# **Answers**- Summarizing & Cleaning

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.

**Finding non-uniform data**

**SELECT DISTINCT rating**

**FROM film**

**GROUP BY rating**

**Graphical user interface, text, application

Description automatically generated**

* Review a few random values to check for inconsistencies using the GROUP BY and DISTINCT keywords.

**Finding Duplicates in film table**

SELECT title,

release\_year,

language\_id,

rental\_duration,

COUNT(\*)

FROM film

GROUP BY title,

release\_year,

language\_id,

rental\_duration

HAVING COUNT(\*) >1

Graphical user interface, text, application

Description automatically generated

**Finding Duplicates in Customer table**

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;

**Graphical user interface, text, application

Description automatically generated**

* **Duplicate – There are 2 options.** Create a virtual table, known as a “view,” where you select only unique records or delete the duplicate record from the table or view (not advised).

**Finding Missing Values in film table**

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;

**Graphical user interface, text, application

Description automatically generated**

**Finding Missing Values in customer table**

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,

Answers 3.6 2

COUNT(\*) AS count\_rows

FROM customer;

**Graphical user interface, text, application

Description automatically generated**

* Missing Values - We can either ignore columns with a high percentage of missing values or *impute* missing values with a column average: Ex.

imputing missing values with the AVG value

UPDATE tablename

SET = AVG(col1)

WHERE col1 IS NULL

1. **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.

**Descriptive Stats for film table (numerical columns)**

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

Graphical user interface, application, Word

Description automatically generated

**Mode value for film table (non-numerical)**

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;

Graphical user interface, text, application, email

Description automatically generated

**Descriptive Stats for customer table(numerical columns)**

**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;**

**Graphical user interface, text, application, email

Description automatically generated**

**Mode value for customer table(numerical column)**

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;

**Graphical user interface, text, application, email

Description automatically generated**

1. **Reflect on your work**

Based on your previous experience, which tool (Excel or SQL) do you think is more effective for data profiling, and why?

SQL is more efficient at computing data profiles. Once the code is written, it can be applied to other data sets, it just takes time to get use to and lot’s of trial and error.