DATA CLEANING & TRANSFORMATION

Data cleaning in SQL involves the process of identifying and correcting errors or inconsistencies in a database to ensure that the data is accurate, complete, and ready for analysis. Common tasks in data cleaning include handling missing values, correcting data types, removing duplicates, and standardizing formats. Here are some examples of data cleaning tasks in SQL:

Transformation involves changing the format or values of existing data in a table to meet specific requirements or standards. This can include tasks such as converting data types, modifying values, or standardizing formats.

Handling Missing Values:

```
mysql> -- Handling Missing Values
mysql> -- Identify rows with missing values in the 'age' column
mysql> SELECT * FROM sample_table WHERE age IS NULL;
+----+------------+------+-----------------------+
| id | name       | age  | email                 |
+----+------------+------+-----------------------+
|  2 | Jane Smith | NULL | jane.smith@example.com |
+----+------------+------+-----------------------+
1 row in set (0.00 sec)

mysql> -- Replace missing values in the 'age' column with a default value (e.g., 0)
mysql> UPDATE sample_table SET age = 0 WHERE age IS NULL;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> -- Correcting Data Types
```

Removing Duplicates

```
mysql> -- Removing Duplicates
mysql> -- Identify and display duplicate rows based on the 'name' column
mysql> SELECT name, COUNT(*) FROM sample_table GROUP BY name HAVING COUNT(*) > 1;
+------------+----------+
| name       | COUNT(*) |
+------------+----------+
| Jane Smith |        2 |
+------------+----------+
1 row in set (0.00 sec)
```

Correcting Data Types

```
mysql> -- Correcting Data Types
mysql> -- Convert the 'age' column to VARCHAR
mysql> ALTER TABLE sample_table MODIFY age VARCHAR(10);
Query OK, 6 rows affected (0.08 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Standardizing Formats

```
mysql> -- Standardizing Formats
mysql> -- Convert the 'email' column values to lowercase
mysql> UPDATE sample_table SET email = LOWER(email);
Query OK, 0 rows affected (0.00 sec)
Rows matched: 6  Changed: 0  Warnings: 0
```

Ranking:

Ranking in SQL involves assigning a rank or position to each row based on certain criteria. Commonly used window functions like ROW_NUMBER(), RANK(), and DENSE_RANK() can be used for ranking.

ROW_NUMBER() & RANK()

Command Prompt - mysql -u root -p

```
mysql> -- Using ROW_NUMBER()
mysql> SELECT
    ->     id,
    ->     name,
    ->     age,
    ->     email,
    ->     ROW_NUMBER() OVER (ORDER BY id) AS row_num
    -> FROM
    ->     sample_table;
+----+-------------+------+-------------------------+---------+
| id | name        | age  | email                   | row_num |
+----+-------------+------+-------------------------+---------+
|  1 | John Doe    | 25   | john.doe@example.com    |       1 |
|  2 | Jane Smith  | 0    | jane.smith@example.com  |       2 |
|  3 | Bob Johnson | 30   | NULL                    |       3 |
|  4 | Alice Brown | 22   | alice.brown@example.com |       4 |
|  5 | Chris Lee   | 28   | chris.lee@example.com   |       5 |
|  6 | Jane Smith  | 25   | jane.smith@example.com  |       6 |
+----+-------------+------+-------------------------+---------+
6 rows in set (0.03 sec)

mysql> -- Using RANK()
mysql> SELECT
    ->     id,
    ->     name,
    ->     age,
    ->     email,
    ->     RANK() OVER (ORDER BY age) AS rank_num
    -> FROM
    ->     sample_table;
+----+-------------+------+-------------------------+----------+
| id | name        | age  | email                   | rank_num |
+----+-------------+------+-------------------------+----------+
|  2 | Jane Smith  | 0    | jane.smith@example.com  |        1 |
|  4 | Alice Brown | 22   | alice.brown@example.com |        2 |
|  1 | John Doe    | 25   | john.doe@example.com    |        3 |
|  6 | Jane Smith  | 25   | jane.smith@example.com  |        3 |
|  5 | Chris Lee   | 28   | chris.lee@example.com   |        5 |
|  3 | Bob Johnson | 30   | NULL                    |        6 |
+----+-------------+------+-------------------------+----------+
6 rows in set (0.03 sec)

mysql>
```

DENSE_RANK() & PERCENT_RANK()

Command Prompt - mysql -u root -p

```
mysql> -- Using DENSE_RANK()
mysql> SELECT
    ->     id,
    ->     name,
    ->     age,
    ->     email,
    ->     DENSE_RANK() OVER (ORDER BY age) AS dense_rank_num
    -> FROM
    ->     sample_table;
+----+-------------+------+-------------------------+----------------+
| id | name        | age  | email                   | dense_rank_num |
+----+-------------+------+-------------------------+----------------+
|  2 | Jane Smith  | 0    | jane.smith@example.com  |              1 |
|  4 | Alice Brown | 22   | alice.brown@example.com |              2 |
|  1 | John Doe    | 25   | john.doe@example.com    |              3 |
|  6 | Jane Smith  | 25   | jane.smith@example.com  |              3 |
|  5 | Chris Lee   | 28   | chris.lee@example.com   |              4 |
|  3 | Bob Johnson | 30   | NULL                    |              5 |
+----+-------------+------+-------------------------+----------------+
6 rows in set (0.00 sec)

mysql> -- Using PERCENT_RANK()
mysql> SELECT
    ->     id,
    ->     name,
    ->     age,
    ->     email,
    ->     PERCENT_RANK() OVER (ORDER BY age) AS percent_rank_num
    -> FROM
    ->     sample_table;
+----+-------------+------+-------------------------+------------------+
| id | name        | age  | email                   | percent_rank_num |
+----+-------------+------+-------------------------+------------------+
|  2 | Jane Smith  | 0    | jane.smith@example.com  |                0 |
|  4 | Alice Brown | 22   | alice.brown@example.com |              0.2 |
|  1 | John Doe    | 25   | john.doe@example.com    |              0.4 |
|  6 | Jane Smith  | 25   | jane.smith@example.com  |              0.4 |
|  5 | Chris Lee   | 28   | chris.lee@example.com   |              0.8 |
|  3 | Bob Johnson | 30   | NULL                    |                1 |
+----+-------------+------+-------------------------+------------------+
6 rows in set (0.00 sec)

mysql>
```

Stored Procedures:

A stored procedure is a precompiled collection of one or more SQL statements that can be executed as a single unit. It is stored in the database and can be called and executed by name.

Command Prompt - mysql -u root -p

```
mysql> -- Create a stored procedure
mysql> DELIMITER //
mysql> CREATE PROCEDURE UpdateEmails()
    -> BEGIN
    ->     -- Replace NULL values in the 'email' column with a default email address
    ->     UPDATE sample_table SET email = 'default@example.com' WHERE email IS NULL;
    -> END //
Query OK, 0 rows affected (0.04 sec)

mysql> DELIMITER ;
mysql> -- Call the stored procedure
mysql> CALL UpdateEmails();
Query OK, 1 row affected (0.03 sec)

mysql> -- Display the updated data in the table
mysql> SELECT * FROM sample_table;
+----+-------------+------+-------------------------+
| id | name        | age  | email                   |
+----+-------------+------+-------------------------+
|  1 | John Doe    | 25   | john.doe@example.com    |
|  2 | Jane Smith  | 0    | jane.smith@example.com  |
|  3 | Bob Johnson | 30   | default@example.com     |
|  4 | Alice Brown | 22   | alice.brown@example.com |
|  5 | Chris Lee   | 28   | chris.lee@example.com   |
|  6 | Jane Smith  | 25   | jane.smith@example.com  |
+----+-------------+------+-------------------------+
6 rows in set (0.00 sec)

mysql>
```