

## Assignment-6

### Data cleaning and Transformation

SQL offers a wide variety of techniques for cleaning and transforming data efficiently.

Some of the techniques include the following:

- Check for Missing Values
- Check for Duplicates
- Standardizing and Transforming Data
- Updating Data / Column Data Types

#### 1) Check for Missing values

SQL provides powerful functions and techniques for handling missing values.

Ex: **IS NULL**: It is used to indicate missing values in a column.

**COALESCE**: It is used to replace missing values with more suitable values.

#### 2) Check for Duplicates

Duplicate records can distort analysis results and skew decision-making. SQL offers efficient ways to identify and remove duplicates.

Ex: **GROUP BY and HAVING**: They are used to find duplicate records.

**DISTINCT**: It eliminates duplicate rows from the result set.

**DELETE FROM**: It removes certain rows from the result set.

#### 3) Standardizing and Transforming Data

Data standardization and transformation are crucial for consistency and accurate analysis. SQL provides functions and expressions for data manipulation.

Ex: **UPPER or LOWER**: They convert a string to uppercase or lowercase.

**REPLACE**: It is used to replace occurrences of a substring in a string with a new substring.

**SUBSTRING**: It returns specified portions of text.

**TRIM**: It removes leading and trailing spaces from a string.

**DATE\_FORMAT**: You can format a date/time value into a specific format using this.

**CASE WHEN**: It performs conditional data transformations. Similar to IF in Excel.

#### 4) Updating Data / Column Data Types

**UPDATE**: You can use the UPDATE statement to update existing data in a table.

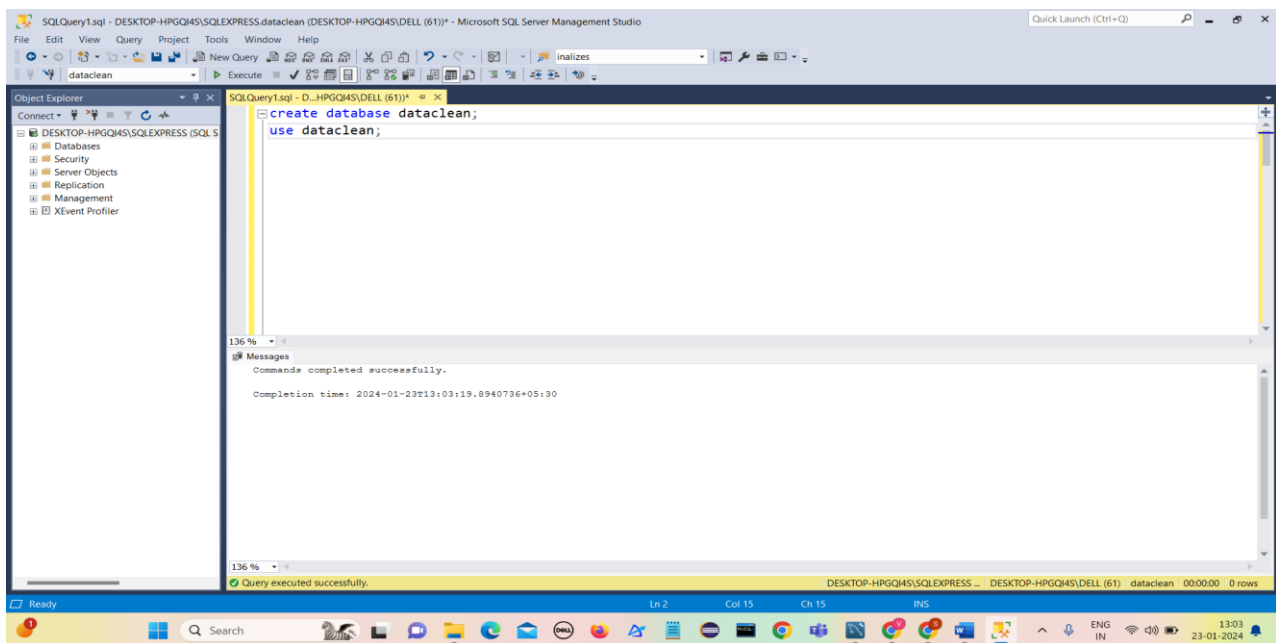
**ALTER TABLE & MODIFY COLUMN:** If you need to modify the data type of a column, you can use the ALTER TABLE statement with the MODIFY COLUMN clause.

**ALTER TABLE & DROP COLUMN:** To remove a column from a table, you can use the ALTER TABLE statement with the DROP COLUMN clause.

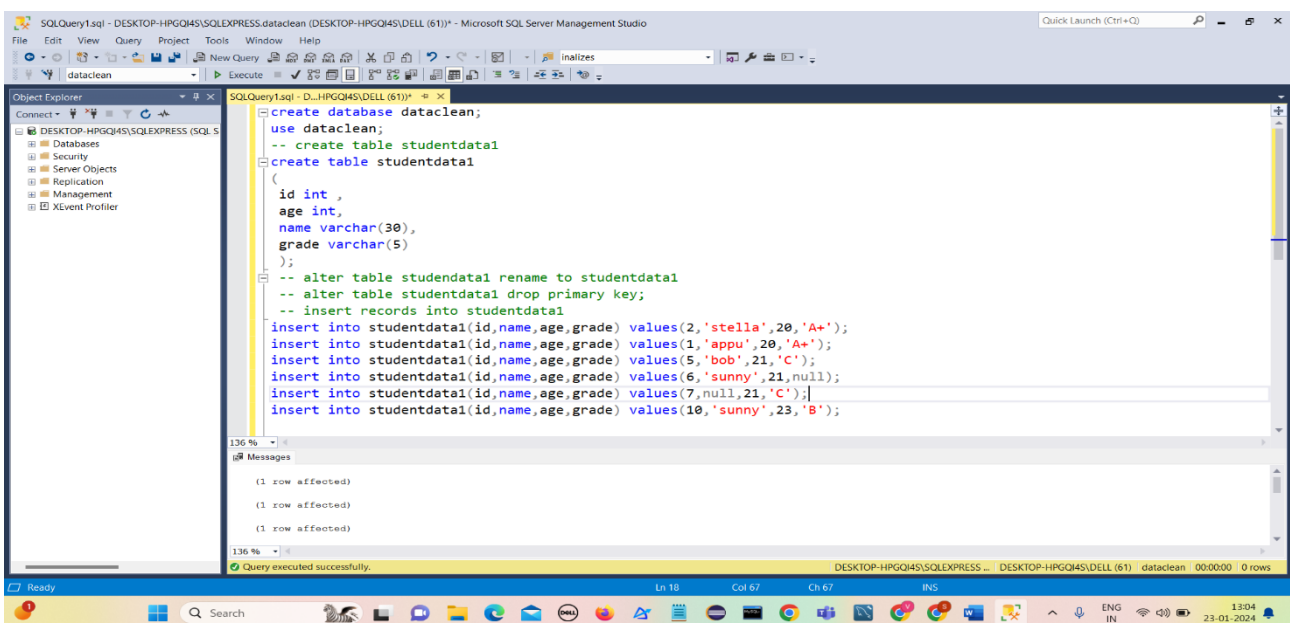
### Example for data cleaning and transformation

create database dataclean

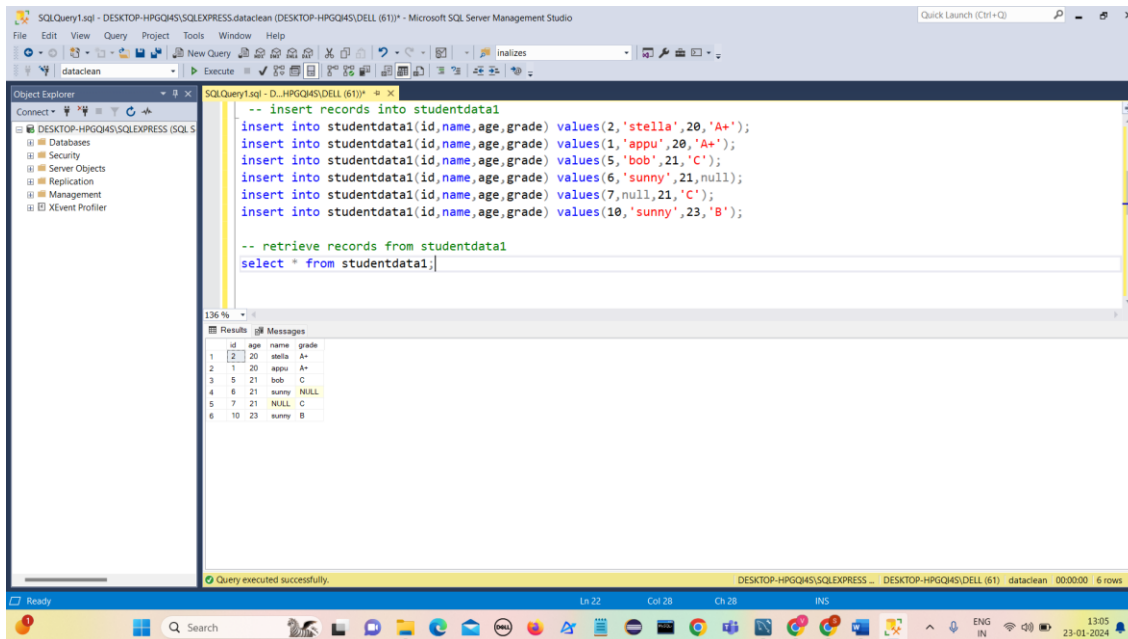
use dataclean to make it as current working database



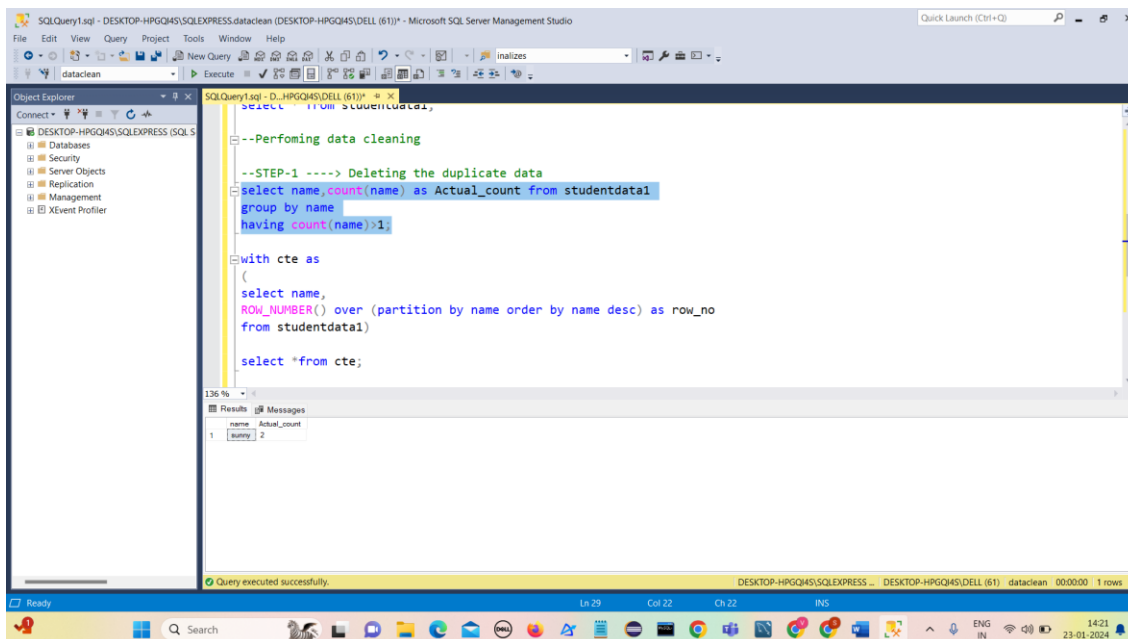
Create table studentdata1 and insert data into it



## Retrieve studentdata1 records



## Performing Data Cleaning

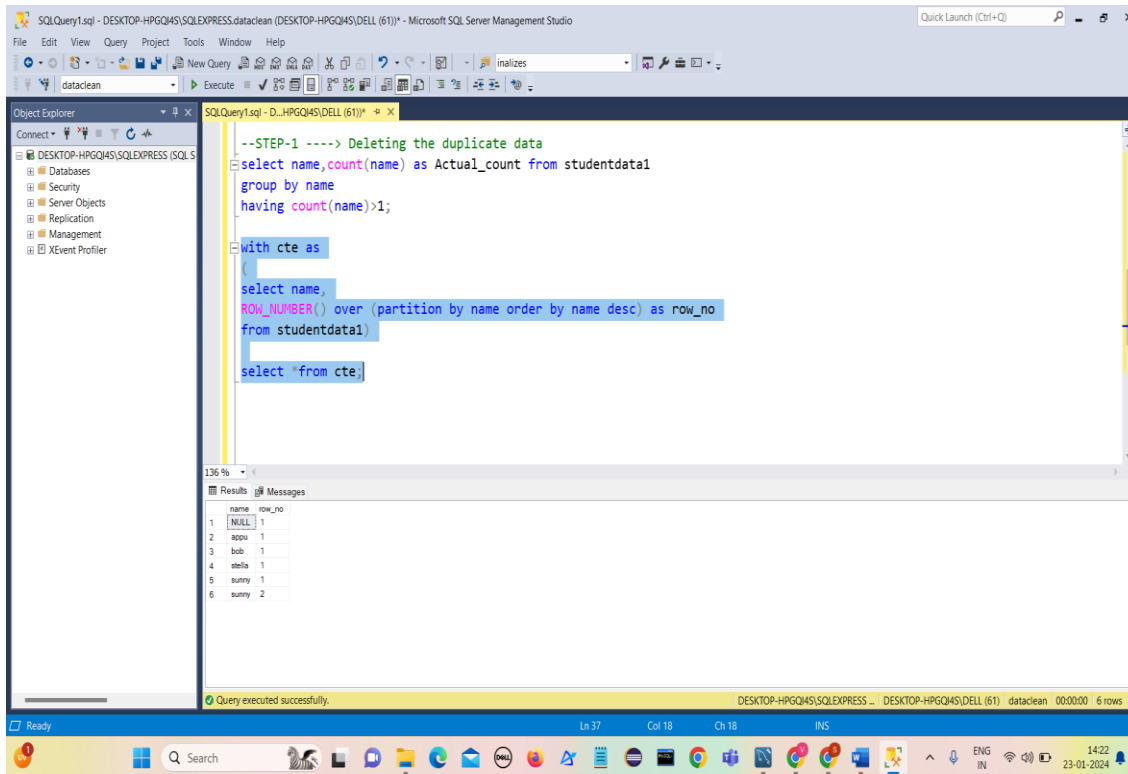


This query gives the name and count of each student.

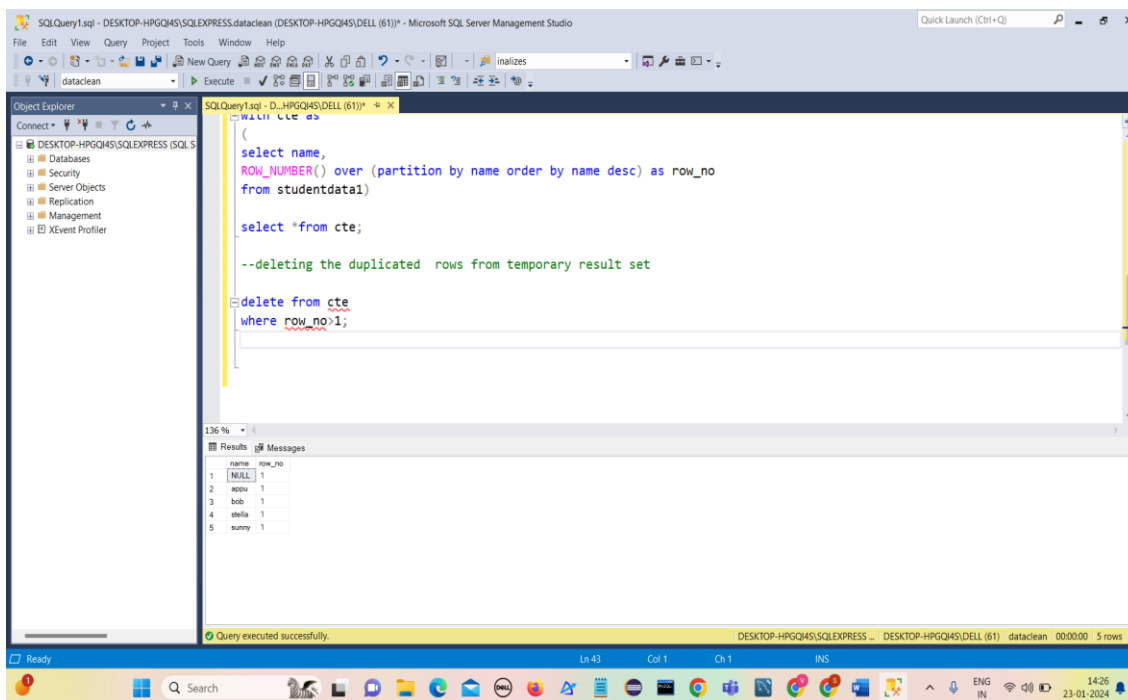
### Step-1 Deleting duplicate data

Creating a Common Table Expression (CTE) named **cte** using the **ROW\_NUMBER()** window function over the studentdata1 table.

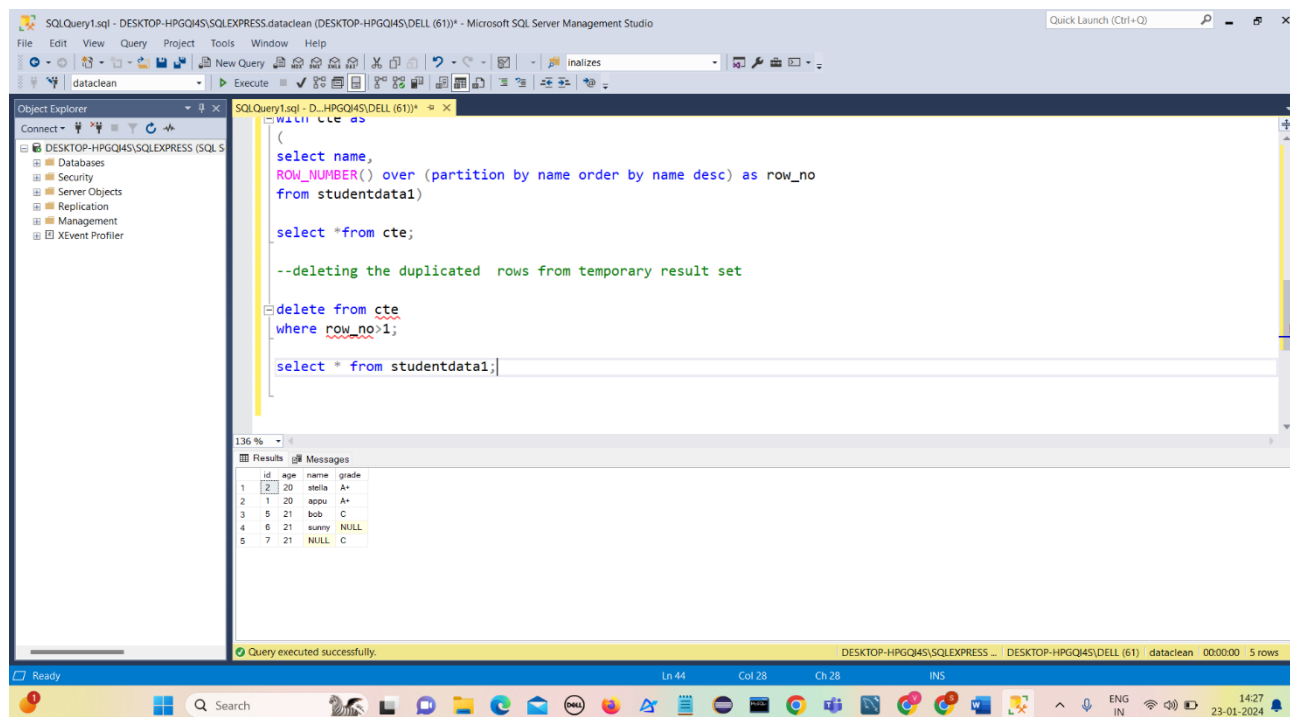
The **ROW\_NUMBER()** function assigns a unique number to each row within a partition of the result set based on the specified ordering.



The row\_no for sunny is 2 i.e., it is repeating and it is duplicate. We need to eliminate duplicate values.

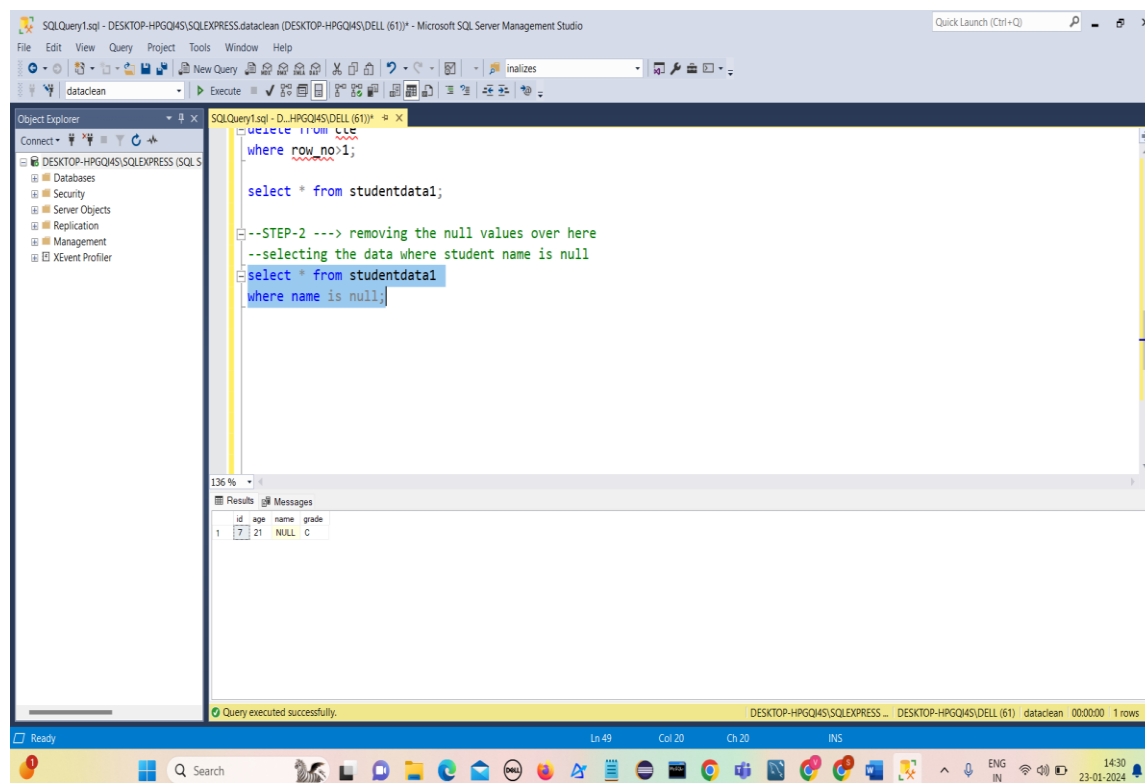


The value of sunny is 1.

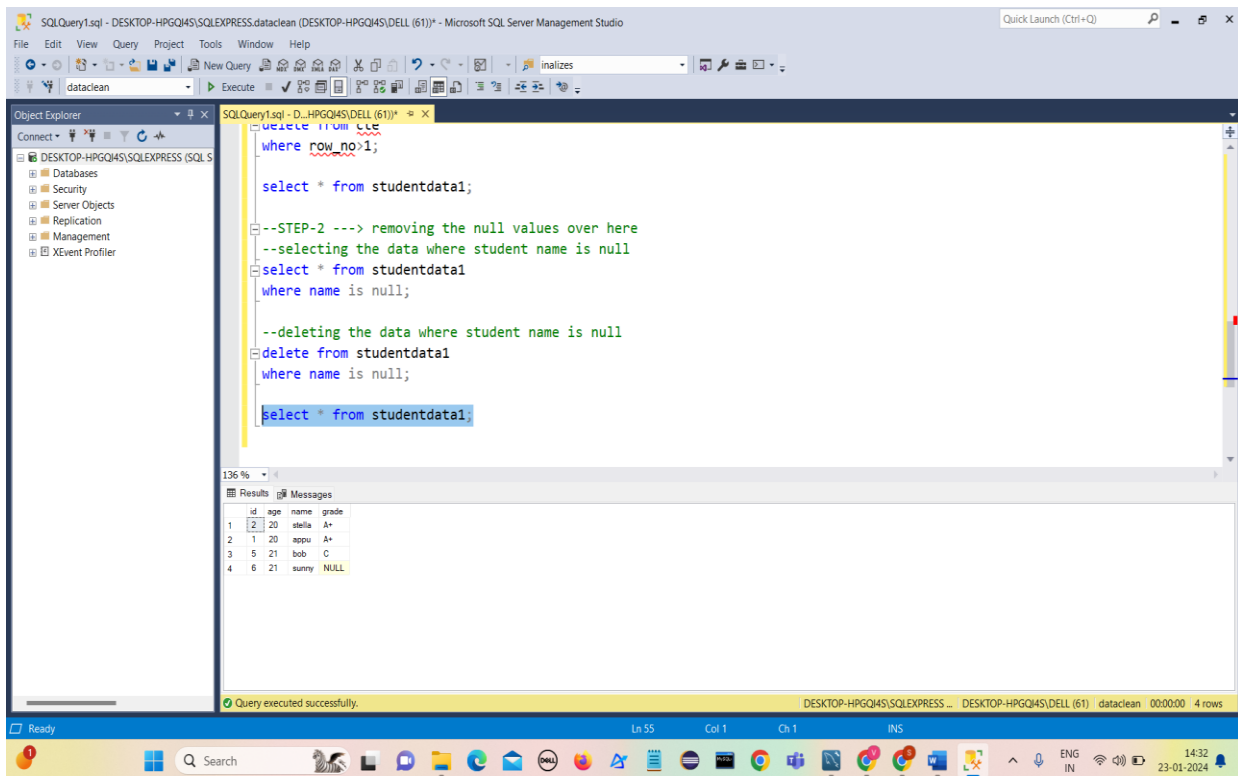


Duplicate values are removed.

## Step-2: Removing null values

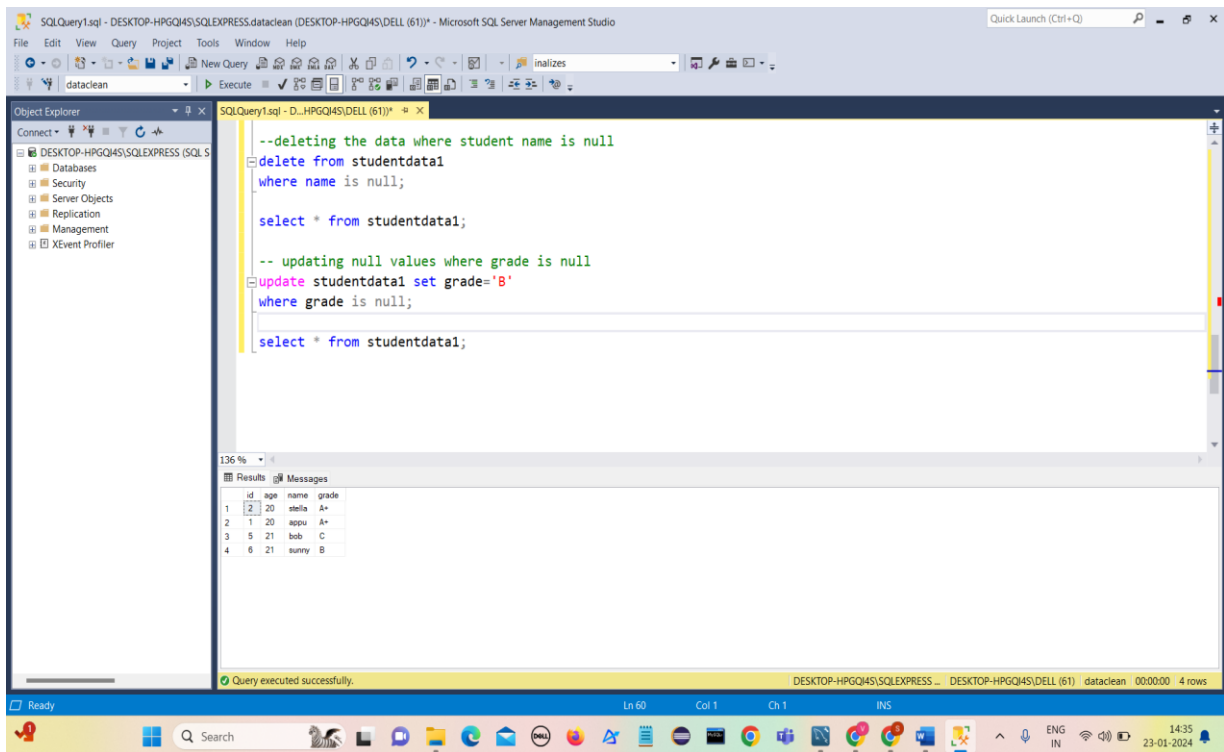


Here name is null for student with id 7. We have to eliminate null values



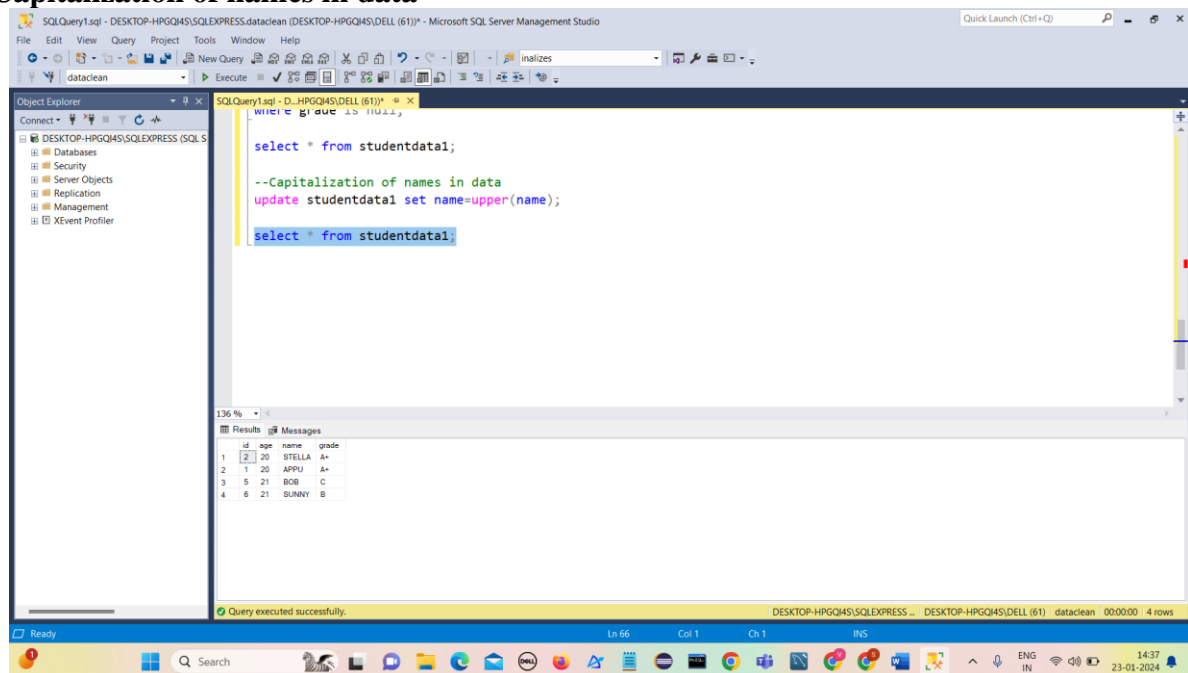
Student with id 7 is deleted.

Updating null values where grade is null



Hence grade is updated.

## Capitalization of names in data



## RANK() FUNCTION:

The **RANK() function** is a window function could be used in SQL Server to calculate a rank for each row within a partition of a result set.

SQL Server provides SQL RANK functions to specify rank for individual fields as per the categorizations.

Rank Functions are:

- 1) **ROW\_NUMBER()**
- 2) **RANK()**
- 3) **DENSE\_RANK()**
- 4) **NTILE()**

### 1) ROW\_NUMBER():

- i) **OVER() clause** - define a set of rows in the result set.
- ii) **ROW\_Number() SQL RANK function** gives a unique sequential number for each row in the specified data.
- iii) It gives the rank one for the first row and then increments the value by one for each row. We get different ranks for the row having similar values as well.

datacleaning & TCL commands employee.sql - DESKTOP-HPGQ4S\SQLEXPRESS\dataclean (DESKTOP-HPGQ4S\DELL (61)) - Microsoft SQL Server Management Studio

Object Explorer

- DESKTOP-HPGQ4S\SQLEXPRESS (SQL S
- Databases
- Security
- Server Objects
- Replication
- Management
- XEvent Profiler

datacleaning & TC...HPGQ4S\DELL (61))

```
-- ROW_NUMBER() function
select id,name,age,grade,ROW_NUMBER() OVER(order by age) RowNumber
from studentdata1;
```

Results

	id	name	age	grade	RowNumber
1	2	STELLA	20	A+	1
2	1	APPU	20	A+	2
3	5	BOB	21	C	3
4	6	SUNNY	21	B	4

Query executed successfully.

DESKTOP-HPGQ4S\SQLEXPRESS ... DESKTOP-HPGQ4S\DELL (61) dataclean 00:00:00 4 rows

Age order by in descending order

datacleaning & TCL commands employee.sql - DESKTOP-HPGQ4S\SQLEXPRESS\dataclean (DESKTOP-HPGQ4S\DELL (61)) - Microsoft SQL Server Management Studio

Object Explorer

- DESKTOP-HPGQ4S\SQLEXPRESS (SQL S
- Databases
- Security
- Server Objects
- Replication
- Management
- XEvent Profiler

datacleaning & TC...HPGQ4S\DELL (61))

```
-- ROW_NUMBER() function
select id,name,age,grade,ROW_NUMBER() OVER(order by age) RowNumber
from studentdata1;

-- Age order by in descending order
select id,name,age,grade,ROW_NUMBER() OVER(order by age desc) RowNumberDesc
from studentdata1;
```

Results

	id	name	age	grade	RowNumberDesc
1	5	BOB	21	C	1
2	6	SUNNY	21	B	2
3	2	STELLA	20	A+	3
4	1	APPU	20	A+	4

Query executed successfully.

DESKTOP-HPGQ4S\SQLEXPRESS ... DESKTOP-HPGQ4S\DELL (61) dataclean 00:00:00 4 rows



## 2) RANK():

RANK() SQL Rank function to specify rank for each row in the result set.

**Using Partition: ( divide data into smaller subset)**

Use partition by age

Each subset should get rank as per id in descending order

Uses order by clause to sort based on name and rank.

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
-- Age order by in descending order
select id,name,age,grade,ROW_NUMBER() OVER(order by age desc) RowNumberDesc
from studentdata1;

-- Using Partition
select id,name,age,grade,RANK() OVER(PARTITION by age order by id desc) Rank
from studentdata1
order by name,rank;
```

The Results pane shows the output of the second query:

id	name	age	grade	Rank
1	APPU	20	A+	2
2	BOB	21	C	2
3	STELLA	20	A+	1
4	BUNNY	21	B	1

## Without partitioning

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
order by name,rank;

-- Without partition
select id,name,age,grade,RANK() OVER(order by id desc) Rank
from studentdata1
order by rank;
```

The Results pane shows the output of the second query:

id	name	age	grade	Rank
1	BUNNY	21	B	1
2	BOB	21	C	2
3	STELLA	20	A+	3
4	APPU	20	A+	4

If ids are same then same rank is assigned.

### 3) DENSE\_RANK()

DENSE\_RANK() function to specify a unique rank number within the partition as per the specified column value.

In the SQL RANK function DENSE\_RANK(), if we have duplicate values, SQL assigns different ranks to those rows as well.

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
-- DENSE_RANK
select id,name,age,grade,DENSE_RANK() OVER(order by id desc) Rank
from studentdata1
order by rank;
```

The Results pane displays the output of the query:

id	name	age	grade	Rank
1	BOB	21	C	1
2	SUNNY	21	B	2
3	STELLA	20	A+	3
4	APPU	20	A+	4

### Using partition

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
-- order by age
select id,name,age,grade,DENSE_RANK() OVER(order by age desc) Rank
from studentdata1
order by rank;

-- use partition
select id,name,age,grade,DENSE_RANK() OVER(partition by grade order by id desc) Rank
from studentdata1
order by rank;
```

The Results pane displays the output of the first query (order by age desc):

id	name	age	grade	Rank
1	BOB	21	C	1
2	SUNNY	21	B	1
3	STELLA	20	A+	2
4	APPU	20	A+	2

The Results pane also displays the output of the second query (partition by grade order by id desc):

id	name	age	grade	Rank
1	STELLA	20	A+	1
2	SUNNY	21	B	1
3	BOB	21	C	1
4	APPU	20	A+	2

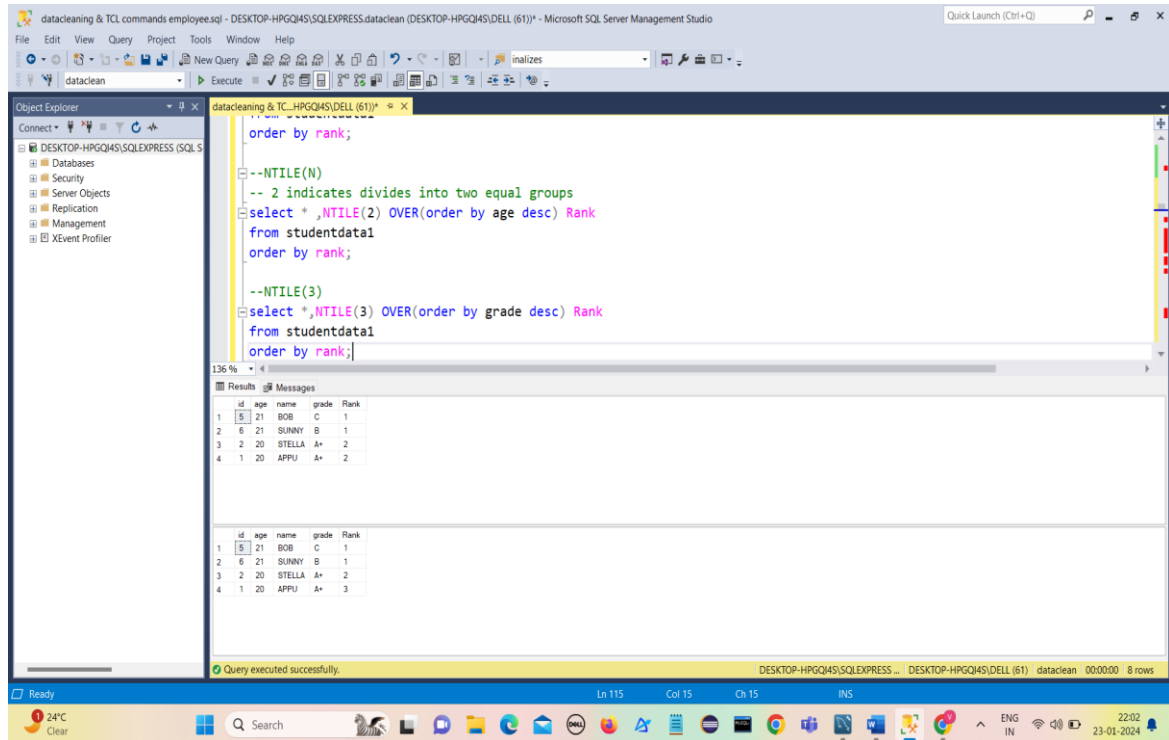
#### 4) NTILE(N) RANK FUNCTION:

NTILE(N) function to distribute the number of rows in the specified (N) number of groups.

Each row group gets its rank as per the specified condition.

We need to specify the value for the desired number of groups.

NTILE(2)-means divides the table into two groups



The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains two SQL queries using the NTILE function. The first query uses NTILE(2) to divide rows into two groups based on age, and the second query uses NTILE(3) to divide rows into three groups based on grade. Both queries use the RANK function to assign ranks within each group.

```
--NTILE(N)
-- 2 indicates divides into two equal groups
select *, NTILE(2) OVER(order by age desc) Rank
from studentdata1
order by rank;

--NTILE(3)
select *, NTILE(3) OVER(order by grade desc) Rank
from studentdata1
order by rank;
```

The Results pane shows two tables of data. The first table is the result of the first query, showing rows ranked by age. The second table is the result of the second query, showing rows ranked by grade.

id	age	name	grade	Rank
1	21	BOB	C	1
2	21	SUNNY	B	1
3	20	STELLA	A+	2
4	20	APPU	A+	2

id	age	name	grade	Rank
1	21	BOB	C	1
2	21	SUNNY	B	1
3	20	STELLA	A+	2
4	20	APPU	A+	3

#### STORED PROCEDURE:

An SQL **stored procedure** is a group of pre-compiled SQL statements (prepared SQL code) that can be reused by simply calling it whenever needed.

##### Syntax:

CREATE PROCEDURE *procedure\_name*

AS

BEGIN

*sql\_statement*

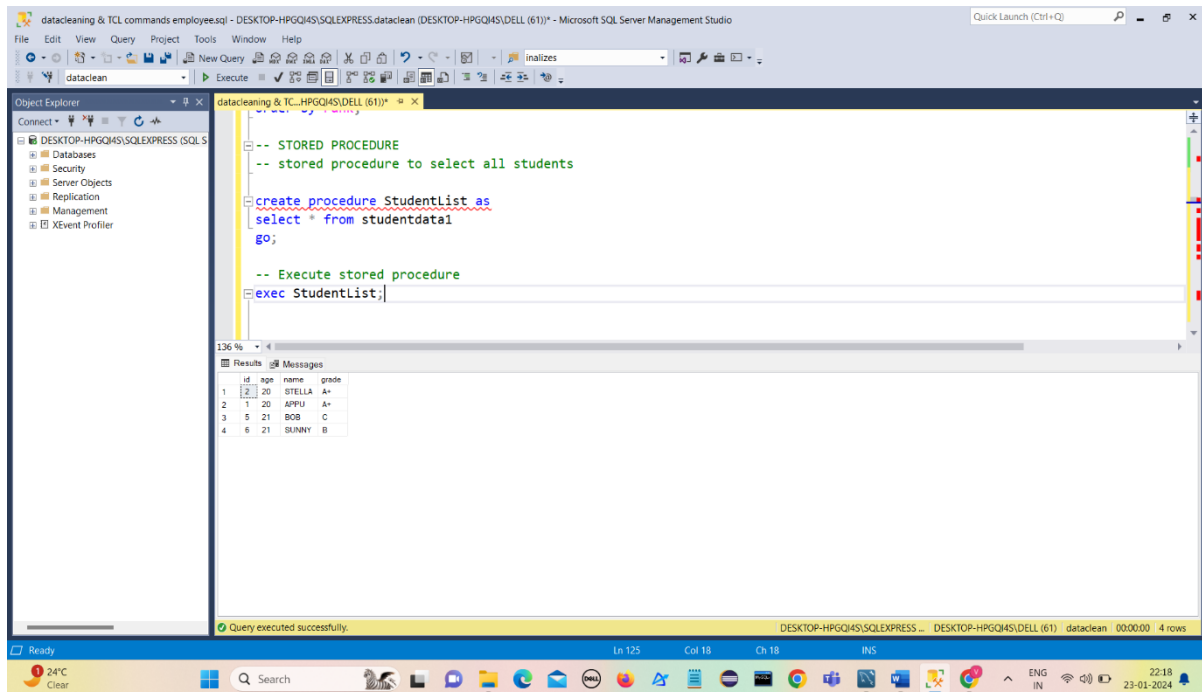
END

GO;

##### Execute a Stored Procedure

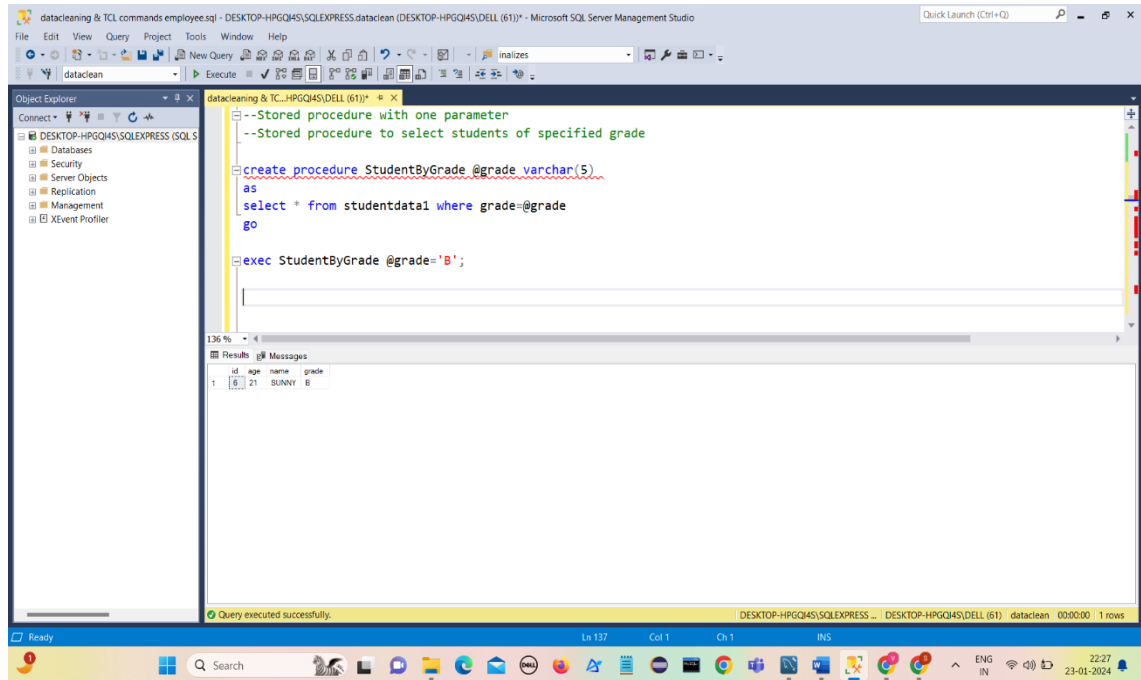
EXEC *procedure\_name*;

## 1) Stored procedure to select all students



## 2) Stored procedure with one parameter

Stored procedure to select students of specified grade.



### 3) Stored procedure with multiple parameters.

Create stored procedure to get students of specific age and id.

