`

AMRITA
VISHWA VIDYAPEETHAM
AHEAD Online

SQL Statements

Objectives

- Familiarize SQL Statements:

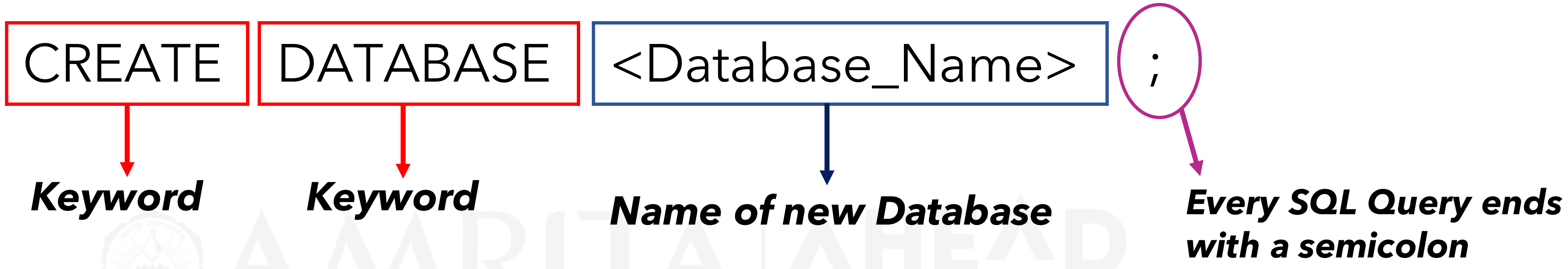
CREATE Command

- Data Definition Language (DDL) SQL command.
- Used to create a table or a database in RDBMS.
- There are two CREATE statements in SQL:
  - CREATE DATABASE
  - CREATE TABLE

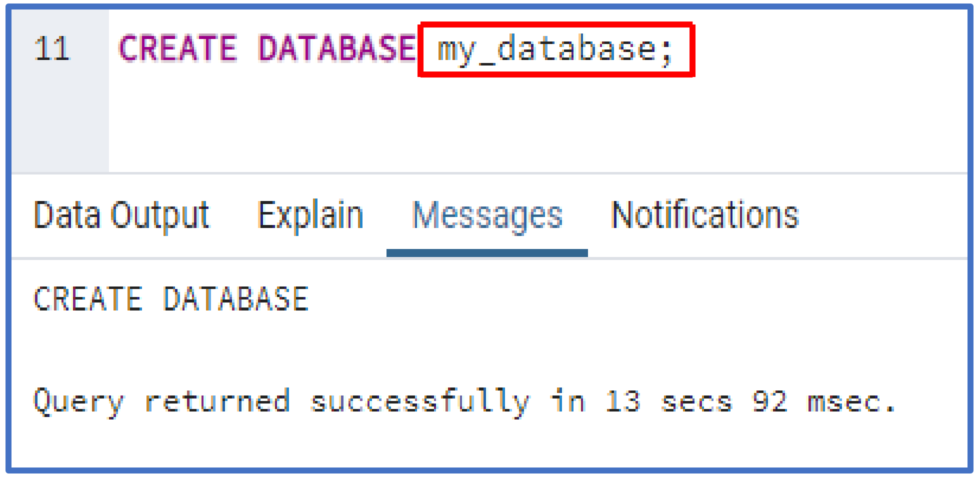
CREATE DATABASE

- A database is defined as a structured set of data.
- To create a database in RDBMS, create command is used.

Syntax :

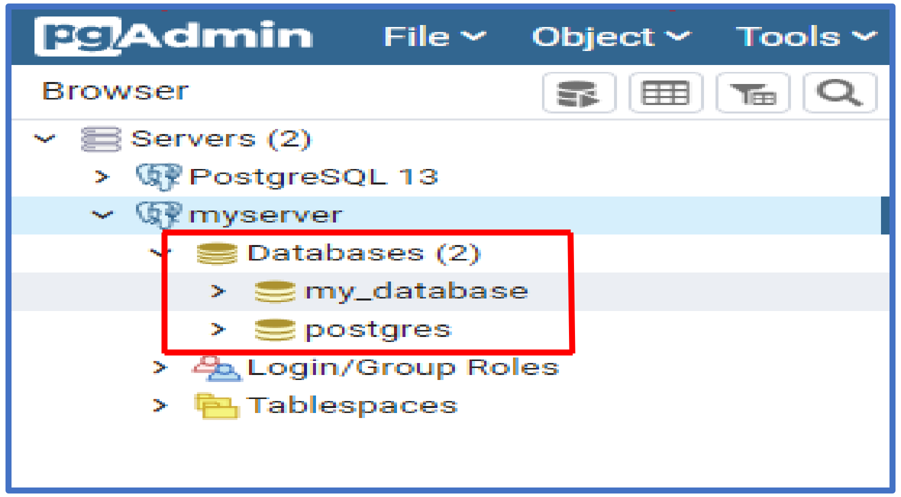


Example: Creating DATABASE in pgAdmin



```
CREATE DATABASE my_database;
```

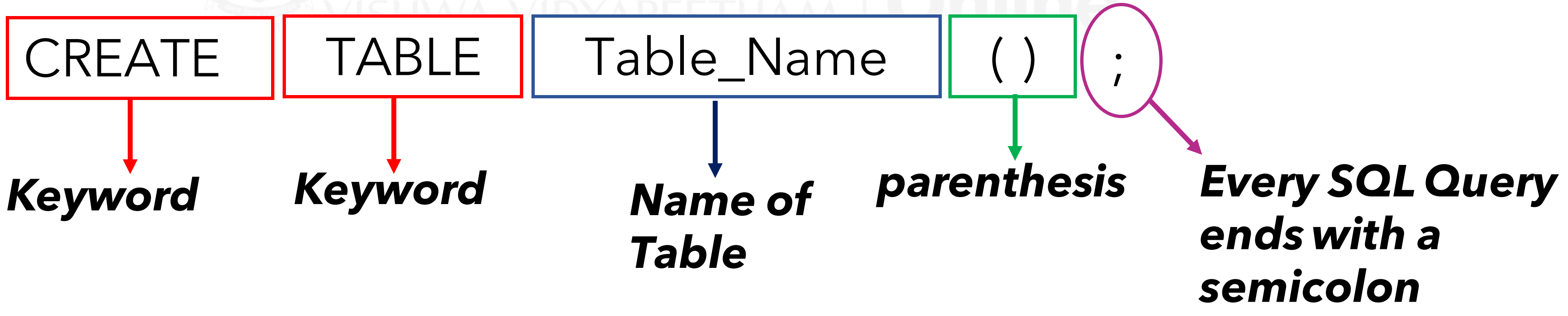
```
CREATE DATABASE

Query returned successfully in 13 secs 92 msec.
```



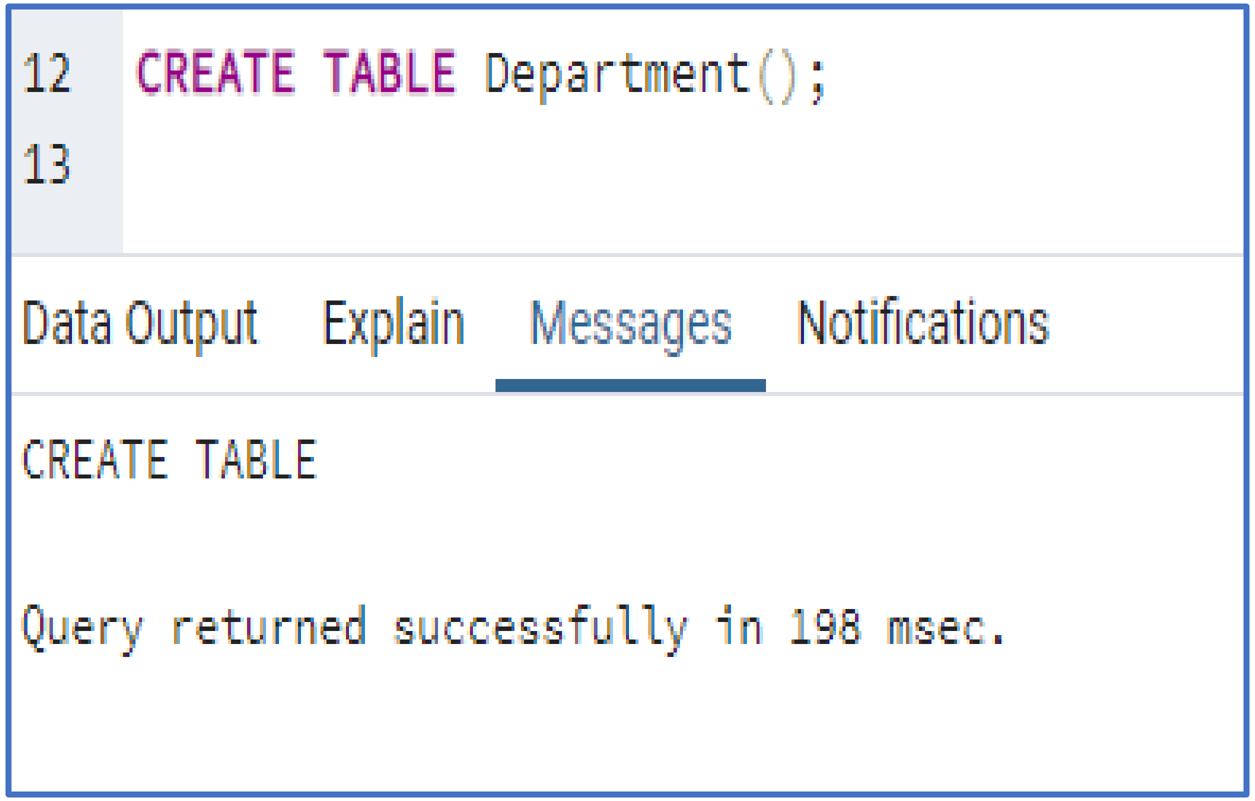
CREATE TABLE

- Create command can also be used to create tables.
- Specify the details of the columns of the tables.
- Specify the **names** and **datatypes** of various columns.

Syntax : CREATE TABLE without table structure



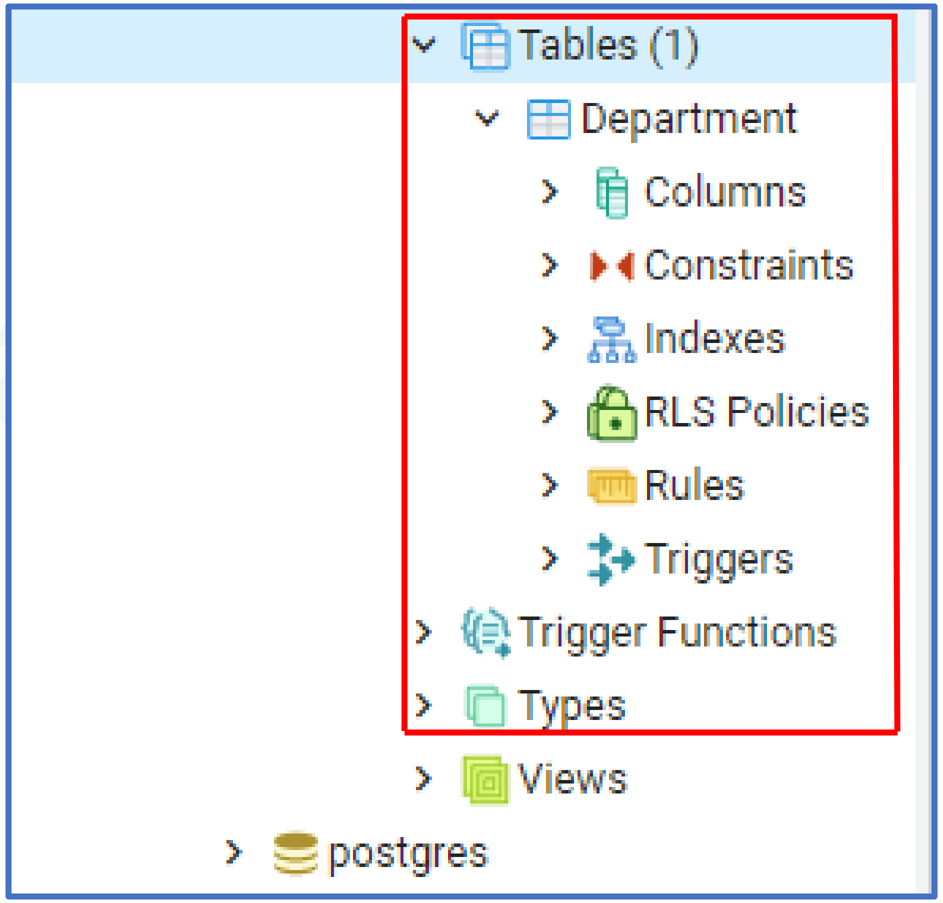
Example:
Creating a TABLE without table structure



```
CREATE TABLE Department();
```

Data Output | Explain | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 198 msec.
```

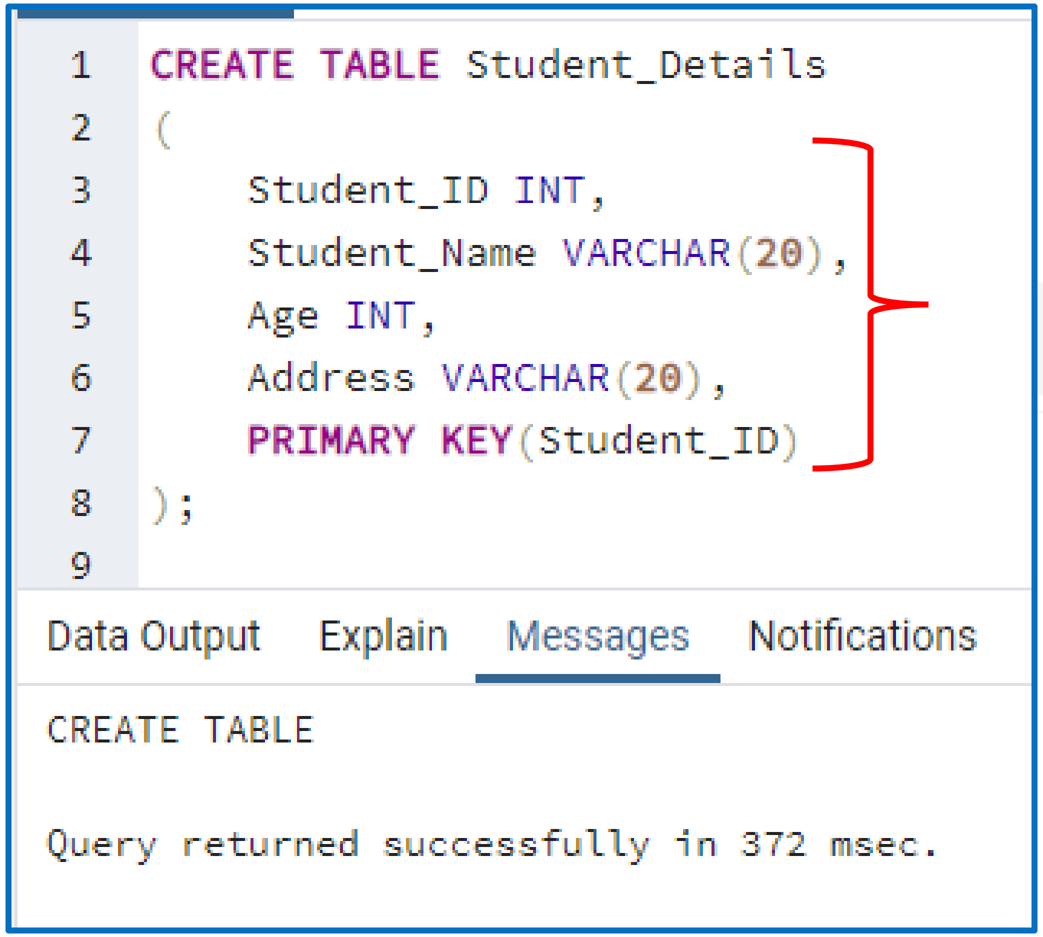


CREATE TABLE

Syntax : CREATE TABLE with table structure

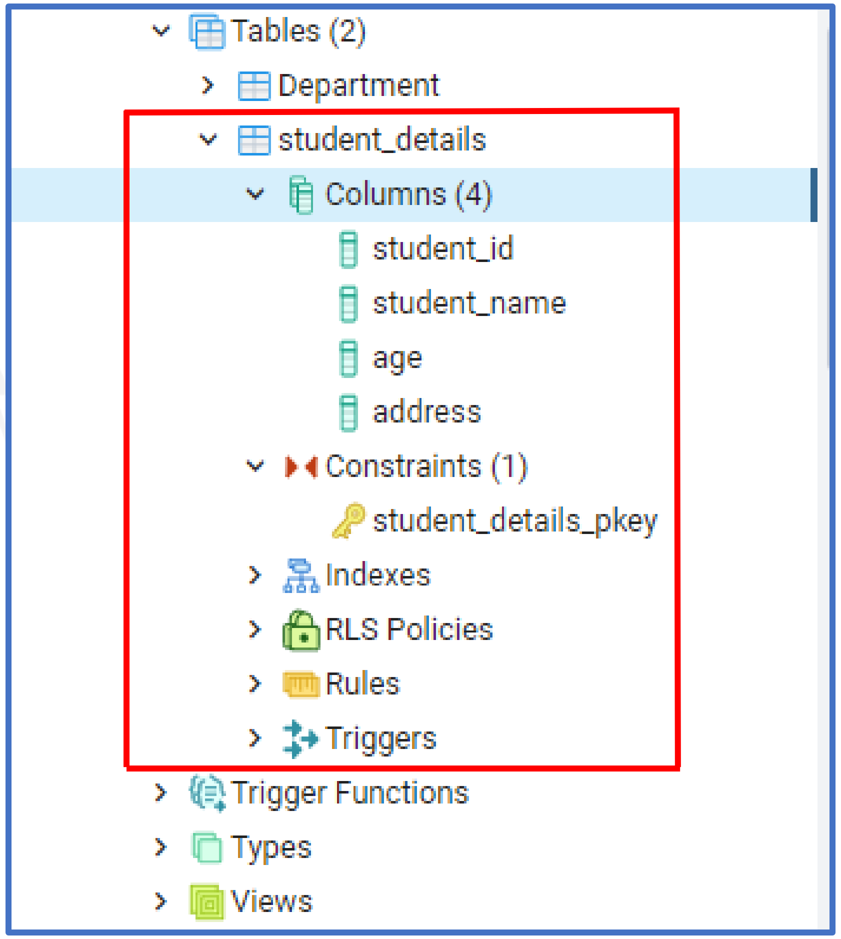
CREATE	TABLE	<Table_Name>	(column1 datatype, column2 datatype,….)	;
Keyword	***Keyword***	***Name of Table***	***Name and Datatype of columns present in the tables***	***Every SQL Query ends with a semicolon***

Example:
Creating a TABLE with table structure



```
CREATE TABLE Student_Details
(
    Student_ID INT,
    Student_Name VARCHAR(20),
    Age INT,
    Address VARCHAR(20),
    PRIMARY KEY(Student_ID)
);
```

```
CREATE TABLE

Query returned successfully in 372 msec.
```



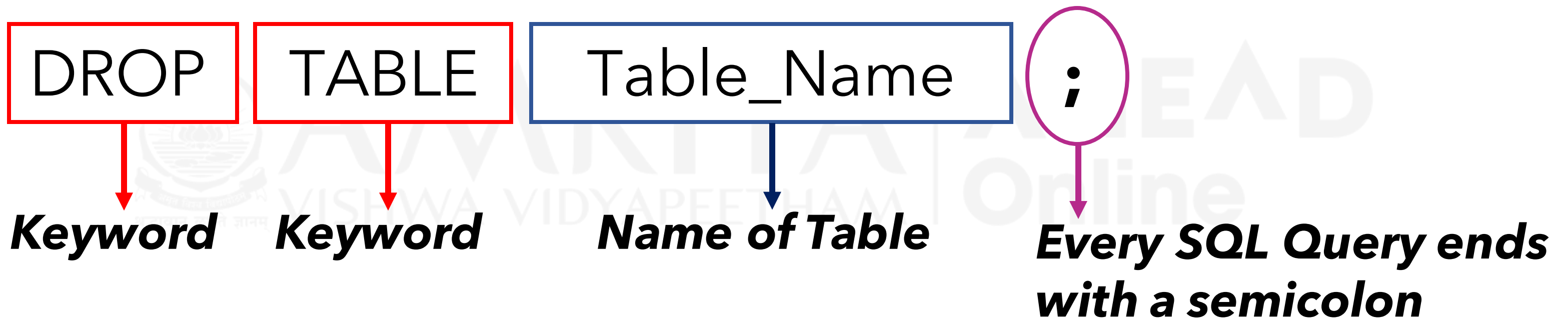
Data Types

Datatype	Use
INT	used for columns which will store integer values.
FLOAT	used for columns which will store float values.
DOUBLE	used for columns which will store float values.
VARCHAR	used for columns which will be used to store characters and integers, basically a string.
CHAR	used for columns which will store char values(single character).
DATE	used for columns which will store date values.
TEXT	used for columns which will store text which is generally long in length.

DROP TABLE

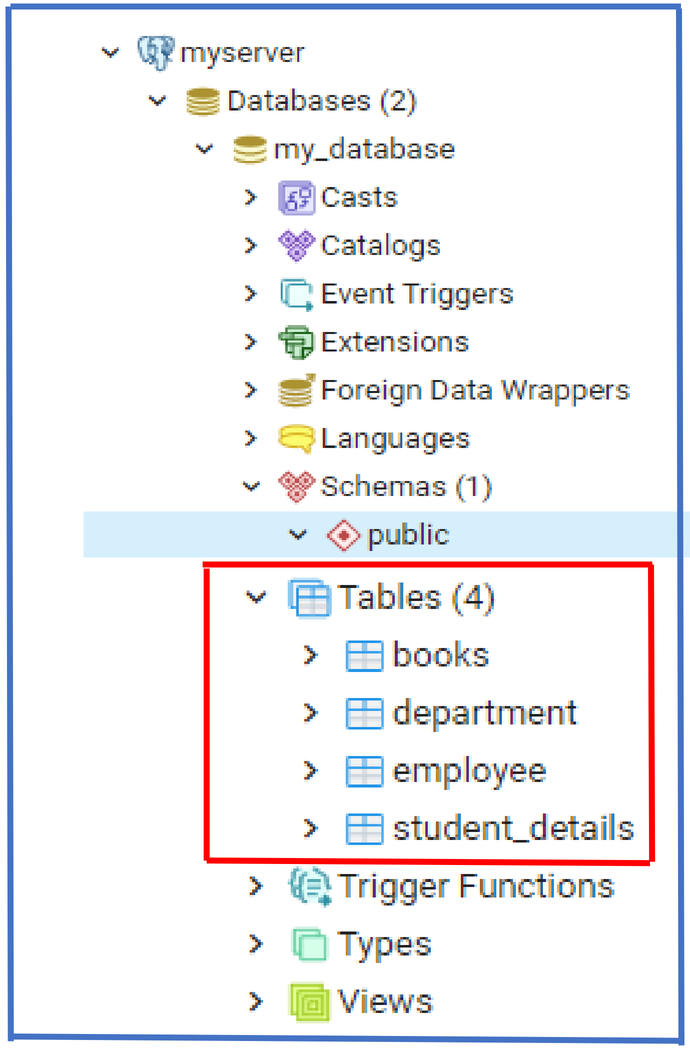
- It will destroy the table and all data which will be recorded in it.

Syntax :

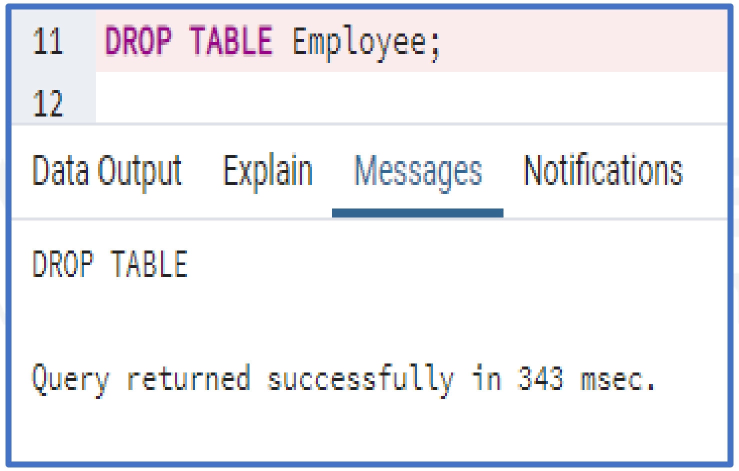


Example: Drop a table in a database

1 ***List of tables in Database***



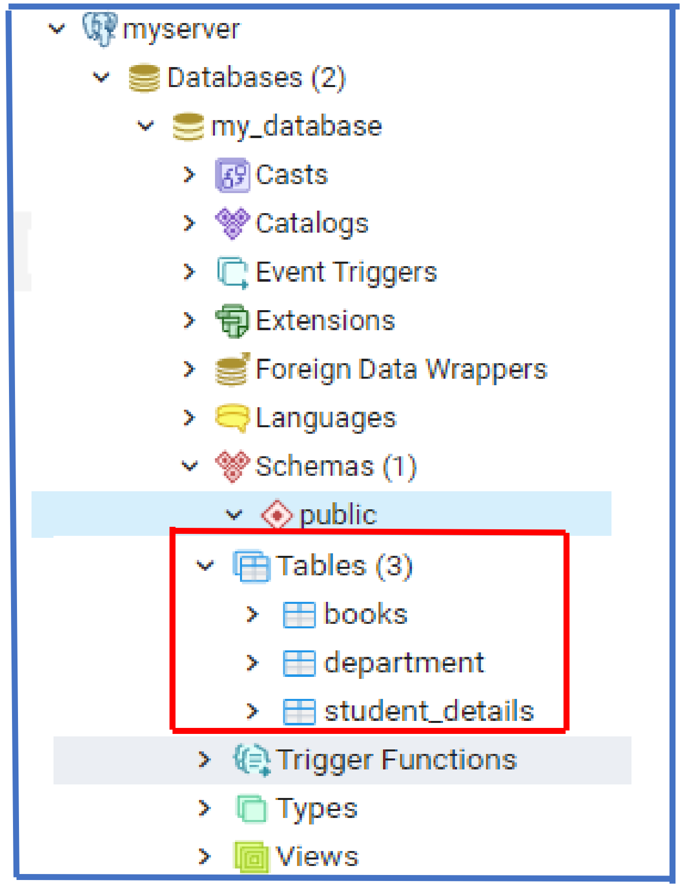
2 ***DROP Command***



```
DROP TABLE Employee;
```

```
DROP TABLE

Query returned successfully in 343 msec.
```

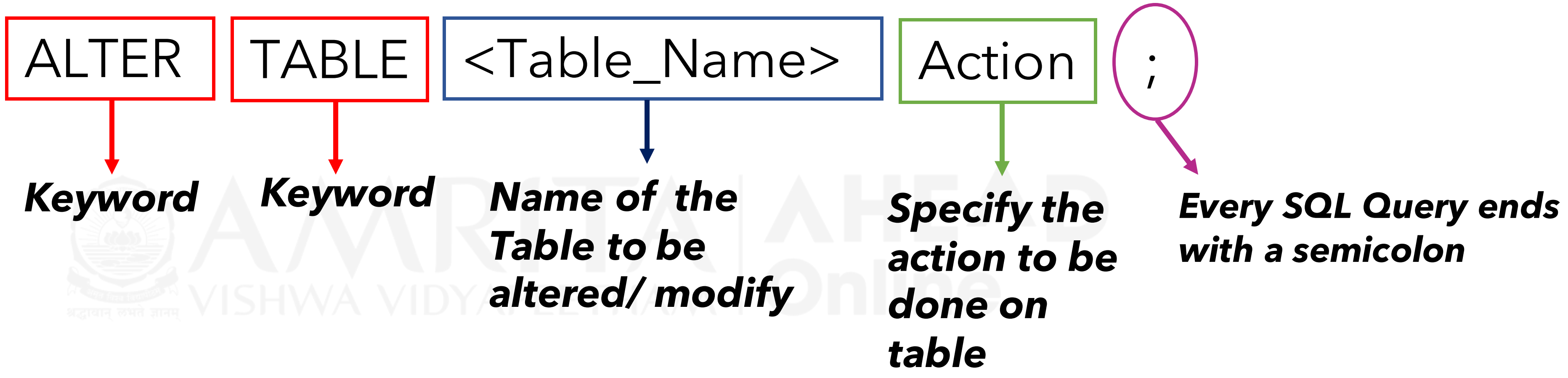
3 ***Updated list of tables in Database***



“ALTER TABLE” Statement

- This statement is used to add, modify or delete constraints or columns.

Syntax :


ALTER
TABLE
<Table_Name>
Action
;
Keyword
Keyword
Name of the Table to be altered/ modify
Specify the action to be done on table
Every SQL Query ends with a semicolon

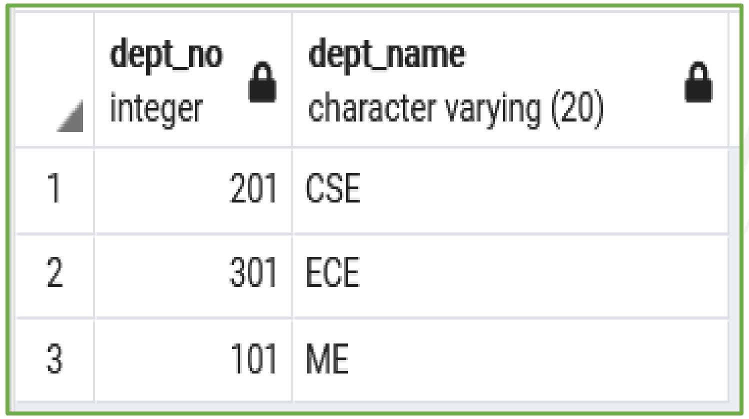
Action

- ➢ Add a column
- ➢ Drop a column
- ➢ Change the data type of a column
- ➢ Rename a column
- ➢ Set a default value for the column.
- ➢ Add a constraint to a column.
- ➢ Rename a table

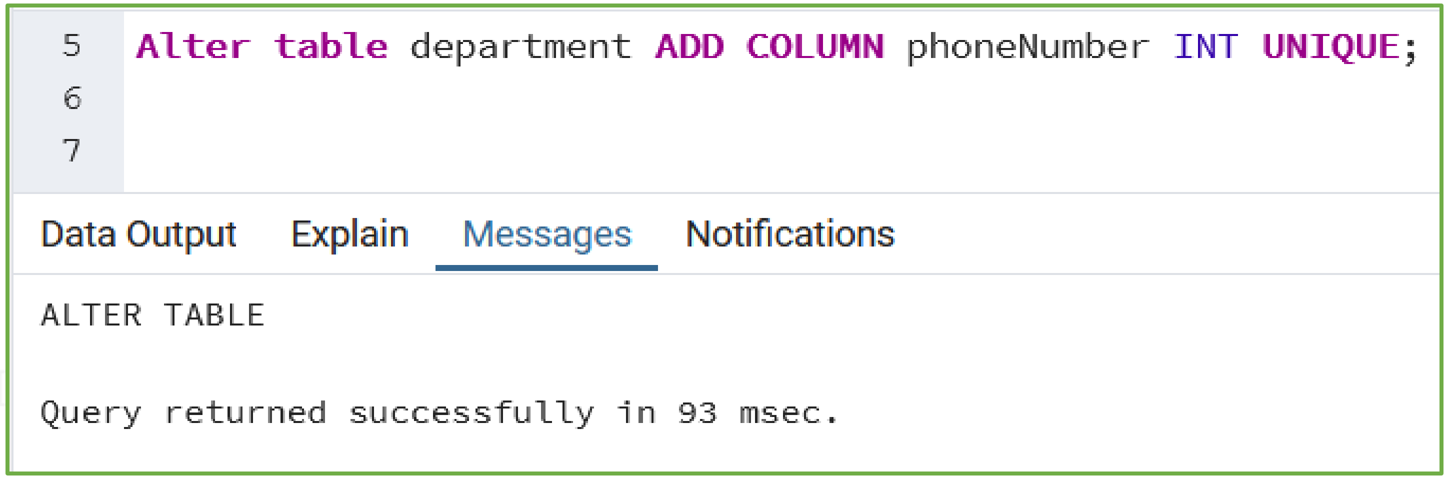
Example: Add a column to a Table

Syntax : ALTER TABLE table_name ADD COLUMN new_column_name data_type constraint;

Table : Department

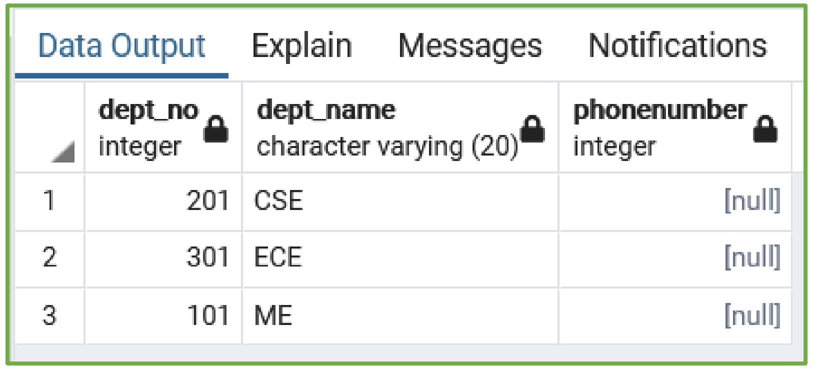


	dept_no integer	dept_name character varying (20)
1	201	CSE
2	301	ECE
3	101	ME



```
Alter table department ADD COLUMN phoneNumber INT UNIQUE;
```

```
ALTER TABLE

Query returned successfully in 93 msec.
```

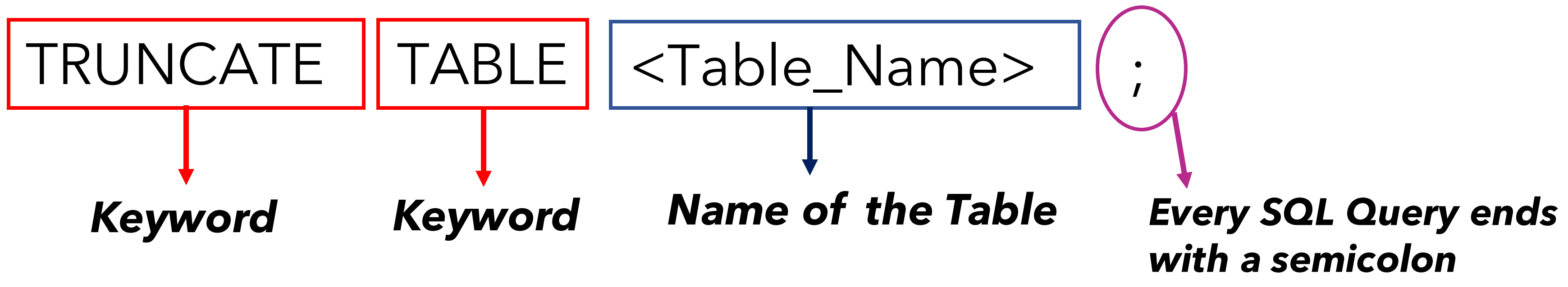


	dept_no integer	dept_name character varying (20)	phonenumber integer
1	201	CSE	[null]
2	301	ECE	[null]
3	101	ME	[null]

TRUNCATE TABLE Statement

- To remove all data from a table, you use the DELETE statement.
- However, when you use the DELETE statement to delete all data from a table that has a lot of data, it is not efficient.
- Therefore, you need to use the TRUNCATE TABLE statement.

Syntax :



TRUNCATE :
Remove all data from multiple tables

- To remove all data from multiple tables at once, you separate each table by a comma (,) .

Syntax :

```
TRUNCATE TABLE <Table_Name1>,
               <Table_Name2>, …  ;
```

TRUNCATE → ***Keyword***

TABLE → ***Keyword***

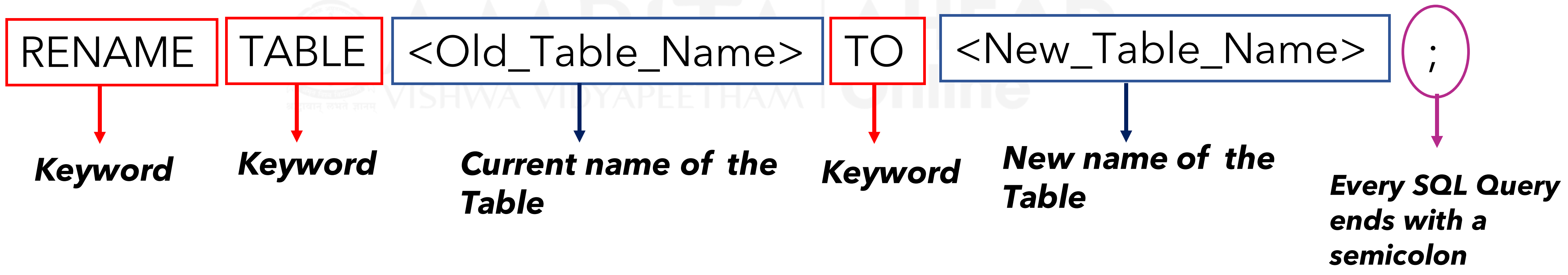
<Table_Name1>, <Table_Name2>, … → ***Names of the Table to delete the data***

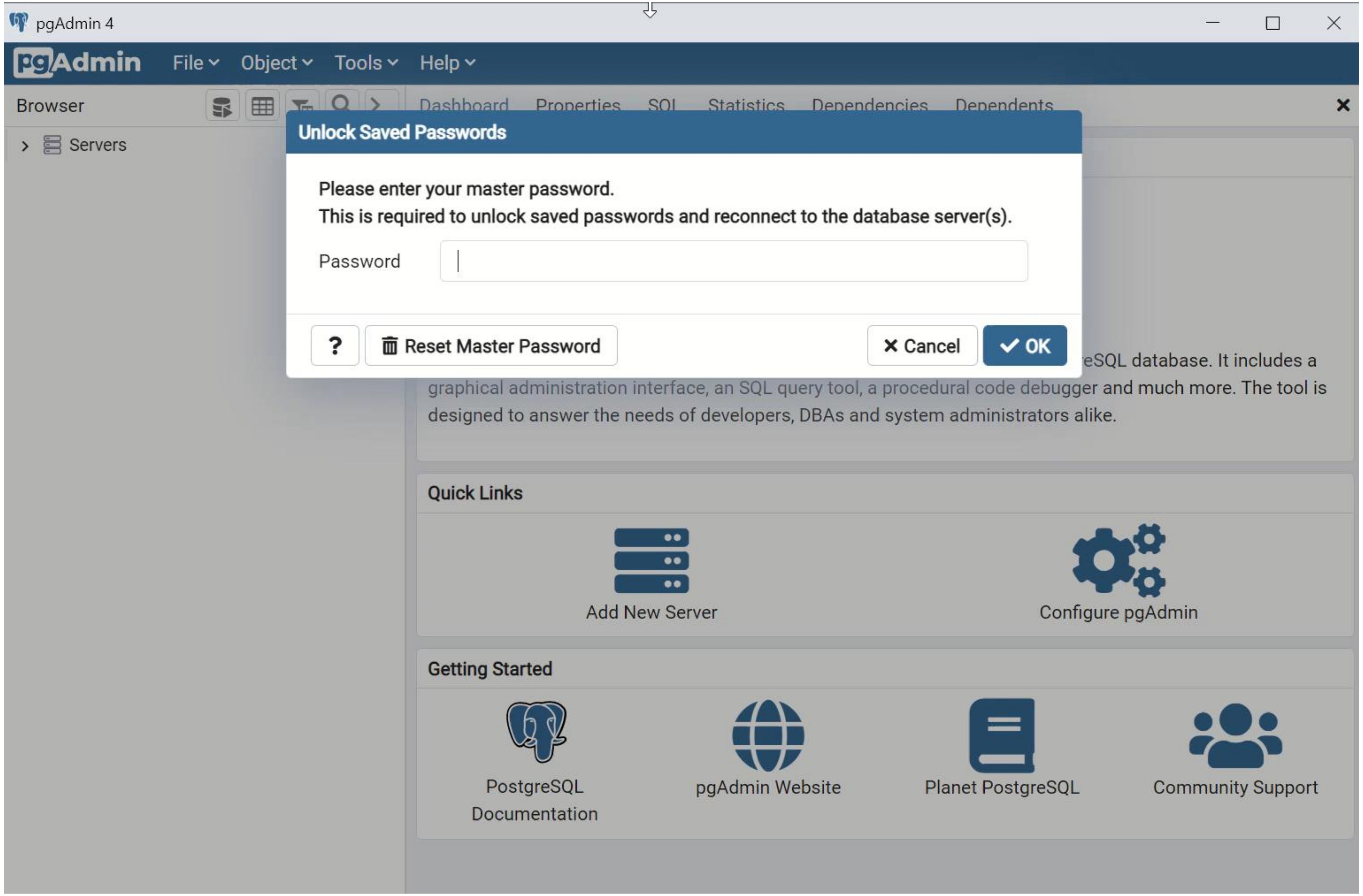
; → ***Every SQL Query ends with a semicolon***

RENAME Command

- It is used to set a new name for any existing table.

Syntax :

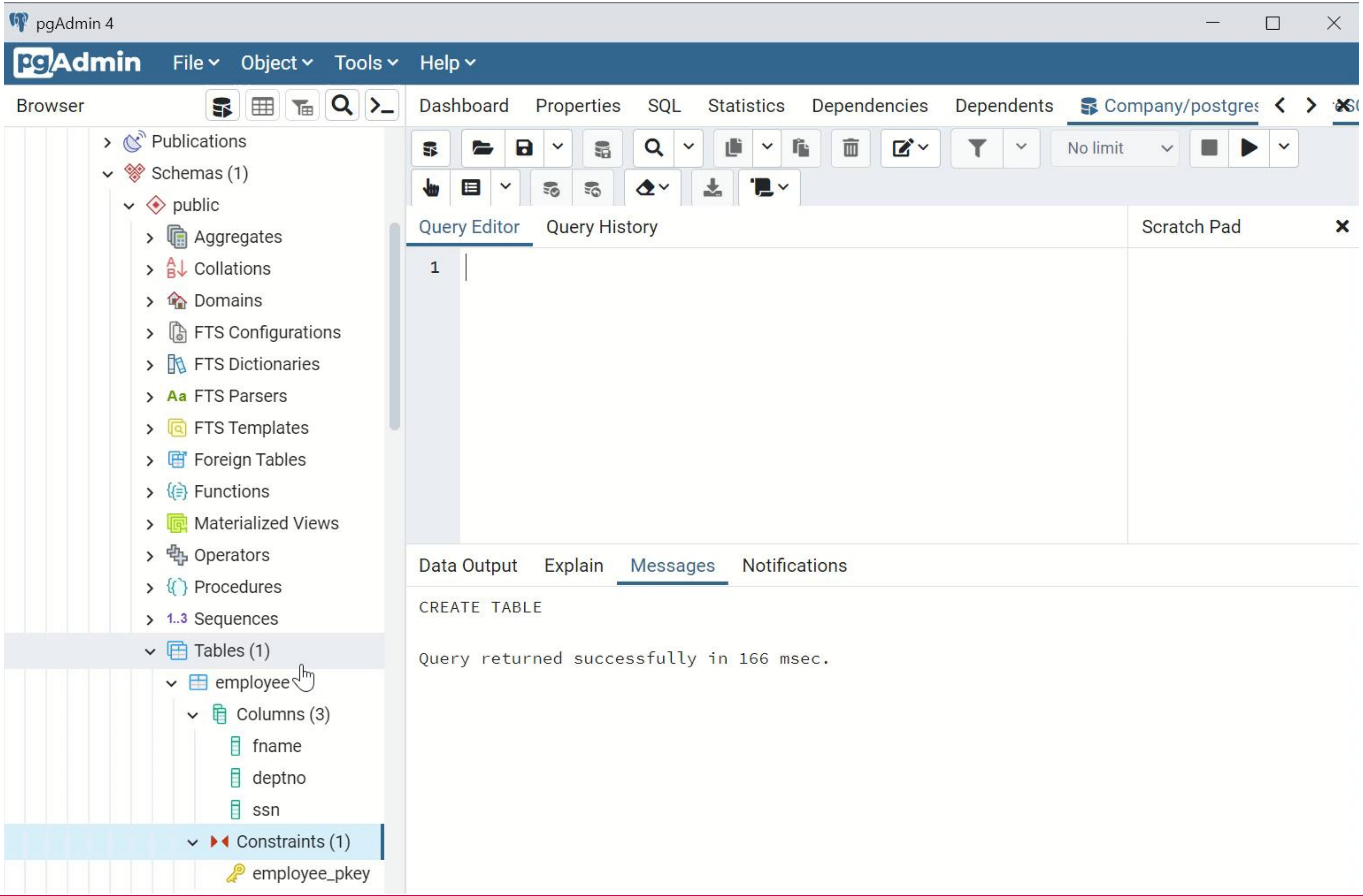




Please enter your master password.
This is required to unlock saved passwords and reconnect to the database server(s).

Password

…eSQL database. It includes a graphical administration interface, an SQL query tool, a procedural code debugger and much more. The tool is designed to answer the needs of developers, DBAs and system administrators alike.



```
CREATE TABLE

Query returned successfully in 166 msec.
```

“INSERT INTO” Statement

- Data Manipulation Language (DML) SQL command.
  - Used to store data in the table.
  - Adds a new record to the table.

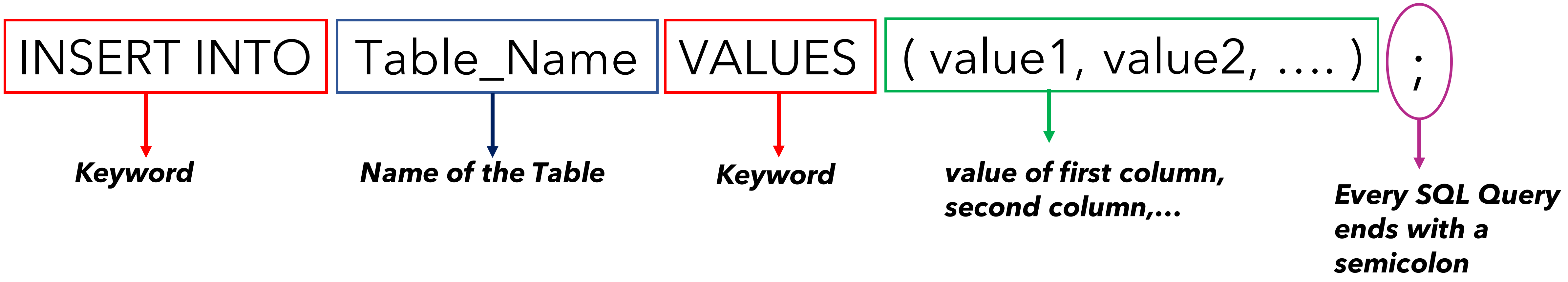
- Two ways of using INSERT INTO statement for inserting rows:

  1. Only values
  2. Column names and values (both)

INSERT INTO : Only Values

- Specify only the value of data to be inserted without the column names.
- ***Only values :*** Takes the advantage of the order of the columns when the table was created.

Syntax:

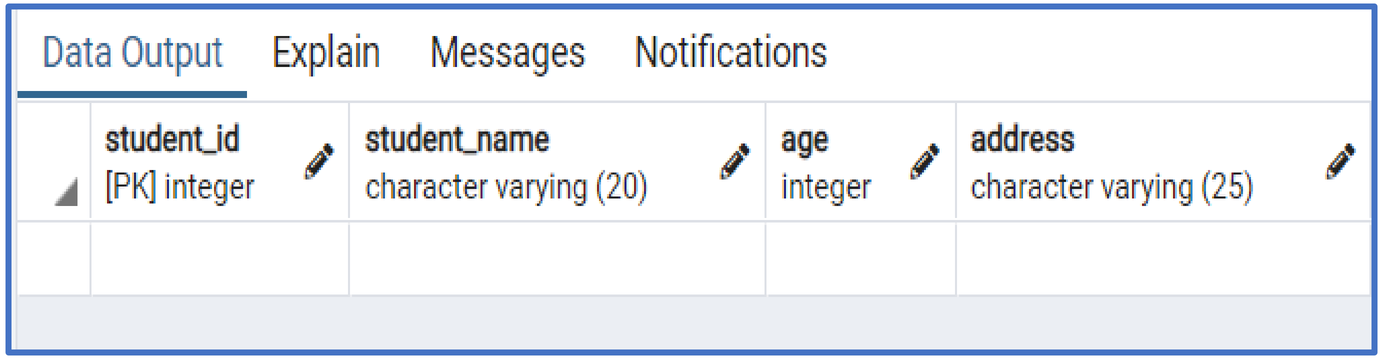


```
INSERT INTO Table_Name VALUES ( value1, value2, .... ) ;
```

Keyword
Name of the Table
Keyword
value of first column, second column,...
Every SQL Query ends with a semicolon

Example

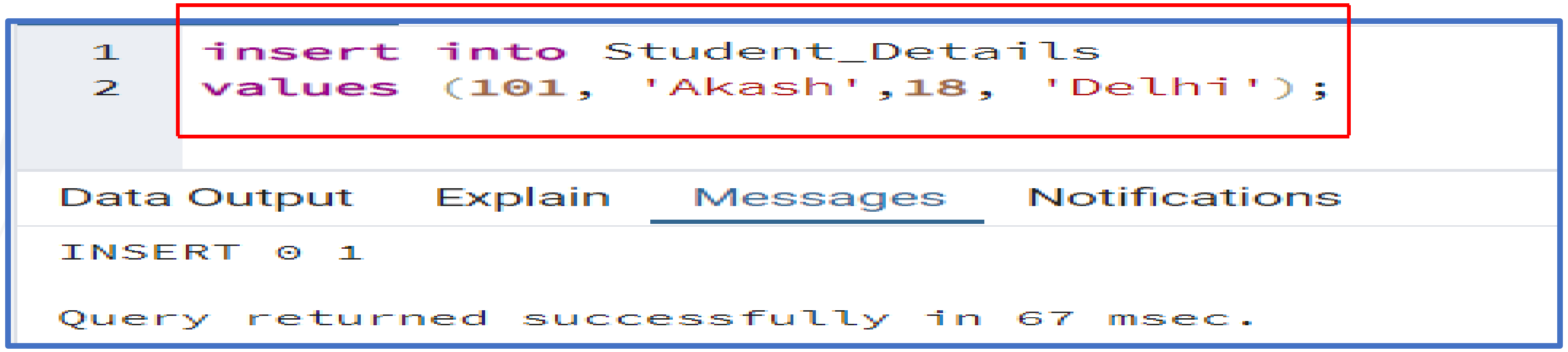
1 Database Table : **Student_Details**



Data Output | Explain | Messages | Notifications

	student_id [PK] integer	student_name character varying (20)	age integer	address character varying (25)

2 INSERT INTO : only values

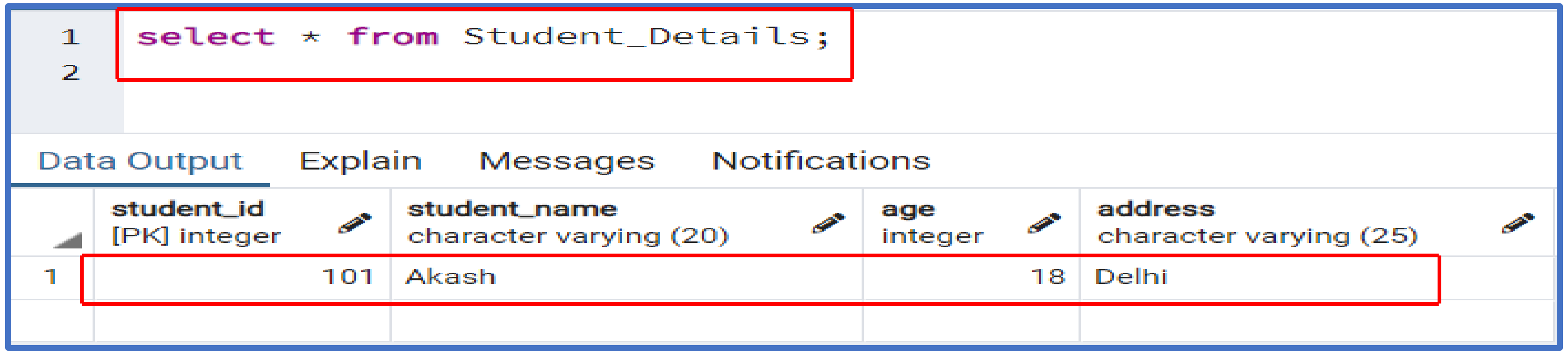


```
insert into Student_Details
values (101, 'Akash',18, 'Delhi');
```

Data Output | Explain | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 67 msec.
```

3 Database Table after inserting one row: **Student_Details**



```
select * from Student_Details;
```

Data Output | Explain | Messages | Notifications

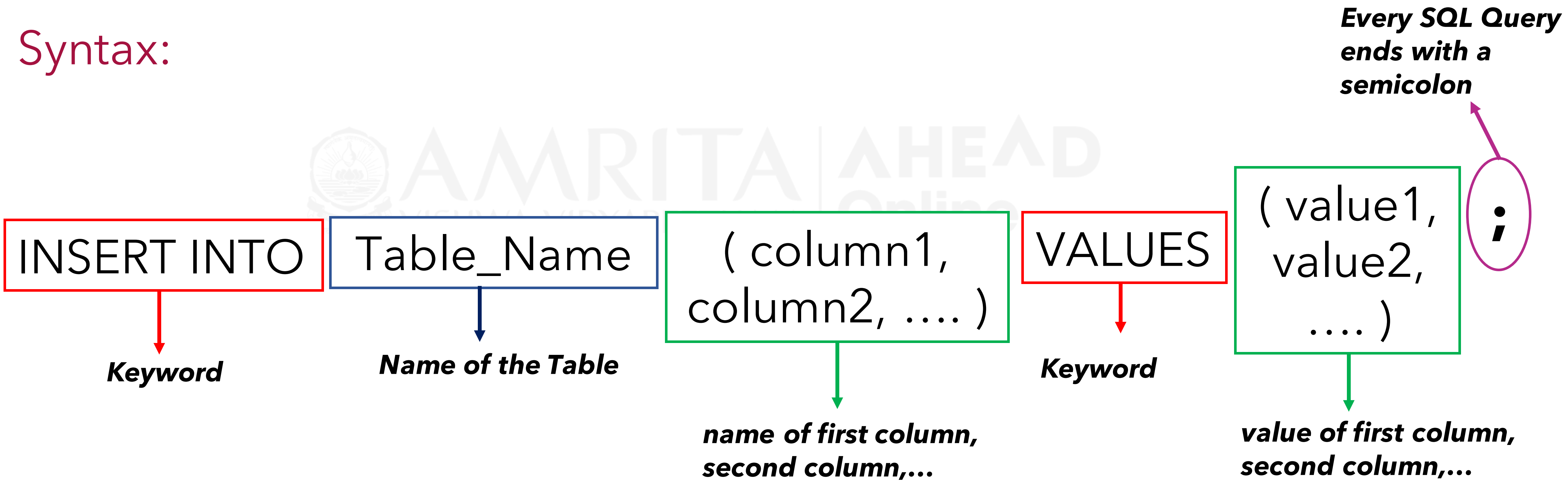
	student_id [PK] integer	student_name character varying (20)	age integer	address character varying (25)
1	101	Akash	18	Delhi

INSERT INTO :
Column names and Values (both)

- Specify both the columns which we want to fill and their corresponding values.
- The number of columns and values must be the same.
- If a column is not specified, the default value for the column is used.
- The values specified must satisfy all the applicable constraints such as primary keys.
- If a syntax error occurs or if any constraints are violated, the new row is not added to the table and an error is returned instead.

INSERT INTO : Column names and Values (both)

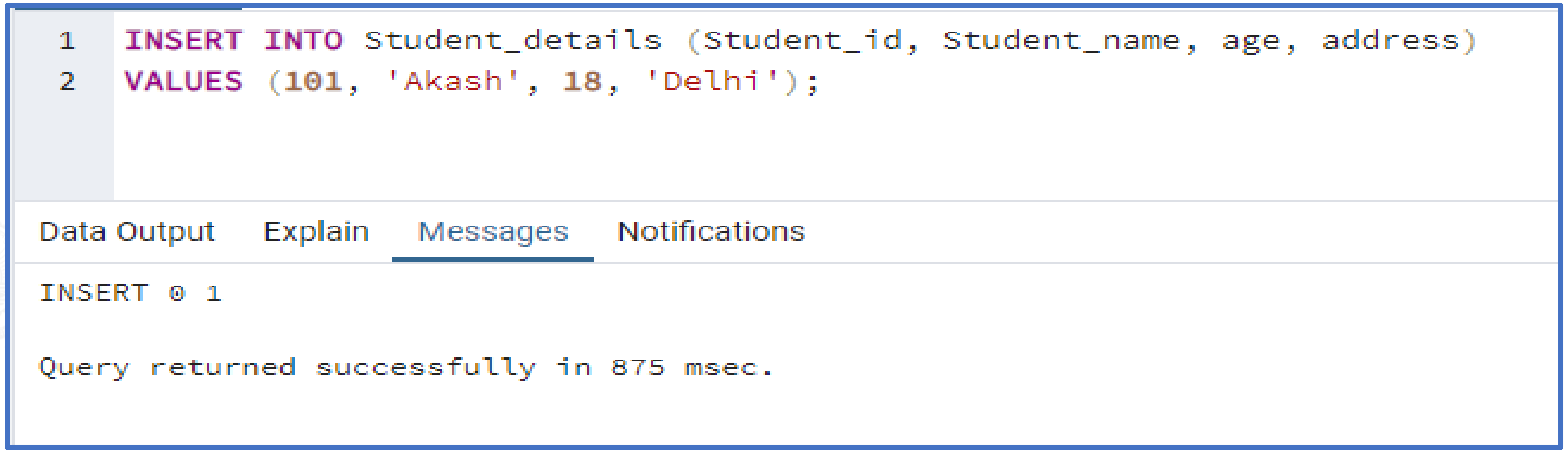
Syntax:



Every SQL Query ends with a semicolon
INSERT INTO
Table_Name
(column1, column2, ...)
VALUES
(value1, value2, ...)
;
Keyword
Name of the Table
name of first column, second column,...
Keyword
value of first column, second column,...

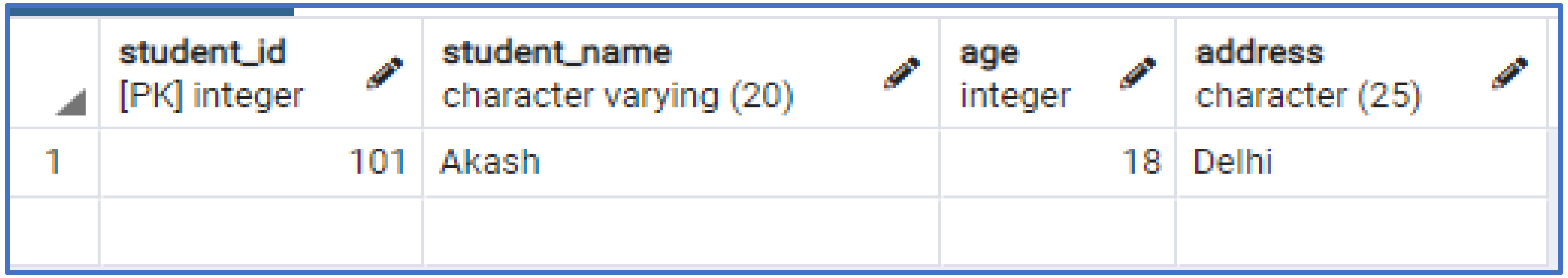
Example :
INSERT INTO column names and values

Query :


```
INSERT INTO Student_details (Student_id, Student_name, age, address)
VALUES (101, 'Akash', 18, 'Delhi');
```

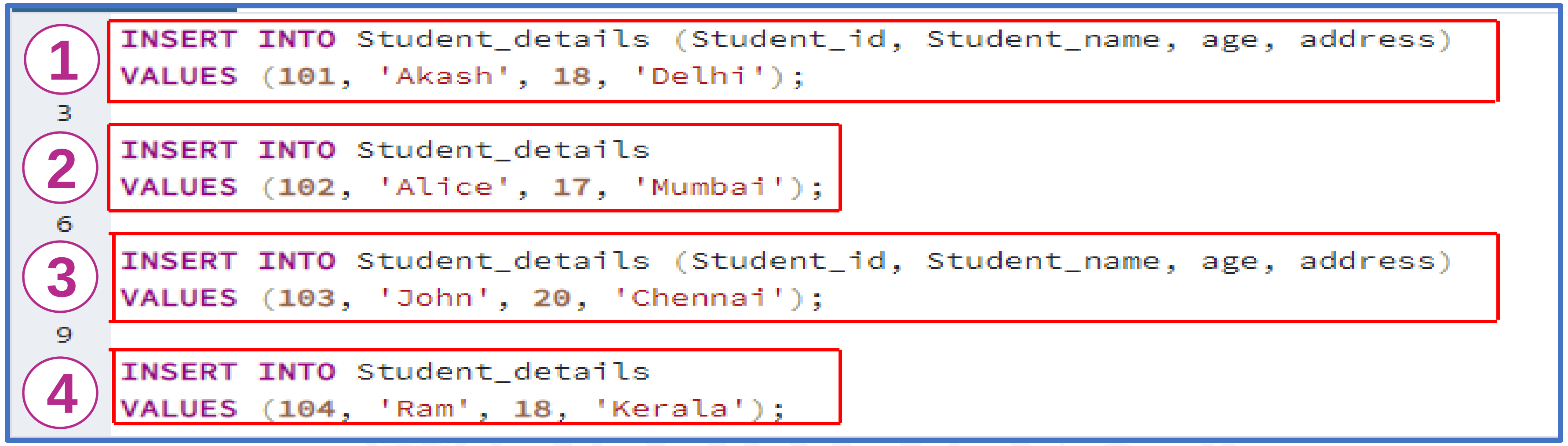
```
INSERT 0 1

Query returned successfully in 875 msec.
```

Database Table: **Student_details**



	student_id [PK] integer	student_name character varying (20)	age integer	address character (25)
1	101	Akash	18	Delhi

Database Table : Student_details



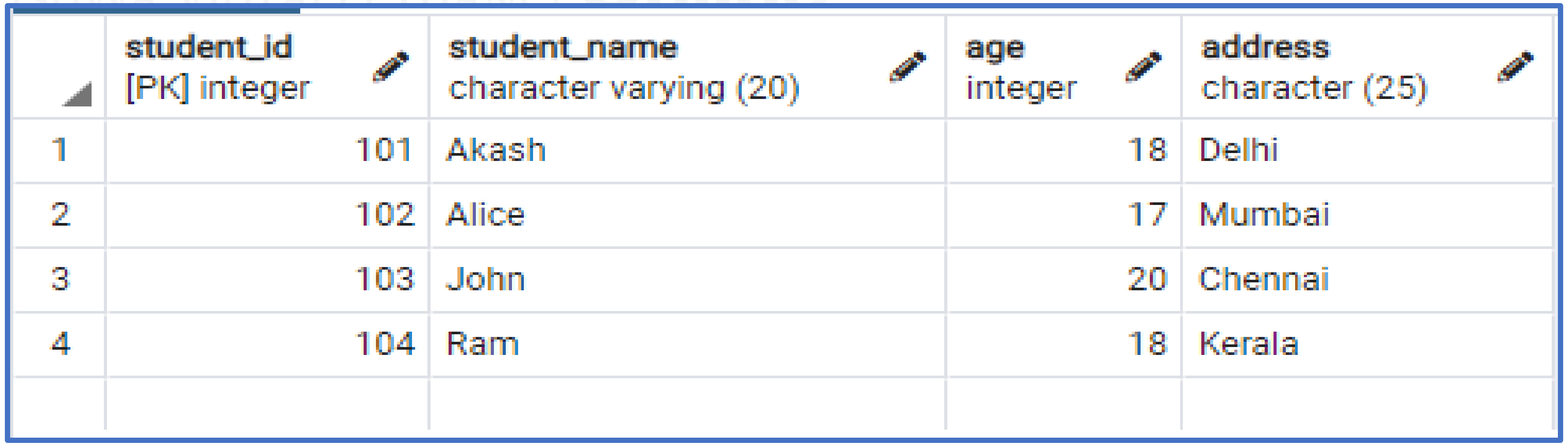
1
```
INSERT INTO Student_details (Student_id, Student_name, age, address)
VALUES (101, 'Akash', 18, 'Delhi');
```

2
```
INSERT INTO Student_details
VALUES (102, 'Alice', 17, 'Mumbai');
```

3
```
INSERT INTO Student_details (Student_id, Student_name, age, address)
VALUES (103, 'John', 20, 'Chennai');
```

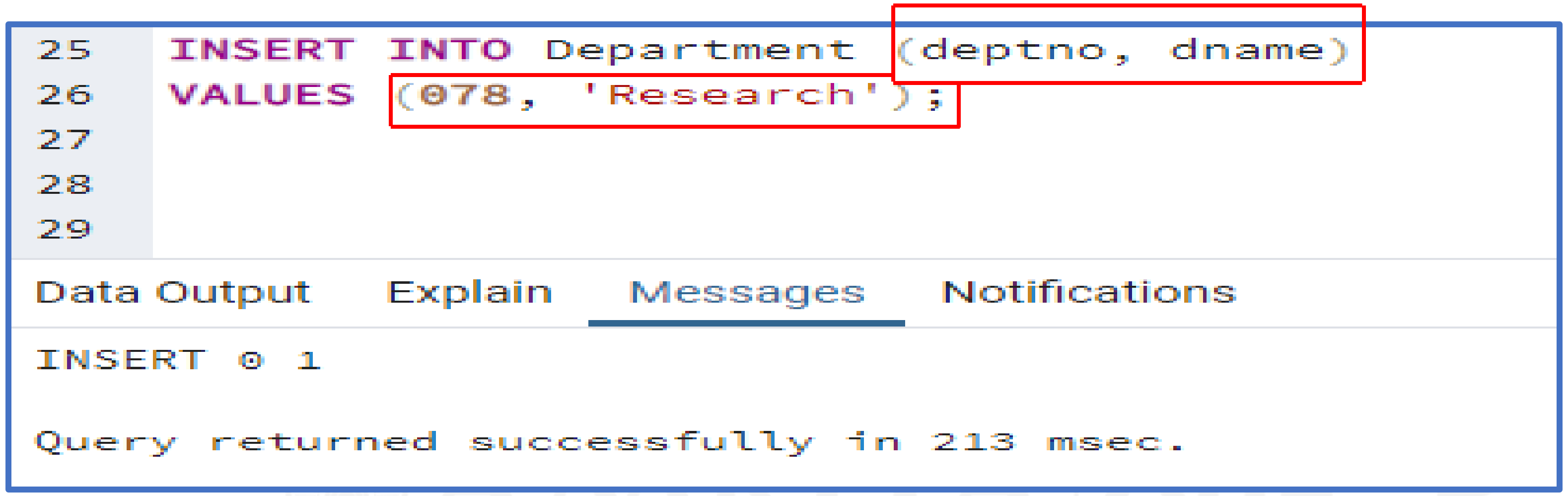
4
```
INSERT INTO Student_details
VALUES (104, 'Ram', 18, 'Kerala');
```

Database Table:
Student_details



	student_id [PK] integer	student_name character varying (20)	age integer	address character (25)
1	101	Akash	18	Delhi
2	102	Alice	17	Mumbai
3	103	John	20	Chennai
4	104	Ram	18	Kerala

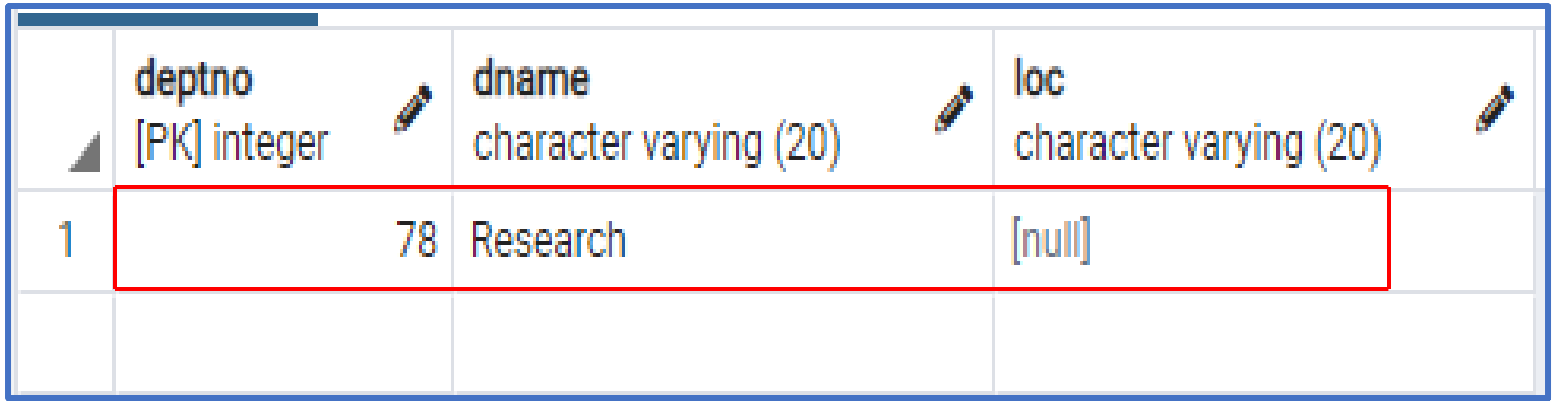
Insert into specific columns in a table



```
INSERT INTO Department (deptno, dname)
VALUES (078, 'Research');
```

```
INSERT 0 1

Query returned successfully in 213 msec.
```

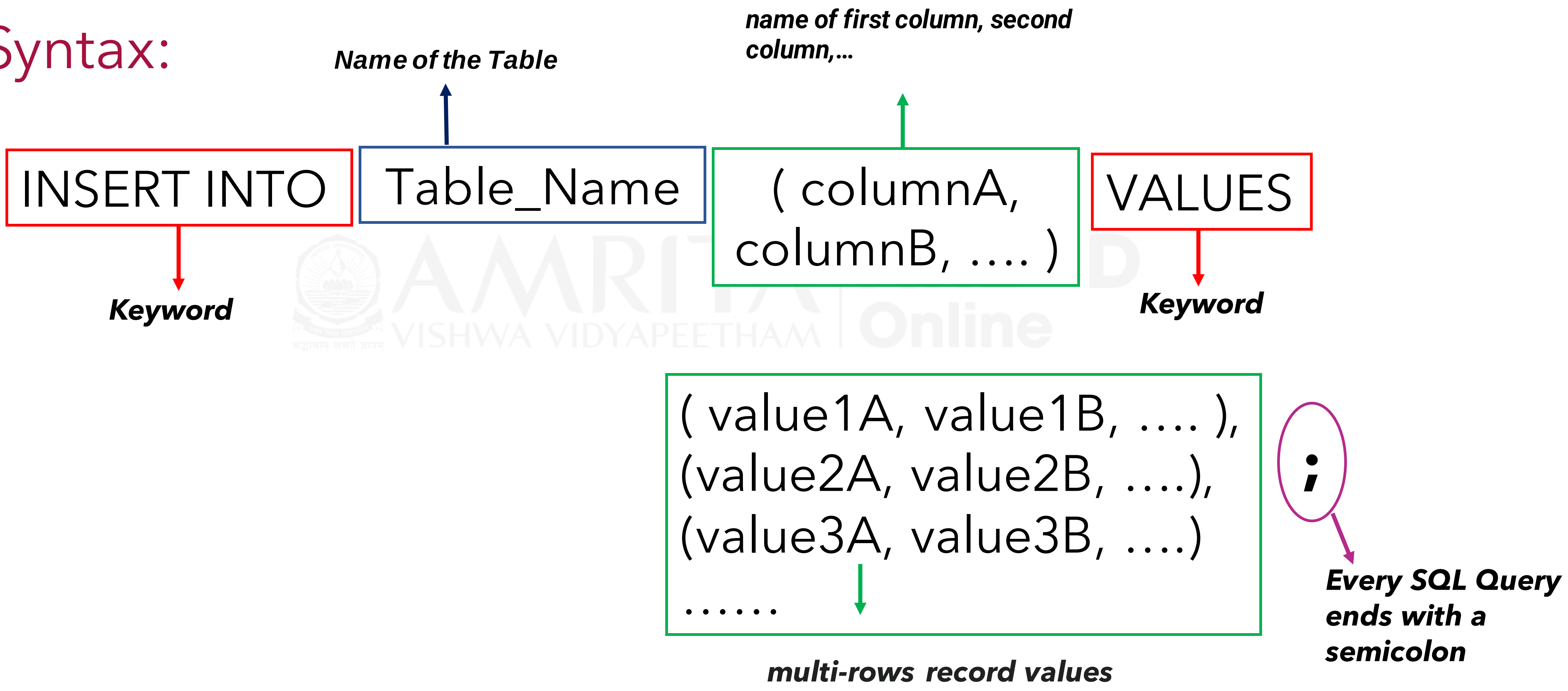
Database Table : Department



	deptno [PK] integer	dname character varying (20)	loc character varying (20)
1	78	Research	[null]

Insert multiple rows at a time in a single SQL statement

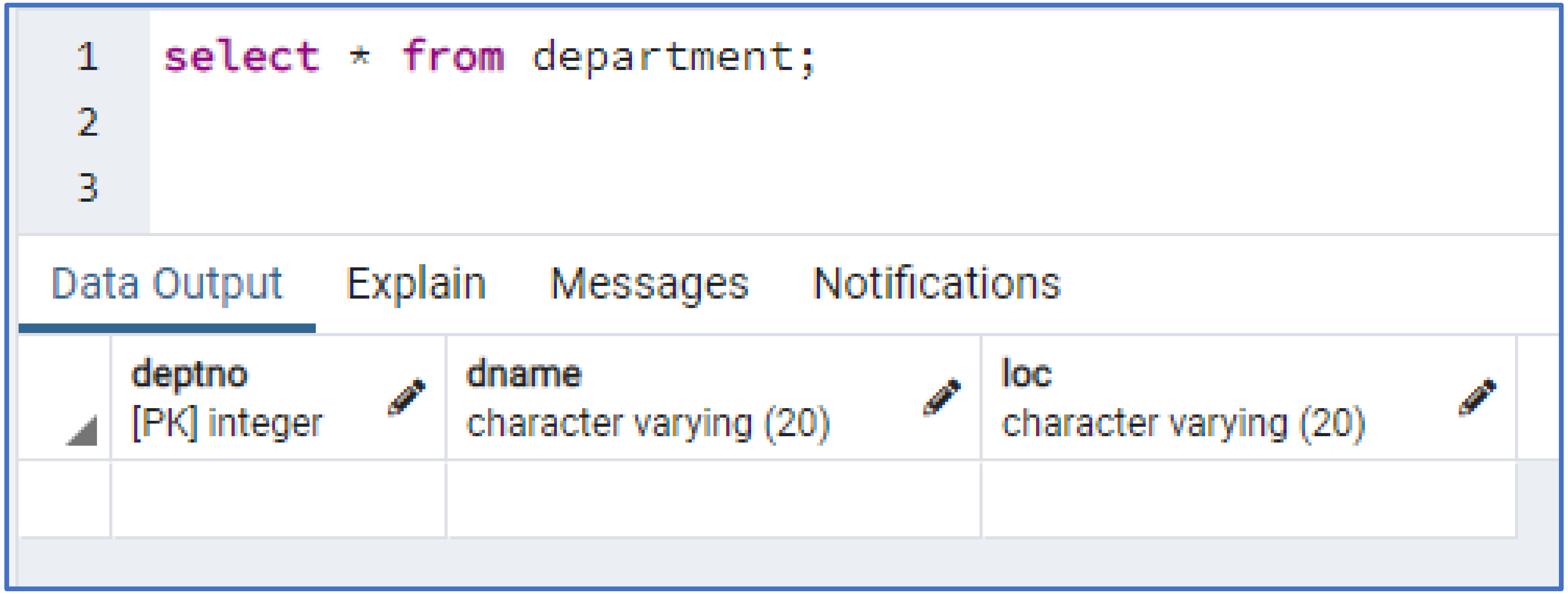
Syntax:



```
INSERT INTO Table_Name ( columnA, columnB, .... ) VALUES
( value1A, value1B, .... ),
(value2A, value2B, ....),
(value3A, value3B, ....)
......
;
```

Name of the Table
name of first column, second column,...
Keyword
Keyword
multi-rows record values
Every SQL Query ends with a semicolon

Database table “Department” with no records

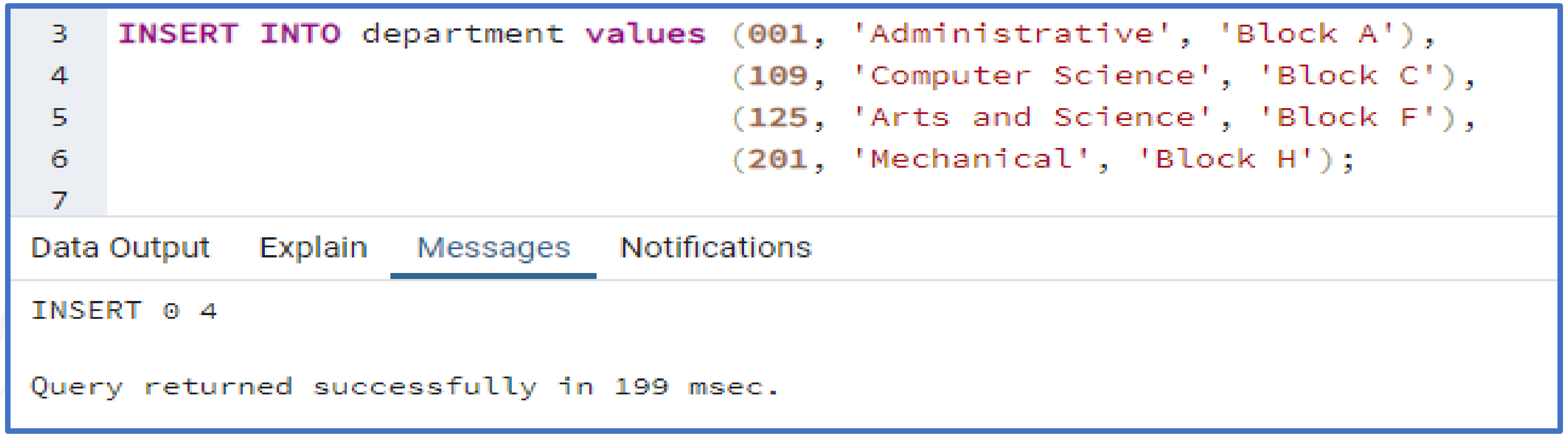


```
select * from department;
```

deptno [PK] integer	dname character varying (20)	loc character varying (20)

Insert multiple rows at a time into a Table

INSERT INTO

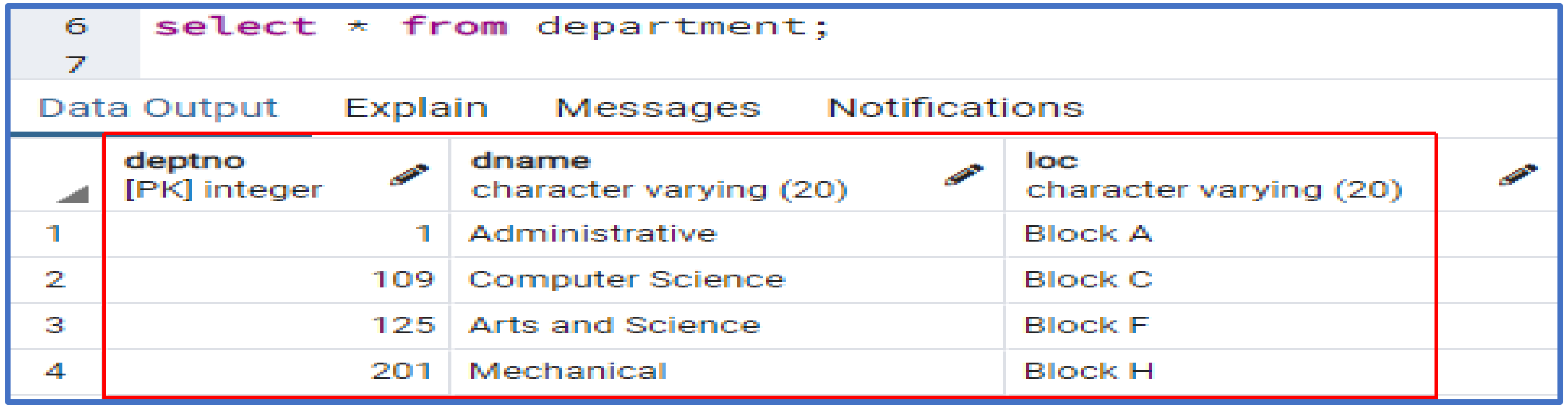


```
INSERT INTO department values (001, 'Administrative', 'Block A'),
                              (109, 'Computer Science', 'Block C'),
                              (125, 'Arts and Science', 'Block F'),
                              (201, 'Mechanical', 'Block H');
```

Data Output Explain Messages Notifications

```
INSERT 0 4

Query returned successfully in 199 msec.
```

Database table after inserting



```
select * from department;
```

Data Output Explain Messages Notifications

	deptno [PK] integer	dname character varying (20)	loc character varying (20)
1	1	Administrative	Block A
2	109	Computer Science	Block C
3	125	Arts and Science	Block F
4	201	Mechanical	Block H

"SELECT" Statement

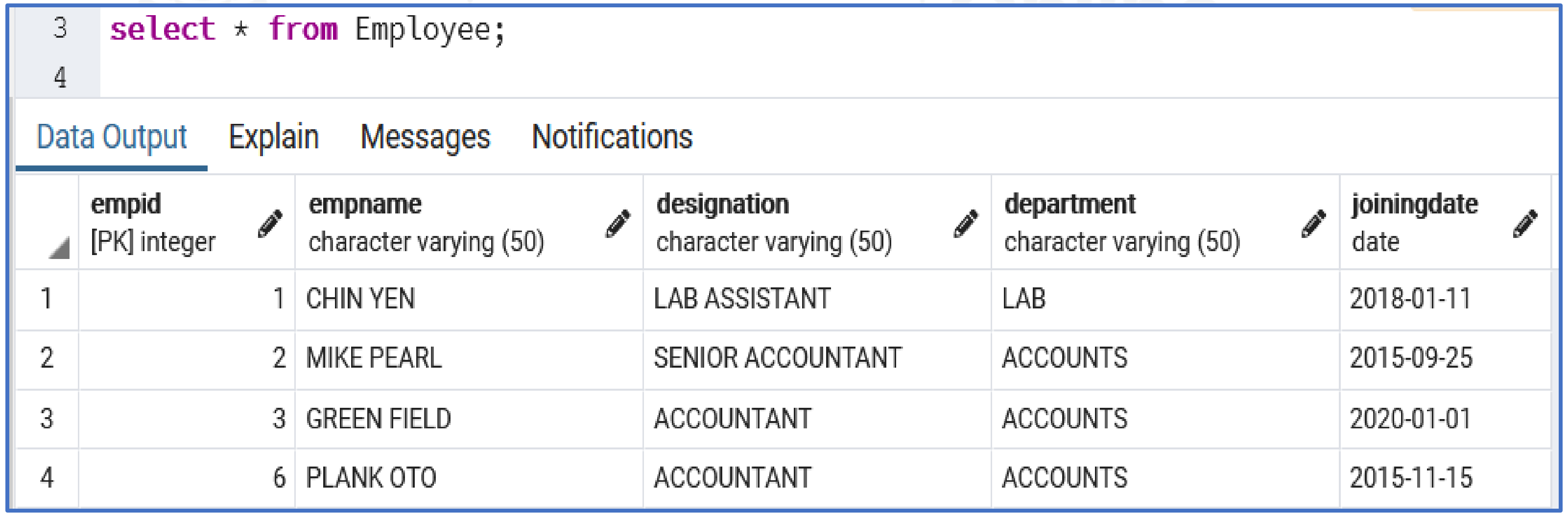
- Data Manipulation Language (DML) SQL command.
- Used to retrieve records from one or more tables.
- The SELECT statement has the following clauses :
  - **DISTINCT:** Select distinct rows in a table.
  - **WHERE:** Select the rows under the specified conditions.
  - **ORDER BY:** Sorts the result according to specified criteria.
  - **HAVING:** The conditions under which a category (group) will be included.
  - **GROUP BY:** Indicate categorization of results.

SELECT all the columns from one table

Syntax : SELECT * FROM Table_Name ;

- SELECT → ***Keyword***
- * → ***"all columns"***
- FROM → ***Keyword***
- Table_Name → ***Name of the Table***
- ; → ***Every SQL Query ends with a semicolon***

Example 1: Select all rows in a table



```
select * from Employee;
```

Data Output | Explain | Messages | Notifications

	empid [PK] integer	empname character varying (50)	designation character varying (50)	department character varying (50)	joiningdate date
1	1	CHIN YEN	LAB ASSISTANT	LAB	2018-01-11
2	2	MIKE PEARL	SENIOR ACCOUNTANT	ACCOUNTS	2015-09-25
3	3	GREEN FIELD	ACCOUNTANT	ACCOUNTS	2020-01-01
4	6	PLANK OTO	ACCOUNTANT	ACCOUNTS	2015-11-15

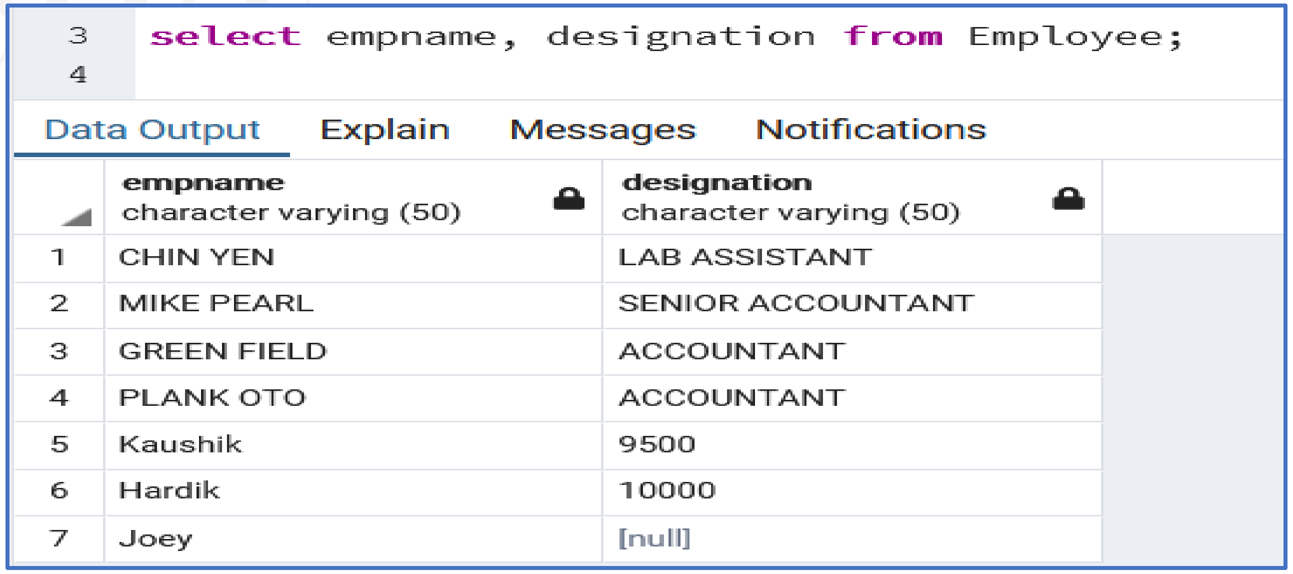
SELECT multiple columns from one table

Syntax :

SELECT | Column1, column2 ,… | FROM | Table_Name | ;

Keyword | ***Selected columns*** | ***Keyword*** | ***Name of the Table*** | ***Every SQL Query ends with a semicolon***

Example 2:
Display multiple columns in a table



```
select empname, designation from Employee;
```

Data Output | Explain | Messages | Notifications

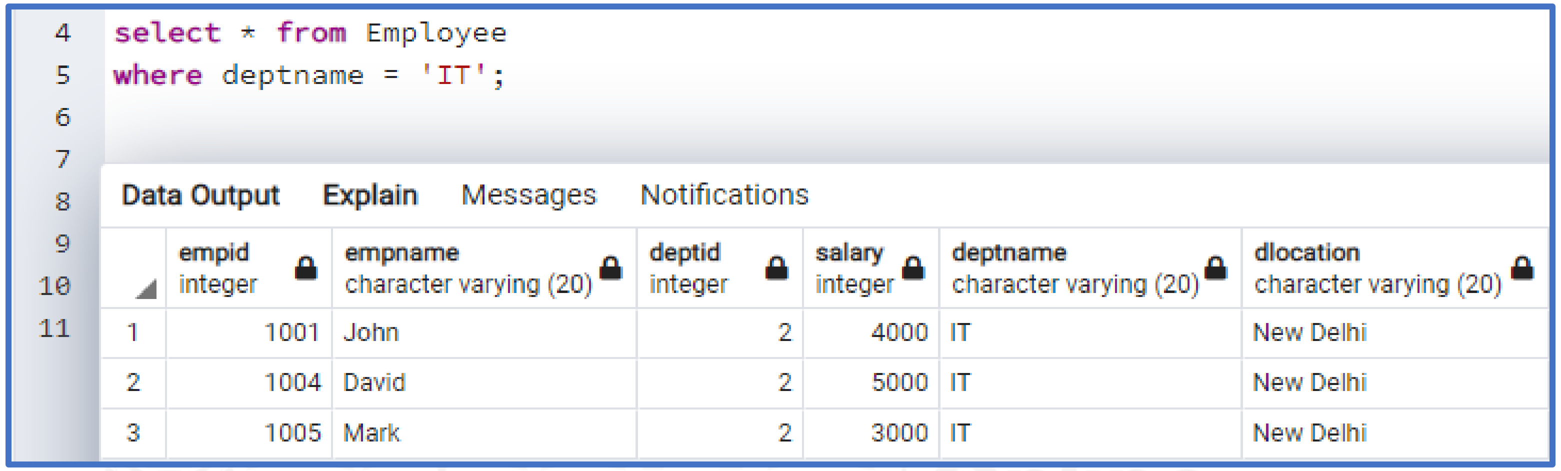
	empname character varying (50)	designation character varying (50)
1	CHIN YEN	LAB ASSISTANT
2	MIKE PEARL	SENIOR ACCOUNTANT
3	GREEN FIELD	ACCOUNTANT
4	PLANK OTO	ACCOUNTANT
5	Kaushik	9500
6	Hardik	10000
7	Joey	[null]

SELECT Statement:
Filter rows using WHERE clause.

- WHERE clause is used to filter the results from a SELECT, INSERT, UPDATE, or DELETE statement.
- The general form : WHERE | Condition
- where ***condition*** is any expression that evaluates to a result of type Boolean.
- Any row that does not satisfy this condition will be eliminated from the output.

Select Statement with Where Clause

Example 3:

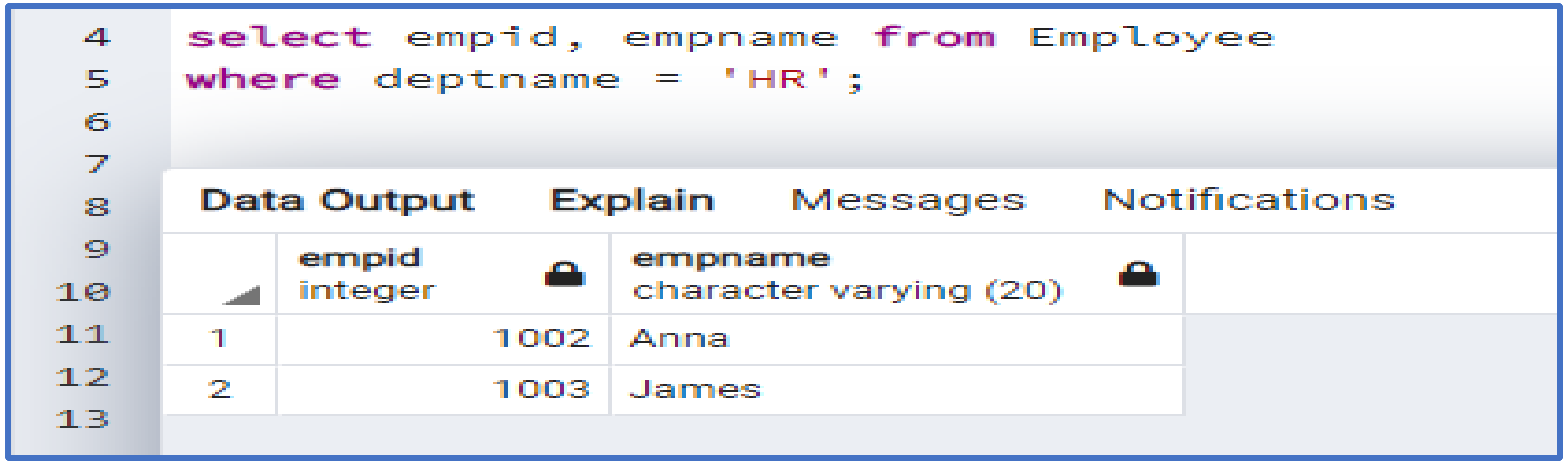


```
select * from Employee
where deptname = 'IT';
```

Data Output | Explain | Messages | Notifications

	empid integer	empname character varying (20)	deptid integer	salary integer	deptname character varying (20)	dlocation character varying (20)
1	1001	John	2	4000	IT	New Delhi
2	1004	David	2	5000	IT	New Delhi
3	1005	Mark	2	3000	IT	New Delhi

Example 4:



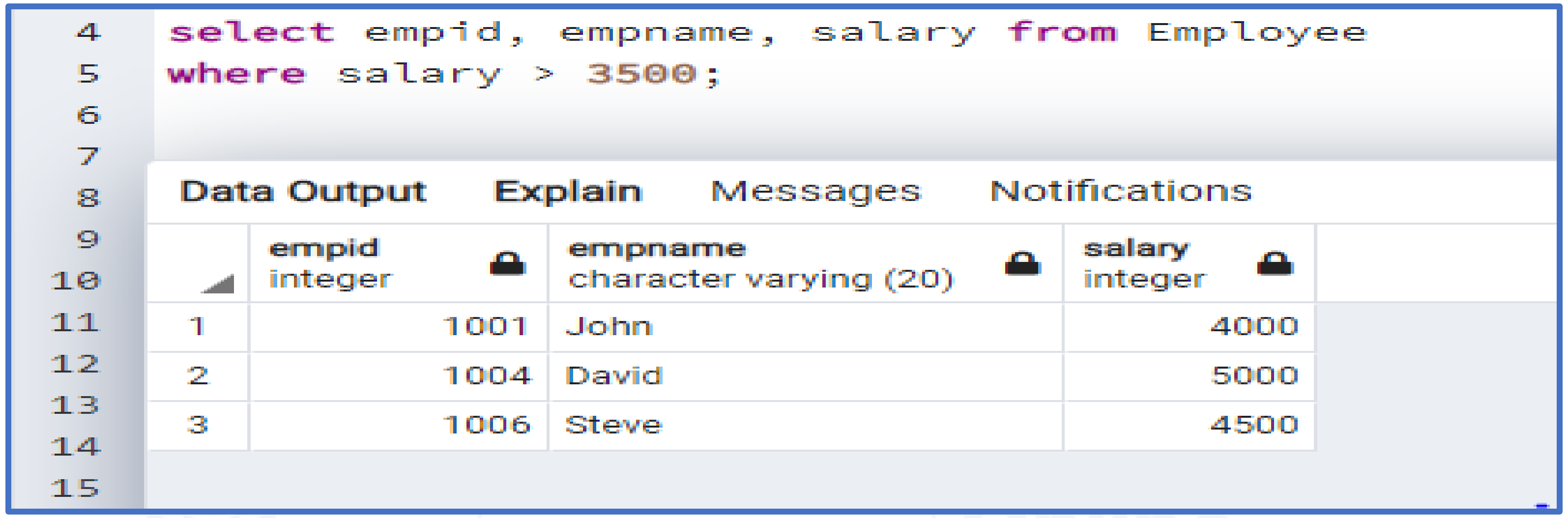
```
select empid, empname from Employee
where deptname = 'HR';
```

Data Output | Explain | Messages | Notifications

	empid integer	empname character varying (20)
1	1002	Anna
2	1003	James

Select data based on comparison operator

Example 5:



```
select empid, empname, salary from Employee
where salary > 3500;
```

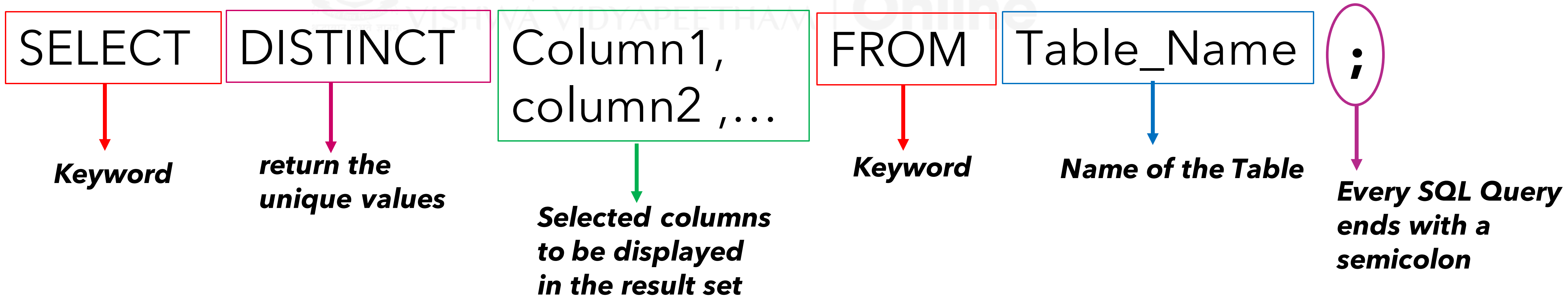
Data Output Explain Messages Notifications

	empid integer	empname character varying (20)	salary integer
1	1001	John	4000
2	1004	David	5000
3	1006	Steve	4500

SELECT distinct rows using DISTINCT operator.

- Used to remove duplicates from the result set.
- The DISTINCT clause can only be used with SELECT statements.

Syntax :

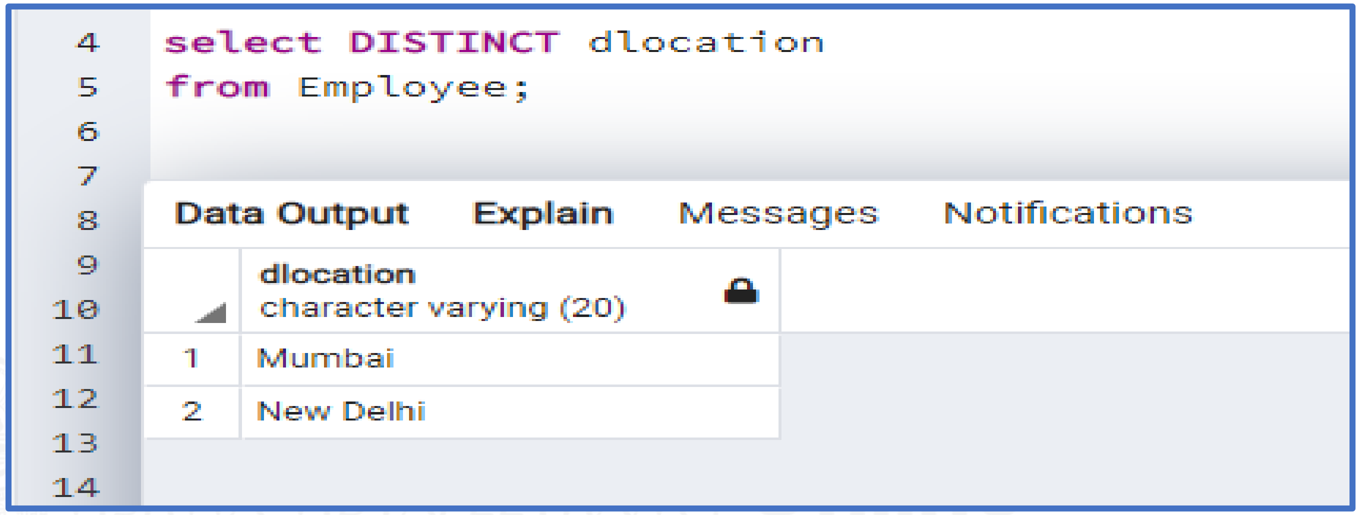


```
SELECT DISTINCT Column1, column2 ,… FROM Table_Name ;
```

Keyword
return the unique values
Selected columns to be displayed in the result set
Keyword
Name of the Table
Every SQL Query ends with a semicolon

Select Statement with DISTINCT Clause

Example :

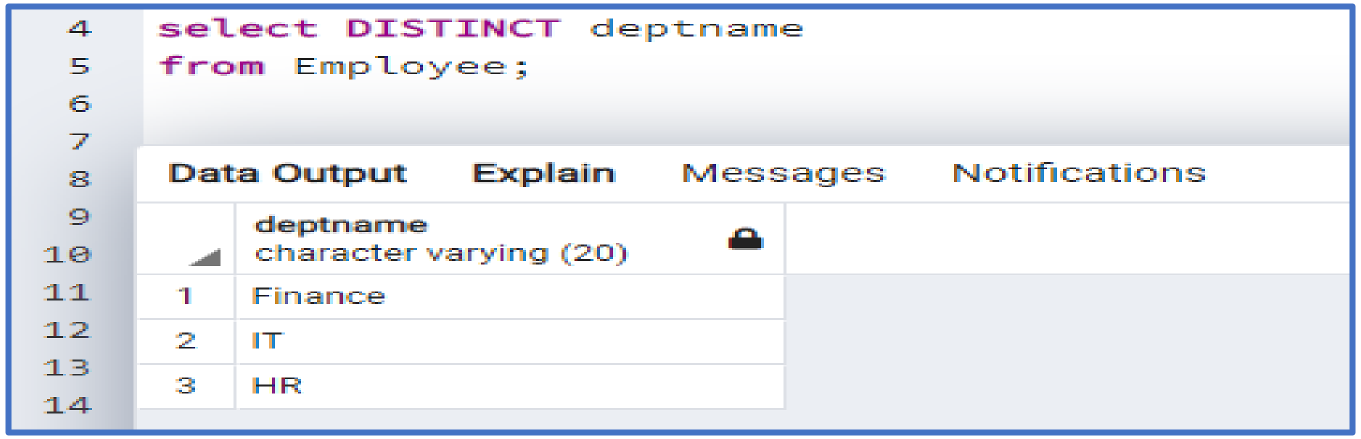


```
select DISTINCT dlocation
from Employee;
```

Data Output | Explain | Messages | Notifications

```
   dlocation
   character varying (20)
1  Mumbai
2  New Delhi
```

Example :



```
select DISTINCT deptname
from Employee;
```

Data Output | Explain | Messages | Notifications

```
   deptname
   character varying (20)
1  Finance
2  IT
3  HR
```

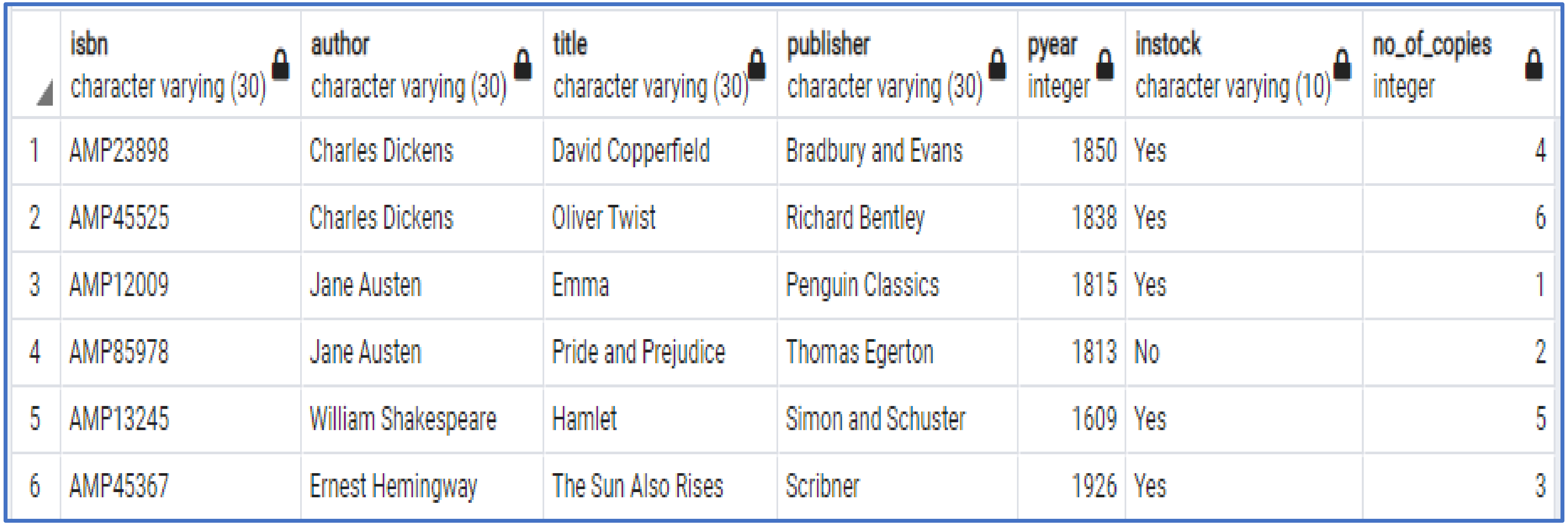
SELECT Statement : Sort rows using ORDER BY clause.

The ORDER BY clause allows you to sort rows returned by a SELECT clause in ascending or descending order based on a sort expression.

Syntax:

```
SELECT <column1, column2, ...>
FROM <Table_name>
[WHERE condition]
[ORDER BY column1, column2, ...] [ASC | DESC];
```

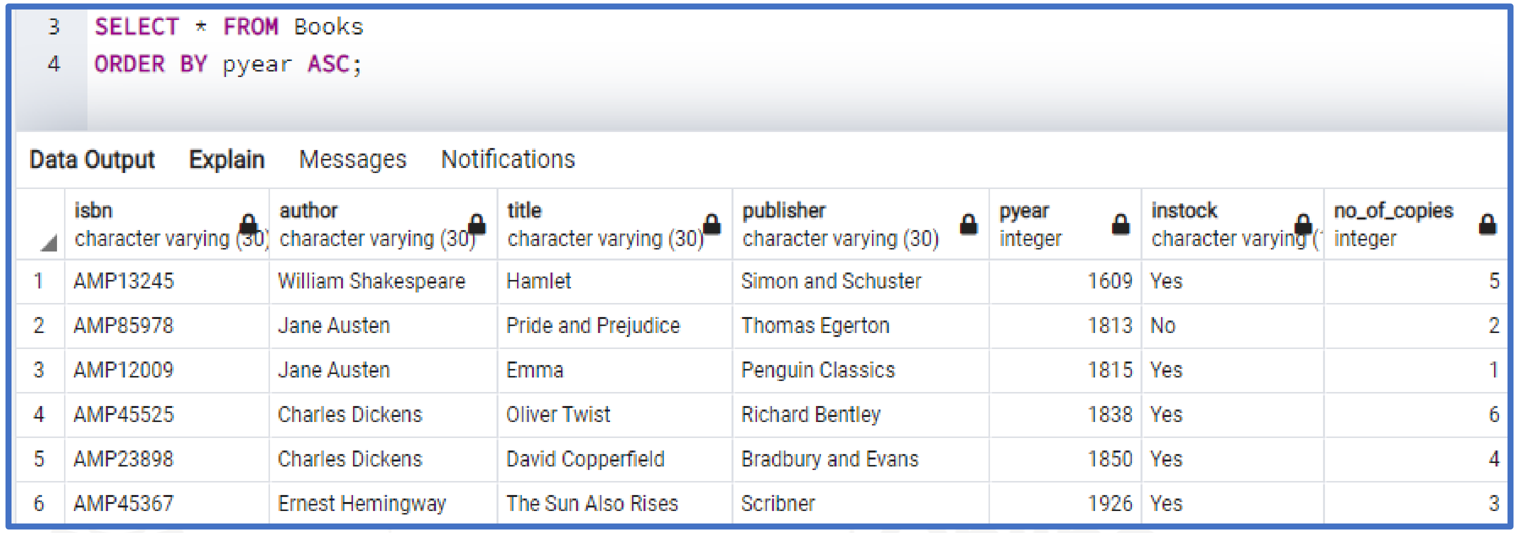
Database Table : Books



	isbn character varying (30)	author character varying (30)	title character varying (30)	publisher character varying (30)	pyear integer	instock character varying (10)	no_of_copies integer
1	AMP23898	Charles Dickens	David Copperfield	Bradbury and Evans	1850	Yes	4
2	AMP45525	Charles Dickens	Oliver Twist	Richard Bentley	1838	Yes	6
3	AMP12009	Jane Austen	Emma	Penguin Classics	1815	Yes	1
4	AMP85978	Jane Austen	Pride and Prejudice	Thomas Egerton	1813	No	2
5	AMP13245	William Shakespeare	Hamlet	Simon and Schuster	1609	Yes	5
6	AMP45367	Ernest Hemingway	The Sun Also Rises	Scribner	1926	Yes	3

Examples

Example :

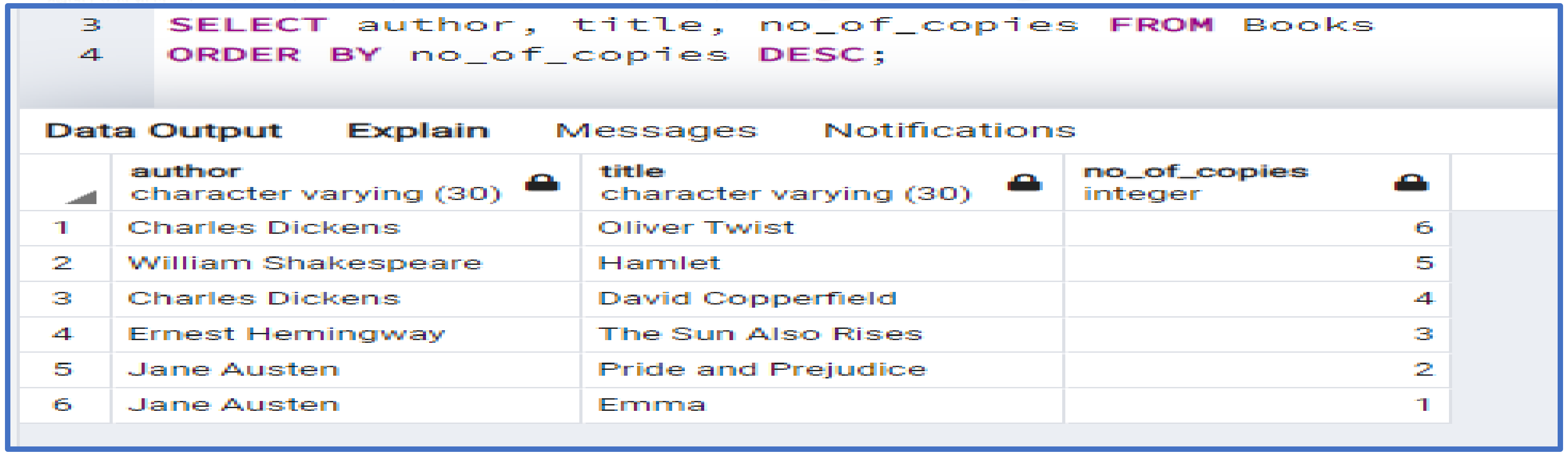


```
SELECT * FROM Books
ORDER BY pyear ASC;
```

Data Output | Explain | Messages | Notifications

	isbn character varying (30)	author character varying (30)	title character varying (30)	publisher character varying (30)	pyear integer	instock character varying (	no_of_copies integer
1	AMP13245	William Shakespeare	Hamlet	Simon and Schuster	1609	Yes	5
2	AMP85978	Jane Austen	Pride and Prejudice	Thomas Egerton	1813	No	2
3	AMP12009	Jane Austen	Emma	Penguin Classics	1815	Yes	1
4	AMP45525	Charles Dickens	Oliver Twist	Richard Bentley	1838	Yes	6
5	AMP23898	Charles Dickens	David Copperfield	Bradbury and Evans	1850	Yes	4
6	AMP45367	Ernest Hemingway	The Sun Also Rises	Scribner	1926	Yes	3

Example :



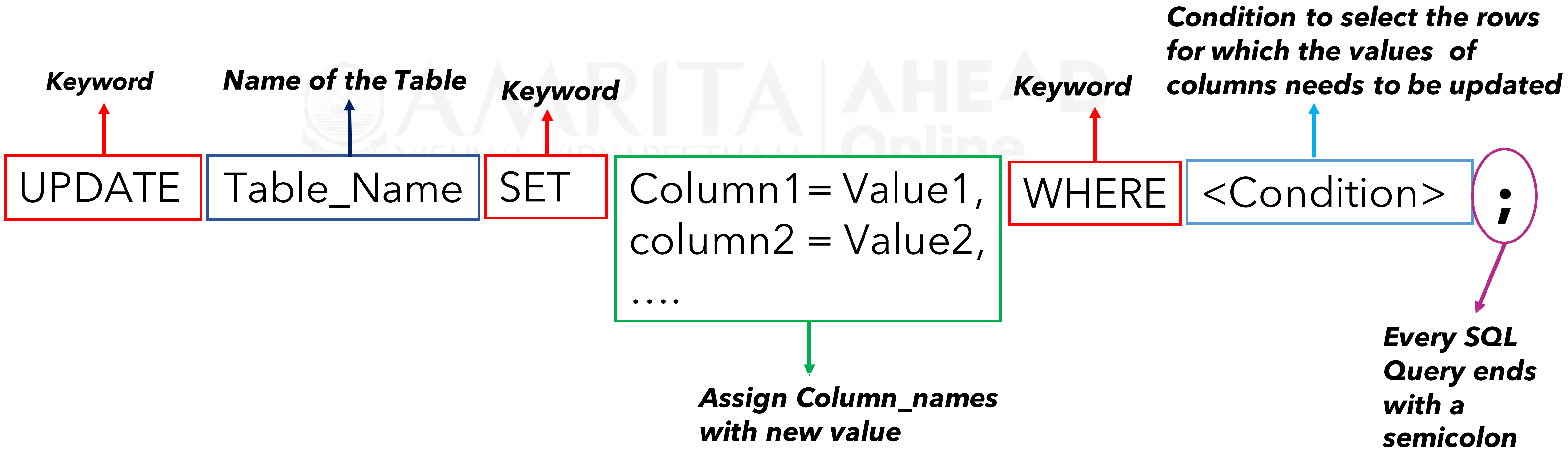
```
SELECT author, title, no_of_copies FROM Books
ORDER BY no_of_copies DESC;
```

Data Output | Explain | Messages | Notifications

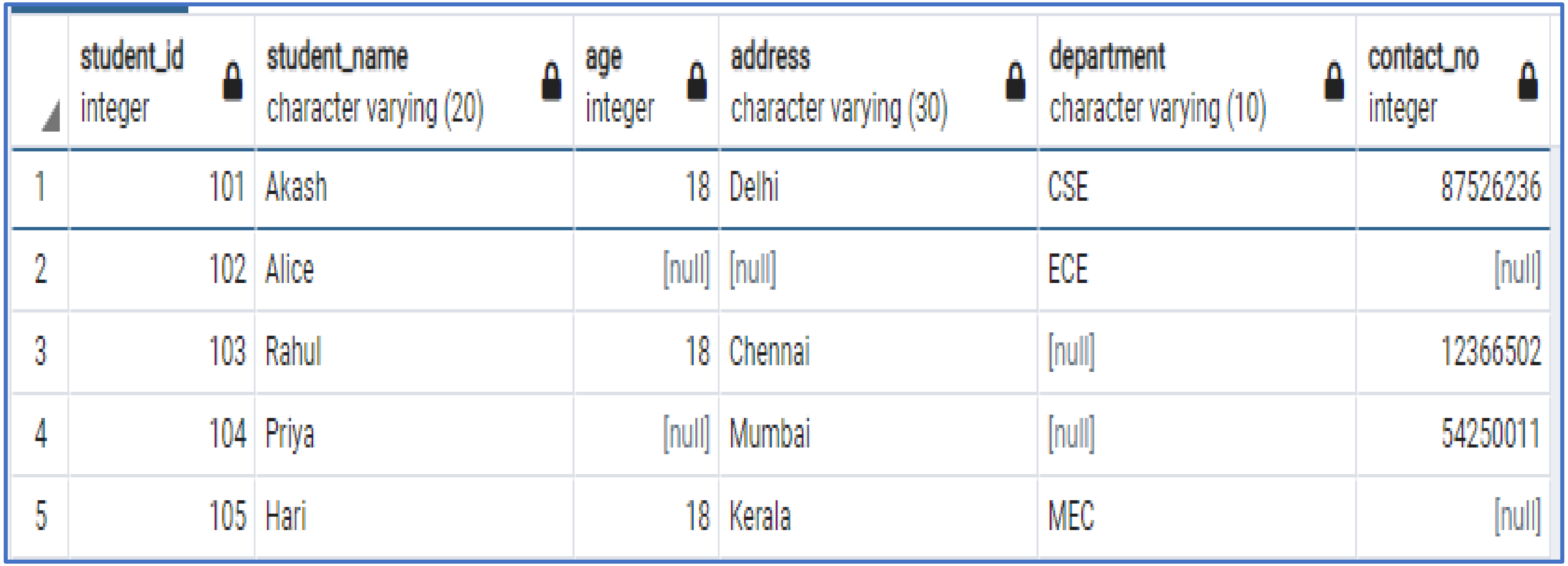
	author character varying (30)	title character varying (30)	no_of_copies integer
1	Charles Dickens	Oliver Twist	6
2	William Shakespeare	Hamlet	5
3	Charles Dickens	David Copperfield	4
4	Ernest Hemingway	The Sun Also Rises	3
5	Jane Austen	Pride and Prejudice	2
6	Jane Austen	Emma	1

Syntax: UPDATE Statement

- Specify both the columns which we want to fill and their corresponding values.

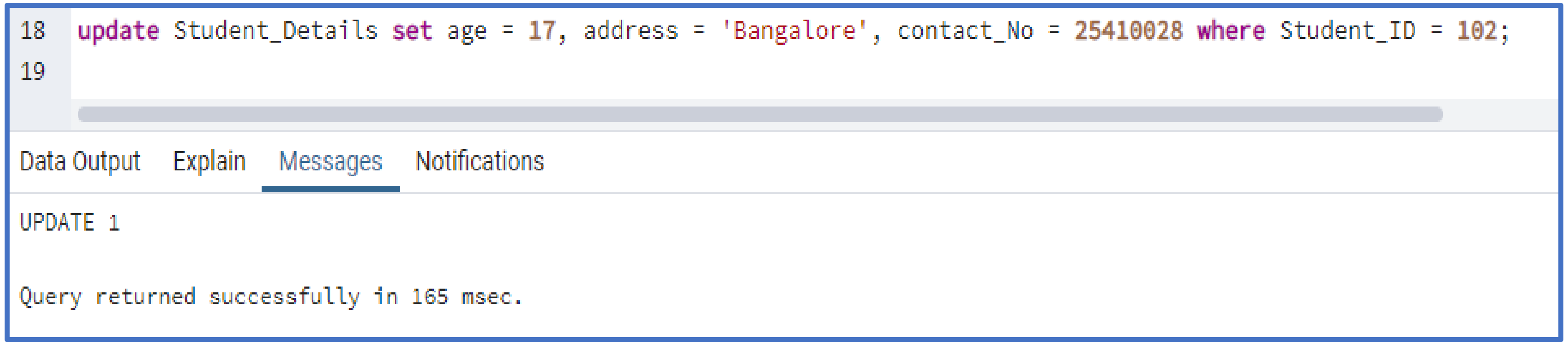

Keyword
Name of the Table
Keyword
Keyword
Condition to select the rows for which the values of columns needs to be updated
UPDATE
Table_Name
SET
Column1= Value1, column2 = Value2, ….
WHERE
<Condition>
;
Assign Column_names with new value
Every SQL Query ends with a semicolon

Database Table : Student_Details



	student_id integer	student_name character varying (20)	age integer	address character varying (30)	department character varying (10)	contact_no integer
1	101	Akash	18	Delhi	CSE	87526236
2	102	Alice	[null]	[null]	ECE	[null]
3	103	Rahul	18	Chennai	[null]	12366502
4	104	Priya	[null]	Mumbai	[null]	54250011
5	105	Hari	18	Kerala	MEC	[null]

Example 1: Update the details of Student_ID = 102

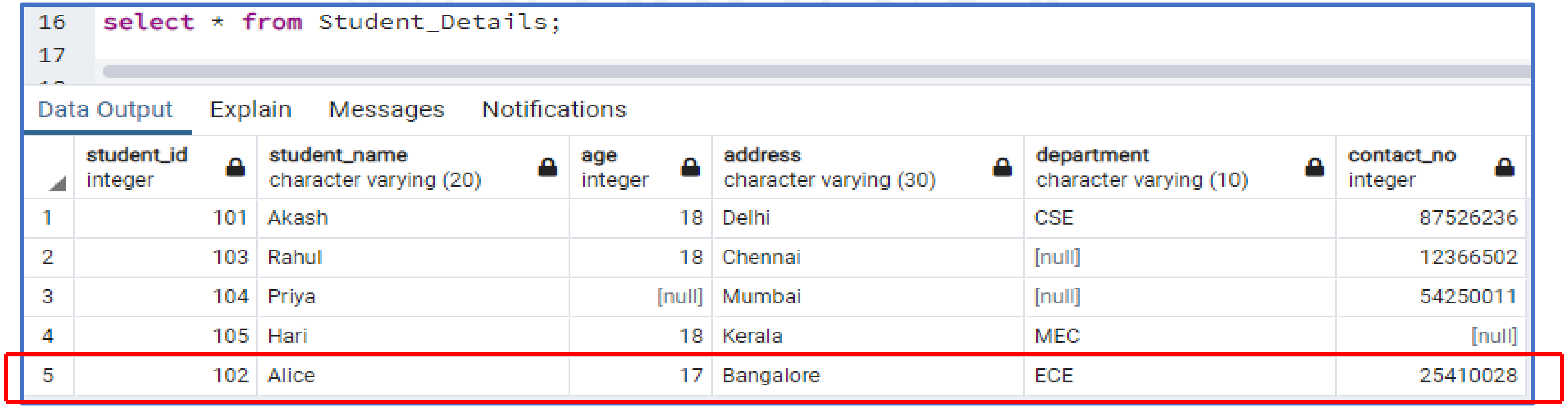


```
update Student_Details set age = 17, address = 'Bangalore', contact_No = 25410028 where Student_ID = 102;
```

Data Output Explain Messages Notifications

```
UPDATE 1

Query returned successfully in 165 msec.
```

Updated Database Table : Student_Details

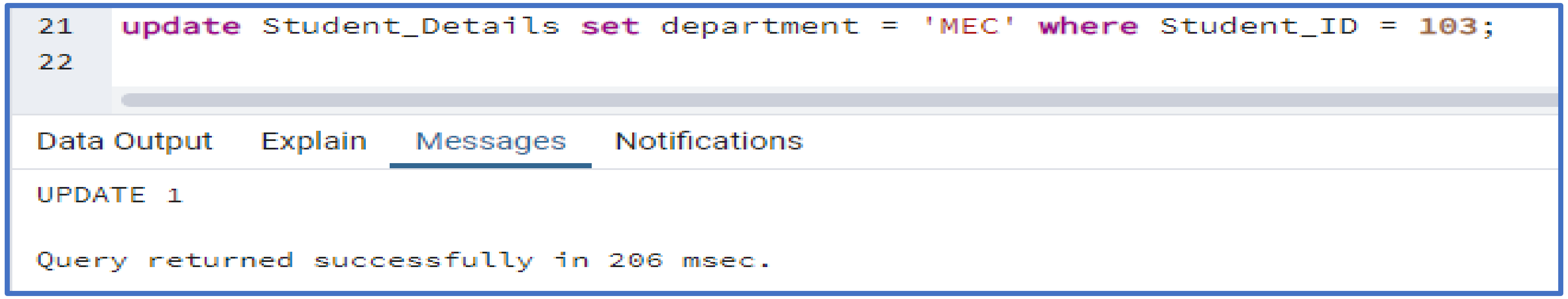


```
select * from Student_Details;
```

Data Output Explain Messages Notifications

	student_id integer	student_name character varying (20)	age integer	address character varying (30)	department character varying (10)	contact_no integer
1	101	Akash	18	Delhi	CSE	87526236
2	103	Rahul	18	Chennai	[null]	12366502
3	104	Priya	[null]	Mumbai	[null]	54250011
4	105	Hari	18	Kerala	MEC	[null]
5	102	Alice	17	Bangalore	ECE	25410028

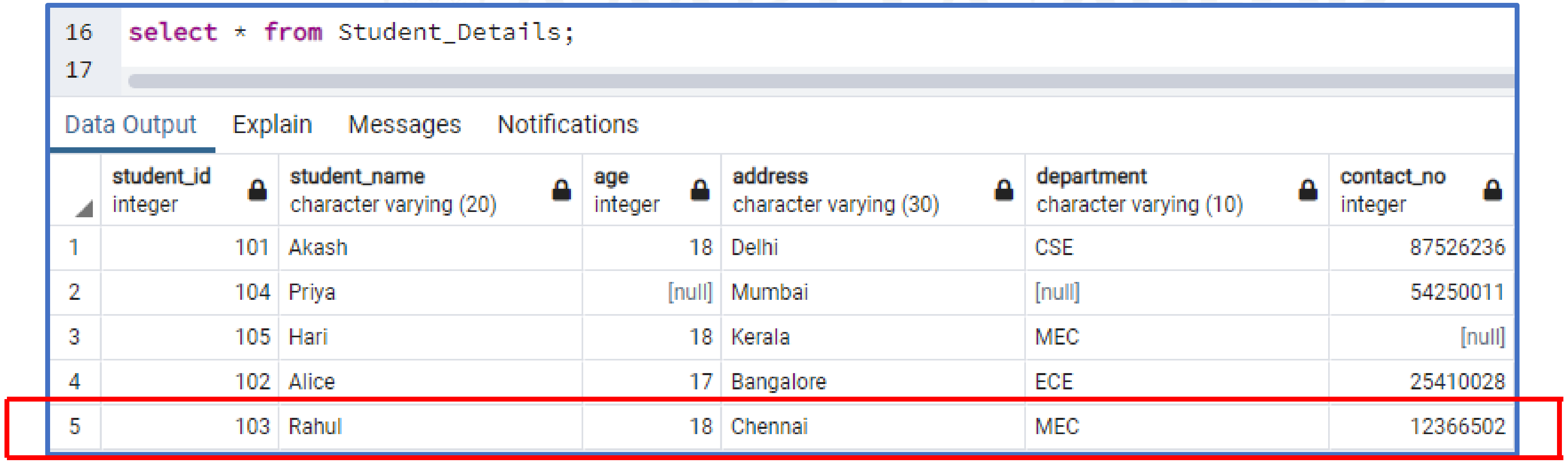
Example 2: Update the details of Student_ID = 103



```
update Student_Details set department = 'MEC' where Student_ID = 103;
```

```
UPDATE 1

Query returned successfully in 206 msec.
```

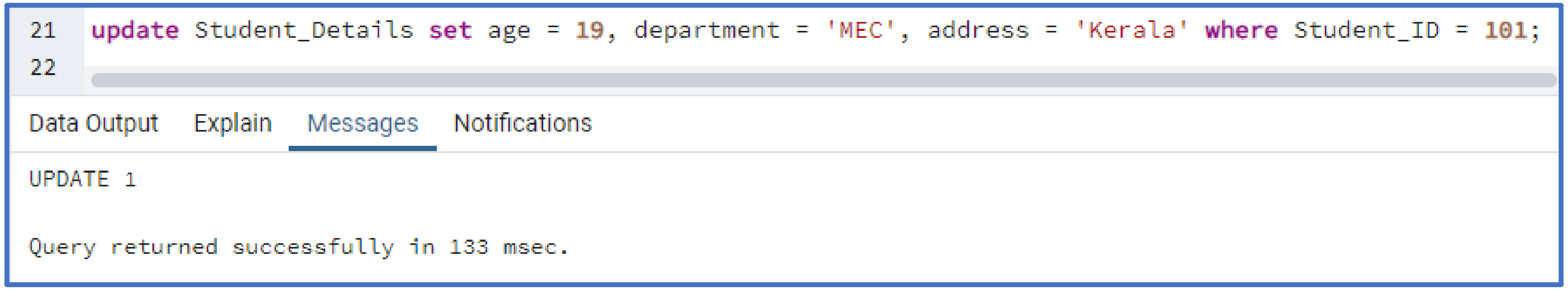
Updated Database Table : Student_Details



```
select * from Student_Details;
```

	student_id integer	student_name character varying (20)	age integer	address character varying (30)	department character varying (10)	contact_no integer
1	101	Akash	18	Delhi	CSE	87526236
2	104	Priya	[null]	Mumbai	[null]	54250011
3	105	Hari	18	Kerala	MEC	[null]
4	102	Alice	17	Bangalore	ECE	25410028
5	103	Rahul	18	Chennai	MEC	12366502

Example 3: Update the already existing data in a record

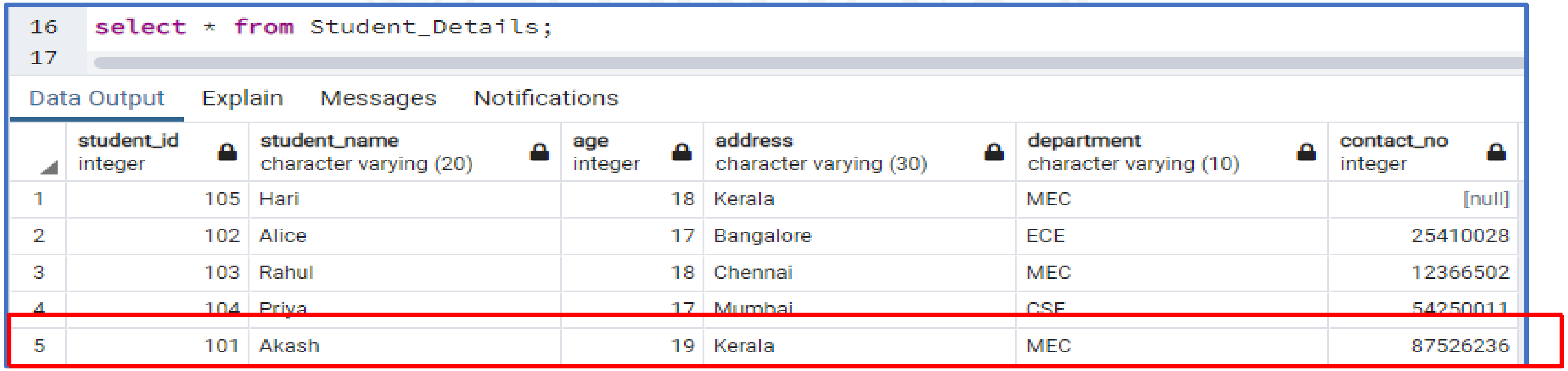


```
update Student_Details set age = 19, department = 'MEC', address = 'Kerala' where Student_ID = 101;
```

Data Output | Explain | Messages | Notifications

```
UPDATE 1

Query returned successfully in 133 msec.
```

Updated Database Table : Student_Details

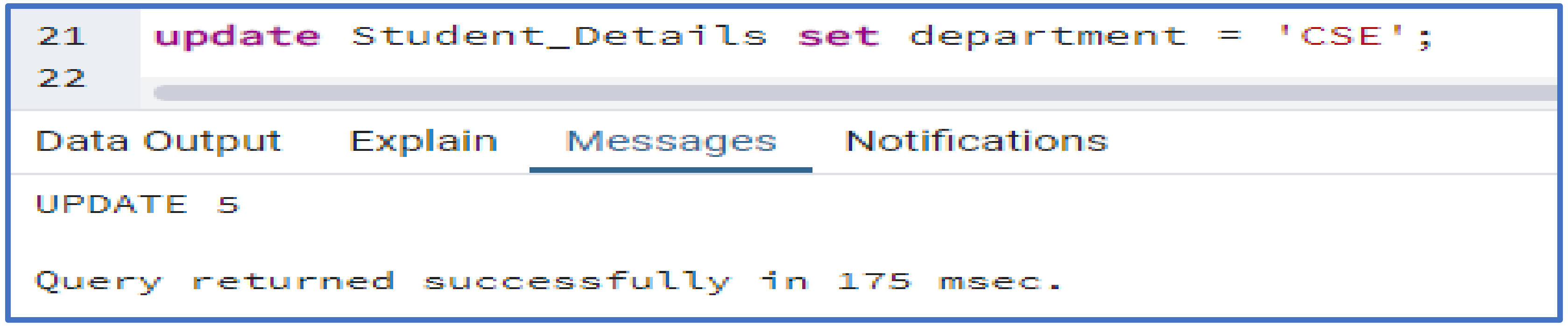


```
select * from Student_Details;
```

Data Output | Explain | Messages | Notifications

	student_id integer	student_name character varying (20)	age integer	address character varying (30)	department character varying (10)	contact_no integer
1	105	Hari	18	Kerala	MEC	[null]
2	102	Alice	17	Bangalore	ECE	25410028
3	103	Rahul	18	Chennai	MEC	12366502
4	104	Priya	17	Mumbai	CSE	54250011
5	101	Akash	19	Kerala	MEC	87526236

Omitting WHERE clause:

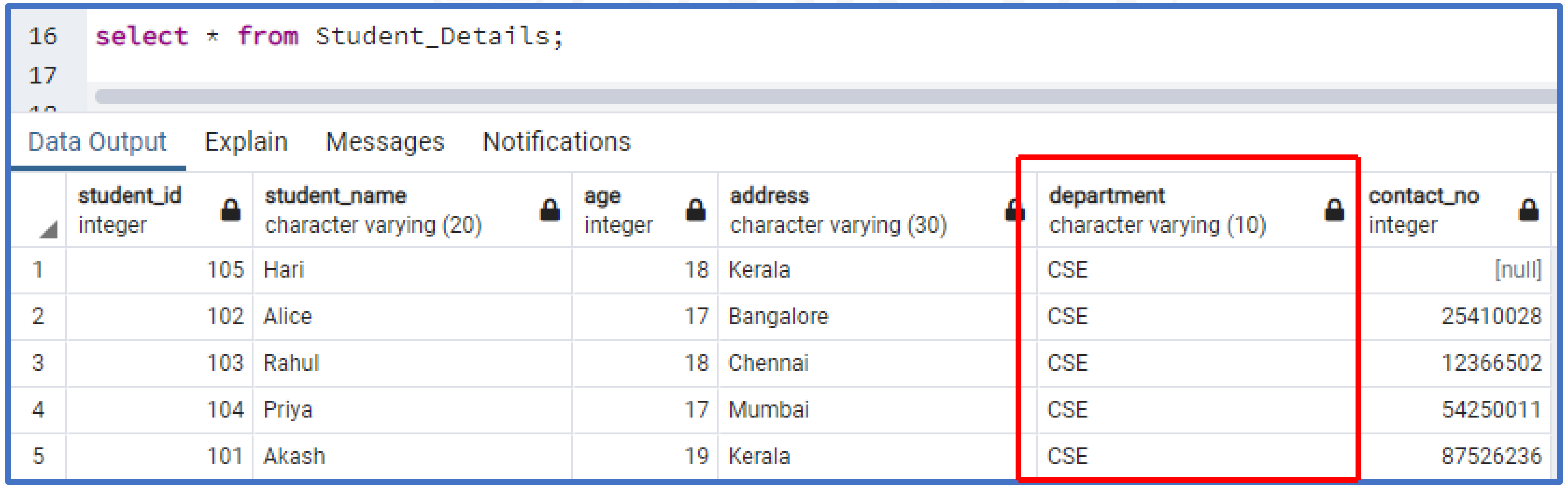


```
update Student_Details set department = 'CSE';
```

Data Output | Explain | Messages | Notifications

```
UPDATE 5

Query returned successfully in 175 msec.
```

Updated Database Table : Student_Details



```
select * from Student_Details;
```

Data Output | Explain | Messages | Notifications

	student_id integer	student_name character varying (20)	age integer	address character varying (30)	department character varying (10)	contact_no integer
1	105	Hari	18	Kerala	CSE	[null]
2	102	Alice	17	Bangalore	CSE	25410028
3	103	Rahul	18	Chennai	CSE	12366502
4	104	Priya	17	Mumbai	CSE	54250011
5	101	Akash	19	Kerala	CSE	87526236

Update Statement: Return Clause

- UPDATE statement returns the following command tag:
  - UPDATE count
- Count is the number of rows updated including rows whose values did not change.
- The UPDATE statement has an optional RETURNING clause that returns the updated rows.

Syntax:

```
UPDATE table_name SET column1=value1, column2=value2, ...
WHERE condition RETURNING * | output_expression AS
output_name;
```

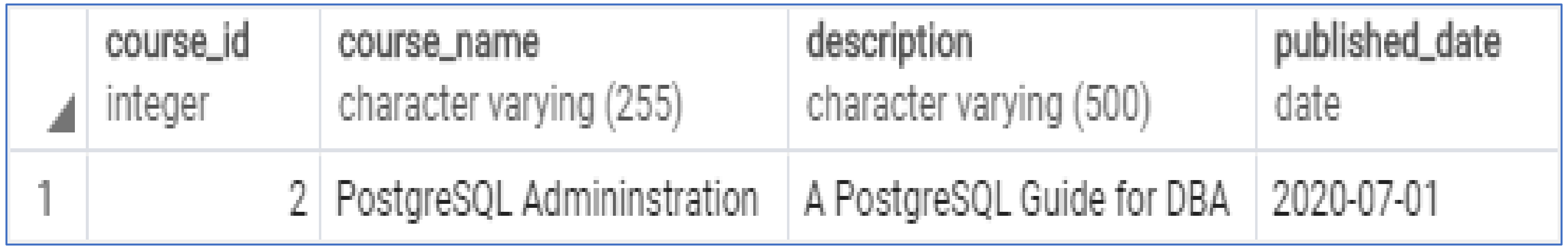
Update Statement: Return Clause

Update a row and return the updated row

Example: Modify the published_date of the course to 2020-07-01 and returns the updated course.

Query: UPDATE course SET published_date = '2020-07-01' WHERE course_id = 2 RETURNING *;

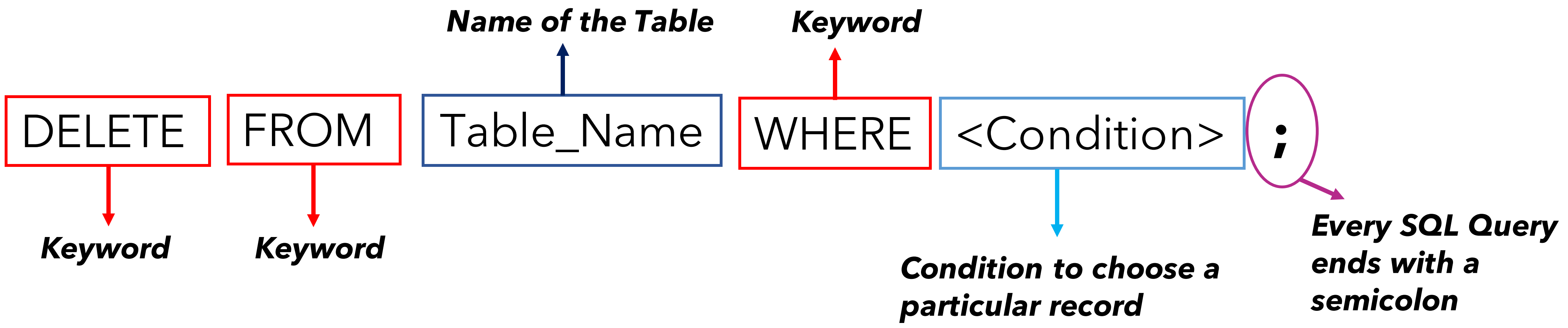
Output Result:



	course_id integer	course_name character varying (255)	description character varying (500)	published_date date
1	2	PostgreSQL Admininstration	A PostgreSQL Guide for DBA	2020-07-01

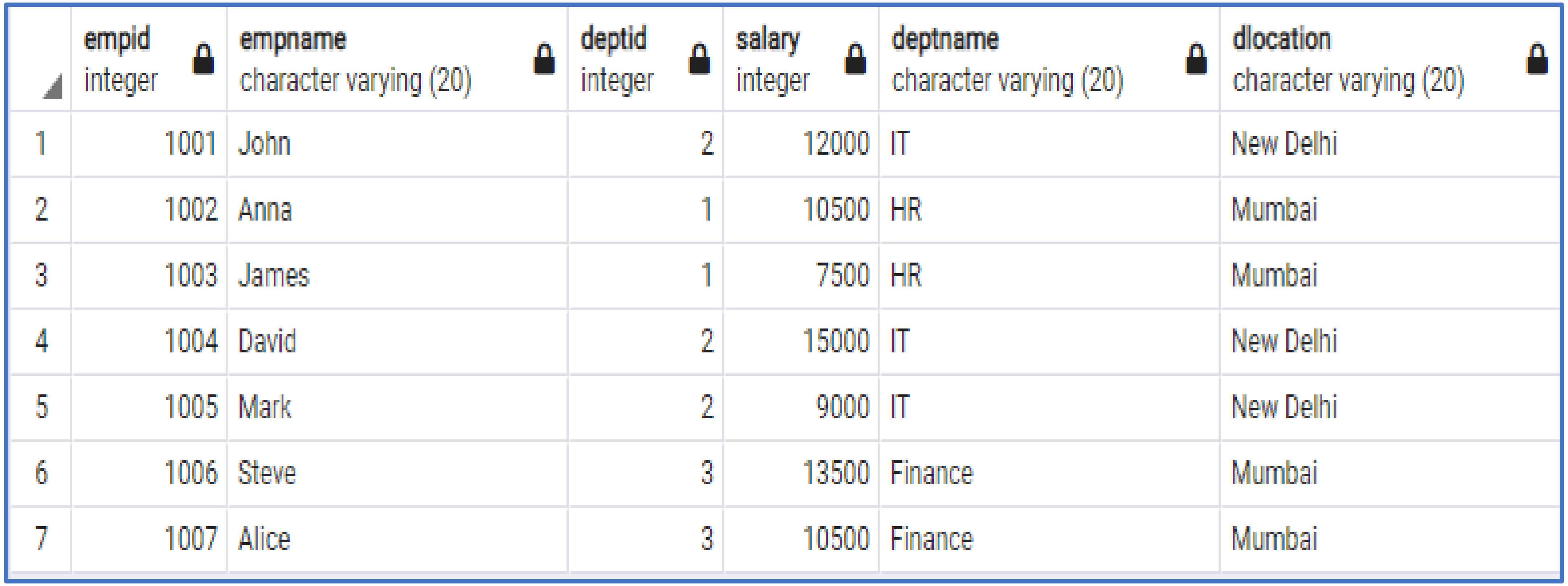
DELETE Statement

- Data Manipulation Language (DML) SQL Command.
- Used to delete existing records from a table.
- We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.


Name of the Table
Keyword
DELETE
FROM
Table_Name
WHERE
<Condition>
;
Keyword
Keyword
Condition to choose a particular record
Every SQL Query ends with a semicolon

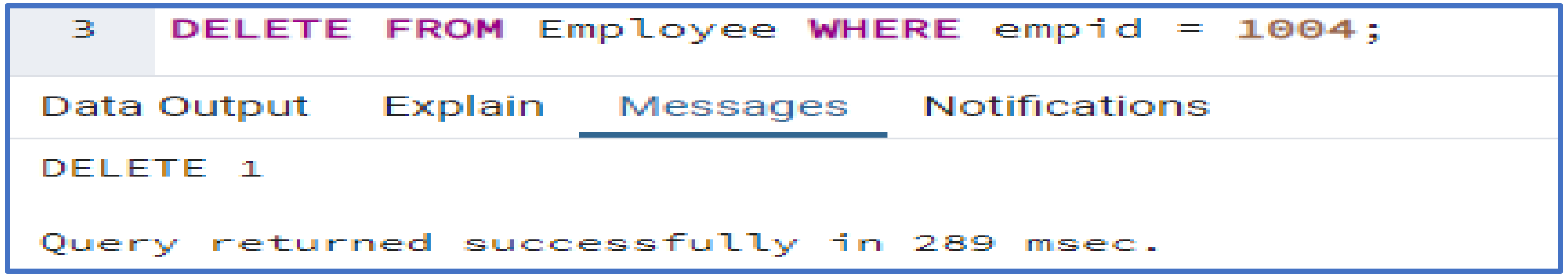
Example: Database Table

Table: Employee



	empid integer	empname character varying (20)	deptid integer	salary integer	deptname character varying (20)	dlocation character varying (20)
1	1001	John	2	12000	IT	New Delhi
2	1002	Anna	1	10500	HR	Mumbai
3	1003	James	1	7500	HR	Mumbai
4	1004	David	2	15000	IT	New Delhi
5	1005	Mark	2	9000	IT	New Delhi
6	1006	Steve	3	13500	Finance	Mumbai
7	1007	Alice	3	10500	Finance	Mumbai

Example 1: Delete one row in a table

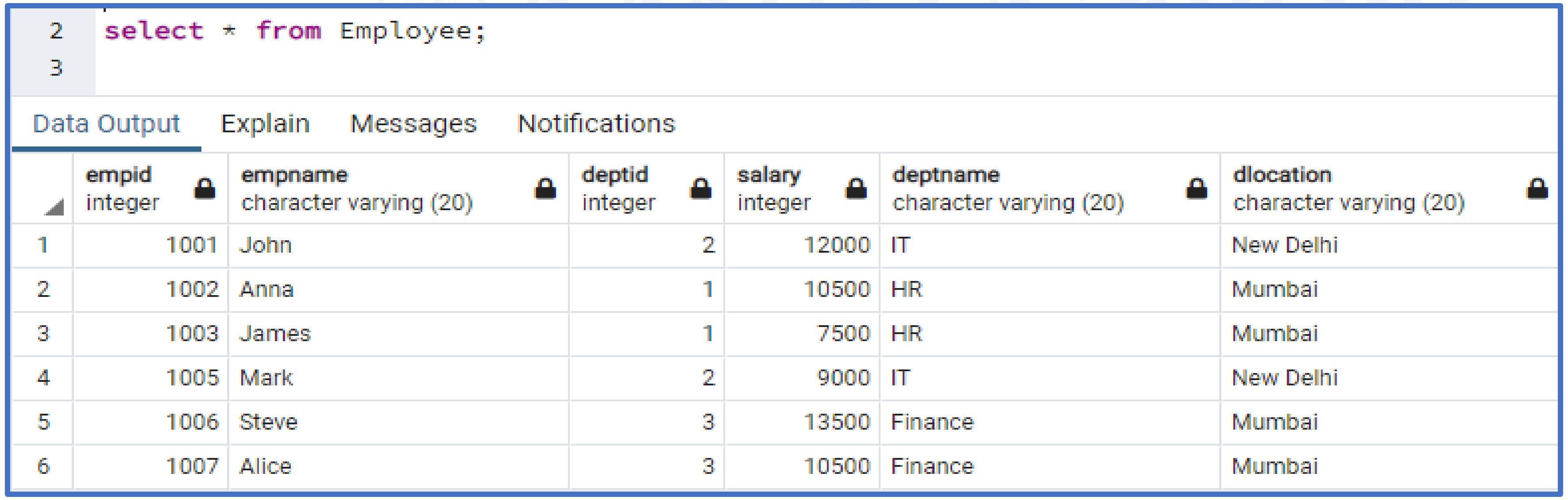


```
DELETE FROM Employee WHERE empid = 1004;
```

Data Output | Explain | Messages | Notifications

```
DELETE 1

Query returned successfully in 289 msec.
```

Updated Database Table : Employee

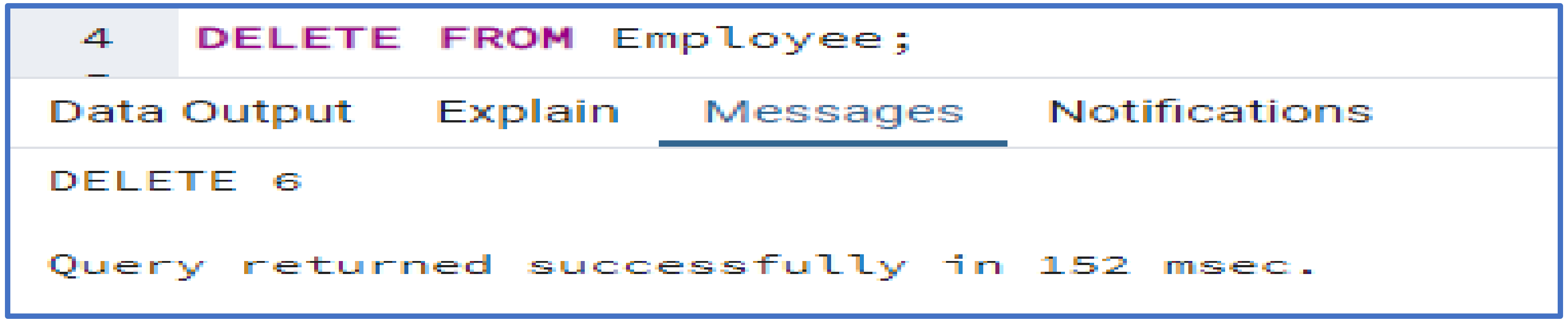


```
select * from Employee;
```

Data Output | Explain | Messages | Notifications

	empid integer	empname character varying (20)	deptid integer	salary integer	deptname character varying (20)	dlocation character varying (20)
1	1001	John	2	12000	IT	New Delhi
2	1002	Anna	1	10500	HR	Mumbai
3	1003	James	1	7500	HR	Mumbai
4	1005	Mark	2	9000	IT	New Delhi
5	1006	Steve	3	13500	Finance	Mumbai
6	1007	Alice	3	10500	Finance	Mumbai

Example 2: Delete all rows in a table

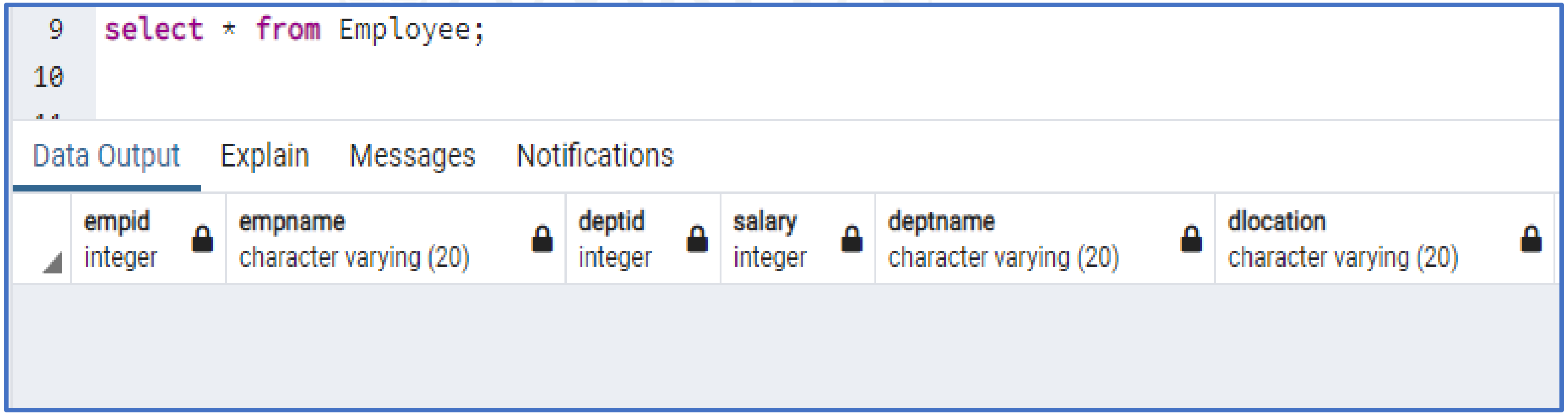


```
DELETE FROM Employee;
```

Data Output Explain Messages Notifications

```
DELETE 6

Query returned successfully in 152 msec.
```

Updated Database Table : Employee



```
select * from Employee;
```

Data Output Explain Messages Notifications

empid integer	empname character varying (20)	deptid integer	salary integer	deptname character varying (20)	dlocation character varying (20)

Delete Statement: Return Clause

- By using the RETURNING clause, you can return the deleted rows to client as follows:
  - DELETE FROM Table_Name WHERE condition
    RETURNING (select_list | *);
- The asterisk (*) allows you to return all columns of the deleted row.
- To return specific columns, you can specify them after the RETURNING keyword.
- DELETE statement only removes data from a table.
- It doesn't modify the structure of the table.

Delete Statement: Return Clause

Delete a row and return deleted row

- Example: Deletes the row with id 7 and returns the deleted row to the client.
  - Query: DELETE FROM links WHERE id = 7 RETURNING *;

Delete multiple rows from the table

- Example: Delete two rows from the links table and return the values in the id column of deleted rows.
  - Query: DELETE FROM links WHERE id IN (6,5) RETURNING *;

SQL : Aggregate Functions

- Performs a calculation on a set of values, and returns a single value.
- Except for COUNT(*), aggregate functions ignore null values.
- Often used with the GROUP BY clause of the SELECT statement.

Aggregate Functions

- COUNT counts how many rows are in a particular column.
- SUM adds together all the values in a particular column.
- MIN and MAX return the lowest and highest values in a particular column, respectively.
- AVG calculates the average of a group of selected values.

SQL COUNT

Count the number of rows in a particular column.

- Syntax : Count all rows

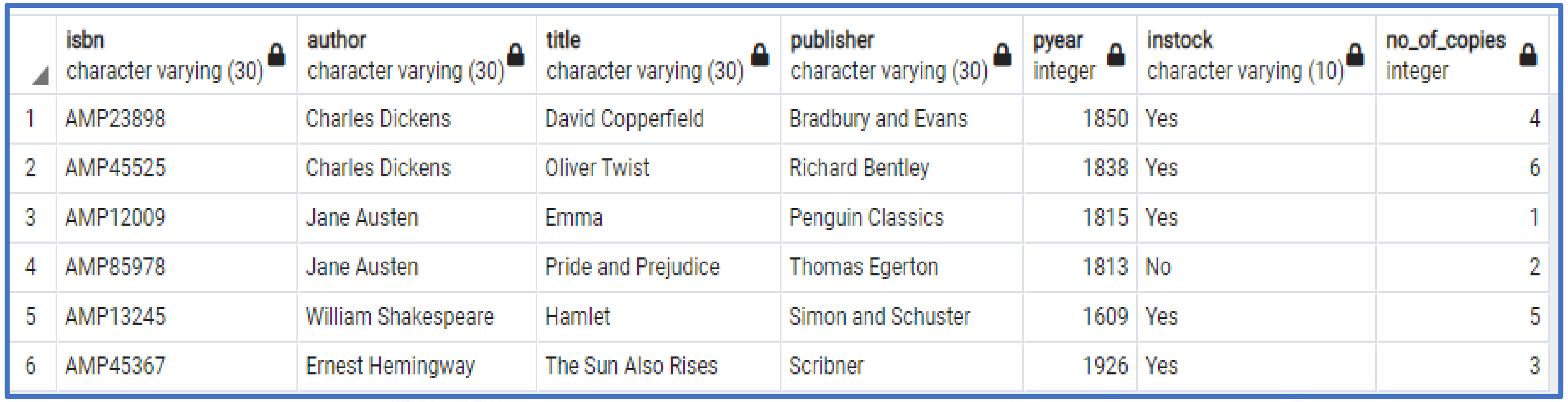
  SELECT COUNT(*) FROM <Table_Name>;

- Syntax : Count individual columns

  SELECT COUNT(<column_name>) FROM <Table_Name>;

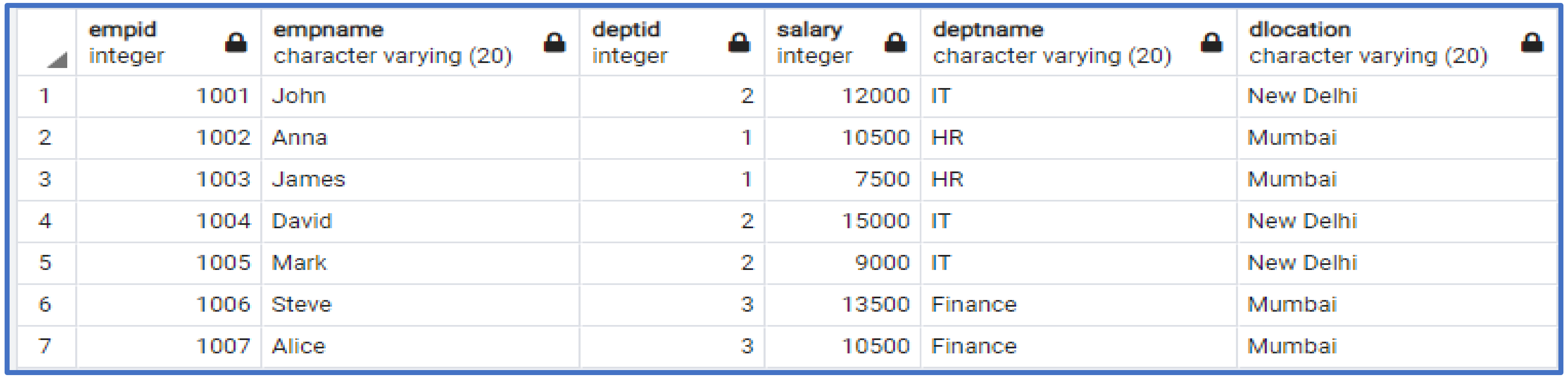
Database Table

Table : Books



	isbn character varying (30)	author character varying (30)	title character varying (30)	publisher character varying (30)	pyear integer	instock character varying (10)	no_of_copies integer
1	AMP23898	Charles Dickens	David Copperfield	Bradbury and Evans	1850	Yes	4
2	AMP45525	Charles Dickens	Oliver Twist	Richard Bentley	1838	Yes	6
3	AMP12009	Jane Austen	Emma	Penguin Classics	1815	Yes	1
4	AMP85978	Jane Austen	Pride and Prejudice	Thomas Egerton	1813	No	2
5	AMP13245	William Shakespeare	Hamlet	Simon and Schuster	1609	Yes	5
6	AMP45367	Ernest Hemingway	The Sun Also Rises	Scribner	1926	Yes	3

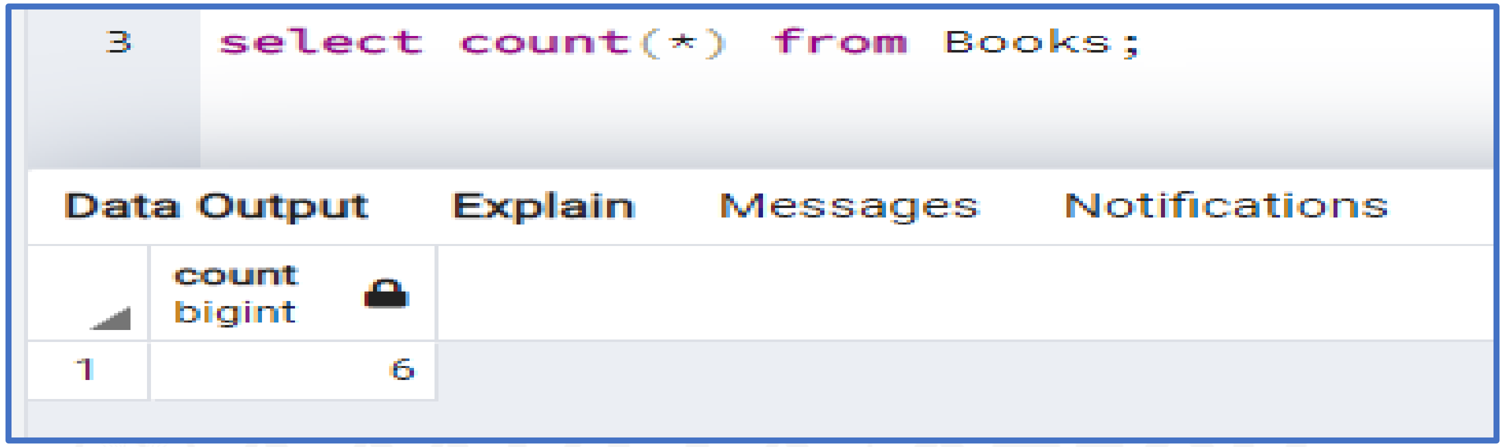
Table : Employee



	empid integer	empname character varying (20)	deptid integer	salary integer	deptname character varying (20)	dlocation character varying (20)
1	1001	John	2	12000	IT	New Delhi
2	1002	Anna	1	10500	HR	Mumbai
3	1003	James	1	7500	HR	Mumbai
4	1004	David	2	15000	IT	New Delhi
5	1005	Mark	2	9000	IT	New Delhi
6	1006	Steve	3	13500	Finance	Mumbai
7	1007	Alice	3	10500	Finance	Mumbai

SQL COUNT

Example 1

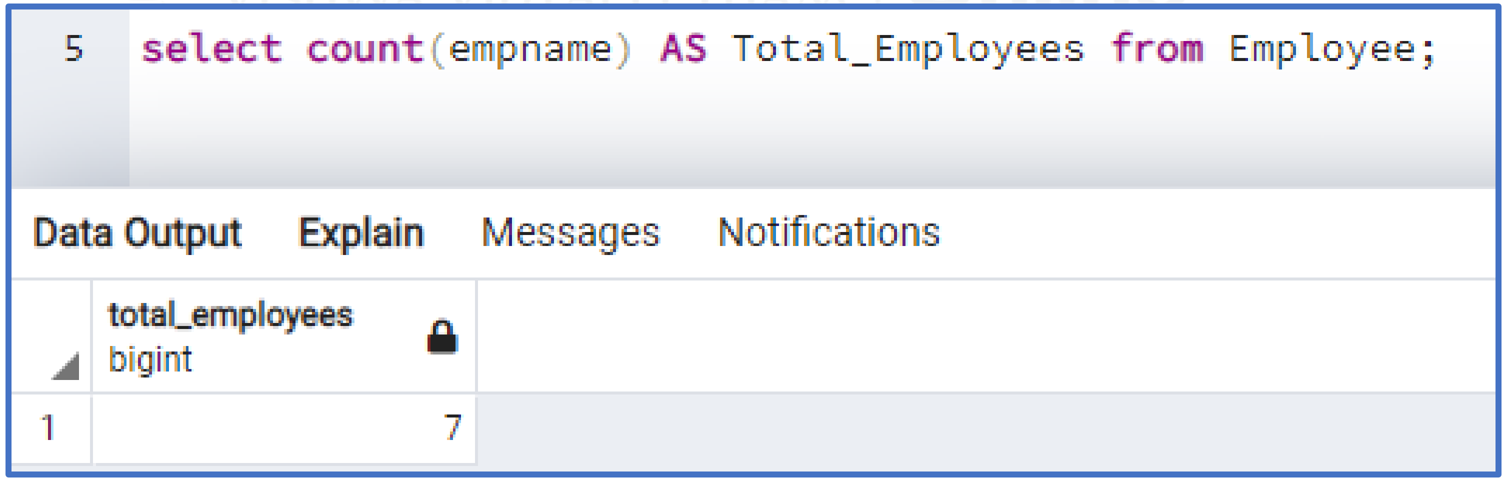


```
select count(*) from Books;
```

Data Output | Explain | Messages | Notifications

```
   count
   bigint
1       6
```

Example 2



```
select count(empname) AS Total_Employees from Employee;
```

Data Output | Explain | Messages | Notifications

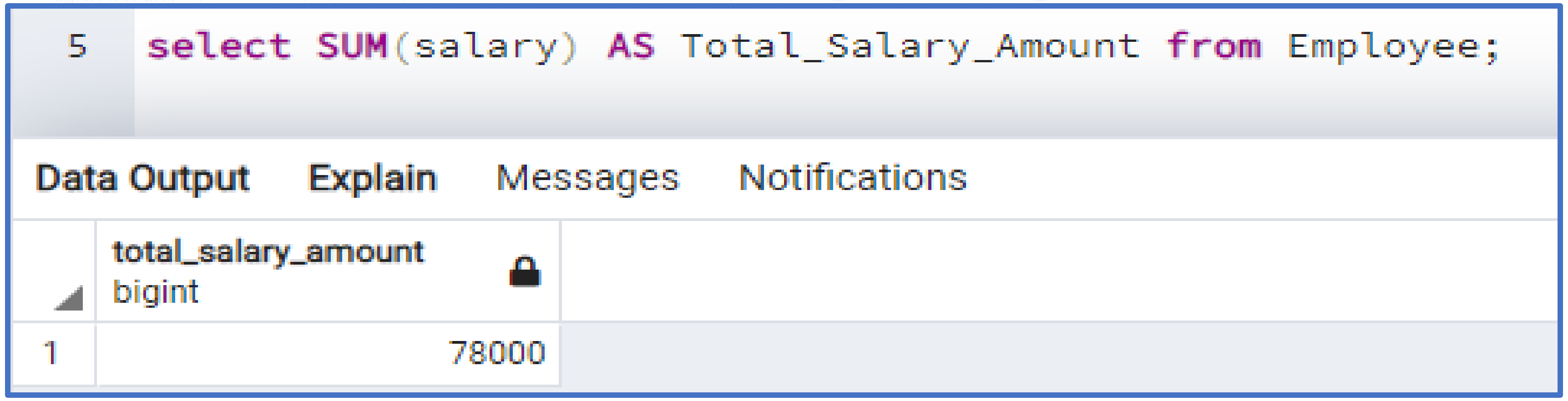
```
   total_employees
   bigint
1                7
```

SQL SUM

- Totals the values in a given column.
- Can only use SUM on columns containing numerical values.

Syntax : SELECT SUM(<column_name>) FROM <Table_Name>;

Example 3:



```
select SUM(salary) AS Total_Salary_Amount from Employee;
```

	total_salary_amount bigint
1	78000

SQL MIN & MAX

- MIN and MAX are SQL aggregation functions that return the lowest and highest values in a particular column.

❑ Syntax :

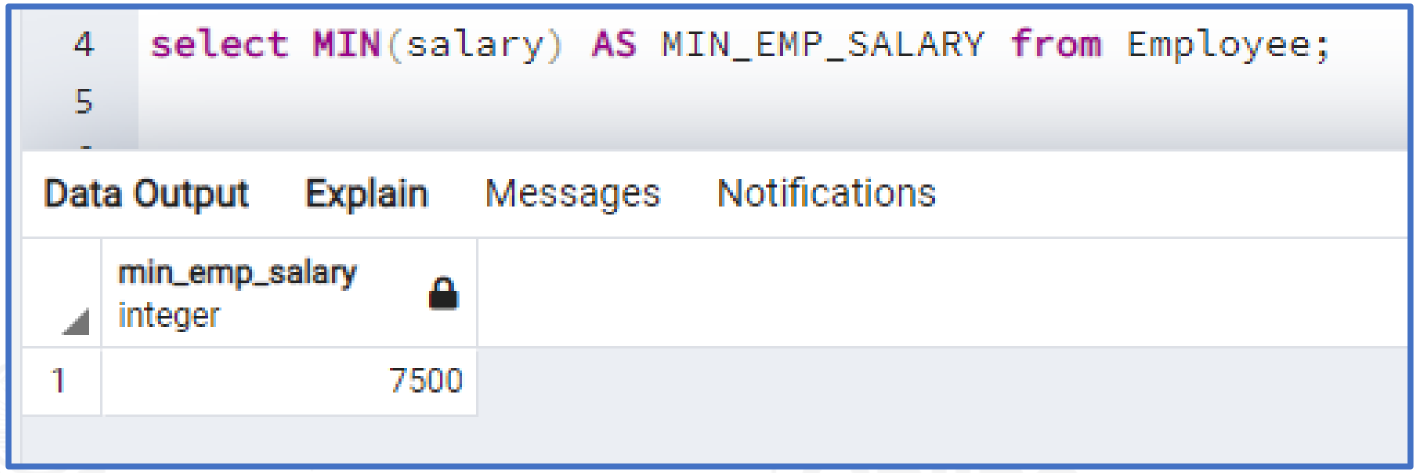
```
SELECT  MIN(<column_name>) FROM <Table_Name>;
```

❑ Syntax :

```
SELECT  MAX(<column_name>) FROM <Table_Name>;
```

Example: SQL MIN / MAX

Example 4 :

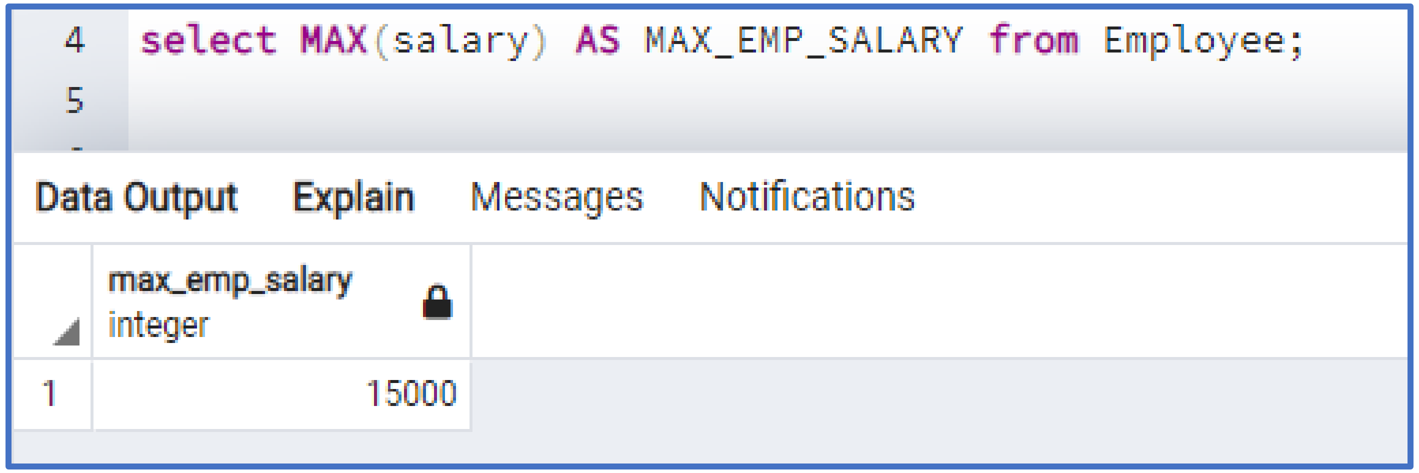


```
select MIN(salary) AS MIN_EMP_SALARY from Employee;
```

Data Output Explain Messages Notifications

	min_emp_salary integer
1	7500

Example 5 :



```
select MAX(salary) AS MAX_EMP_SALARY from Employee;
```

Data Output Explain Messages Notifications

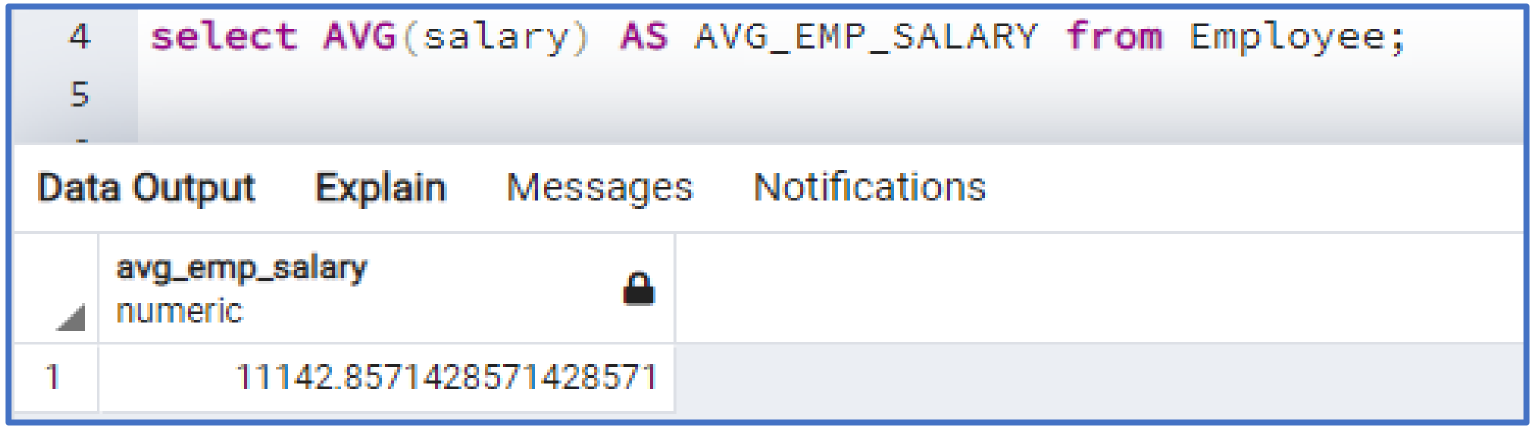
	max_emp_salary integer
1	15000

SQL AVG

- Calculates the average of a selected group of values.
- It's very useful, but has some limitations.
  - First, it can only be used on numerical columns.
  - Second, it ignores null values completely.

Syntax : SELECT AVG(<column_name>) FROM <Table_Name>;

Example 6 :



```
select AVG(salary) AS AVG_EMP_SALARY from Employee;
```

Data Output | Explain | Messages | Notifications

```
  avg_emp_salary
  numeric
1      11142.8571428571428571
```

Summary

- Discussed SQL Statements.

Reference

- Modern database management / Jeffrey A. Hoffer, V. Ramesh, Heikki Topi. – 10th edition. Pearson Publication.