SAFNA S

Importing and Cleaning Data

IN SQL

DATA IMPORT

- Using SQL INSERT Statements
- Using SQL LOAD DATA INFILE
- Using a GUI Tool
- Using ETL Tools

To import data into a database, you generally have a few options depending on the specific database management system (DBMS) you are using. Here are the common methods for importing data:

1. Using SQL INSERT Statements:
   - If you have a small amount of data, you can manually write SQL INSERT statements to add data row by row. Here's an example:

```
INSERT INTO table_name (column1, column2, column3) VALUES (value1, value2, value3);
```

- You would need to repeat this statement for each row of data you want to import. While this method is feasible for small datasets, it can be time-consuming for larger datasets.

2.Using SQL LOAD DATA INFILE:
- Some database systems, like MySQL, offer the LOAD DATA INFILE statement, which allows you to import data from a text file (such as a CSV file) directly into a table. This method used for large data set
- Here's an example of using LOAD DATA INFILE in MySQL:

```sql
LOAD DATA INFILE '/path/to/file.csv'
INTO TABLE table_name
FIELDS TERMINATED BY ',' -- Specify the field delimiter
LINES TERMINATED BY '\n' -- Specify the line terminator
IGNORE 1 LINES; -- Skip the header row if present
```

3. Using a GUI Tool:
   - Many database management systems provide graphical user interface (GUI) tools that allow you to import data easily. These tools often have import wizards or functionalities specifically designed for importing data from various file formats, such as CSV, Excel, or JSON.
   - For example, if you're using MySQL, you can use the MySQL Workbench tool, which provides an Import Wizard for importing data.

4. Using ETL Tools:
   - For complex data integration tasks, you may consider using Extract, Transform, Load (ETL) tools. These tools allow you to extract data from various sources, perform transformations on the data, and load it into the database.
   - Popular ETL tools include Apache NiFi, Talend, Pentaho and SSIS.

Choose the method that suits your needs based on the size of the dataset, the database management system you are using, and the available tools.

DATA CLEANING

- Removing Duplicates
- Handling Missing Values
- Standardizing and Formatting Data
- Data Validation and Correction
- Handling Outliers

Removing Duplicates:

- Identify duplicate records in a table based on specific columns using the GROUP BY clause and the HAVING clause.
- Delete duplicate records using the DELETE statement or update them based on your requirements.

```sql
-- Identify duplicate records
SELECT column1, column2, COUNT(*)
FROM table_name
GROUP BY column1, column2
HAVING COUNT(*) > 1;

-- Delete duplicate records
DELETE FROM table_name
WHERE column1 = 'value' AND column2 = 'value';

-- Update duplicate records
UPDATE table_name
SET column1 = 'new_value'
WHERE column1 = 'duplicate_value';
```

Handling Missing Values:

- Identify missing values in a table using the IS NULL or IS NOT NULL conditions.
- Replace or update missing values using the UPDATE statement or fill them with appropriate values.

```sql
-- Identify rows with missing values
SELECT *
FROM table_name
WHERE column1 IS NULL;

-- Replace missing values with a default value
UPDATE table_name
SET column1 = 'default_value'
WHERE column1 IS NULL;

-- Fill missing values with a calculated value
UPDATE table_name
SET column1 = column2 * 2
WHERE column1 IS NULL;
```

Standardizing and Formatting Data:

- Convert data to a consistent format by using SQL functions like UPPER, LOWER, INITCAP, or REPLACE.
- Remove unwanted characters, leading/trailing spaces, or non-alphanumeric characters using functions like TRIM, LTRIM, RTRIM, and REGEXP_REPLACE

```sql
-- Convert data to uppercase
UPDATE table_name
SET column1 = UPPER(column1);

-- Remove leading/trailing spaces
UPDATE table_name
SET column1 = TRIM(column1);

-- Replace specific characters
UPDATE table_name
SET column1 = REPLACE(column1, 'old_value', 'new_value');

-- Remove non-alphanumeric characters
UPDATE table_name
SET column1 = REGEXP_REPLACE(column1, '[^a-zA-Z0-9]', '')
```

Data Validation and Correction:

- Validate data against specific rules using SQL functions and conditions.
- Correct inconsistent or incorrect data using UPDATE statements.

```sql
-- Validate data using conditions
SELECT *
FROM table_name
WHERE column1 < 0;

-- Correct incorrect data
UPDATE table_name
SET column1 = ABS(column1)
WHERE column1 < 0;
```

Handling Outliers:

- Identify outliers using statistical functions like AVG, STDDEV, or percentile functions.
- Decide on the appropriate action based on your analysis, such as removing outliers or adjusting their values.

```sql
-- Identify outliers
SELECT column1
FROM table_name
WHERE column1 > (SELECT AVG(column1) + 2 * STDDEV(column1) FROM table_name);

-- Remove outliers
DELETE FROM table_name
WHERE column1 > (SELECT AVG(column1) + 2 * STDDEV(column1) FROM table_name);

-- Adjust outlier values
UPDATE table_name
SET column1 = (SELECT AVG(column1) FROM table_name)
WHERE column1 > (SELECT AVG(column1) + 2 * STDDEV(column1) FROM table_name);
```

Thank You So Much