Data cleaning process

Non-uniform values from film table

--looking for non-uniform values from film table

SELECT film_id, title, description, release_year, language_id, rental_duration, rental_rate, replacement_cost, rating, last_update

FROM film

There is no non-uniform data in film table. If I found that the rating PG was entered as P'G, P,G, or P.G. I will correct it with the following query;

--Update P'G, P,G, and P.G as PG in the film table

UPDATE film SET rating ='G' WHERE rating IN ('P'G', 'P.G ', 'P.G ')

Duplicate data from film table

--looking for duplicate data from film table

SELECT film_id, title, description, release_year, language_id, rental_duration, rental_rate, replacement_cost, rating, last_update,

COUNT(*) FROM film

GROUP BY film_id, title, description, release_year, language_id, rental_duration, rental_rate, replacement_cost, rating, last_update

HAVING COUNT(*) >1;

No duplicate was found on the film table. To fix the duplicate records in film table, we can either Create a virtual table, known as a "view," where you select only unique records or delete the duplicate record from the table or view. Deleting records is not a good option, the best option is to create the unique record. The following query will create a unique record;

--Shows only unique records from the film table

SELECT DISTINCT film_id, title, description, release_year, language_id,

rental_duration, rental_rate, replacement_cost, rating, last_update

FROM film

Missing values from film table

--looking for missing values from film table

SELECT film_id, title, description, release_year, language_id, rental_duration, rental_rate, replacement_cost, rating, last_update

FROM film

There are no missing values in the film table. If there are high percentage of missing values, its best to ignore the column in a query. If there are few missing values in a column, then, we can impute values such as estimates to fill in the missing values. The following query can be used;

--imputing missing values in language_id with the MODE value in film table

UPDATE film SET = MODE(language_id) WHERE language_id IS NULL

Non-uniform values from customer table

--looking for non-uniform values data from customer table

SELECT customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, last_update

FROM customer

There is no non-uniform data in customer table. If I found that the customer_id 2 was entered as .2, /2, and ;2. I will correct it with the following query;

--Update 2, /2, and ;2 as 2 in the customer table

UPDATE customer SET rating ='2' WHERE customer_id IN ('.2', '/2 ', ';2 ')

Duplicate data from customer table

--looking for duplicate data from customer table

SELECT customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, last_update,

COUNT(*) FROM customer

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

No duplicate was found on the customer table. To fix the duplicate records in customer table, we can either Create a virtual table, known as a "view," where you select only unique records or delete the duplicate record from the table or view. Deleting records is not a good option, the best option is to create the unique record. The following query will create a unique record;

CREATE VIEW customer_table2

AS SELECT customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, last_update FROM customer

GROUP BY customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, last_update

--Group by will make each row unique

Missing values from customer table

--looking for missing values from customer table

SELECT customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, last_update

FROM customer

There are no missing values in the customer table. If there are high percentage of missing values, its best to ignore the column in a query. If there are few missing values in a column, then, we can impute values such as estimates to fill in the missing values. The following query can be used;

--imputing missing values in language_id with the MODE value in customer table

UPDATE customer SET = MODE(activebool) WHERE activebool IS NULL

Descriptive Analysis

2a. Film table

--select and display minimum, maximum and average of film_id, release_year, language_id, rental_duration, rental_rate, replacement_cost, count of rows, and modes of title, description, rating, last_update, special_features and fulltext from the film table

```
SELECT MIN(film_id) AS min_film_id,
    MAX(film_id) AS max_film_id,
    AVG(film_id) AS avg_film_id,
    COUNT(film_id) AS count_rent_values,
        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(replacement_cost) AS MIN_replacement_cost,
    MAX(replacement_cost) AS max_replacement_cost,
    AVG(replacement_cost) AS avg_replacement_cost,
        COUNT(*) AS count_rows,
        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 last_update) AS mode_last_update,
        MODE () WITHIN GROUP (ORDER BY special_features) AS mode_special_features,
        MODE () WITHIN GROUP (ORDER BY fulltext) AS mode_fulltext FROM film;
```

2b. Customer table

--select and display minimum, maximum and average of customer_id, store_id, address_id, active, count of rows, and modes of first_name, last_name, email, activebool and create_date from the 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(active) AS min_active,
    MAX(active) AS max_active,
    AVG(active) AS avg_active,
          COUNT(*) AS count_rows,
          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 activebool) AS mode_activebool,
          MODE () WITHIN GROUP (ORDER BY create_date) AS mode_create_date
         FROM customer;
```

INNER JOIN

--select and display customer_id and count of country, count of country top 10, arranged in descending order.

```
SELECT D.country,  COUNT(A.customer_id) AS customer_numbers
FROM customer A
INNER JOIN address B ON A.address_id = B.address_id
INNER JOIN city C ON B.city_id = C.city_id
INNER JOIN country D ON C.country_ID = D.country_ID
GROUP BY D.country
ORDER BY COUNT (A.customer_id) DESC LIMIT 10;
```

--select and display customer_id, count of country, top 10 cities from count of country top 10, arranged in descending order.

SELECT D.country, C.city,

COUNT(A.customer_id) AS customer_numbers

FROM customer A

INNER JOIN address B ON A.address_id = B.address_id

INNER JOIN city C ON B.city_id = C.city_id

INNER JOIN country D ON C.country_ID = D.country_ID

WHERE D.country IN('India', 'China', 'United States', 'Japan', 'Mexico',

'Brazil', 'Russian Federation', 'Phillipines', 'Turkey', 'Indonesia')

GROUP BY D. country, C.city

*ORDER BY COUNT (A.customer_id) DESC LIM**IT 10;***

--select and display first name, last name, customer_id, sum of amount from 5 top customers from top 10 cities, city, and country, arranged in descending order.

SELECT C.city AS City, D.country AS Country, A.customer_id AS Customer_ID,

A.first_name AS Customer_First_Name, A.last_name as Last_Name,

SUM(F.amount) AS Total_Amount_Paid

FROM customer A

INNER JOIN address B ON A.address_id = B.address_id

INNER JOIN city C ON B.city_id = C.city_id

INNER JOIN country D ON C.country_ID = D.country_ID

INNER JOIN rental E ON A.customer_id = E.customer_id

INNER JOIN payment F ON E.rental_id = F.rental_ID

WHERE C.city IN('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei',

'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')

```
GROUP BY C.city, D. country, A.customer_id, A.first_name, A.last_name

ORDER BY SUM(F.amount) DESC LIMIT 5;
```

LEFT JOIN

```
SELECT DISTINCT(A.country),

COUNT(DISTINCT D.customer_id) AS all_customer_count,

COUNT(DISTINCT A.country)AS top_customer_count

FROM country A

INNER JOIN city B ON A.country_ID =B.country_ID

INNER JOIN address C ON B.city_id =C.city_id

INNER JOIN customer D ON C.address_id = D.address_id

LEFT JOIN(SELECT C.city AS City, D.country AS Country, A.customer_id AS Customer_ID,

A.first_name AS Customer_First_Name, A.last_name as Last_Name,

 SUM(F.amount) AS Total_Amount_Paid

FROM customer A

INNER JOIN address B ON A.address_id = B.address_id

INNER JOIN city C ON B.city_id = C.city_id

INNER JOIN country D ON C.country_ID = D.country_ID

INNER JOIN rental E ON A.customer_id = E.customer_id

INNER JOIN payment F ON E.rental_id = F.rental_ID

WHERE D.country IN('India', 'China', 'United States', 'Japan', 'Mexico',

'Brazil', 'Russian Federation', 'Phillipines', 'Turkey', 'Indonesia')

GROUP BY C.city, D. country, A.customer_id, A.first_name, A.last_name

ORDER BY SUM(F.amount) DESC LIMIT 5) AS top_5_customers

ON A.country=top_5_customers.country
```

GROUP BY A.country, top_5_customers

ORDER BY all_customer_count DESC

LIMIT 5;

CTE Queries

WITH average_total_amount_cte(city, country, customer_id, first_name, last_name, amount) AS

(SELECT C.city, D.country, A.customer_id, A.first_name, A.last_name, SUM(F.amount)

FROM customer A

INNER JOIN address B ON A.address_id =B.address_id

INNER JOIN city C ON B.city_id =C.city_id

INNER JOIN country D ON C.country_Id =D.country_Id

INNER JOIN rental E ON A.customer_id =E.customer_id

INNER JOIN payment F ON E.rental_id =F.rental_Id

WHERE C.city IN('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei',

'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')

GROUP BY C.city, D.country, A.customer_id, A.first_name, A.last_name

ORDER BY SUM(F.amount) DESC LIMIT 5)

SELECT AVG(amount)

FROM average_total_amount_cte

B. WITH top_5_customers_cte(amount, customer_id, first_name, Last_name, city, country, Total_Amount_Paid) AS

(SELECT B.customer_id, B.first_name, A.amount, B.last_name,

D.city, E.country, SUM(amount) AS Total_Amount_Paid

FROM payment A

INNER JOIN customer B ON A.customer_id =B.customer_id

```
INNER JOIN address  C ON B.address_id =C.address_id

INNER JOIN city D ON C.city_id =D.city_id

INNER JOIN country E ON D.country_id =E.country_id

WHERE country IN('India', 'China', 'United States', 'Japan', 'Mexico',

'Brazil', 'Russian Federation', 'Phillipines', 'Turkey', 'Indonesia')

GROUP BY D.city, E.country, B.customer_id, B.first_name, B.last_name, A.amount

ORDER BY SUM(amount) DESC LIMIT 5), top_5_customers AS (SELECT D.country,

COUNT(DISTINCT A.customer_id) AS all_customer_count, COUNT(DISTINCT D.country) AS
top_customer_count

FROM customer A

INNER JOIN address B ON A.address_id =B.address_id

INNER JOIN city C ON B.city_id =C.city_id

INNER JOIN country D ON C.country_id =D.country_id

GROUP BY D.country)

SELECT D.country,

COUNT(DISTINCT A.customer_id) AS all_customer_count,

COUNT(DISTINCT top_5_customers_cte.customer_id) AS top_customer_count

FROM customer A

INNER JOIN address B ON A.address_id =B.address_id

INNER JOIN city C ON B.city_id =C.city_id

INNER JOIN country D ON C.country_id =D.country_id

LEFT JOIN top_5_customers_cte ON D.country =top_5_customers_cte.country

GROUP BY D.country

ORDER BY all_customer_count DESC
```