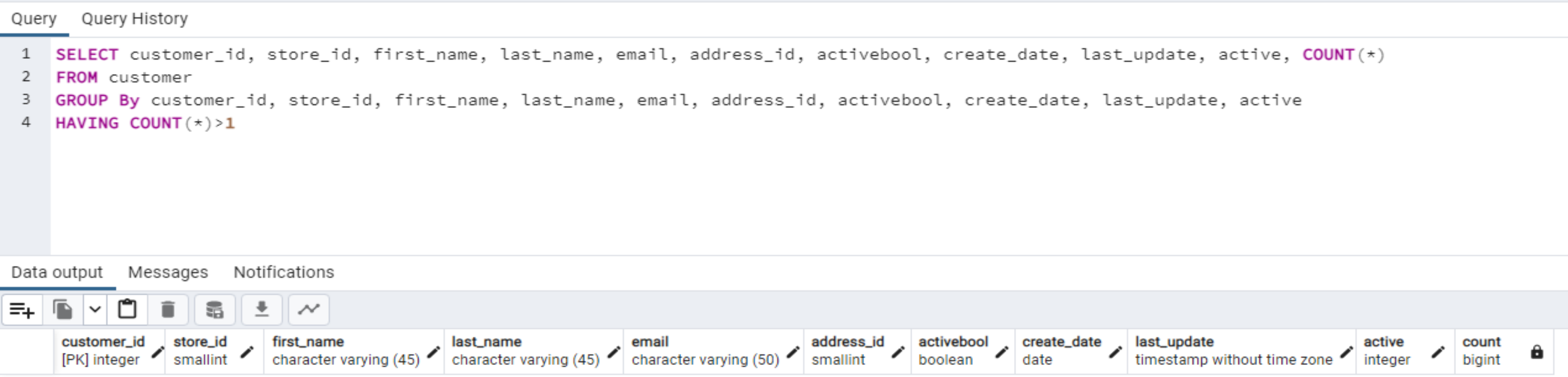
- Duplicates – None found, if found duplicate data can be cleaned by creating a view with unique records, or duplicate records can be deleted.
  - ○ Film Table



```sql
SELECT film_id, title, description, release_year, language_id, rental_duration, rental_rate, length,
replacement_cost, rating, last_update, special_features, fulltext, 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, fulltext
HAVING COUNT(*)>1
```

| film_id<br>[PK] integer | title<br>character varying (255) | description<br>text | release_year<br>integer | language_id<br>smallint | rental_duration<br>smallint | rental_rate<br>numeric (4,2) | length<br>smallint | replacement_cost<br>numeric (5,2) | rating<br>mpaa_rating | last_update<br>timestamp without time zone | special_features<br>text[] | fulltext<br>tsvector | count<br>bigint |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

  - ○ Customer Table

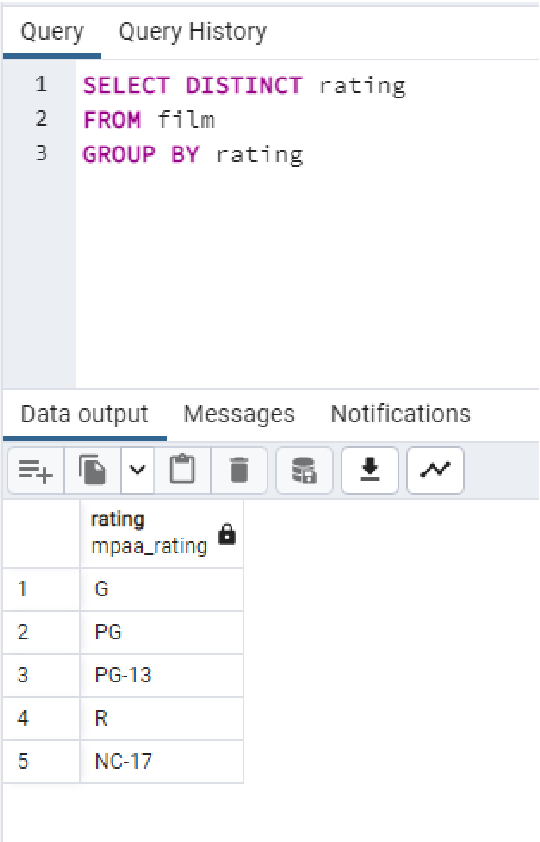


```sql
SELECT customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, last_update, active, COUNT(*)
FROM customer
GROUP By customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, last_update, active
HAVING COUNT(*)>1
```

| customer_id<br>[PK] integer | store_id<br>smallint | first_name<br>character varying (45) | last_name<br>character varying (45) | email<br>character varying (50) | address_id<br>smallint | activebool<br>boolean | create_date<br>date | last_update<br>timestamp without time zone | active<br>integer | count<br>bigint |
|---|---|---|---|---|---|---|---|---|---|---|

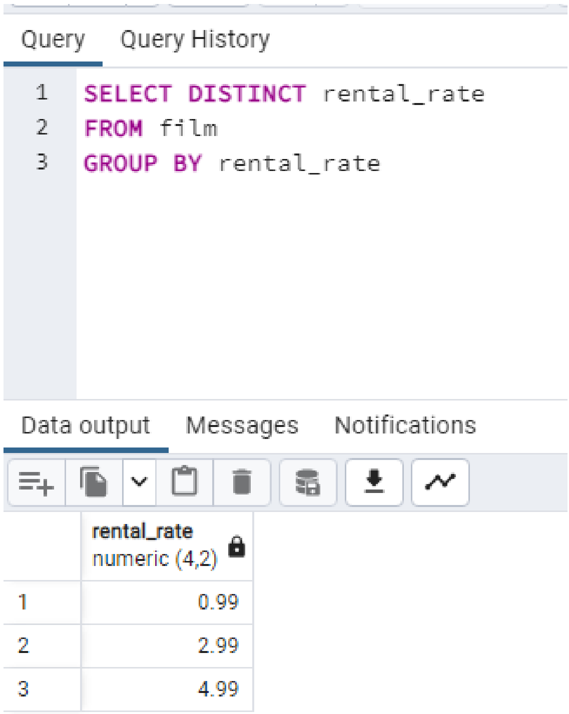
- Non-Uniform Values – None found, if found while searching through a few random values to check for inconsistencies then the record can be verified and updated to match similar records

- Film Table



```
SELECT DISTINCT rating
FROM film
GROUP BY rating
```

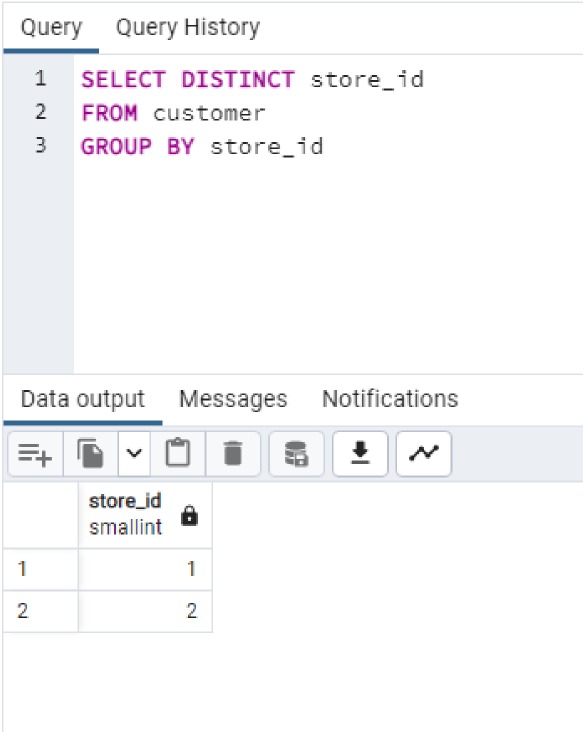
| | rating<br>mpaa_rating |
|---|---|
| 1 | G |
| 2 | PG |
| 3 | PG-13 |
| 4 | R |
| 5 | NC-17 |



```
SELECT DISTINCT rental_rate
FROM film
GROUP BY rental_rate
```

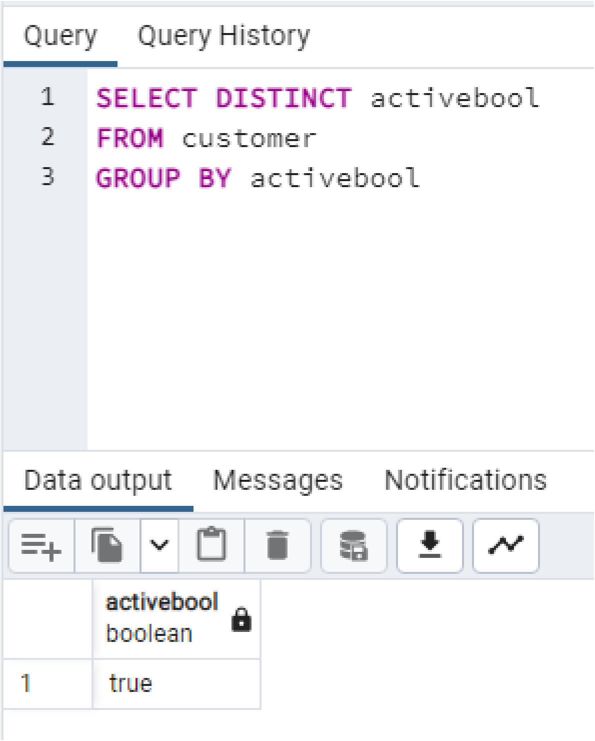
| | rental_rate<br>numeric (4,2) |
|---|---|
| 1 | 0.99 |
| 2 | 2.99 |
| 3 | 4.99 |

- Customer Table



```
SELECT DISTINCT store_id
FROM customer
GROUP BY store_id
```

| | store_id<br>smallint |
|---|---|
| 1 | 1 |
| 2 | 2 |

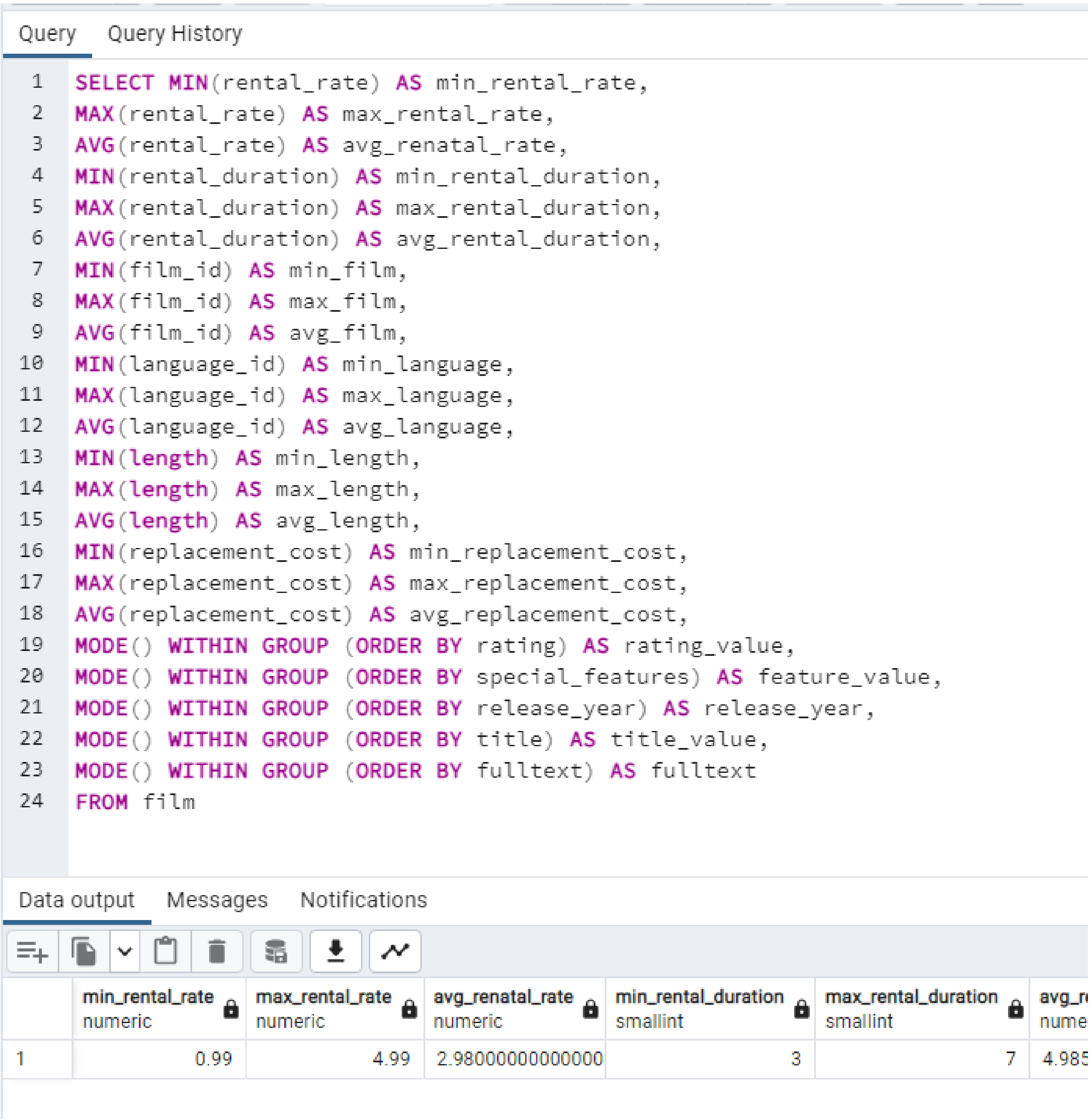


```
SELECT DISTINCT activebool
FROM customer
GROUP BY activebool
```

| | activebool<br>boolean |
|---|---|
| 1 | true |

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.

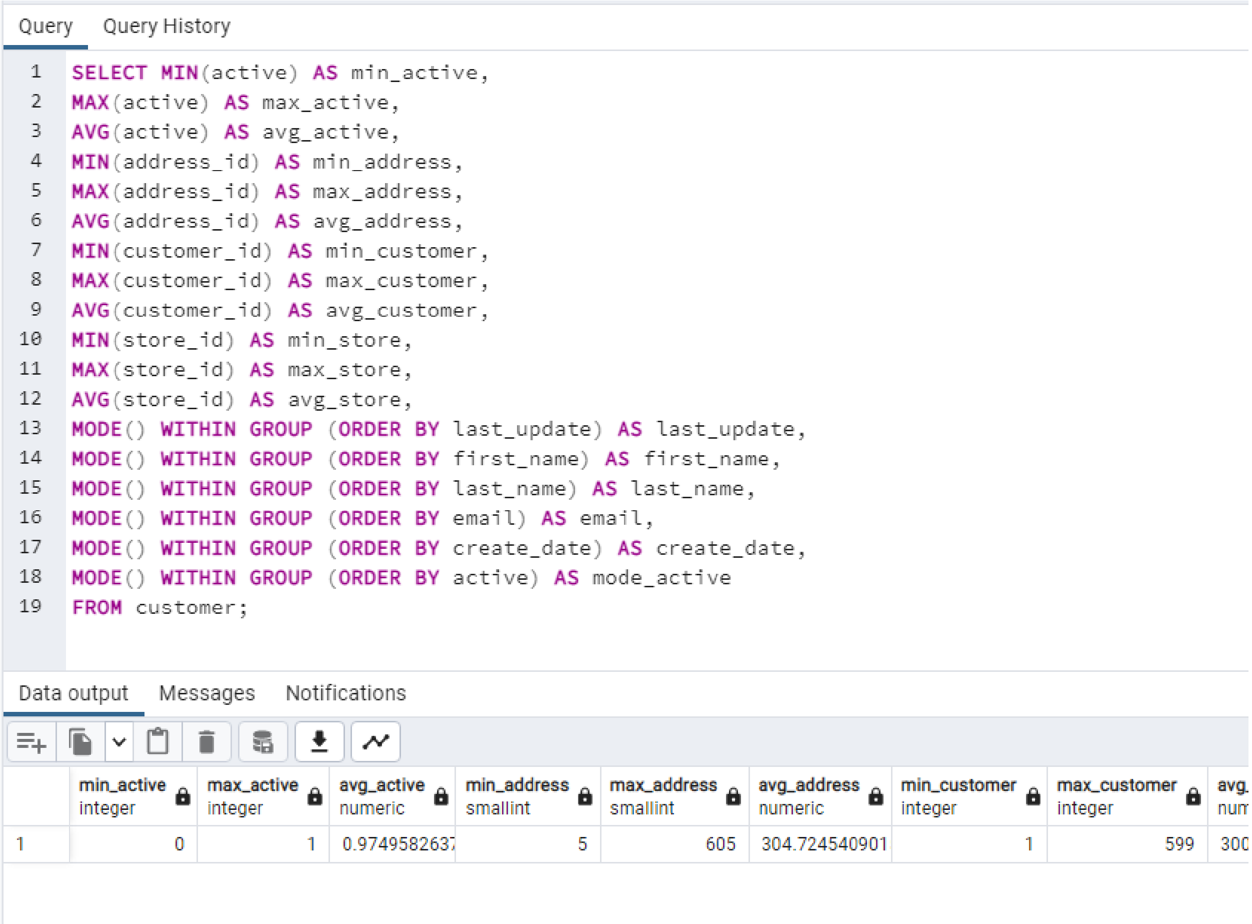
- Film Table



```sql
SELECT MIN(rental_rate) AS min_rental_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,
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 release_year,
MODE() WITHIN GROUP (ORDER BY title) AS title_value,
MODE() WITHIN GROUP (ORDER BY fulltext) AS fulltext
FROM film
```

| | min_rental_rate<br>numeric | max_rental_rate<br>numeric | avg_renatal_rate<br>numeric | min_rental_duration<br>smallint | max_rental_duration<br>smallint | avg_r<br>nume |
|---|---|---|---|---|---|---|
| 1 | 0.99 | 4.99 | 2.98000000000000 | 3 | 7 | 4.985 |

- Customer Table



```sql
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,
MODE() WITHIN GROUP (ORDER BY last_update) AS last_update,
MODE() WITHIN GROUP (ORDER BY first_name) AS first_name,
MODE() WITHIN GROUP (ORDER BY last_name) AS last_name,
MODE() WITHIN GROUP (ORDER BY email) AS email,
MODE() WITHIN GROUP (ORDER BY create_date) AS create_date,
MODE() WITHIN GROUP (ORDER BY active) AS mode_active
FROM customer;
```

| | min_active<br>integer | max_active<br>integer | avg_active<br>numeric | min_address<br>smallint | max_address<br>smallint | avg_address<br>numeric | min_customer<br>integer | max_customer<br>integer | avg<br>num |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0.9749582637 | 5 | 605 | 304.724540901 | 1 | 599 | 300 |

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 is more effective than Excel at data profiling when the data set is very large, and when the data is housed in a shared storage method. With smaller data sets, Excel and SQL are both proficient in data profiling but sharing and collaboration would be hindered with excel. SQL's language allows analysts to efficiently process, clean, and analyze data in a streamlined wat making it overall more effective than Excel.

4. Save your "Answers 3.6" document as a PDF and upload it here for your tutor to review.