Juliana Lee

Task 3.6: Summarizing and Cleaning Data in SQL

1. **Check for and clean dirty data.**

Non-Uniform for Film table:

Query:	Explanation:
SELECT DISTINCT film_id, title, description, release_year, language_id, rental_duration, rental_rate, length, replacement_cost, rating, last_update, special_features FROM film;	1) This shows you whole entire film table and each column within the film table.

	film_id [PK] integer	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)	ra m
1	216	Day Unfaithful	A Stunning ...	2006	1	3	4.99	113	16.99	G
2	163	Clyde Theory	A Beautiful ...	2006	1	4	0.99	139	29.99	P
3	571	Metal Armageddon	A Thrilling ...	2006	1	6	2.99	161	26.99	P
4	514	Lebowski Soldiers	A Beautiful ...	2006	1	6	2.99	69	17.99	P
5	67	Berets Agent	A Taut Sag...	2006	1	5	2.99	77	24.99	P
6	394	Hamlet Wisdom	A Touching...	2006	1	7	2.99	146	21.99	R
7	444	Hustler Party	A Emotiona...	2006	1	3	4.99	83	22.99	N
8	206	Dancing Fever	A Stunning ...	2006	1	6	0.99	144	25.99	G
9	173	Confessions Maguire	A Insightful...	2006	1	7	4.99	65	25.99	P
10	795	Siege Madre	A Boring Ta...	2006	1	7	0.99	111	23.99	R
11	739	Rocky War	A Fast-Pac...	2006	1	4	4.99	145	17.99	P
12	849	Storm Happiness	A Insightful...	2006	1	6	0.99	57	28.99	N
13	297	Extraordinary Conquerer	A Stunning ...	2006	1	6	2.99	122	29.99	G
14	104	Bugsy Song	A Awe-Insp...	2006	1	4	2.99	119	17.99	G

Total rows: 1000 of 1000 Query complete 00:00:00.103 Ln 1, Col 177

SELECT DISTINCT rating FROM film GROUP BY rating;	2) This shows you the individual column, rating, of the film table. This can be done for each individual column.

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

Non-Uniform for Customer table:

Query:	Explanation:
SELECT DISTINCT customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, active FROM customer;	1) This shows you whole entire customer table and each column within the customer table.

Juliana Lee

<table>
<tr><td></td><td>

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	create_date date	active integer
1	275	2	Carole	Barnett	carole.barnett@sakila...	280	true	2006-02-14	1
2	14	2	Betty	White	betty.white@sakilacu...	18	true	2006-02-14	1
3	552	2	Hugh	Waldrop	hugh.waldrop@sakila...	558	true	2006-02-14	1
4	68	1	Nicole	Peterson	nicole.peterson@saki...	72	true	2006-02-14	1
5	250	2	Jo	Fowler	jo.fowler@sakilacust...	254	true	2006-02-14	1
6	183	2	Ida	Andrews	ida.andrews@sakilac...	187	true	2006-02-14	1
7	61	2	Katherine	Rivera	katherine.rivera@saki...	65	true	2006-02-14	1
8	214	1	Kristin	Johnston	kristin.johnston@saki...	218	true	2006-02-14	1
9	541	2	Darren	Windham	darren.windham@sak...	547	true	2006-02-14	1
10	473	2	Jorge	Olivares	jorge.olivares@sakila...	478	true	2006-02-14	1
11	114	2	Grace	Ellis	grace.ellis@sakilacus...	118	true	2006-02-14	1
12	343	1	Douglas	Graf	douglas.graf@sakilac...	348	true	2006-02-14	1
13	580	1	Ross	Grey	ross.grey@sakilacust...	586	true	2006-02-14	1
14	356	2	Gerald	Fultz	gerald.fultz@sakilacu...	361	true	2006-02-14	1
15	210	2	Ella	Oliver	ella.oliver@sakilacust...	214	true	2006-02-14	1

Total rows: 599 of 599 Query complete 00:00:00.094 Ln 1, Col 105

</td></tr>
<tr><td>

SELECT DISTINCT activebool FROM customer GROUP BY activebool;

</td><td>

2) This shows you the individual column, activebool, of the customer table. This can be done for each individual column.

	activebool boolean
1	true

</td></tr>
</table>

Duplicate Data for Film table:

<table>
<tr><th>Query:</th><th>Result screenshot:</th></tr>
<tr><td>

SELECT film_id, title, release_year, language_id, rental_duration,

COUNT(*)

FROM film

GROUP BY film_id, title, release_year, language_id, rental_duration

HAVING COUNT(*) >1;

</td><td>

film_id [PK] integer	title character varying (255)	release_year integer	language_id smallint	rental_duration smallint	count bigint

</td></tr>
</table>

Duplicate Data for Customer table:

Juliana Lee

Query:	Result screenshot:
SELECT **customer_id,** **first_name,** **last_name,** **email,** **address_id,** **COUNT(*)** **FROM** **customer** **GROUP BY** **customer_id,** **first_name,** **last_name,** **email,** **address_id** **HAVING** **COUNT(*) >1;**	customer_id [PK] integer | first_name character varying (45) | last_name character varying (45) | email character varying (50) | address_id smallint | count bigint

There are no duplicates for either table (film/customer). If there were duplicates, there are a couple ways to deal with duplicates. You can create a virtual table "View" where unique records can be selected, or duplicate records can be deleted. This would require permission to alter the table. If permission to alter the table is not granted, GROUP BY or DISTINCT functions can be used to select unique records.

Missing Data:

Queries:	Explanations:
SELECT column 1, column 3 **FROM** tablename --column 2 is ignored in select because of the numerous missing values	1) If there is a column with too much missing data, it is best to leave it alone as erasing/replacing will cause more problems later.
UPDATE tablename **SET** = **AVG** (column1) **WHERE** column1 **IS** NULL	2) If there a few missing values, those missing values can be imputed with the average value.

2. **Summarize your data.**

Descriptive Statistics for 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;
```

Descriptive Statistics for 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(create_date) AS min_create_date,

MAX(create_date) AS max_create_date,
```

```
MIN(last_update) AS min_last_update,

MAX(last_update) AS max_last_update,

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 create_date) AS mode_create_date,

MODE () WITHIN GROUP (ORDER BY last_update) AS mode_last_update

FROM customer;
```

3. **Reflect on your work:** I think that SQL is more effective for data profiling. With the correct SQL query, the result will be given immediately. However, with Excel there are multiple steps involved in the process of cleaning your data. SQL is easier to use and gives faster results. In addition, SQL is the better choice for large datasets.