What is SQL?

- SQL is a language to communicate with DATABASE
- SQL - Structured Query Language (S-E-Q-U-E-L)
- SQL is Domain Specific Language (DSLs) or it is like a scripting language.

-SQL/DS(data System) was - 1981 by IBM , 1st commercial RDBMS.

-SQL helps us to access - STORE, PROCESS, ANALYZE and MANIPULATE the datas.

--CRUD - [CREATE, READ, UPDATE & DELETE] operations that can be used in the SQL.

Introduction to SQL SERVER

SQL stands for the structured query language. SQL Server is a relational database management system (RDBMS) developed by Microsoft. By using SQL Server we can store and retrieve data from one place called database and use that data whenever it required in other programming applications. SQL is same as other relational database management systems like Sybase, Mysql, pl/sql, etc.

Uses of SQL Server

- It will allow us to access data in database and perform different operations on data - CRUD
- We can create databases, tables, and stored procedures to access data in required format
- We can do insert, update, delete, select, etc. to communicate with data stored in databases.
- We can create views, functions, indexes, keys, etc. in a database to get data in an efficient way
- In SQL we can create roles and user-level permissions to access data in databases
- We can access SQL database data from other languages using SQL modules, libraries

History of SQL Server

The journey of SQL started in 1989, originally code base of SQL sold by Sybase SQL Server to Microsoft. They launched this product as a competitor for IBM, Oracle, etc.

What is Latesh version of sql server?
The current version is Microsoft SQL Server 2022, released November 16, 2022. The RTM version is 16.0. 1000.6. (varies based on the updates)
https://support.microsoft.com/en-us/topic/kb4518398-sql-server-2019-build-versions-782ed548-1cd8-b5c3-a566-8b4f9e20293a

MICROSOFT SERVER / MS-SQL

Why MS-Server?

-- its a RDBMS tool.
-- Introduced by Micorsoft Company in 1989.
-- SQL server is a backend tool which helps to develop backend application

-- SQL server will provide more GUI facility.

- 1. GUI - Graphical User Interface
  Without Query Statement

- 2. CUI - Character User Interface
  With Query Statement

HOW to work with SQL server?

-- INSTANCE
- Collection of Objects
- Objects can be - Tables, views, procedures etc...
- in single instace - we can have 32767 DBs.
- The instance will allcate/create a memory in server

** SQL server is collection of DBs <<------>> every DBs is collection of Objects

Server Types

SSMS tool:
1. Database Engine

MSBI tool ---> to intract with MSBI tool we need developing tool (BIDS)
MSBI -- Microsoft Business Intelligence Tool
BIDS -- Business intelligence Development Studio

2. Analysis Services (SSAS)
3. Reporting Services (SSRS)
4. Intergration Services (SSIS)

**SSMS will not support MSBO ttol (SSAS, SSRS, SSIS)

SERVER TYPES

DATABASE ENGINE :
--its a core compentent of SQL Server
--its a main service/core service of SSMS

'CORE' -- Major part of Database Engine
--STORES -- Stores data in table format
--PROCESSING -- Insert, Update, Delete datas
--SECURITY -- Provides security for datas

Database Engine is most important server that MSBI tools.

ANALYSIS SERVICES (SSAS):
-- Creation of cubes with data marts/data warehouses for insightful and quicker data analysis

-- Two major alternatives – Multidimensional (for corporate analytics) and Tabular (for personal and team analytics)
-- Mostly used in Data Warehousing environment

REPORTING SERVICES (SSRS):
-- Faster processing of reports on both relational and multidimensional data
-- SSRS allows reports to be exported in different formats. You can deliver SSRS reports using emails
-- SSRS provides a host of security features, which helps you to control, who can access which report

INTERGRATION SERVICES (SSIS):
-- SSIS is a fast & flexible data warehousing tool used for data extraction, loading and transformation like cleaning, aggregating, merging data, etc.
--It makes it easy to move data from one database to another database. SSIS can extract data from a wide variety of sources like SQL Server databases, Excel files, Oracle and DB2 databases, etc.
-- SSIS tool helps you to merge data from various data stores
-- Automates Administrative Functions and Data Loading

SERVER NAME

-- Servername is nothing but Instance Name given to your SQLServer instance
-- For default instance, it would be same as the Server Name or the Machine Name
-- For named instance, it would be ServerName\InstanceName or MachineName\InstanceName
-- You can also use IPAddress instead of ServerName

AUTHENTICATION

There are two authentication modes in SQL Server using which you can login and connect with the SQL Server.

1. WINDOWS Authentications:
-- Windows authentication mode enables local Windows authentication with SQL Server, where you can login with your local Windows credentials.

2. SERVER Authentication:
-- Database administrators create SQL logins and provide appropriate permissions for users to authenticate themselves to SQL Server. Users need to specify the login and password while connecting to SQL Server

Data Types in SQL Server

In SQL, the data type is an attribute used to specify a type of data the column can hold, like numeric type, character type, binary type, date and time type, etc.

==========
==========
==========

SQL Exact Numeric Data Types

The exact numeric data types in sql are useful to store an integer type of data. Please check the following table for exact numeric data types in sql server.

Data Type	Description	Storage Size
bit	It’s single bit integer that can take values of 0, 1 or null	
tinyint	It’s single Byte integer we can store values from 0 to 255 (Minval: 0, Maxval: 255)	1 Byte
smallint	It’s 16 bit integer we can store values from -2^15 (-32,768) to 2^15 - 1 (32,767)	2 Bytes
int	It’s 32 bit integer we can store values from -2^15 (-32,768) to 2^15 - 1 (32,767)	4 Bytes
bigint	It’s a 64-bit integer we can store values from -2^63 (-9223372036854775808) to 2^63-1 (9223372036854775807)	8 Bytes
decimal	We can store values from -10^38 1 to 10^38 –1	5 - 17 Bytes
numeric	We can store values from -10^38 1 to 10^38 –1	5 - 17 Bytes
smallmoney	We can store monetary values from - 214,748.3648 to 214,748.3647	4 Bytes
money	We can store monetary values from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 Bytes

==========
SQL Approximate Numeric Data Types

In sql, Approximate numeric data types are used to store numeric data with floating-point i.e. floating-point data is approximate therefore not all values in the data type range can be represented exactly.

Data Type	Description	Storage Size
Float(n)	We can store values from - 1.79E 308 to 1.79E 308	Depends on the value of n
real	We can store values from - 3.40E 38 to 3.40E 38	4 Bytes

==========
==========
==========

SQL Date and Time Data Types

In sql, datetime data types are used to store a date or date and time data in particular column. We have a different type of date and time data types are available in sql server, those are

Data Type	Description	Storage Size
smalldatetime	We can store values From January 1, 1900 to June 6, 2079, and it’s having an accuracy of 1 minute	4 Bytes
datetime	We can store values from January 1, 1753, to December 31, 9999 and it’s having an accuracy of 3.33 milliseconds	8 Bytes
datetime2	We can store values from January 1, 0001 to December 31, 9999 and it’s having accuracy of 100 nanoseconds	6 - 8 Bytes
date	We can store only date format like May 05, 1986, and the range is from January 1, 0001 to December 31, 9999	3 Bytes
time	We can store only time format like 12.00 PM	3 - 5 Bytes
datetimeoffset	It’s same as datetime2 with the addition of a time zone offset	8 - 10 Bytes

==========

String Data Types in SQL Server

In sql, string data types are used to store any kind of data in the table. In string data types, we have an option to allow users to store either the fixed length of characters or huge length data based on their requirements.

In SQL we have a different type of string data types available, those are

- Character String Data Types
- Unicode Character String Data Types

SQL Character String Data Types

- char datatype

In sql, char datatype is used to store a fixed length of characters. Suppose if we declare a char(50), then it will allocate a memory for 50 characters to hold 50 characters of data. In case, if we insert only 10 characters of a string, then only 10 characters of memory will be used and remaining 40 characters of memory will be wasted.

- varchar datatype

In sql, varchar means variable characters and it is used to store non-unicode characters. It will allocate the memory based on the number characters inserted. Suppose if we declare varchar(50), then it will allocate memory of 0 characters at the time of declaration. Now if we insert only 10 characters of a string, then it will allocate memory for only 10 characters.

Data Type	Description	Storage
char	It’s a fixed-length character string and we can store maximum 8000 characters	Based on defined width
varchar(n)	It’s a variable-length character string and we can store maximum 8000 characters	2 Bytes number of characters
varchar(max)	It’s a variable-length character string and we can store maximum 2^31-1 characters (upto 2 GB)	2 Bytes number of characters

==========

SQL Unicode Character String Data Types

In sql, Unicode character string data types are used in a situation where we required to store huge data. In Unicode character string, we have a different type of string data types available, those are

- nchar
- nvarchar
- nvarchar(max)

Data Type	Description
nchar	It’s a fixed-length character string and we can store maximum 4000 characters
nvarchar(n)	It’s a variable-length character string and we can store maximum 4000 characters
nvarchar(max)	It’s variable-length character string and we can store maximum 2^30-1 characters (upto 2 GB)

==========

==========

Binary Data Types in SQL Server

In SQL, binary data types are used to store any kind of binary data like images, word files, text files, etc. in the table. In binary data types, we have an option like allowing users to store fixed-length or variable length of bytes based on requirements.

In SQL, we have a different type of binary data types available, those are

- binary datatype we can use binary datatype whenever the size of data entries are consistent.
- varbinary datatype we can use varbinary datatype whenever the size of data entries are considerably varies.
- varbinary(max) datatype we can use varbinary(max) datatype whenever the size of data entries exceeds 8000 bytes.

Data Type	Description	Storage
binary(n)	It’s fixed-length binary data and we can store maximum 8000 bytes	Based on the defined size
varbinary	It’s variable-length binary data and we can store maximum 8000 bytes	Based on number of bytes
varbinary(max)	It’s a variable-length character string and we can store maximum 2GB data	Based on the number of bytes

==========

Miscellaneous Data Types in SQL Server

we have different datatypes that will not come under string datatypes, binary data types, date and time and numeric datatypes those will be called miscellaneous or other data types.

we have a different type of miscellaneous data types available, those are

- cursor datatype we can use cursor datatype to hold a reference of cursor objects. Any variables created with cursor datatype are nullable and cursor datatype cannot be used for a column while creating a table.

- XML datatype we will use XML datatype to store XML formatted data in the column. In the XML datatype column, we can store a maximum of 2 GB data

- table datatype table datatype will be useful to store a table result set in variable for processing later time. The table variables can be used in stored procedures, functions and these variables cleaned up automatically at the end of function or stored procedures.

- uniqueidentifier datatype the uniqueidentifier datatypes are used to store a globally unique identifier (GUID).

- sql_variant datatype sql_variant datatype is used to store a various sql supported data types like int, char, binary, etc.

Data Type	Description
cursor	It is used to store a reference to cursor object
XML	It is used to store XML formatted data and we can store maximum 2GB data
table	It is used to store table result set for processing at a later time
uniqueidentifier	It is used to store globally unique identifier (GUID)
sql_variant	It is used to store values of various sql supported data types except text,ntext, varchar(max), xml, etc… We can store a maximum of 8000 bytes

==========

What is DATA?
-- Data can be anything
-- You and I are a DATA.
-- Data can be in the form of text, numbers or date formats
-- A storing of a data in a file is called database
-- Data can be sotred in Notepad, excel or SQL server

Challenges in File Data Storage
-- Size of Data
-- Ease of updating
-- Accuracy
-- Security
-- Redundancy
-- Incomplete Data

==========

```
==================================
What is Database?
-- Database is a storage system that has collection of data
-- Relstional Database, Stores data in form of a table that can be easily retrived.

================================================================================
================================================================================
==================================
What is SQL?
-- SQL is a language to communicate with database
-- Its a DOMAIN SPECIFIC LANGUAGE (DSL)/Scripting Langugae
-- SEQUEL - Structured English Query Language
-- SEQUEL was developed by IBM corporation
-- SEQUEL later become [SQL] but still prononced as "SEQUEL" or "S-Q-L"
-- SQL/DS (data System) was released in 1981 by IBM, its commercial Database
-- SQL commands/syntax helps us to do CRUD operations

================================================================================
================================================================================
==================================
What is Tables?
--Tables are Database Objects(DBO) that contains all the data.
--In a table, Datas are logically organized in a row-and-columns
      -Each ROW - UNIQUE RECORD
      -Each COL - Fields in the record

================================================================================
================================================================================
==================================
Features of SQL
-- Access any data with the releational Database
-- SQL is very fast in doing CRUD operations
-- SQL is very versatile as it works with database systems
-- SQL helps us to manage database without knowing lot of coding
```

--SUB LANGUGAES of SQL.

DDL, DML, DQL, DCL, TCL
--
DDL - Data Definition Language
- works with the Structure of the table
What is structure?

table, name of the table, columns, Datatypes and constraints

DDL ----
CREATE
ALTER
DROP
TRUNCATE
RENAME

What is DDL?
--Data Definition Language
--Works on the structure of the table (columns, datatypes, table name, columns name, constraints)
--DDL will not affect the data's present in the table
--DDL involves in [creation of new table][updating/inserting into existing table - such as changing the table/column name, delete/remove - column/table]
[remove the table from database]

==========
==========
==========
CREATE - to create a object

SYNTAX:
CREATE DATABASE <database_name>
CREATE TABLE <table_name>

==========
==========
==========

ALTER - Modify created object

1. MODIFY - Alter column - change the datatype and size of the datatype
2. ADD - we can add a new column
3. RENAME - we can change the column name
4. DROP - we can drop a column

SYNTAX:

1. ALTER COLUMN:
   ALTER TABLE <table_name>
   ALTER COLUMN <col_name><new datatype>[new size]

2. ADD:
   ALTER TABLE <table_name>
   ADD <new_col_name><datatype>[size]

3. RENAME: ----dont use in office environment[optional]
   SP_RENAME'<Table_name>.<old_col_Name>','<new_col_name>'

4.DROP:
   ALTER TABLE <table_name>
   DROP COLUMN <col_name>,<col_name2>,<col_name3>.....

==========

DROP - Deletes the object ----dont use in office environment[optional]

SYNTAX:
DROP DATABASE <database_name>
DROP TABLE <table_name>

==========

TRUNCATE - Delete all records and keeps the table ----dont use in office environment[optional]

SYNTAX:
TRUNCATE TABLE <table_name>

==========

RENAME - Rename the table/Column ----dont use in office environment[optional]

SYNTAX:

SP_RENAME '<old_tablename>','<new_tablename>'

--DDL
--CREATE
-- I can create a DATABASE
-- I can create a TABLE

Other Syntax:
USE <Database> - make sure your using the desired database
sp_help (sp means stored procedure(object)) - to see the structure of the table

SYNTAX

1. CREATE
CREATE DATABASE <DATABASENAME>
CREATE TABLE <TABLENAME> (<col_name>[datatype], <col_name2>[Datatype])

2. ALTER
1
2
3
4

3. DROP

DROP DATABASE <database_name>
DROP TABLE <table_name>

4. TRUNCATE

TRUNCATE TABLE <table_name>

5. RENAME
1. RENAME COL
2. RENAME TABLE

--SUB LANGUGAES of SQL.

DDL, DML, DQL, DCL, TCL
--
DQL/DRL - Data Query/Reterival Language

main purpose is to

1. Reterive data, Read Data or display data in result window

- SELECT

    SELECT * FROM <table_Name>
    * - ALL (all the columns)

[SELECT *] - i sheling in examining the columns and datas of a table.

3 types of reterival:

1. PROJECTION METHOD - single table - WHERE CONDITION
2. SELECTION METHOD - single table - WITHOUT CONDITION
3. JOINS - Multiple tables

==========

DML - Data Manipulation Language

- modify(s) the data present in the table.
- it has no relation with structure of the table.

DML involes in :

- INSERT

    INSERT INTO <Table_name> VALUES <Col_Value1, Col_values2, Col_values3.....>

- UPDATE

    UPDATE <table_name> SET <Col_name1> = Value1, ... [WHERE]

- DELETE

    DELETE FROM <Table_name>
    [WHERE]

-** Any changes which is made by DML statement will be called as "TRANSACTION - TCL" / in order to handle the changes done by the database by DML, we need TCL.

-** the changes made by DML are permenant

**** What is diffrence between DELETE, DROP and TRUNCATE

OPERATORS:

In SQL, operators is asymbol which is used to specify a particular action that is performed on one or more expression

We have diff. types of OPERATORS:
1. SQL Arithmatic Operators
2. SQL Logical Operators
3. SQL Camparison Operators
4. Assignment Operators

==========

1. SQL Arithmatic Operators :

- Addition -- (+)
- Subtraction -- (-)
- Multiplication -- (*)
- Division -- (/)
- Modulos -- (%)

SYNTAX :

SELECT <Column_Name>, <column_nameX> (arithmatic Operators like +, -, /, %, *) <column_nameY> AS <Final_Col_Name> FROM <TABLE NAME>

Ex: SELECT emp_name, emp_basic_salary + emp_bonus_salary AS emp_total_salary
FROM emp_details

==========

2. SQL Logical Operatorss :

--Logical is a symbol or word used to connect two or more expression

-- AND -- AND operator is used to compare data with more than one condition, and it will return records only when all the defined conditions are TRUE
-- OR -- OR for Comparing data with more than one condition and it will return records when EITHER of the conditions is TRUE
-- BETWEEN // NOT BETWEEN -- BETWEEN Operatorsis usefull to get the values within the defined range // NOT BETWEEN - opposite of BETWEEN show rest other values other than the given range
-- IN // NOT IN -- IN operator is usefull to search for the specifed value that matches with any value in the set of multiple values or NOT // NOT IN - retrives rest other values

-- LIKE // NOT LIKE -- LIKE operator searchs for the character string with specifed pattern using wildcards in Column. Pattern means - string of characters with wildcards to search matching expressions

SYNTAX : AND : SELECT * FROM <TN>
WHERE condition1 = 'expression1' AND condition2 = 'expression2'.....

ex: SELECT * FROM <TN>
WHERE location = 'Guntur' and location = 'Chennai'

==========

SYNTAX : OR : SELECT * FROM <TN>
WHERE condition1 = 'expression1' OR condition2 = 'expression2'.....

ex: SELECT * FROM <TN>
WHERE location = 'Guntur' OR Salary > 50000

==========

LIKE :

SYNTAX : LIKE : SELECT * FROM <TN>
WHERE condition1 LIKE '%[Char]' --- Anything which BEGINs with

SYNTAX : LIKE : SELECT * FROM <TN>
WHERE condition1 LIKE '[Char]%' --- Anything which ENDs with

SYNTAX : LIKE : SELECT * FROM <TN>
WHERE condition1 LIKE '%[Char]%' --- Contains a WORD

NOT LIKE :

SYNTAX : NOT LIKE : SELECT * FROM <TN>
WHERE condition1 NOT LIKE '%[Char]' --- Anything which does not BEGINs with

SYNTAX : NOT LIKE : SELECT * FROM <TN>
WHERE condition1 NOT LIKE '[Char]%' --- Anything which does not ENDs with

SYNTAX : NOT LIKE : SELECT * FROM <TN>
WHERE condition1 NOT LIKE '%[Char]%' --- Does not contain a WORD

==========

IN :

SYNTAX : IN : SELECT * FROM <TN>
WHERE condition1 IN (exp1, exp2, exp3, exp4.....)

NOT IN :

SYNTAX : NOT IN : SELECT * FROM <TN>
WHERE condition1 NOT IN (exp1, exp2, exp3, exp4.....)

```
================================================

BETWEEN :

      SYNTAX : IN : SELECT * FROM <TN>
                  WHERE condition1 BETWEEN range1  AND range2    -- for numeric


      SYNTAX : IN : SELECT * FROM <TN>
                  WHERE condition1 BETWEEN 'Char1'  AND 'Char2'  -- for
alphabet/character

NOT BETWEEN :


      SYNTAX : IN : SELECT * FROM <TN>
                  WHERE condition1 NOT BETWEEN range1  AND range2


      SYNTAX : IN : SELECT * FROM <TN>
                  WHERE condition1 NOT BETWEEN 'Char1'  AND 'Char2'

================================================================================
================================================================================
================================================================
================================================================================
================================================================================
================================================================

3. SQL Camparison Operators

 -- NOT EQUAL                 -- != OR <>
 -- EQUAL               -- =
 -- GREATER THAN EQUAL TO     -- >=
 -- LESS THAT EQUAL TO        -- <=
 -- GREATER THAN        -- >
 -- LESS THAN                 -- <

================================================================================
================================================================================
================================================================
================================================================================
================================================================================
================================================================

4. Assignment Operators

 -- EQUAL TO            -- =
```

```
CLAUSES:
      0. WHERE

      1. GROUP BY
            -- It is used to GROUPING the similar data based on the
columns

            SYNTAX :

            SELECT col_1, col_2....., aggregrate function(col_X)
            FROM <TN>
            [WHERE]
            GROUP BY expX;

//////// //////// //////// //////// ////////  --- run it in the ssms

CREATE DATABASE groupBYClause;
Go

USE groupBYClause;
Go

CREATE TABLE sales (
      OrderId int,
      Product varchar(50),
      category varchar(50),
      qty int,
      price decimal(10, 2)
);
Go

INSERT INTO sales VALUES
(1, 'product A', 'Category X', 10, 100.00),
(2, 'Product B', 'Category X', 5, 50.00),
(3, 'Product C', 'Category Y', 8, 80.00),
(4, 'Product D', 'Category Y', 3, 120.00),
(5, 'Product E', 'Category Z', 12, 100.00),
(6, 'Product F', 'Category Z', 6, 90.00)
Go
INSERT INTO sales VALUES
(7, 'product A', 'Category X', 100, 100.00),
(8, 'product A', 'Category X', 500, 100.00);

SELECT * FROM sales;
Go

SELECT category,  sum(Price)
FROM sales
GROUP BY category ;

--------------

--AGG FUNCTION
SELECT count(EmpName) as Countof FROM <TN>;
SELECT SUM(Salary)  as sumofsalary FROM <TN>
SELECT MAX(Salary) FROM <TN>
SELECT MIN(Salary) FROM <TN>
SELECT AVG(Salary) FROM <TN>
```

\\\\\\\\ \\\\\\\\ \\\\\\\\ \\\\\\\\ \\\\\\\\

aggregrate function : SUM, MAX, MIN, COUNT, AVG

2. HAVING -- The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

SYNTAX :

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

--The HAVING clause places the condition in the groups defined by the GROUP BY clause in the SELECT statement.

--This SQL clause is implemented after the 'GROUP BY' clause in the 'SELECT' statement.

--This clause is used in SQL because we cannot use the WHERE clause with the SQL aggregate functions. Both WHERE and HAVING clauses are used for filtering the records in SQL queries.

3. ORDER BY

- Order by expression is optional
- it is used to sort the records in your table/reslt in the form of ASC (Ascending) order or DESC (Descending) order

SYNTAX :

```
SELECT COL1, COL2, CO13 FROM <TN>
ORDER BY COL_NAME [ASC || DESC]
```

DIFFERENCE BETWEEN WHERE CLAUSE and HAVING CLAUSE

HAVING

1. The HAVING clause is used in database systems to fetch the data/values from the groups according to the given condition.
2. The HAVING clause is always executed with the GROUP BY clause.
3. The HAVING clause can include SQL aggregate functions in a query or statement.
4. We can only use SELECT statement with HAVING clause for filtering the records.
5. The HAVING clause is used in SQL queries after the GROUP BY clause.
6. We can implements this SQL clause in column operations.
7. It is a post-filter.
8. It is used to filter groups.

WHERE

1. The WHERE clause is used in database systems to fetch the data/values from the tables according to the given condition.
2. The WHERE clause can be executed without the GROUP BY clause.
3. We cannot use the SQL aggregate function with WHERE clause in statements.
4. Whereas, we can easily use WHERE clause with UPDATE, DELETE, and SELECT statements.
5. The WHERE clause is always used before the GROUP BY clause in SQL queries.
6. We can implements this SQL clause in row operations.
7. It is a pre-filter.
8. It is used to filter the single record of the table.

```
/*

What is SET OPERATORS?
      Set Operators is used to combine two or more than 2 SELECT
STATEMENTS

1. UNION
2. UNION ALL
3. INTERSECT
4. EXCEPT

1. UNION
      -- Union operator is using to combine two or more than two selected
statements as a SINGLE STATEMENT or SINGLE UNIT of VALUES
      -- That's WITHOUT DUPLICATE VALUES

      SYNTAX :

            ----------------------------------------
            |                                      |
            |     SELECT * FROM <TABLE_NAME>       |
            |     [WHERE condition]                |
            |     UNION                            |
            |                                      |
            |     SELECT * FROM <TABLE_NAME2>      |
            |     [WHERE condition]                |
            ----------------------------------------

2. UNION ALL
      -- Union operator is using to combine two or more than two selected
statements as a SINGLE STATEMENT or SINGLE UNIT of VALUES
      -- That's WITH DUPLICATE VALUES

      SYNTAX :

            ----------------------------------------
            |                                      |
            |     SELECT * FROM <TABLE_NAME>       |
            |     [WHERE condition]                |
            |     UNION ALL                        |
            |     [WHERE condition]                |
            |     SELECT * FROM <TABLE_NAME>       |
            |                                      |
            ----------------------------------------


3. INTERSECT
      -- To return Common Values from the table

      SYNTAX :

            ----------------------------------------
            |                                      |
```

```
            |     SELECT * FROM <TABLE_NAME>              |
            |     [WHERE condition]                       |
            |     INTERSECT                                     |
            |     [WHERE condition]                       |
            |     SELECT * FROM <TABLE_NAME>              |
            |                                                   |
            ----------------------------------------
```

4. EXCEPT
      -- To return all values from LEFT HAND side table which are not found in RIGHT HAND table

      SYNTAX :

```
            ----------------------------------------
            |                                                   |
            |     SELECT * FROM <TABLE_NAME>              |
            |     [WHERE condition]                       |
            |     EXCEPT                                        |
            |     [WHERE condition]                       |
            |     SELECT * FROM <TABLE_NAME>              |
            |                                                   |
            ----------------------------------------
```

==> Rules to Use [Set Operators]
-> No. Of Columns should be same with both select statements
-> ORDER of the columns should be same
-> DataTypes of the columns must be same

*/

TCL - Transcation Control Language

-----DML ---
Any changes made by the DML (Insert, update, Delete) statement will be called as [TRANSACTION] -- TCL.

In order to handle the changes done to the database by DML, we need TCL.

whatever we are doing in sql like create insert update modification after that if we get proper result is transaction.

--+++++++++++++++

--we need to control.

What is TRANSACTION? -- to perform some operation/actions/task

OPERATION? -- INSERT, UPDATE, DELETE (DML statements)
        -- Controlling these DML using TCL
        -- DML Operations are used to control the datas present in the table.

Control Points :
--BEGIN TRANSCTIONS
--COMMIT
      --Chance of ROLLING BACK without COMMIT keyword
      BEGIN TRANSCTION
      <write statement>

      --No chance of ROLLING BACK with COMMIT keyword
      BEGIN TRANSCTION
      <write statement>
      COMMIT

--ROLLBACK

      BEGIN TRANSACTION
      ROLLBACK

--SAVEPOINT
      --It is used to create a temp memory for storing a values which we want to conditionally ROLLBACK

      BEGIN TRANSACTION
      <Write statement>
      SAVE TRANSACTION <POINTER NAME>
      <Write statement>

When, [
      BEGIN TRANSCTION
      ROLLBACK
] - there is no going back, because, by DEFAULT - the records are permanently affected by the DML operations(insert, update, delete) / its perm. executed.

AUTOCOMMIT : --by the implicitly (system) /SQL seerver commited the DML operations which was performed by user on table. (automatically)

-- to Control the date of the table :
we must now use a COMMIT

-- COMMIT: To make the transction -- explicitly (by the user)

--BEGIN TRANSCION
  <Write a statement>
  COMMIT

****with begin transaction (new transaction) if we did some mistake we can rollback but in commit is to make current change permanent.

```
CREATE DATABASE TCL_languages;
USE TCL_languages;

--CREATE TABLE
CREATE TABLE dept (
 branch_id int,
 branch_name varchar(100),
 branch_location varchar(100)
);

SELECT * FROM dept
--DML of --INSERT
INSERT INTO dept VALUES (101, 'SBI', 'CHN')

BEGIN TRANSACTION
ROLLBACK;

--DML 0f --UPDATE
UPDATE dept
SET branch_location = 'VLC'
WHERE branch_id = 101;

BEGIN TRANSACTION
ROLLBACK;

--DML of --DELETE
DELETE FROM dept
WHERE branch_id = 101;

BEGIN TRANSACTION
ROLLBACK;
```

```
--COMMIT

--DQL
SELECT * FROM dept;

BEGIN TRANSACTION
```

```
INSERT INTO dept VALUES (1, 'SBI', 'CHN');
COMMIT

-- this transaction will be perment, when "COMMIT" statement is given

BEGIN TRANSACTION
ROLLBACK

------------------------------------------------------------
--SAVEPOINT

INSERT INTO dept VALUES
(101, 'SBI', 'HYD'),
(102, 'HDFC', 'CHN'),
(103, 'PNJ', 'DL'),
(104, 'ICICI', 'PNJ'),
(105, 'BOI', 'GJ');

SELECT * FROM dept;

BEGIN TRANSACTION
      DELETE FROM dept WHERE branch_id = 102
      DELETE FROM dept WHERE branch_id = 103
SAVE TRANSACTION XYZ
      DELETE FROM dept WHERE branch_id = 104

--from named memory (XYZ)
BEGIN TRANSACTION
ROLLBACK TRANSACTION XYZ

--form unnamed memory - buffer memory
BEGIN TRANSACTION
ROLLBACK
```

DQL -
SELECTION - With [WHERE] condition
PROJECTION - Without [WHERE] condition
JOINS - Reterive the data from two or more tables

JOIN - Reterive the data from two or more tables

-- Two types :

1. ANSI-FORMAT JOINS
-- with 'ON' keyword join condition

---- INNER JOIN

Reterive data based on [EQUALITY] condition/matching values
Should have common column and common column should be/must be have same datatypes
Matching data/rows from multiple tables

----SYNTAX :

SELECT */<LIST COL_NAMES> FROM <TABLE_NAME>
<JOIN KEY> <TB2> ON (JOINING CONDITION);

==========
==========
---- OUTER JOIN

Reterives matching data and unmatching data from the table

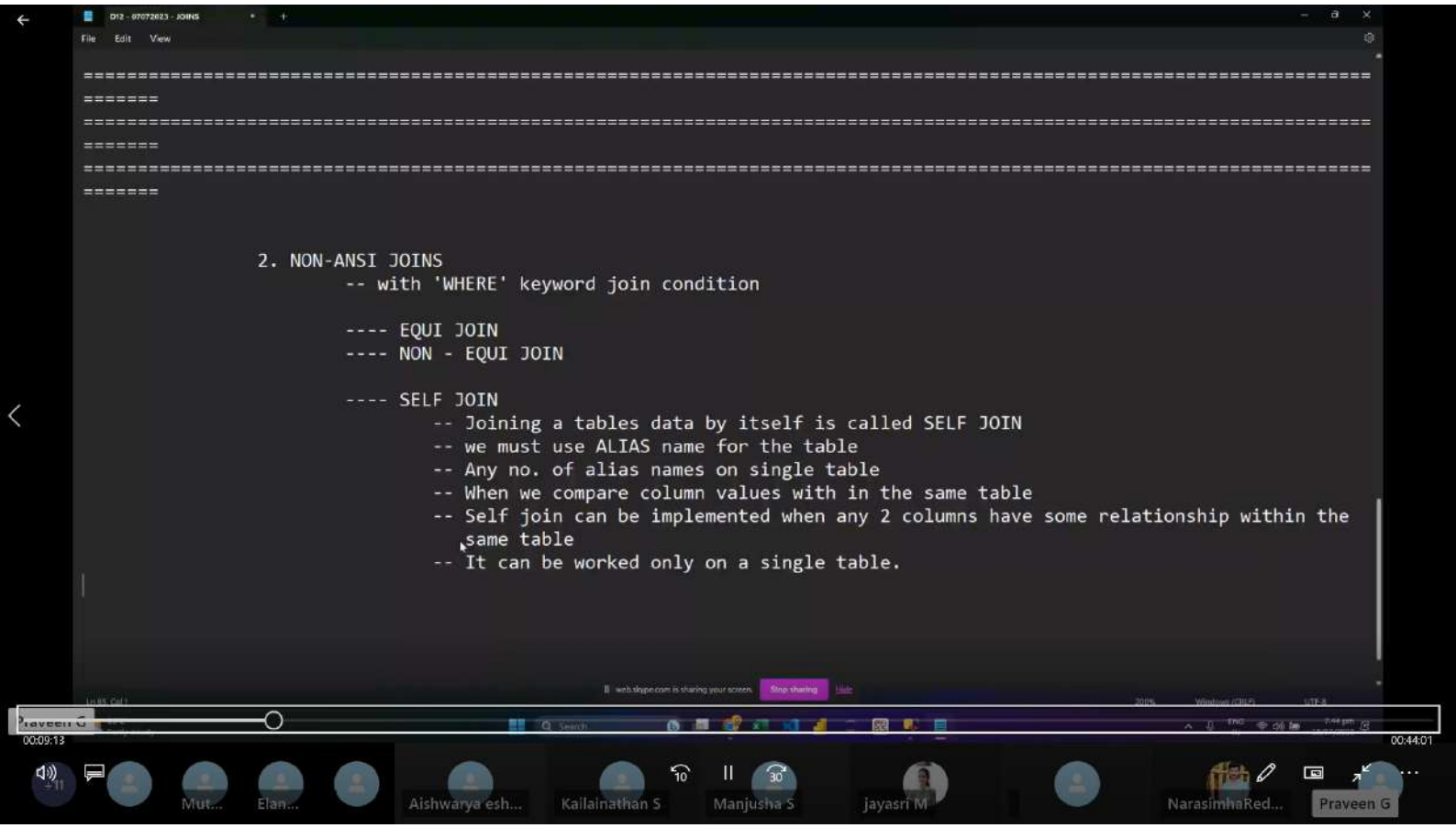
---- LEFT OUTER JOIN
-- Reterives the matching data from all table but unmatching data from left hand side only
-- Left table values will be fully shown but on right hand side it will show on the matching values but
for remaining values it will [NULL]

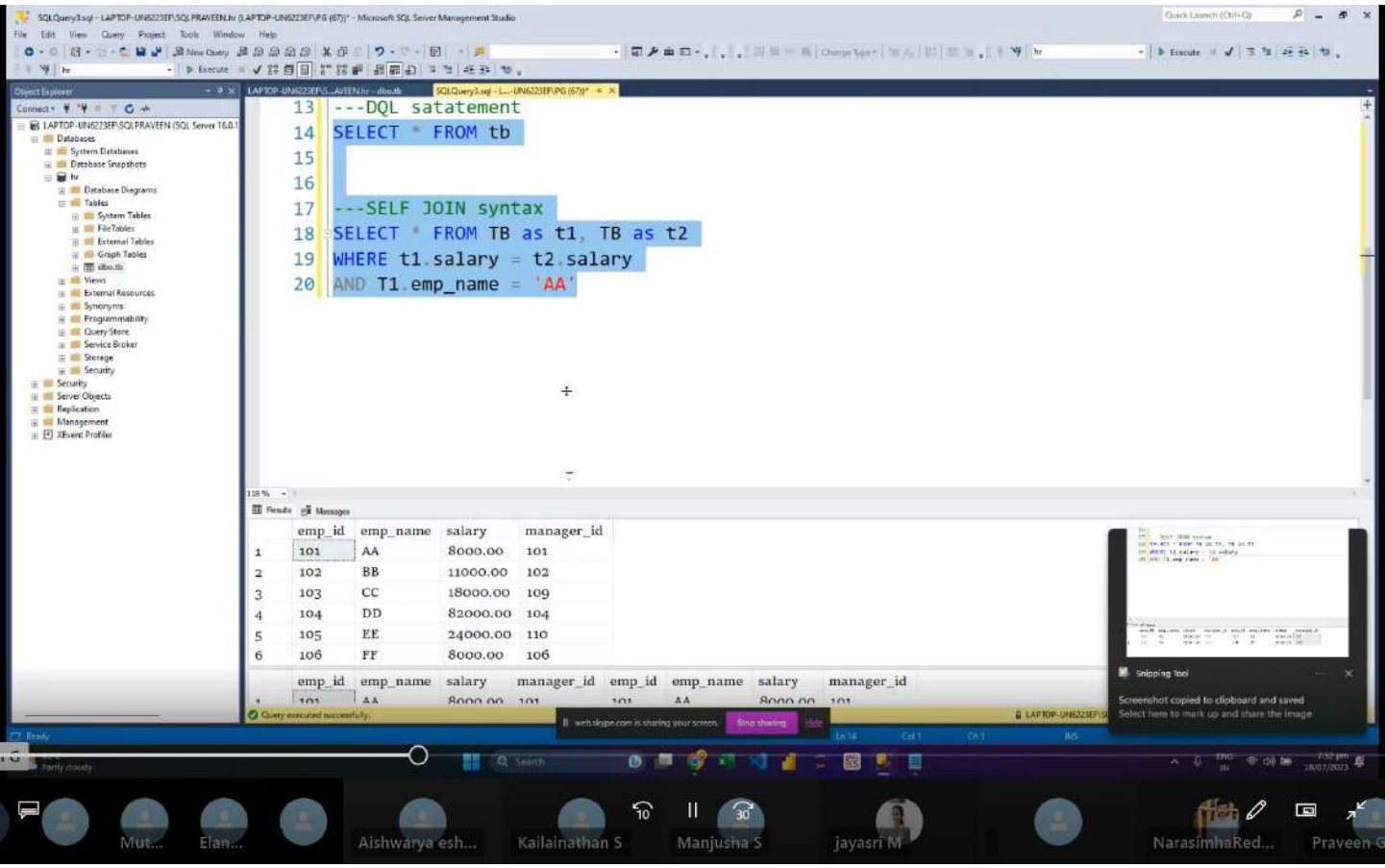
---- RIGTH OUTER JOIN
--Reterives the matching data from all table but unmatching data from the right side table only.
-- in this case, unmatching data will be showns [NULL]

---- FULL OUTER JOIN
-- it will show matching data and also unmatching data

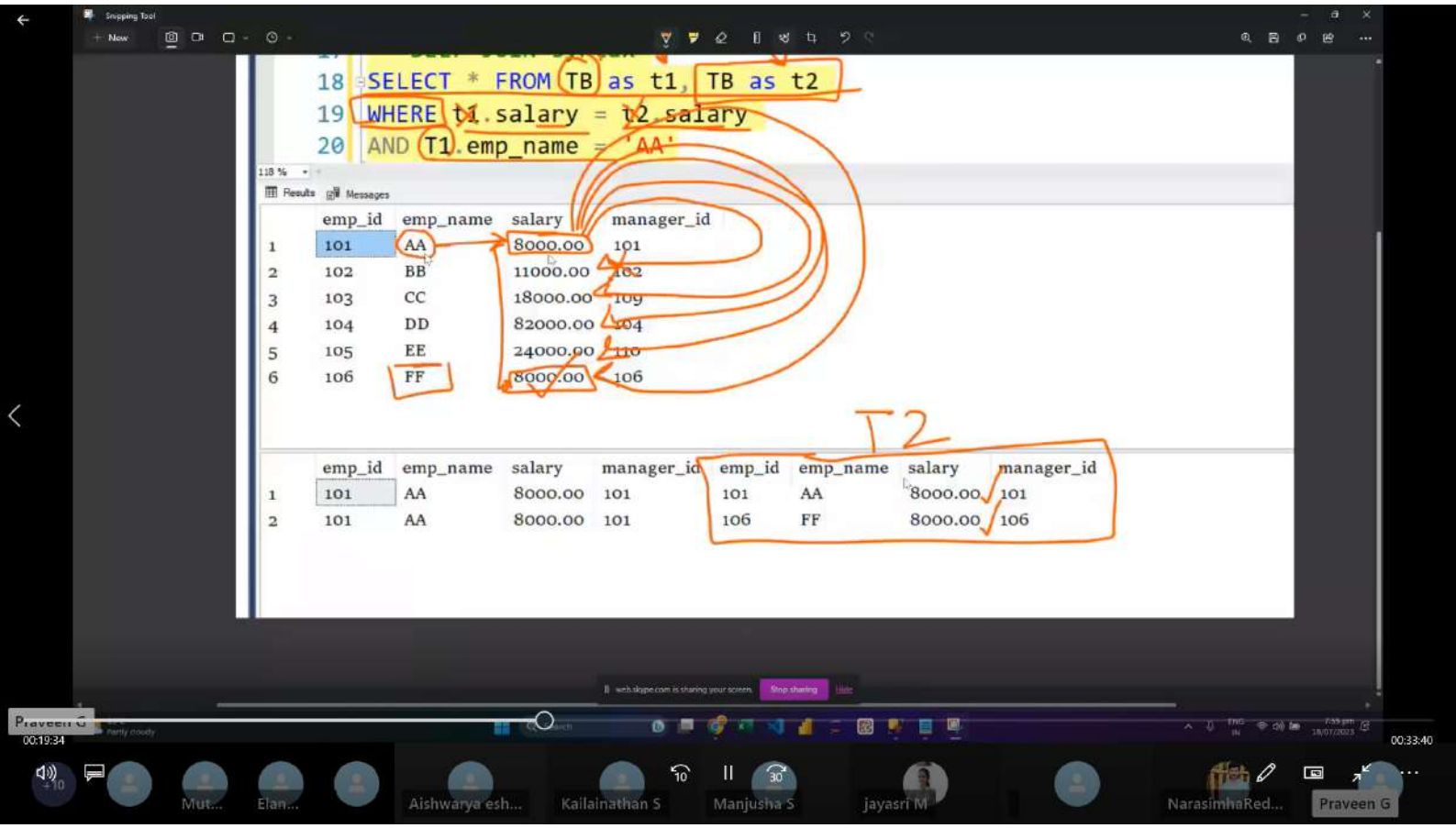
```
================================================================================
================================================
      ---- CROSS JOIN
                        -- The PRODUCT of rows(MxN) rows

                        ***-- CROSS JOIN is used to join two or more than
two tables data without  JOIN CONDITION
                        ***-- There is no required a common column in the
table


                  ---- SYNTAX:

                              SELECT * FROM <TABLE_NAME> CROSS JOIN
<TABLE_NAME2>

================================================================================
================================================
      ---- NATURAL JOIN
                        -- we can use [EQUI JOIN]

                  -- To avoid duplicate columns from JOIN TABLES
                  -- NATURAL JOIN is supported by (ORACLE) but not by SSMS
                  -- Instead of NATURAL JOIN, we can use [EQUI-JOIN]









SELECT * FROM student;
SELECT * FROM courses;


/*

SELECT star/<LIST COL_NAMES> FROM <TABLE_NAME>
<JOIN KEY> <TB2> ON (JOINING CONDITION);

*/

--INNERT JOIN

SELECT * FROM student INNER JOIN courses
ON
student.courseid = courses.courseid
```

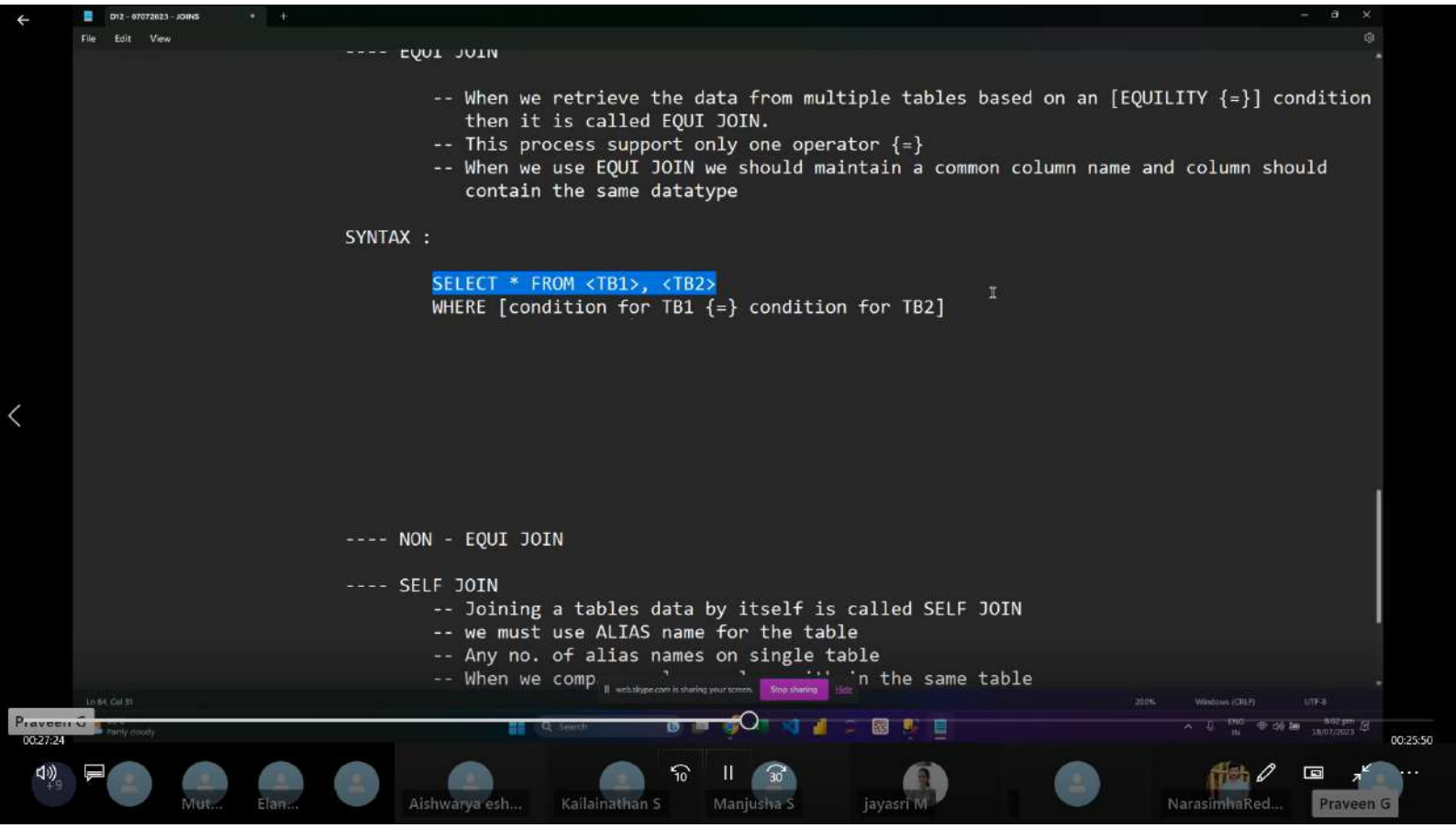
```
--OUTER JOIN
-- 2.1 LEFT OUTER JOIN
-- Reterives the matching data from all table but unmatching data from
left hand side only
SELECT * FROM courses LEFT OUTER JOIN student
ON
courses.courseid = student.courseid;

SELECT * FROM student LEFT OUTER JOIN courses
ON
student.courseid = courses.courseid;

-- 2.2 RIGHT OUTER JOIN
-- Reterives the matching data from all table but unmatching data from
the right side table only.

SELECT * FROM student RIGHT OUTER JOIN courses
ON
Student.courseid = courses.courseid


-- 2.3  FULL OUTER JOIN

SELECT * FROM student FULL OUTER JOIN courses
ON
Student.courseid = courses.courseid


--3. CROSS JOIN
SELECT * FROM student CROSS JOIN courses

================================================================================
================================================
================================================================================
================================================
================================================================================
================================================


           2. NON-ANSI JOINS
                 -- with 'WHERE' keyword join condition

                 ---- EQUI JOIN
                 ---- NON - EQUI JOIN
                 ---- SELF JOIN
```

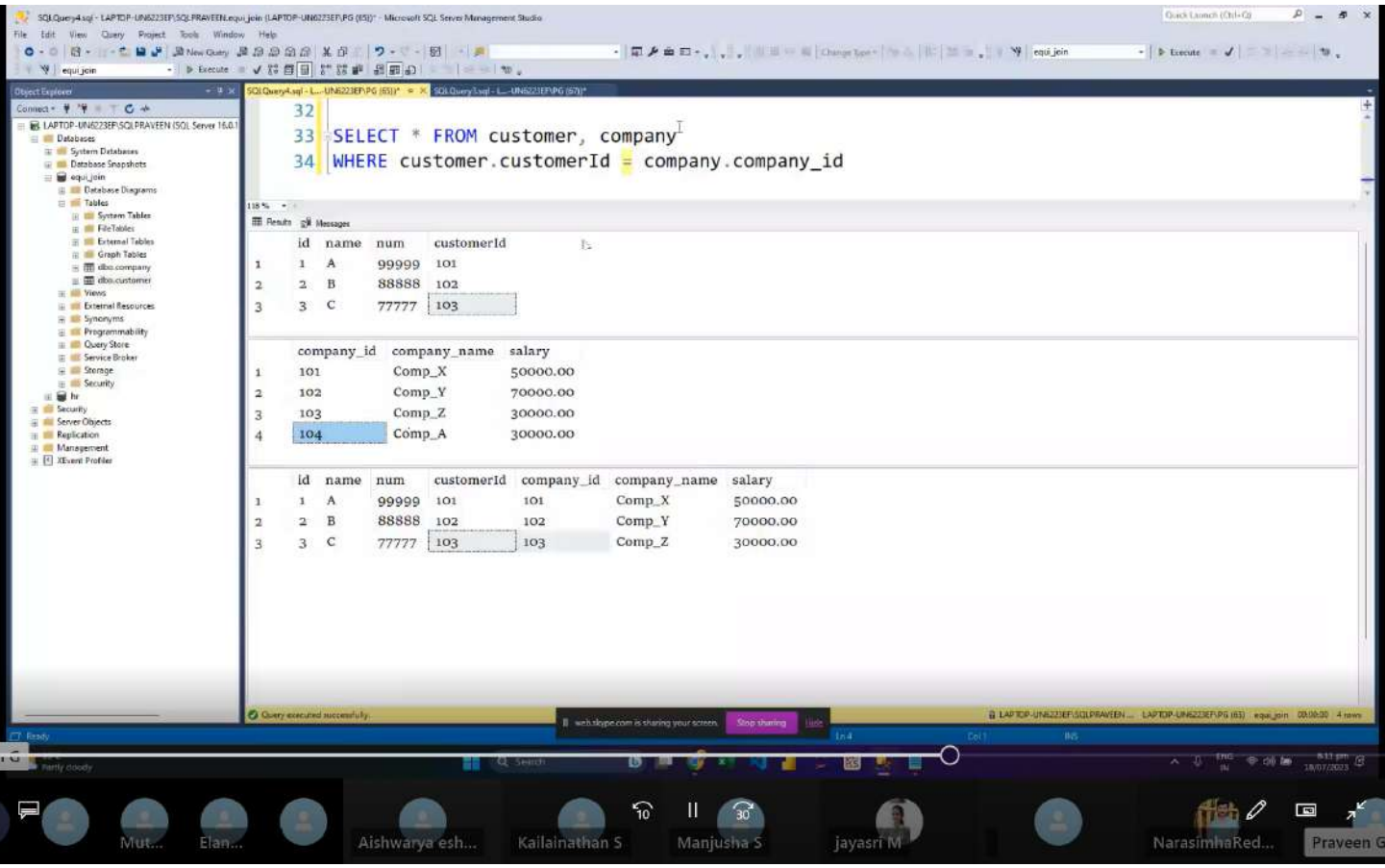

D12 - 07072023 - JOINS
File Edit View
==========
==========
==========
2. NON-ANSI JOINS
-- with 'WHERE' keyword join condition
---- EQUI JOIN
---- NON - EQUI JOIN
---- SELF JOIN
-- Joining a tables data by itself is called SELF JOIN
-- we must use ALIAS name for the table
-- Any no. of alias names on single table
-- When we compare column values with in the same table
-- Self join can be implemented when any 2 columns have some relationship within the same table
-- It can be worked only on a single table.
web.skype.com is sharing your screen.
Stop sharing
Hide
Praveen G
00:09:13
00:44:01
Mut...
Elan...
Aishwarya esh...
Kailainathan S
Manjusha S
jayasri M
NarasimhaRed...
Praveen G



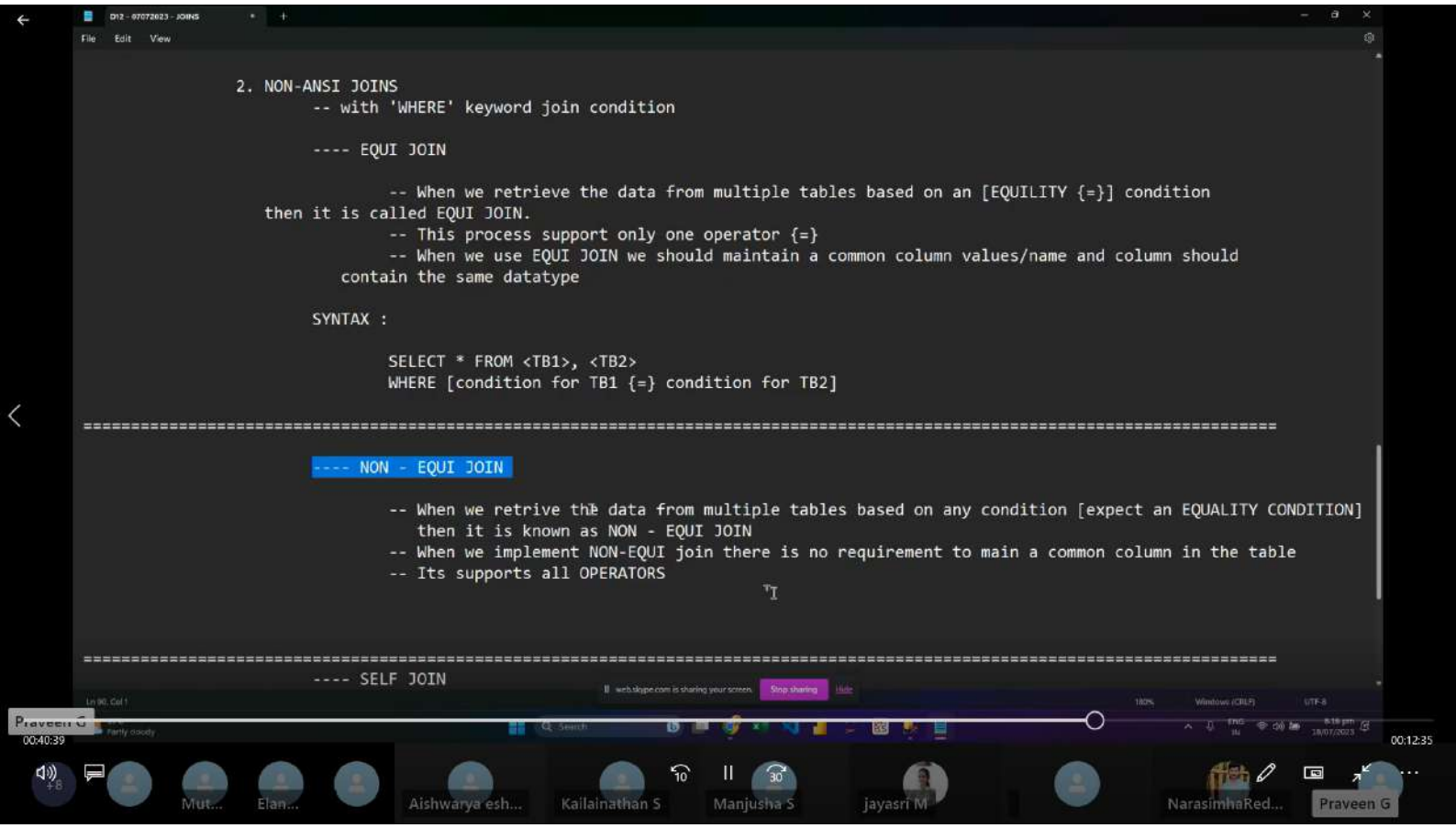
Praveen G
00:16:35
00:36:39
Mut...
Elan...
Aishwarya esh...
Kailainathan S
Manjusha S
jayasri M
NarasimhaRed...
Praveen G

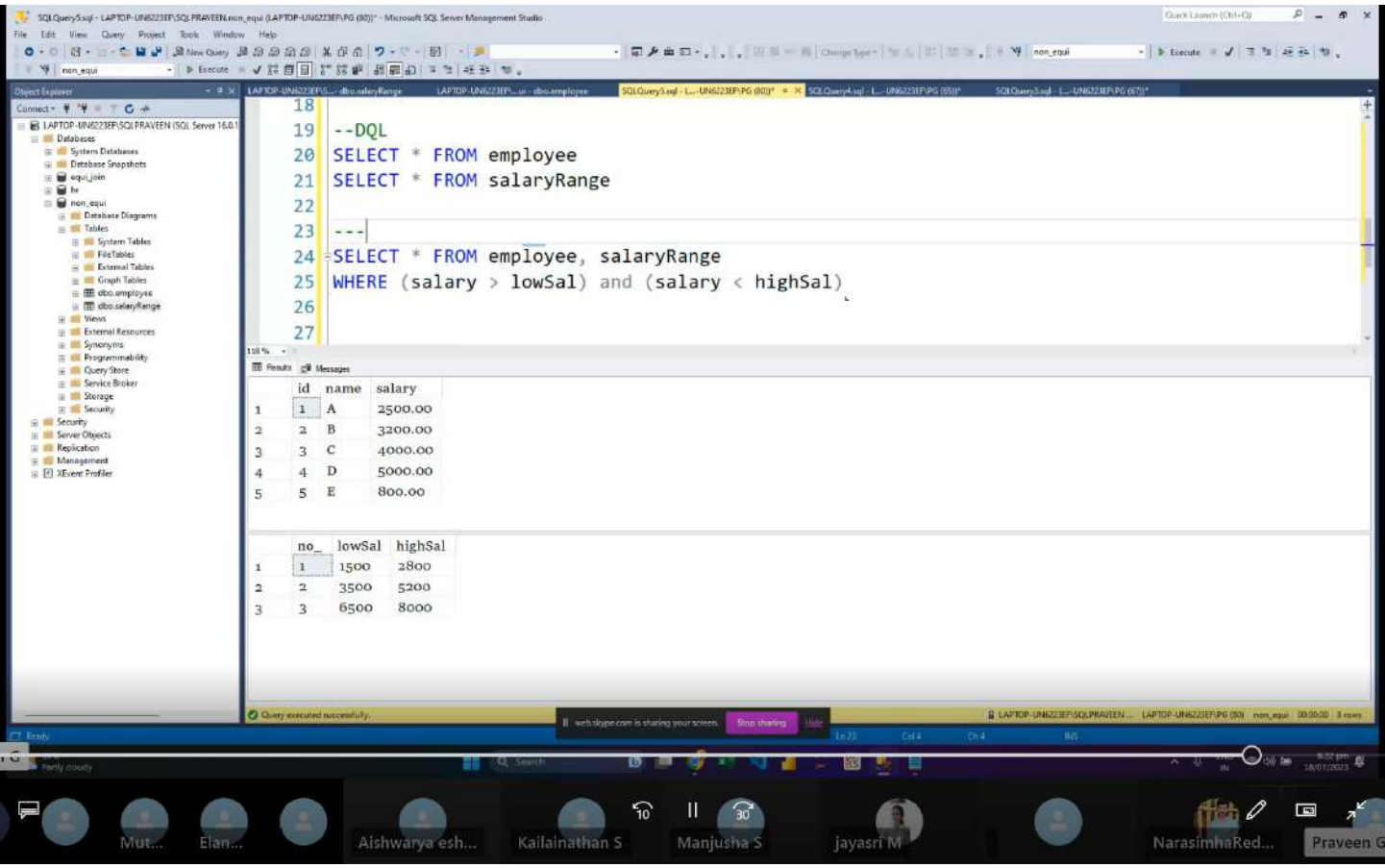

SELECT * FROM TB as t1, TB as t2
WHERE t1.salary = t2.salary
AND T1.emp_name = 'AA'
Results
Messages
emp_id emp_name salary manager_id
1 101 AA 8000.00 101
2 102 BB 11000.00 102
3 103 CC 18000.00 109
4 104 DD 82000.00 104
5 105 EE 24000.00 110
6 106 FF 8000.00 106
T2
emp_id emp_name salary manager_id emp_id emp_name salary manager_id
1 101 AA 8000.00 101 101 AA 8000.00 101
2 101 AA 8000.00 101 106 FF 8000.00 106


D12 - 07072023 - JOINS
File Edit View
---- EQUI JOIN
-- When we retrieve the data from multiple tables based on an [EQUILITY {=}] condition then it is called EQUI JOIN.
-- This process support only one operator {=}
-- When we use EQUI JOIN we should maintain a common column name and column should contain the same datatype
SYNTAX :
SELECT * FROM <TB1>, <TB2>
WHERE [condition for TB1 {=} condition for TB2]
---- NON - EQUI JOIN
---- SELF JOIN
-- Joining a tables data by itself is called SELF JOIN
-- we must use ALIAS name for the table
-- Any no. of alias names on single table
-- When we comp
n the same table
web.skype.com is sharing your screen.
Stop sharing
Hide
Praveen G
00:27:24
00:25:50
Mut...
Elan...
Aishwarya esh...
Kailainathan S
Manjusha S
jayasri M
NarasimhaRed...
Praveen G

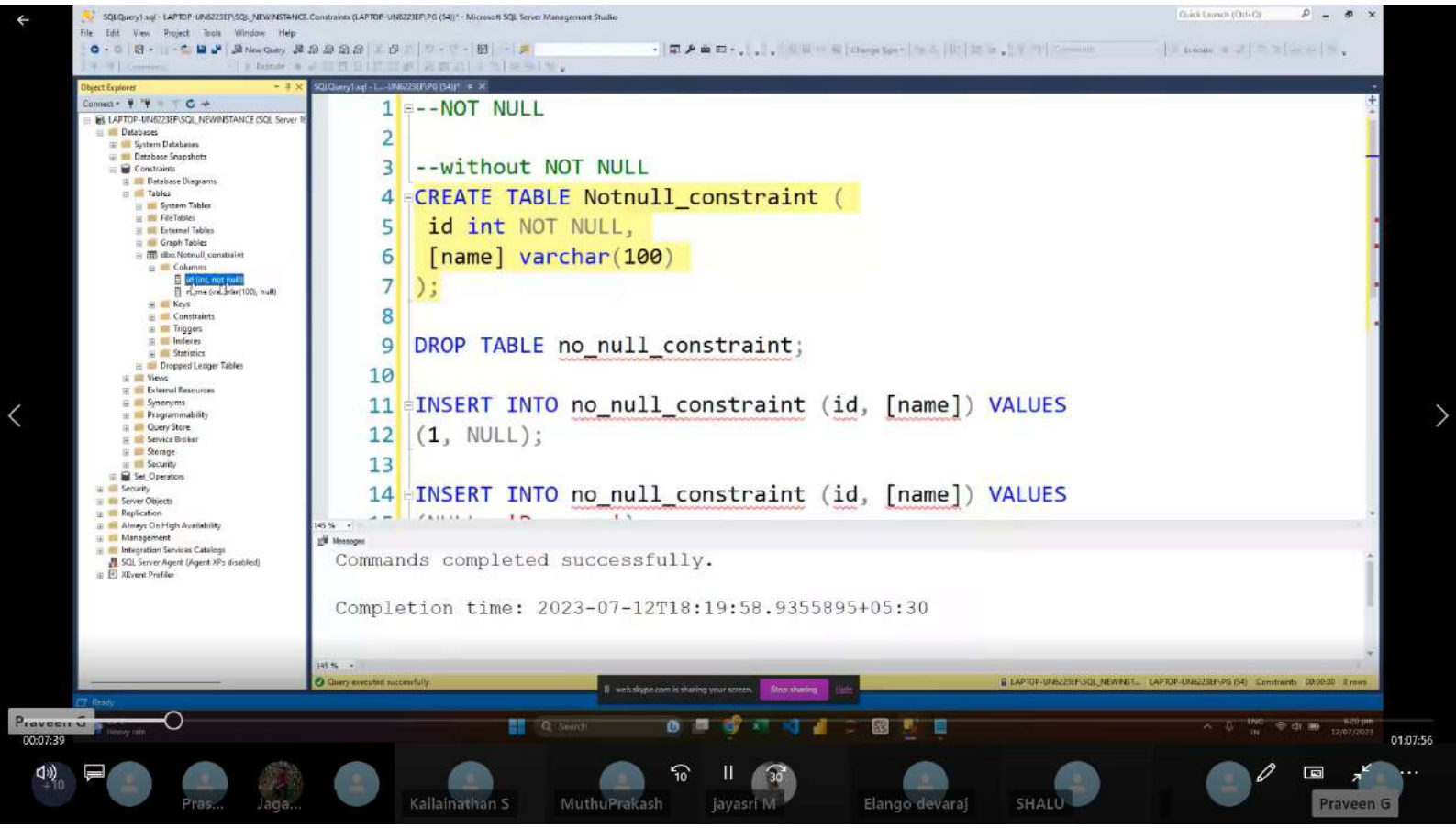


Praveen G
00:35:59
00:17:15
Mut...
Elan...
Aishwarya esh...
Kailainathan S
Manjusha S
jayasri M
NarasimhaRed...
Praveen G

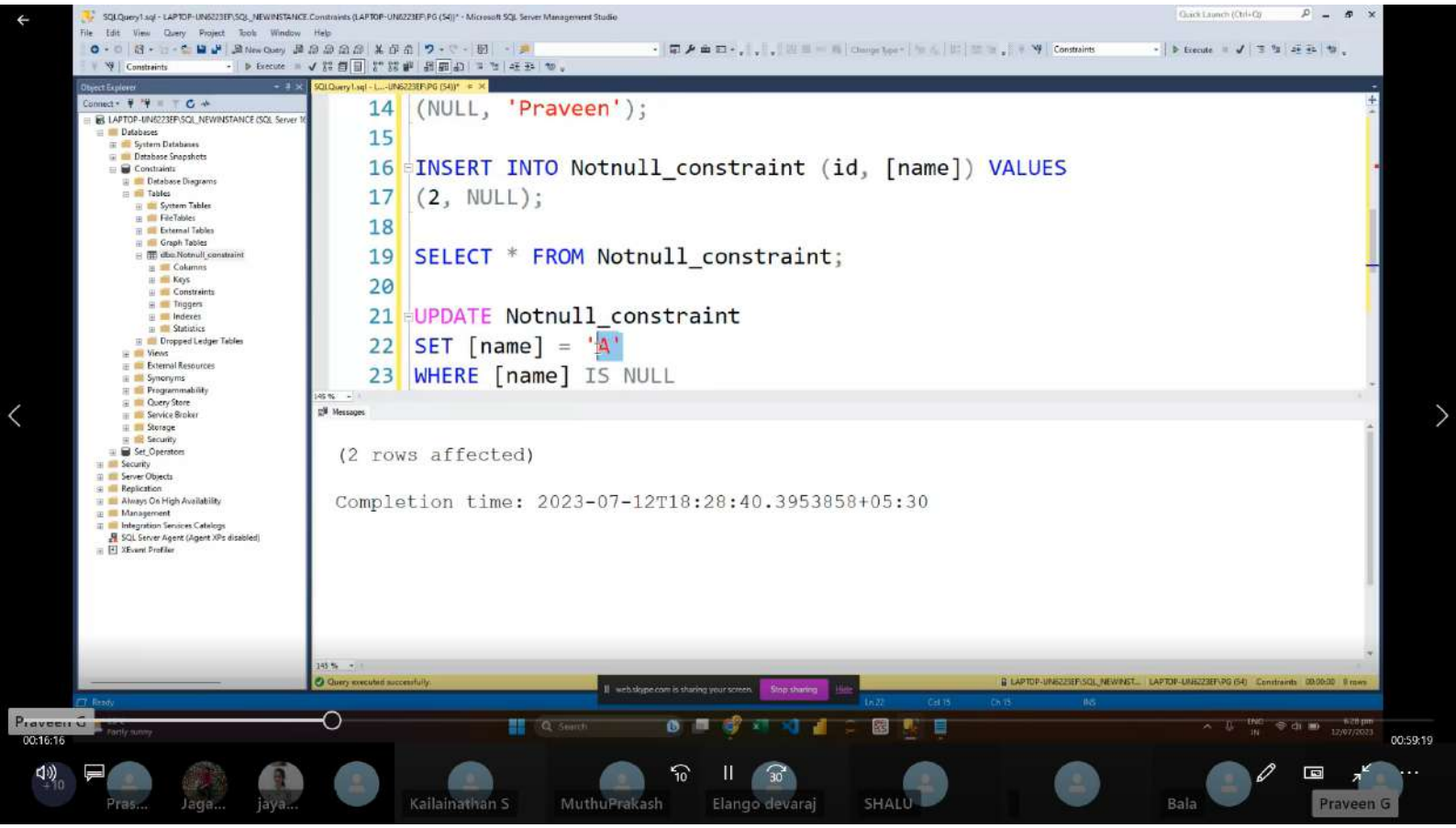

D12 - 07072023 - JOINS
File Edit View
2. NON-ANSI JOINS
-- with 'WHERE' keyword join condition
---- EQUI JOIN
-- When we retrieve the data from multiple tables based on an [EQUILITY {=}] condition
then it is called EQUI JOIN.
-- This process support only one operator {=}
-- When we use EQUI JOIN we should maintain a common column values/name and column should
contain the same datatype
SYNTAX :
SELECT * FROM <TB1>, <TB2>
WHERE [condition for TB1 {=} condition for TB2]
---- NON - EQUI JOIN
-- When we retrive the data from multiple tables based on any condition [expect an EQUALITY CONDITION]
then it is known as NON - EQUI JOIN
-- When we implement NON-EQUI join there is no requirement to main a common column in the table
-- Its supports all OPERATORS
---- SELF JOIN
web.skype.com is sharing your screen.
Stop sharing
Hide
Praveen G
00:40:39
00:12:35
Mut...
Elan...
Aishwarya esh...
Kailainathan S
Manjusha S
jayasri M
NarasimhaRed...
Praveen G



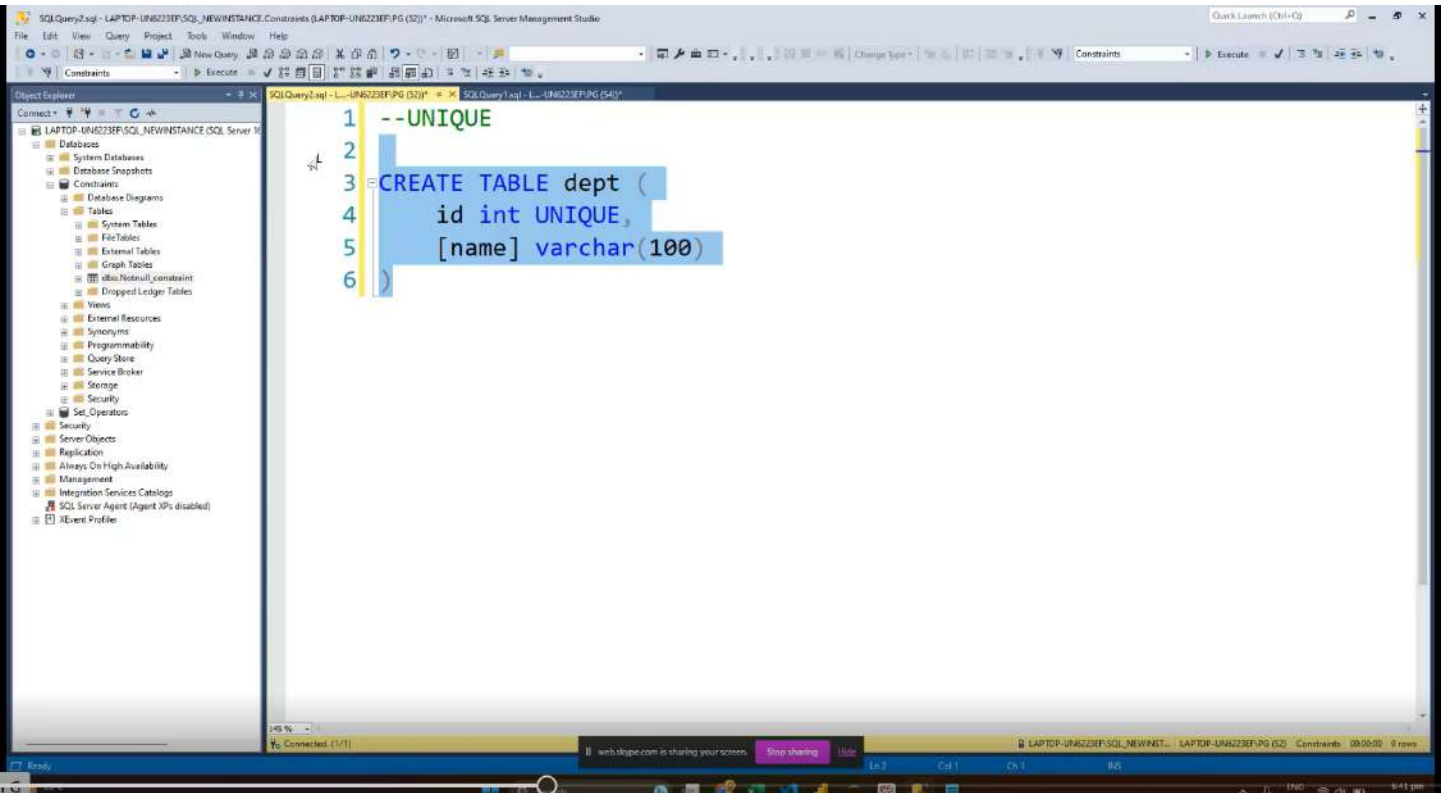
Praveen G
00:47:02
00:06:12
Mut...
Elan...
Aishwarya esh...
Kailainathan S
Manjusha S
jayasri M
NarasimhaRed...
Praveen G



Praveen G
00:07:39
01:07:56
Pras...
Jaga...
Kailainathan S
MuthuPrakash
jayasri M
Elango devaraj
SHALU
Praveen G



Praveen G
00:16:16
00:59:19
Pras...
Jaga...
jaya...
Kailainathan S
MuthuPrakash
Elango devaraj
SHALU
Bala
Praveen G

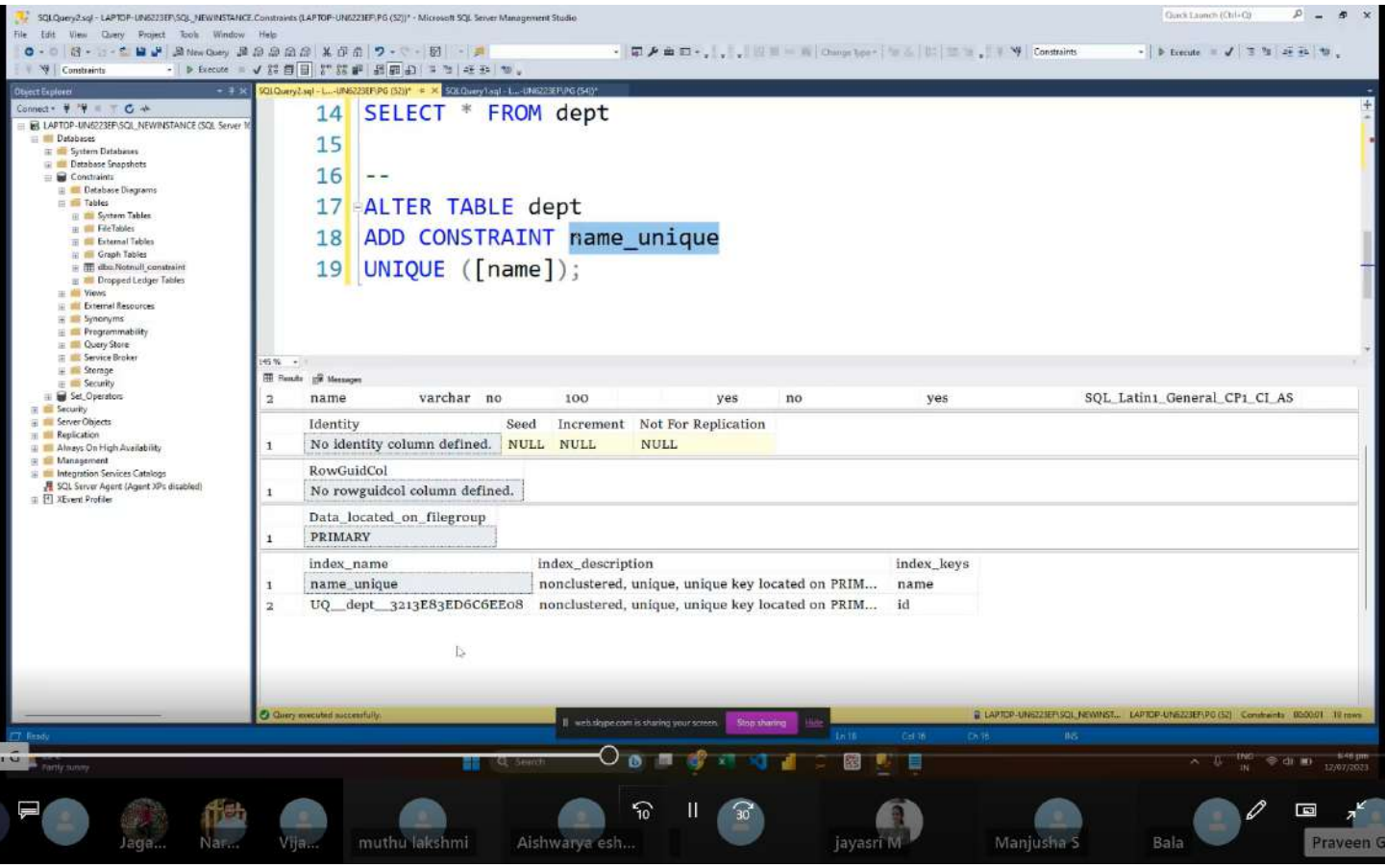


Praveen G

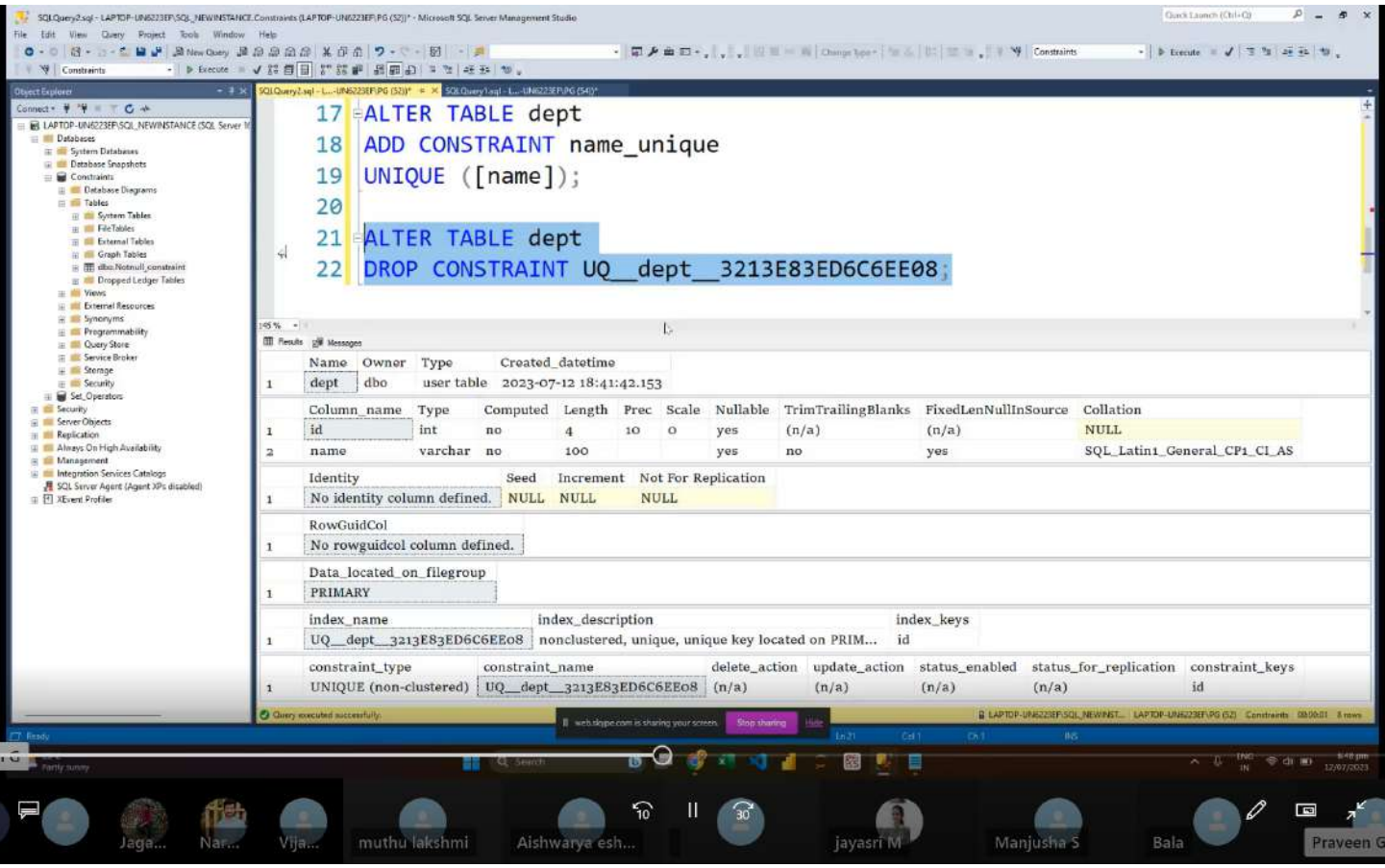
00:29:07

00:46:28

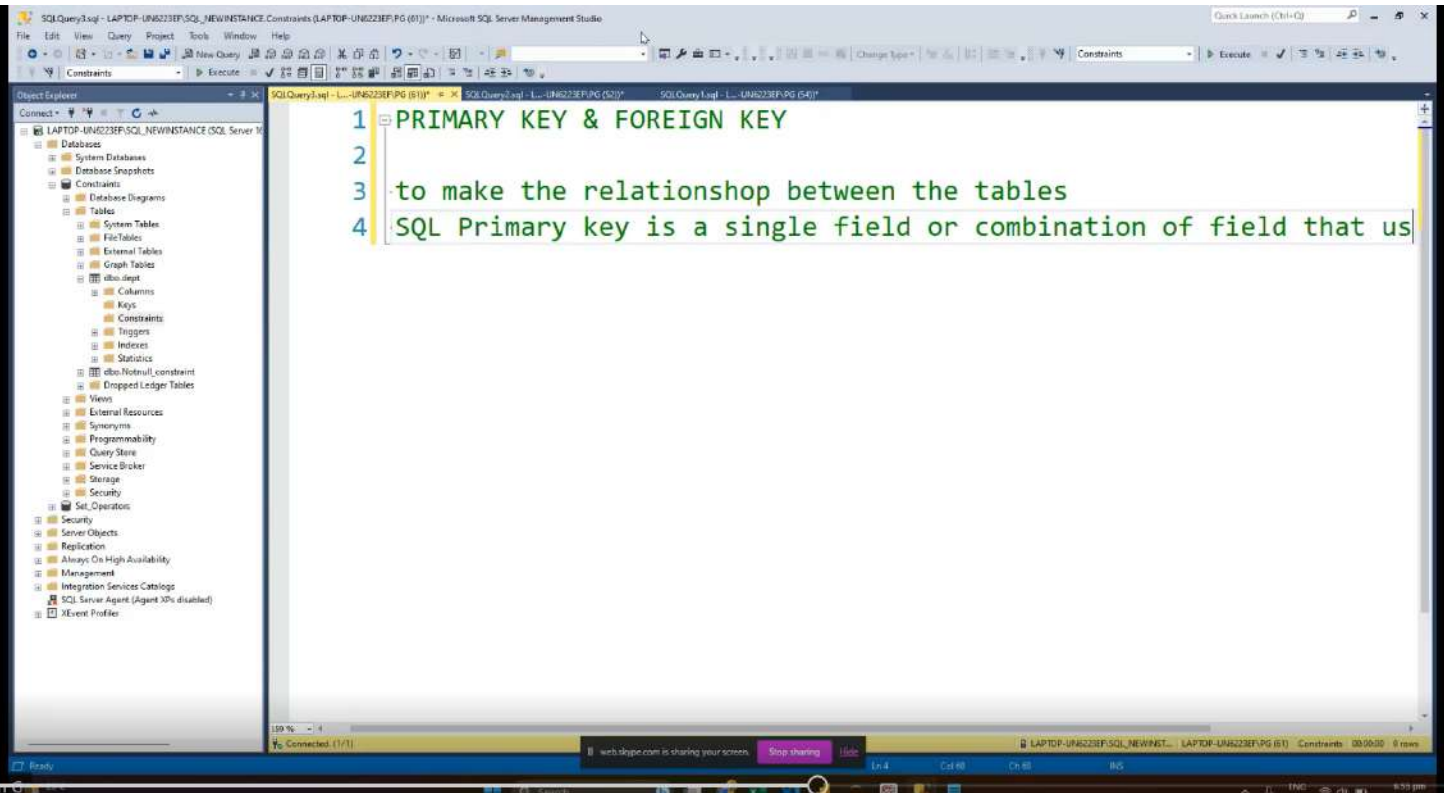
Nar... Vija... muthu lakshmi Aishwarya esh... jayasri M Manjusha S Bala Praveen G



Praveen G
00:33:27
00:42:08
Jaga...
Nar...
Vija...
muthu lakshmi
Aishwarya esh...
jayasri M
Manjusha S
Bala
Praveen G

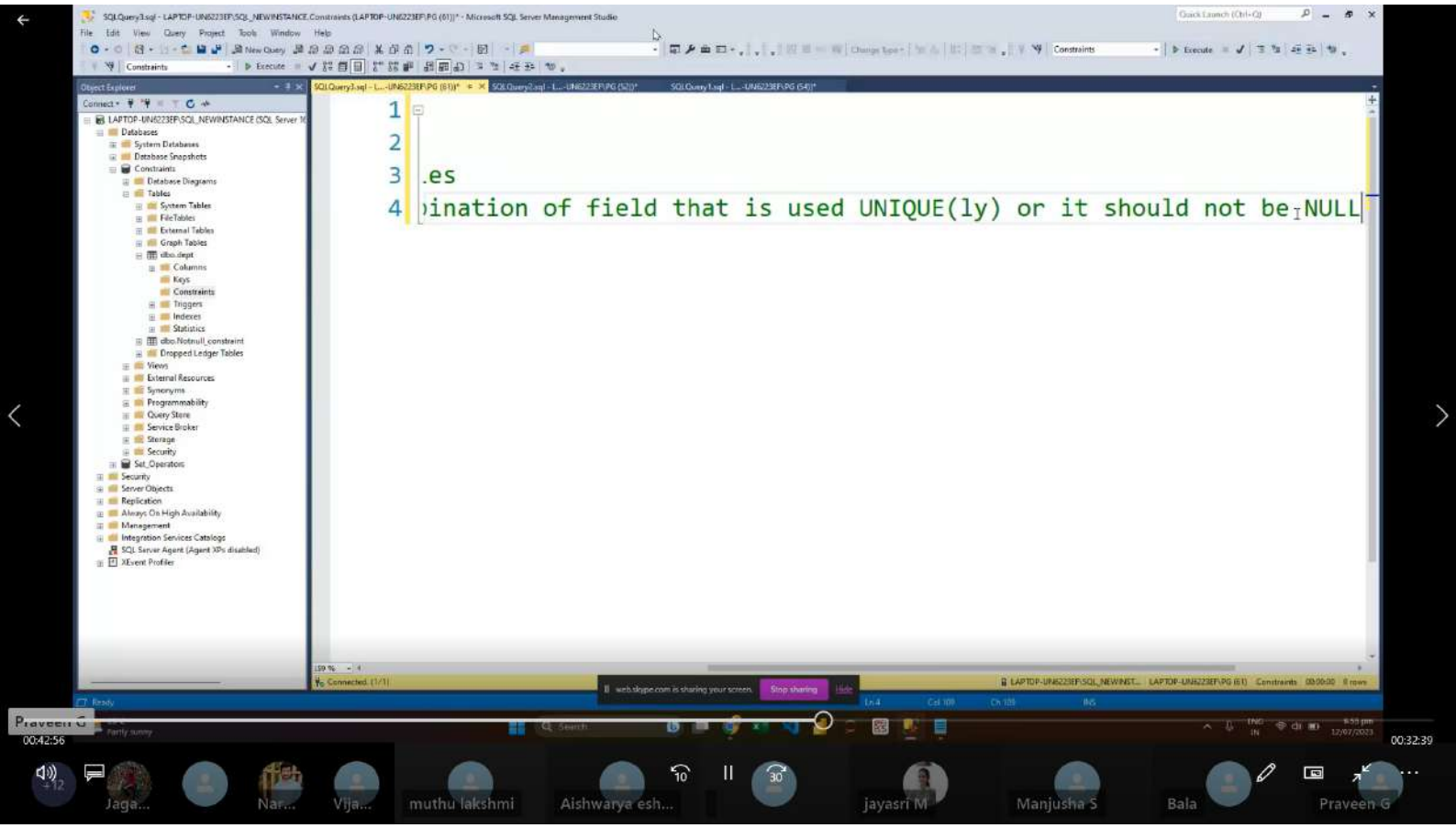


Praveen G
00:36:12
00:39:23
Jaga...
Nar...
Vija...
muthu lakshmi
Aishwarya esh...
jayasri M
Manjusha S
Bala
Praveen G

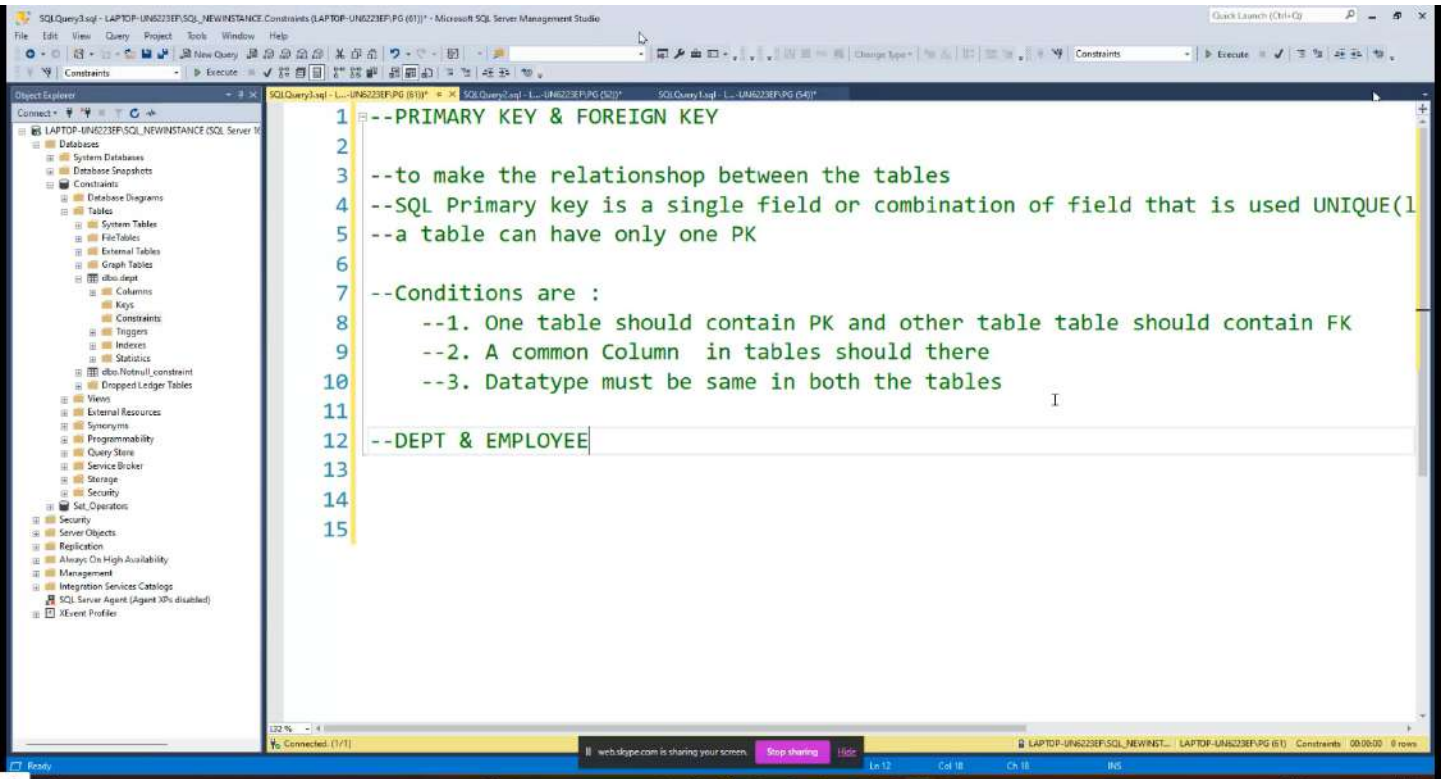


Praveen G
00:42:29
00:33:06

Jaga... Nar... Vija... muthu lakshmi Aishwarya esh... jayasri M Manjusha S Bala Praveen G

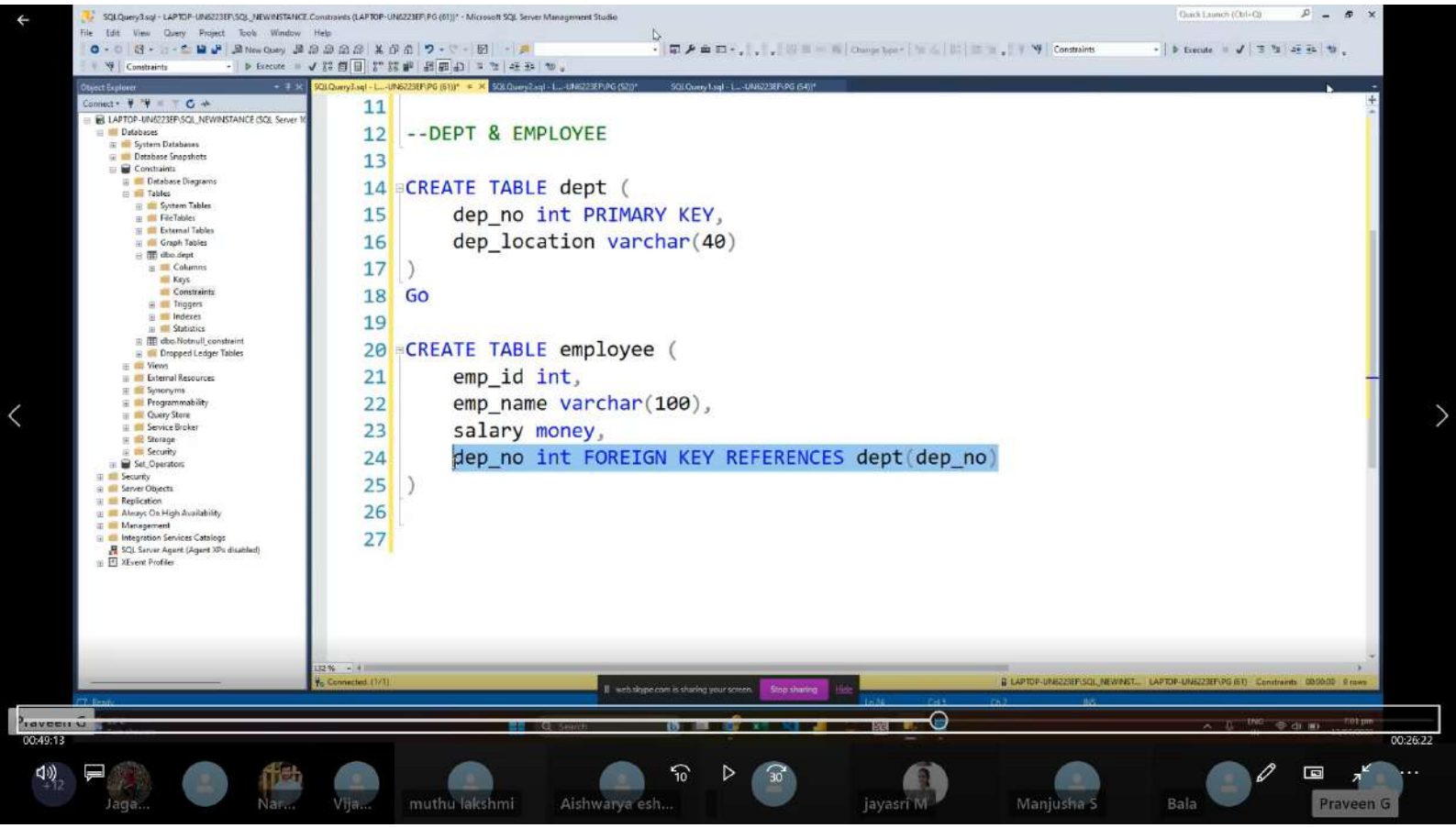


Praveen G
00:42:56
00:32:39
Jaga...
Nar...
Vija...
muthu lakshmi
Aishwarya esh...
jayasri M
Manjusha S
Bala
Praveen G

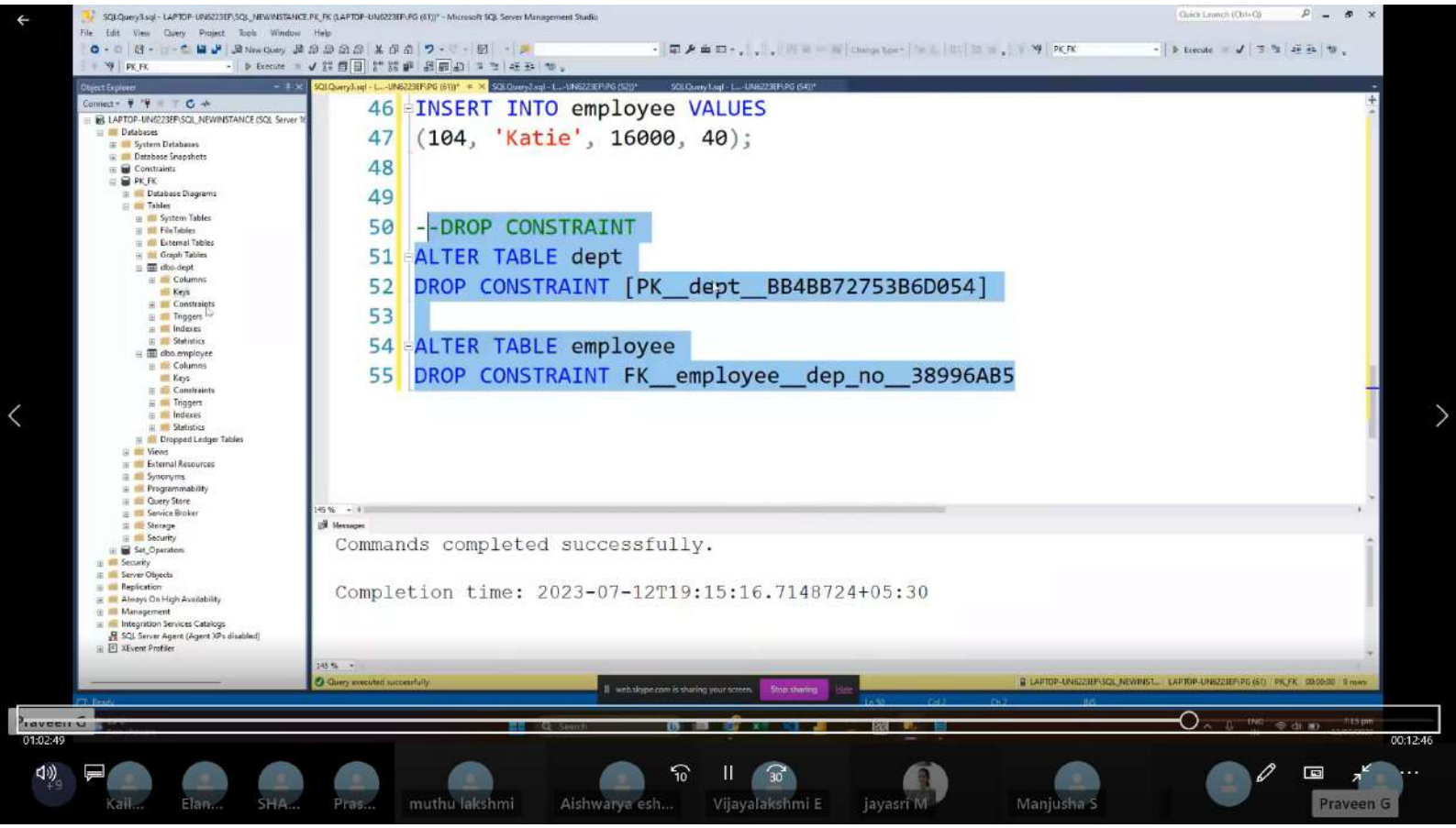


Praveen G

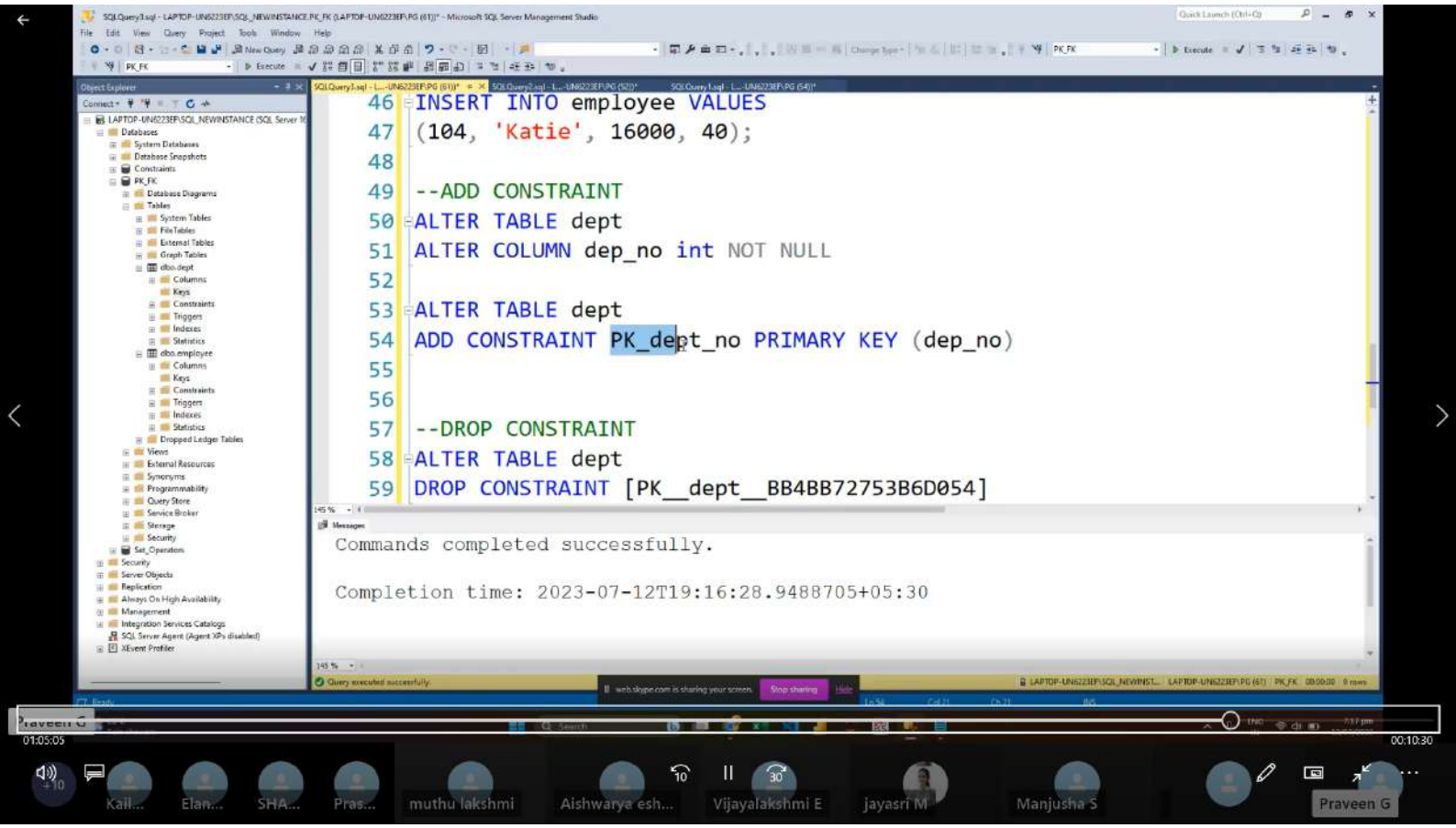
+12 Jaga... Nar... Vija... muthu lakshmi Aishwarya esh... jayasri M Manjusha S Bala Praveen G



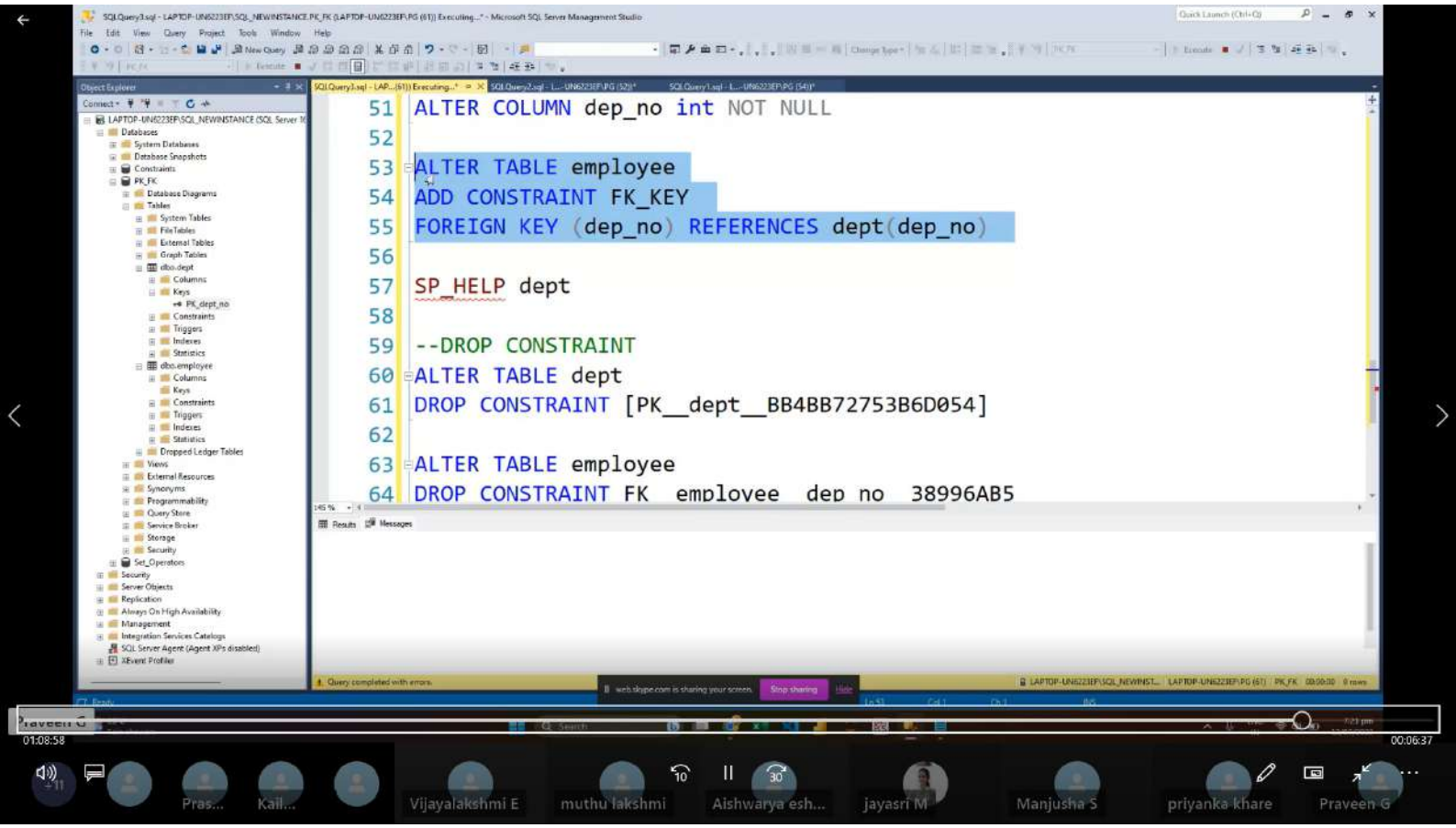
Praveen G
00:49:13
00:26:22
Jaga...
Nar...
Vija...
muthu lakshmi
Aishwarya esh...
jayasri M
Manjusha S
Bala
Praveen G



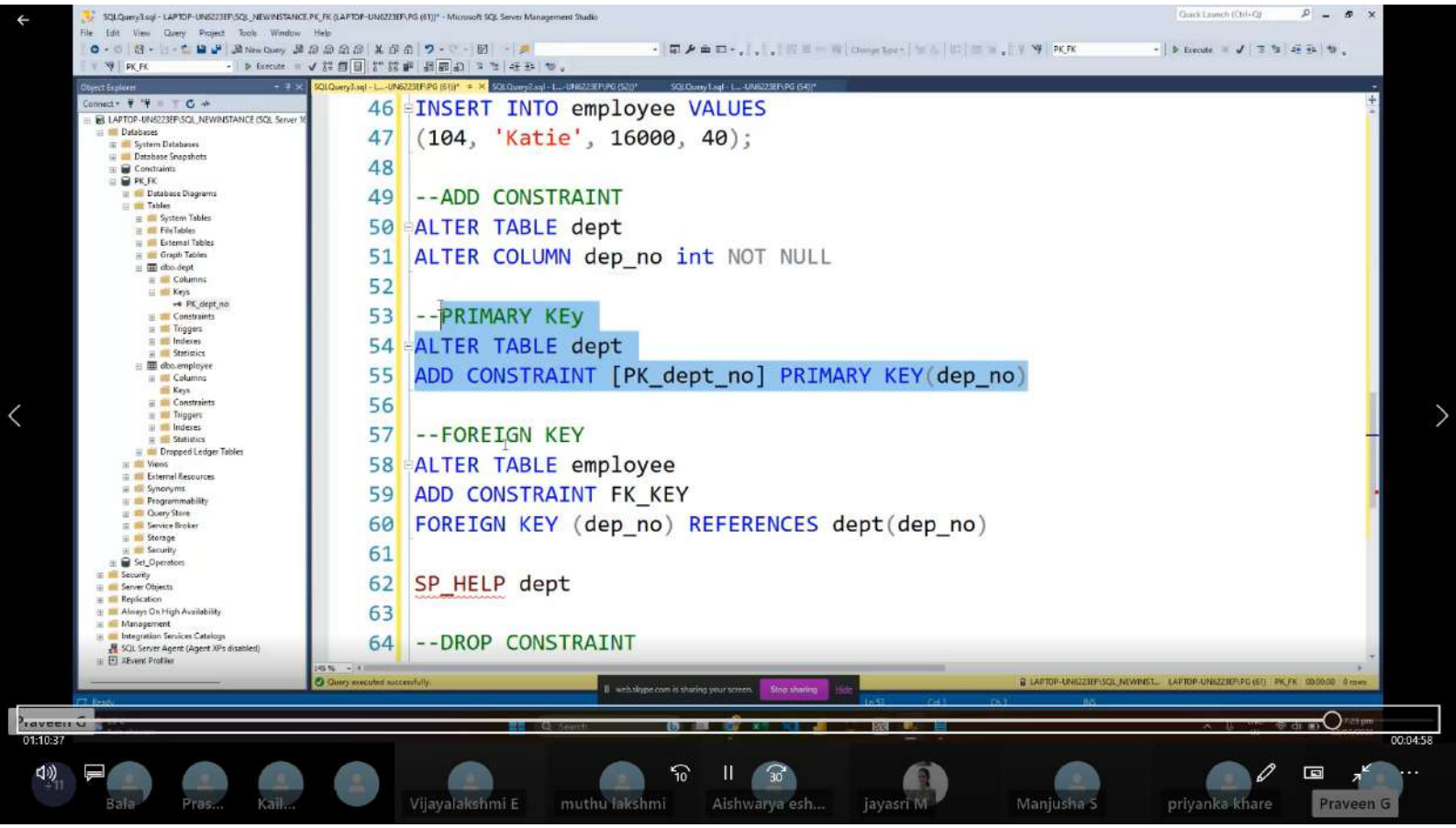
Praveen G
01:02:49
00:12:46
Kali...
Elan...
SHA...
Pras...
muthu lakshmi
Aishwarya esh...
Vijayalakshmi E
jayasri M
Manjusha S
Praveen G



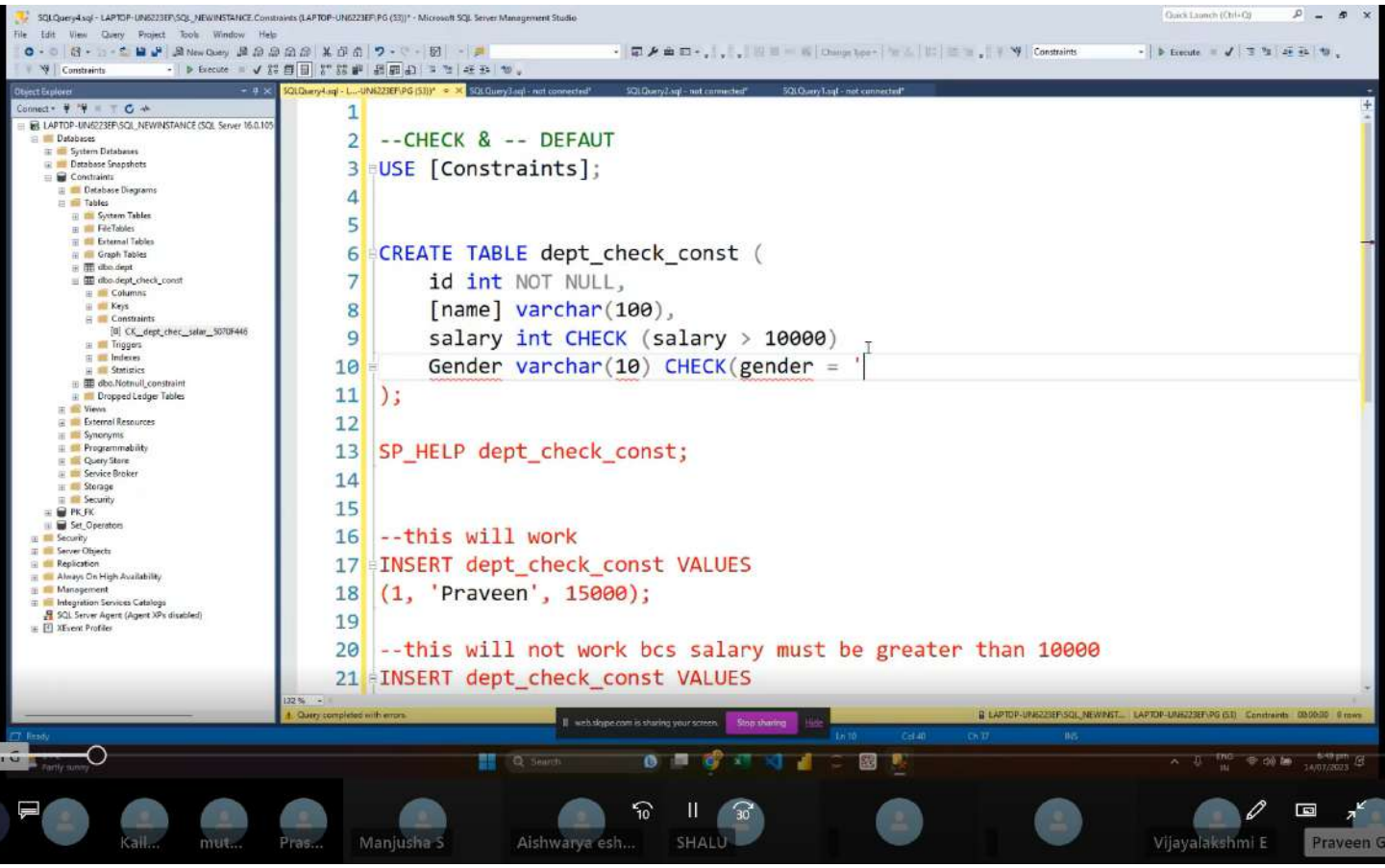
Praveen G
01:05:05
00:10:30
Kali...
Elan...
SHA...
Pras...
muthu lakshmi
Aishwarya esh...
Vijayalakshmi E
jayasri M
Manjusha S
Praveen G



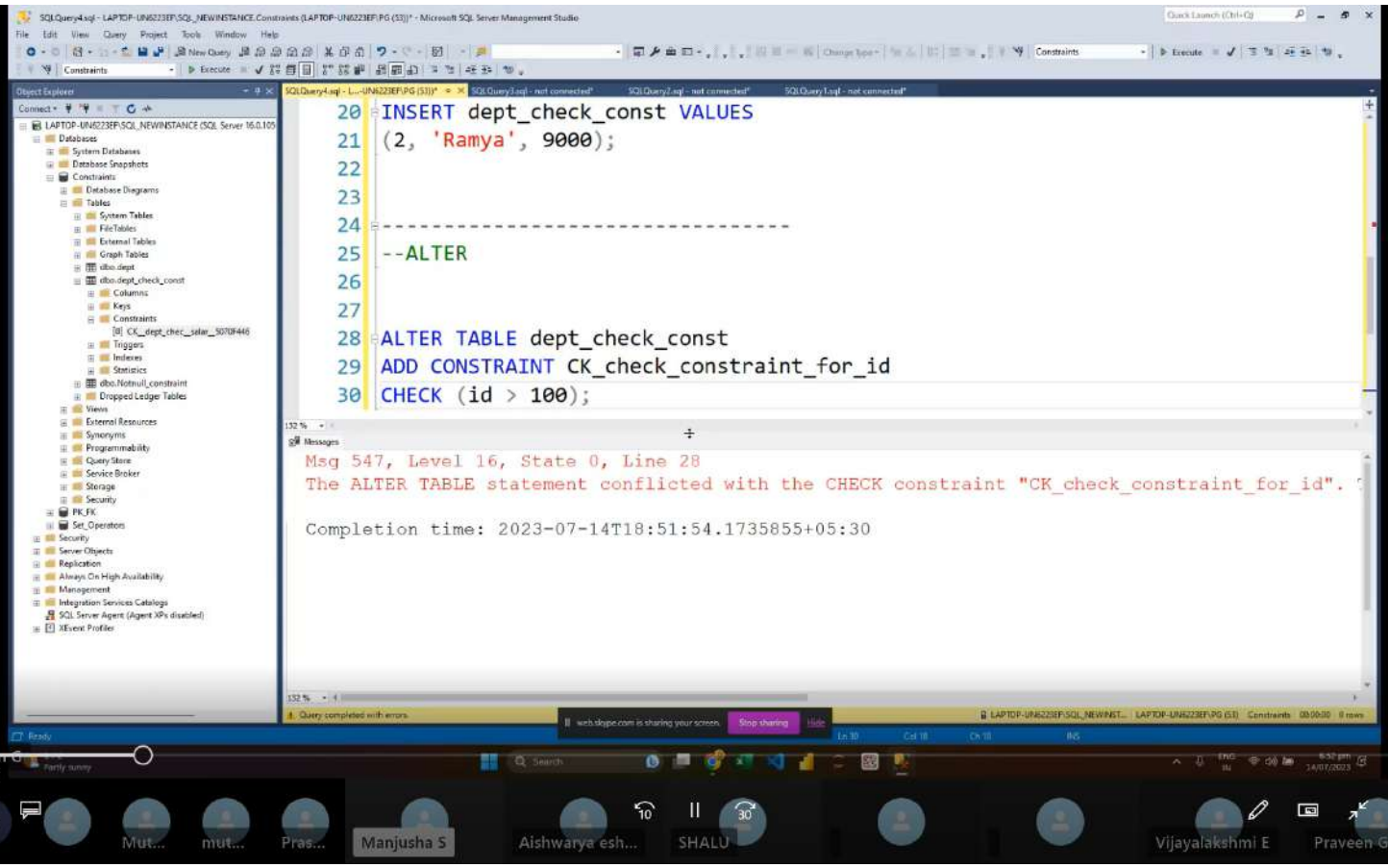
Praveen G
01:08:58
00:06:37
Pras...
Kail...
Vijayalakshmi E
muthu lakshmi
Aishwarya esh...
jayasri M
Manjusha S
priyanka khare
Praveen G



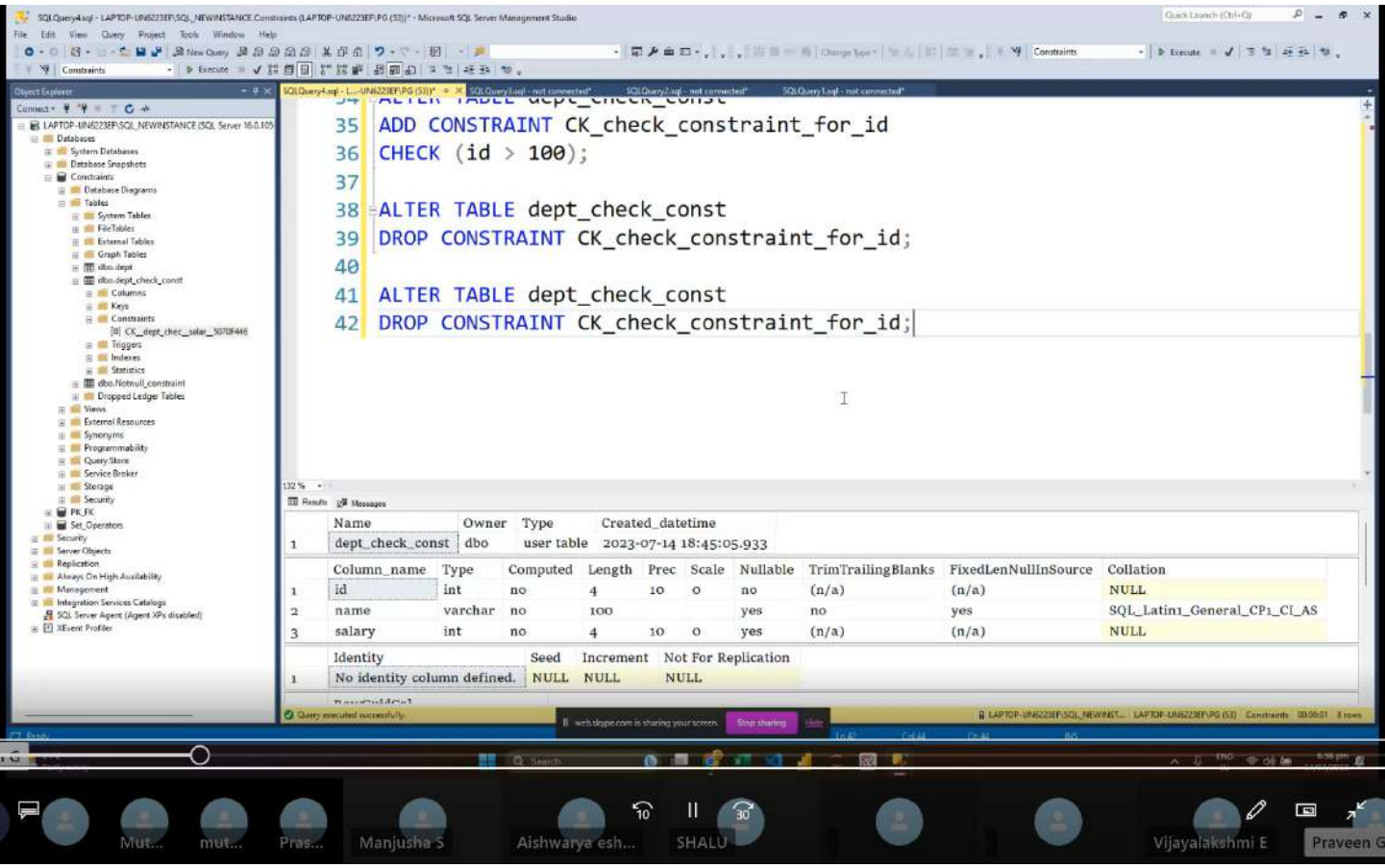
Praveen G
01:10:37
00:04:58
Bala
Pras...
Kail...
Vijayalakshmi E
muthu lakshmi
Aishwarya esh...
jayasri M
Manjusha S
priyanka khare
Praveen G



Praveen G
00:09:01
01:30:23
Kail...
mut...
Pras...
Manjusha S
Aishwarya esh...
SHALU
Vijayalakshmi E
Praveen G

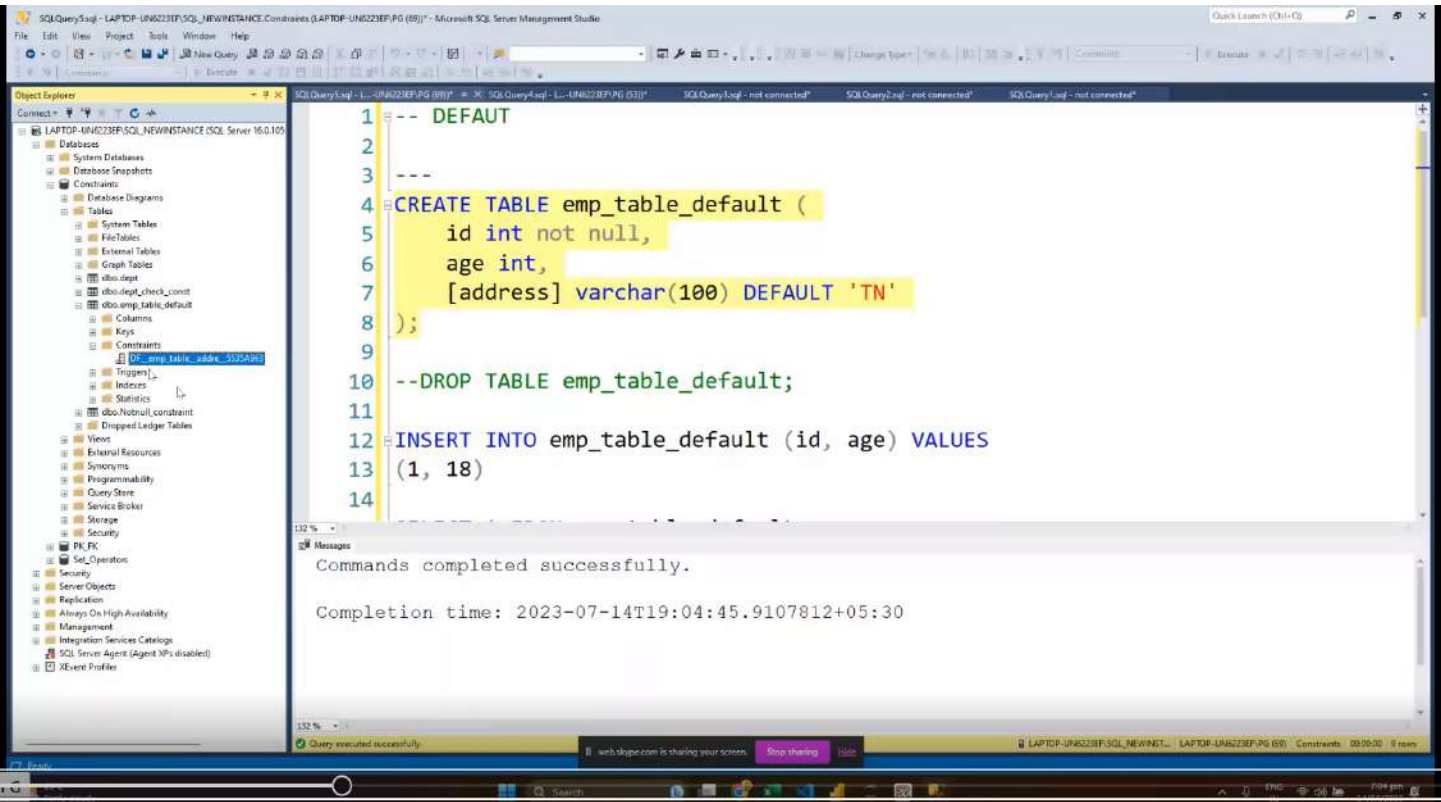


Praveen G
00:12:05
01:27:19
Mut...
mut...
Pras...
Manjusha S
Aishwarya esh...
SHALU
Vijayalakshmi E
Praveen G

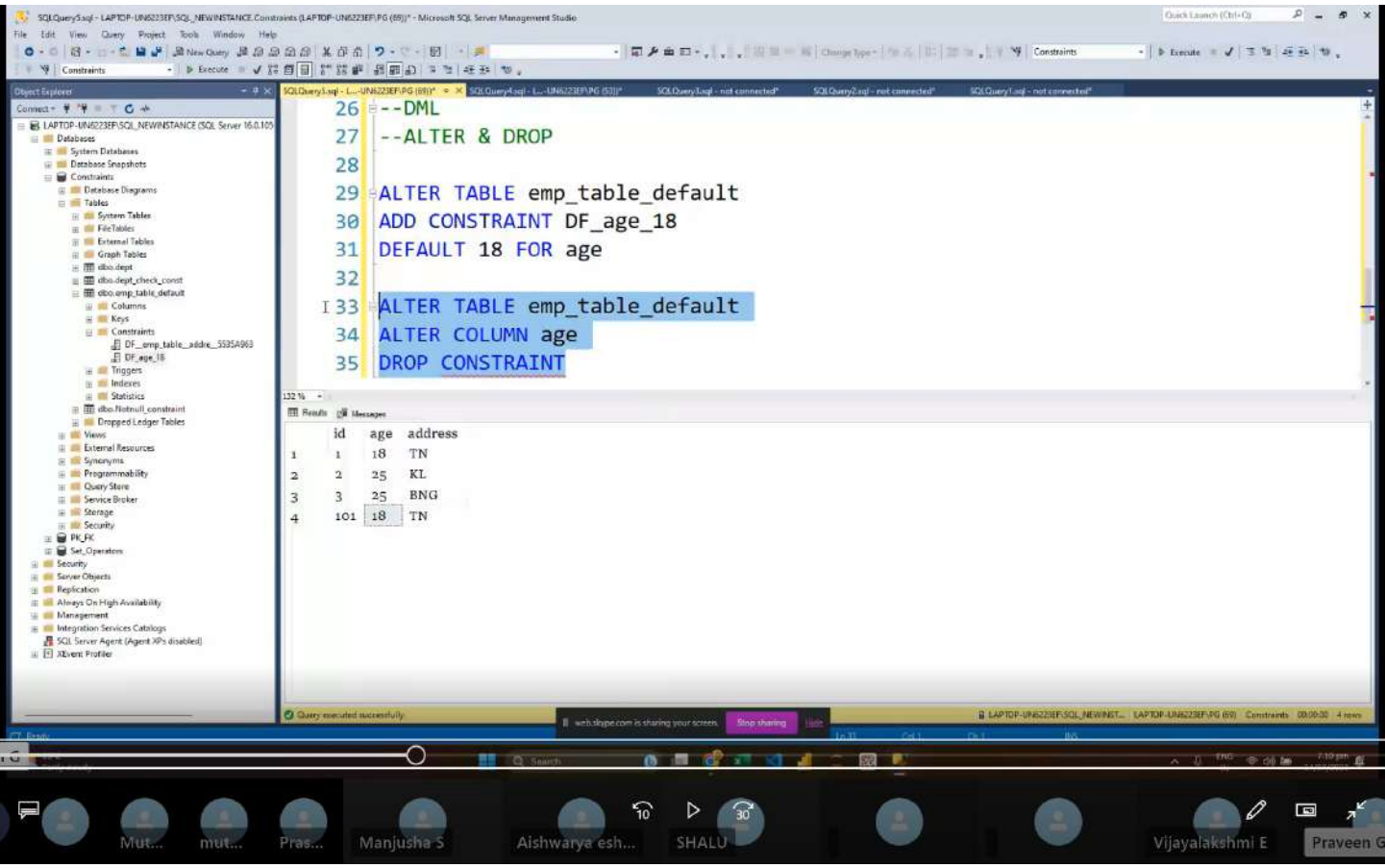


00:16:06 01:23:18

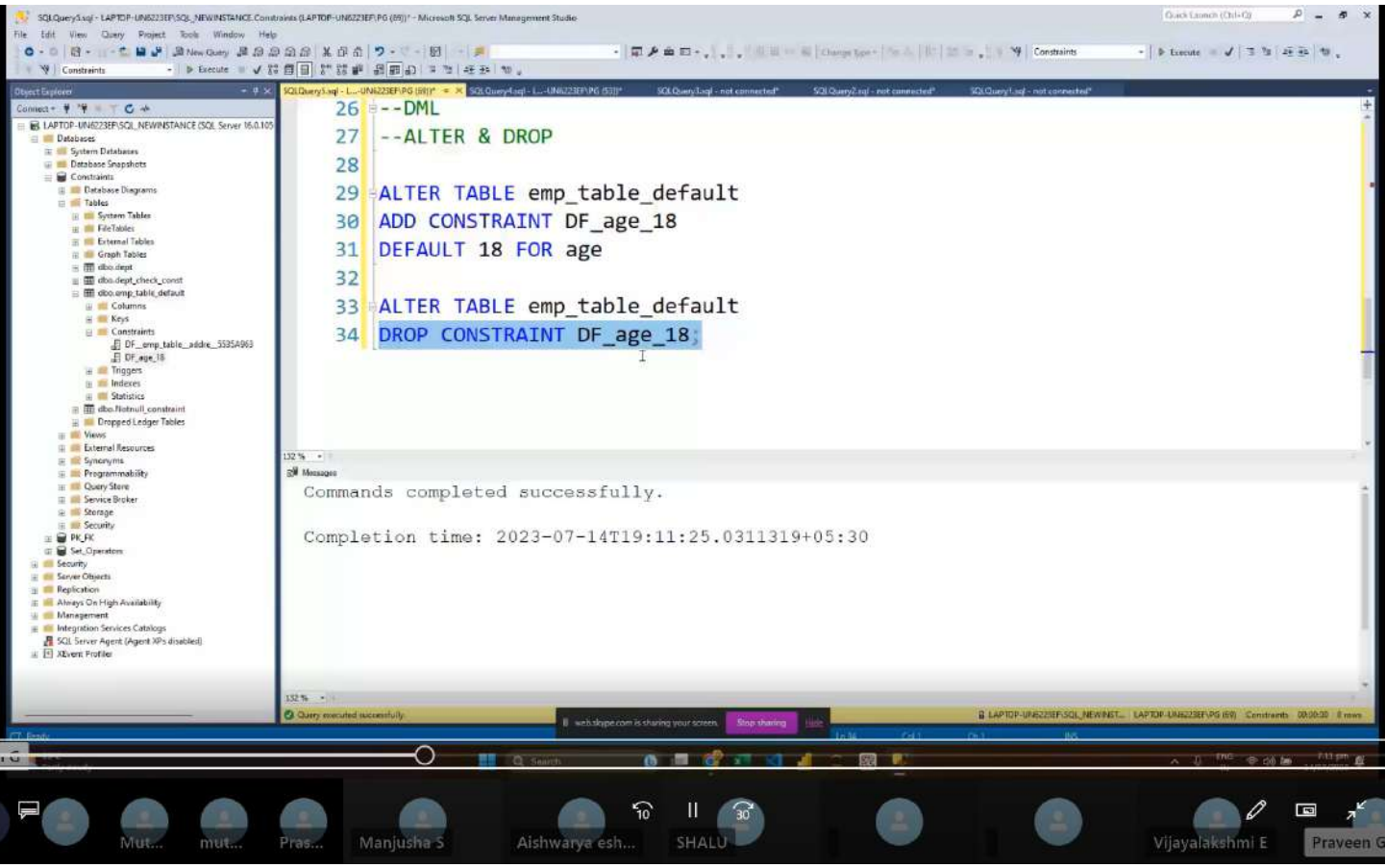
Mut... mut... Pras... Manjusha S Aishwarya esh... SHALU Vijayalakshmi E Praveen G



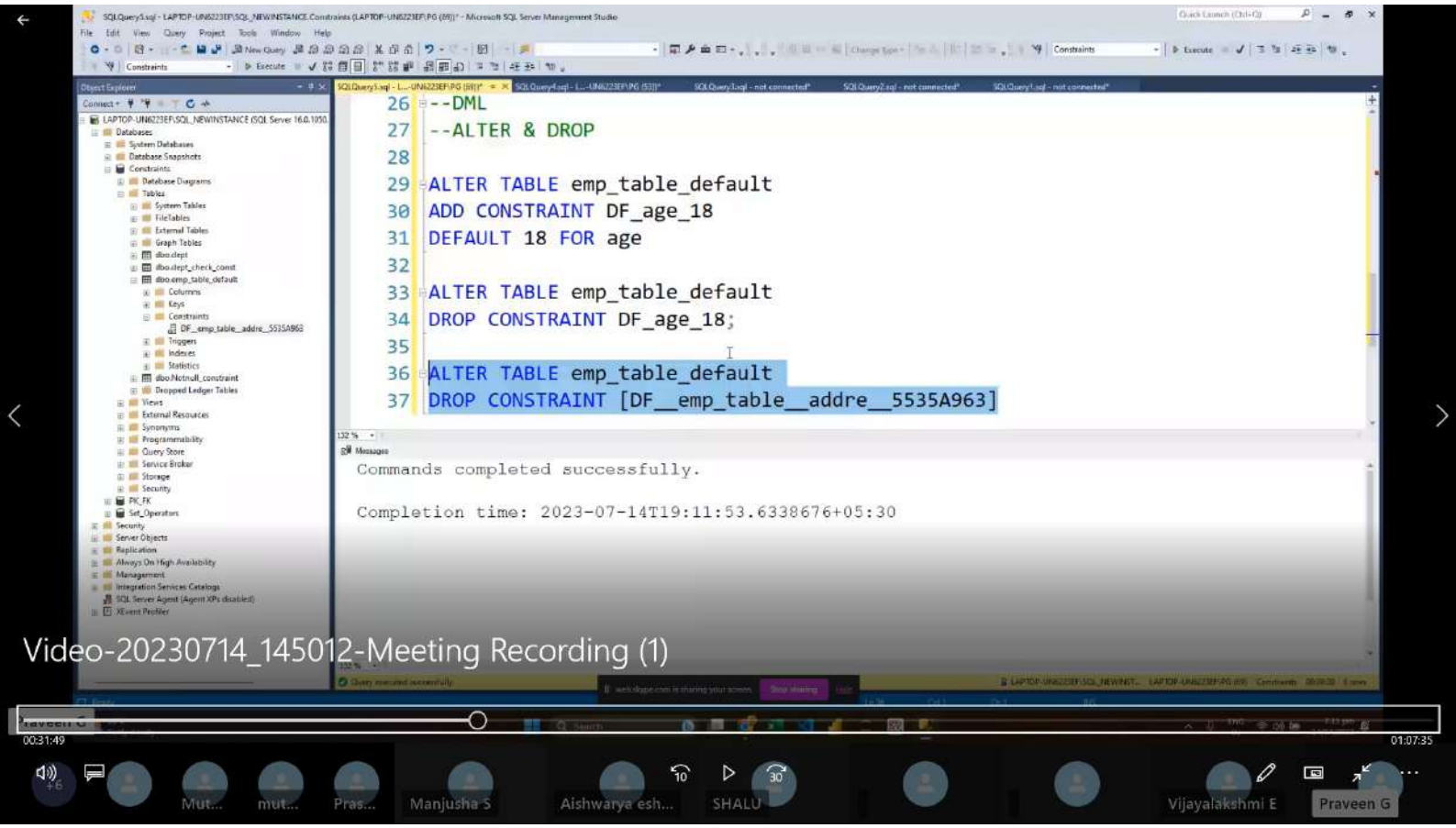
Praveen G
00:24:52
01:14:32
Mut...
mut...
Pras...
Manjusha S
Aishwarya esh...
SHALU
Vijayalakshmi E
Praveen G



00:30:49
01:08:35
Mut... mut... Pras... Manjusha S Aishwarya esh... SHALU Vijayalakshmi E Praveen G



00:31:26
01:07:58
Mut...
mut...
Pras...
Manjusha S
Aishwarya esh...
SHALU
Vijayalakshmi E
Praveen G



Video-20230714_145012-Meeting Recording (1)

00:31:49 01:07:35

Mut... mut... Pras... Manjusha S Aishwarya esh... SHALU Vijayalakshmi E Praveen G

```
----RANKING FUNCTION

> to assign the ranks to each Row wise or Group of row wise
      -- Row_Number()
      -- Rank()
      -- Dense_rank()


ORDER BY
PARTITION ORDER BY


SYNTAX :

      OVER        - pre defined function (it is a mandatory - ORDER BY is
also mandatory)
      Group of row wise - PARTITION   - OPTIONAL CLAUSE
      Row wise      - ORDER BY    - mandatory

      RANKING FUNCTION () OVER ([PARTITION BY <Col_name>] ORDER BY
<col_name> [ASC/DESC])

-----------------------------------------------------------------------
CREATE DATABASE rankFunctions;

CREATE TABLE emp (
      e_id int,
      e_name varchar(100),
      e_salary money,
      e_dept_no int
);

--DQL
SELECT * FROM emp

----  ORDER BY
---ROW_NUMBER()  - it will not skip the numbers
SELECT e_name, e_salary, ROW_NUMBER()
OVER (ORDER BY e_salary DESC)
FROM emp


---Rank()  - will skip a number
SELECT e_name, e_salary, Rank()
OVER (ORDER BY e_salary DESC)
FROM emp

---Dense_rank() - it will not skip a number and it mantains a order
SELECT e_name, e_salary, Dense_rank()
OVER (ORDER BY e_salary DESC)
FROM emp

---- PARTITION BY ORDER BY
---ROW_NUMBER()
```

```
SELECT e_name, e_dept_no, e_salary, ROW_NUMBER()
OVER (PARTITION BY e_dept_no ORDER BY e_salary)
FROM emp;
```

```
CREATE DATABASE string_functions;
USE string_functions;

CREATE TABLE hr_dept(
      emp_id int,
      first_name varchar(30),
      last_name varchar(40),
      email_id varchar(100),
      [address] varchar(50),
      [state] varchar(50),
      phone varchar(100)
);

INSERT INTO hr_dept VALUES (1, 'RAm', 'Kumar', 'Ram.kumar@gmail.com',
'Chennai', 'tn', '999628, 82486')
INSERT INTO hr_dept VALUES (2, 'KiShore', 'Raj', 'KIShoRe.Raj@gmail.com',
'Chennai', 'tn', '999628, 82486')
INSERT INTO hr_dept VALUES (3, 'RaVina', 'Kasana',
'Ravina.kasana@yahoo.com', 'Salem', 'TN', '999628, 82486')
INSERT INTO hr_dept VALUES (4, 'LaVnya', 'Ashok',
'Lavnya.ASHOK@gmail.com', 'Tanjur', 'TN', '999628, 82486'
INSERT INTO hr_dept VALUES (5, 'PRiya', 'Redhu', 'Priya.redhu@gmail.com',
'ECII', 'Bng', '999628, 82486')
INSERT INTO hr_dept VALUES (6, 'JAYA', 'Lakshmi',
'Jaya.lakshmi@gmail.com', 'Tanisandra', 'bNg', '999628, 82486')
INSERT INTO hr_dept VALUES (7, 'Arjun', 'PRAKesh',
'Arjun.prakesh@gmail.com', 'Chennai', 'tN', '999628, 82486')

SELECT * FROM hr_dept;

--------------------------------------------------------------------------
---------------------------------------


-- ASCII - American Standard Code for Information Interchange

---ASCII
SELECT ASCII('a') -- 97
SELECT ASCII('A') -- 65
SELECT ASCII('2') -- 50

---CHAR
SELECT CHAR(97) -- a
SELECT CHAR(65) -- A

--CONCAT
SELECT CONCAT('Hello', ' ', 'World', ' ', 'Praveen')
-- + operator
SELECT 'Hello' + 'World'  -- used before 2008

SELECT first_name, Last_name, CONCAT(first_name, ' ',  Last_name) as
FULLNAME
FROM hr_dept
```

```
--CONCAT_WS
---CONCAT_WS(seperator, exp1, exp2, exp3......)
SELECT CONCAT_WS('-', 'Hello', 'World', 'how', 'are', 'you')

SELECT *, concat_ws('-', first_name, last_name ) FROM hr_dept

--LEFT
---LEFT(string, no_of_character)

SELECT LEFT('Hello World', 7)
SELECT LEFT(First_name, 3) FROM hr_dept

--LEN()
---LEN(string)
---- The LEN() returns the number of character of an string, excluding
the TRAINING BLANKS

SELECT len('Hello World')                                        -- 11
SELECT len('Hello World hello are you    ')            -- 25

SELECT len(email_id) from hr_dept


-- LOWER()
SELECT LOWER('HELLO')
SELECT LOWER('HElLo')

SELECT LOWER(first_name) as firstName, LOWER(last_name) as lastname,
LOWER(CONCAT(first_name, ' ' ,last_name)) as fullname FROM hr_dept


---LTRIM()
---- is going to remove all the LEADING SPACES
SELECT LTRIM('        Hello')


---REPLACE
---- REPLACE(string, substring, new_string)

SELECT REPLACE('HOW ARE UOU', 'UOU', 'YOU')


--REPLICATE
--repeats a string N(users input) number of times

SELECT REPLICATE('Hello', 3)


--REVERSE
SELECT REVERSE('HELLO')


--RIGHT()
---RIGHT(string, no_of_chart.)
```

```
SELECT RIGHT('SQL SERVER', 6)


--RTRIM()
---- is going to remove all the TRAILING SPACES

SELECT RTRIM('HELLO WORLD                       ')

--SPACE()
---Space function returns a string of repeated spaces
--- if COUNT is negative int, the output will be NULL
SELECT 'Praveen' + space(4) + 'Kumar'

SELECT first_name + SPACE(1) + last_name FROM hr_dept


--STRING_SPLIT()
---STRING_SPLIT(string, seperator)

SELECT
      VALUE
FROM
      string_split('hello- world- earth', '-')

-----

SELECT
      *, VALUE
FROM
      hr_dept
      CROSS APPLY string_split(phone, ',')


--SUBSTRING
--- SUBSTRING(input_string, start, lenth)

SELECT SUBSTRING('SQL SERVER SUBSTRING', 5, 10)


--TRIM()
----TRIM([removes_char FROM] string)
SELECT TRIM('        Hello  World Earth              ')
SELECT TRIM('_$#' FROM '___#____HELLO$$$___$$$$')

--UPPER()

SELECT * FROM hr_dept
SELECT UPPER('sql server')

UPDATE hr_dept
SET first_name = CONCAT(UPPER(SUBSTRING(first_name, 1, 1)),
LOWER(SUBSTRING(first_name, 2, LEN(first_name))))
```

```
UPDATE hr_dept
SET state = UPPER(state)

UPDATE hr_dept
SET email_id = lower(email_id)

SELECT CONCAT(UPPER(SUBSTRING(first_name, 1, 1)),
LOWER(SUBSTRING(first_name, 2, LEN(first_name)))) FROM hr_dept

SELECT SUBSTRING(first_name, 2, LEN(first_name)) FROM hr_dept

--CHARINDEX()
--- CHARINDEX(substring, string[, start location])

SELECT CHARINDEX('HELLO', 'World hello earth hello')       -- 7
SELECT CHARINDEX('HELLO', 'World hello earth hello', 10)  -- 19
```

```
SQL Server SYSDATETIME, SYSDATETIMEOFFSET and SYSUTCDATETIME Functions
SQL Server High Precision Date and Time Functions have a scale of 7 and
are:

SYSDATETIME – returns the date and time of the machine the SQL Server is
running on
SYSDATETIMEOFFSET – returns the date and time of the machine the SQL
Server is running on plus the offset from UTC
SYSUTCDATETIME - returns the date and time of the machine the SQL Server
is running on as UTC
*/

-- higher precision functions
SELECT SYSDATETIME()       AS 'DateAndTime';        -- return
datetime2(7)
SELECT SYSDATETIMEOFFSET() AS 'DateAndTime+Offset'; -- datetimeoffset(7)
SELECT SYSUTCDATETIME()    AS 'DateAndTimeInUtc';   -- returns
datetime2(7)

-------------------------------------------------------------------------
--------------------------------------------------

/*
SQL Server CURRENT_TIMESTAMP, GETDATE() and GETUTCDATE() Functions
SQL Server Lesser Precision Data and Time Functions have a scale of 3 and
are:

CURRENT_TIMESTAMP - returns the date and time of the machine the SQL
Server is running on
GETDATE() - returns the date and time of the machine the SQL Server is
running on
GETUTCDATE() - returns the date and time of the machine the SQL Server is
running on as UTC
*/

-- lesser precision functions - returns datetime
SELECT CURRENT_TIMESTAMP AS 'DateAndTime'; -- note: no parentheses
SELECT GETDATE()         AS 'DateAndTime';
SELECT GETUTCDATE()      AS 'DateAndTimeUtc';

-------------------------------------------------------------------------
--------------------------------------------------

/*
SQL Server DATENAME Function
*/

-- date and time parts - returns nvarchar
SELECT DATENAME(YEAR, '2023-12-12')     AS 'Year';
SELECT DATENAME(QUARTER, GETDATE())     AS 'Quarter';
SELECT DATENAME(MONTH, GETDATE())       AS 'Month Name';
SELECT DATENAME(DAYOFYEAR, GETDATE())   AS 'DayOfYear';
SELECT DATENAME(DAY, GETDATE())         AS 'Day';
SELECT DATENAME(WEEK, GETDATE())        AS 'Week';
```

```
SELECT DATENAME(WEEKDAY, GETDATE())     AS 'Day of the Week';
SELECT DATENAME(HOUR, GETDATE())        AS 'Hour';
SELECT DATENAME(MINUTE, GETDATE())      AS 'Minute';
SELECT DATENAME(SECOND, GETDATE())      AS 'Second';
SELECT DATENAME(MILLISECOND, GETDATE()) AS 'MilliSecond';
SELECT DATENAME(MICROSECOND, GETDATE()) AS 'MicroSecond';
SELECT DATENAME(NANOSECOND, GETDATE())  AS 'NanoSecond';
SELECT DATENAME(ISO_WEEK, GETDATE())    AS 'Week';



-------------------------------------------------------------------------
-----------------------------------------------------

/*
SQL Server DATEPART Function
*/

-- date and time parts - returns int
SELECT DATEPART(YEAR, GETDATE())        AS 'Year';
SELECT DATEPART(QUARTER, GETDATE())     AS 'Quarter';
SELECT DATEPART(MONTH, GETDATE())       AS 'Month';
SELECT DATEPART(DAYOFYEAR, GETDATE())   AS 'DayOfYear';
SELECT DATEPART(DAY, GETDATE())         AS 'Day';
SELECT DATEPART(WEEK, GETDATE())        AS 'Week';
SELECT DATEPART(WEEKDAY, GETDATE())     AS 'WeekDay';
SELECT DATEPART(HOUR, GETDATE())        AS 'Hour';
SELECT DATEPART(MINUTE, GETDATE())      AS 'Minute';
SELECT DATEPART(SECOND, GETDATE())      AS 'Second';
SELECT DATEPART(MILLISECOND, GETDATE()) AS 'MilliSecond';
SELECT DATEPART(MICROSECOND, GETDATE()) AS 'MicroSecond';
SELECT DATEPART(NANOSECOND, GETDATE())  AS 'NanoSecond';
SELECT DATEPART(ISO_WEEK, GETDATE())    AS 'Week';

-------------------------------------------------------------------------
-----------------------------------------------------

/*
SQL Server DAY, MONTH and YEAR Functions
DAY - returns an integer corresponding to the day specified
MONTH- returns an integer corresponding to the month specified
YEAR- returns an integer corresponding to the year specified
*/

--SQL Server DAY, MONTH and YEAR Functions
SELECT DAY(GETDATE())   AS 'Day';
SELECT MONTH(GETDATE()) AS 'Month';
SELECT YEAR(GETDATE())  AS 'Year';


-------------------------------------------------------------------------
-----------------------------------------------------

/*
```

```
SQL Server DATEFROMPARTS, DATETIME2FROMPARTS, DATETIMEFROMPARTS,
DATETIMEOFFSETFROMPARTS, SMALLDATETIMEFROMPARTS and  TIMEFROMPARTS
Functions
DATEFROMPARTS – returns a date from the date specified
DATETIME2FROMPARTS – returns a datetime2 from part specified
DATETIMEFROMPARTS – returns a datetime from part specified
DATETIMEOFFSETFROMPARTS - returns a datetimeoffset from part specified
SMALLDATETIMEFROMPARTS - returns a smalldatetime from part specified
TIMEFROMPARTS - returns a time from part specified
*/

-- date and time from parts
SELECT DATEFROMPARTS(2024,8,1)                          AS 'Date';
-- returns date
SELECT DATETIME2FROMPARTS(2019,1,1,6,0,0,0,1)           AS 'DateTime2';
-- returns datetime2
SELECT DATETIMEFROMPARTS(2019,1,1,6,0,0,0)              AS 'DateTime';
-- returns datetime
SELECT DATETIMEOFFSETFROMPARTS(2019,1,1,6,0,0,0,0,0,0) AS 'Offset';
-- returns datetimeoffset
SELECT SMALLDATETIMEFROMPARTS(2019,1,1,6,0)             AS
'SmallDateTime'; -- returns smalldatetime
SELECT TIMEFROMPARTS(6,0,0,0,0)                         AS 'Time';
-- returns time


------------------------------------------------------------------------
--------------------------------------------------

/*
SQL Server DATEDIFF and DATEDIFF_BIG Functions

DATEDIFF - returns the number of date or time datepart boundaries crossed
between specified dates as an int
DATEDIFF_BIG - returns the number of date or time datepart boundaries
crossed between specified dates as a bigint
*/

--Date and Time Difference
SELECT DATEDIFF(DAY, 2023-31-01, 2023-01-01)      AS 'DateDif'    --
returns int
SELECT DATEDIFF_BIG(DAY, 2023-31-01, 2023-01-01)  AS 'DateDifBig' --
returns bigint


------------------------------------------------------------------------
--------------------------------------------------

/*
SQL Server DATEADD, EOMONTH, SWITCHOFFSET and TODATETIMEOFFSET Functions

DATEADD - returns datepart with added interval as a datetime
EOMONTH – returns last day of month of offset as type of start_date
SWITCHOFFSET - returns date and time offset and time zone offset
```

```
TODATETIMEOFFSET - returns date and time with time zone offset
*/

-- modify date and time
SELECT DATEADD(MONTH,3,GETDATE())        AS 'DatePlus1';          --
returns data type of the date argument
SELECT EOMONTH(GETDATE(),2)            AS 'LastDayOfNextMonth'; --
returns start_date argument or date
SELECT SWITCHOFFSET(GETDATE(), -6)     AS 'NowMinus6';          --
returns datetimeoffset
SELECT TODATETIMEOFFSET(GETDATE(), -2) AS 'Offset';             --
returns datetimeoffset

-------------------------------------------------------------------------
-----------------------------------------------------

/*
SQL Server ISDATE Function to Validate Date and Time Values

ISDATE – returns int - Returns 1 if a valid datetime type and 0 if not
*/

-- validate date and time - returns int
SELECT ISDATE(GETDATE()) AS 'IsDate';
SELECT ISDATE(NULL) AS 'IsDate';


-- NUMBER/MATH Functions


--ABS() - Abdolute - returns the output as numeric express
SELECT ABS(22)    -- 22
SELECT ABS(-22)   -- 22

--SQRT
SELECT SQRT(16)  --  4
SELECT SQRT(4)   --  2

--PI
SELECT PI()       -- 3.14159265358979

--CEILING()
-- CEILING FUNCTION returns the ouput after rounding of the decimals,
which is the next  highest value of the table
SELECT CEILING(2.1)  -- 3
SELECT CEILING(2.5)  -- 3
SELECT CEILING(2.7)  -- 3

SELECT CEILING(252.7)
SELECT CEILING(252.1)

--FLOOR()
-- this function will give a output after rounding the decimals which is
equal to or less than the expression.
```

```
SELECT FLOOR(2.1)  -- 2
SELECT FLOOR(2.5)  -- 2
SELECT FLOOR(2.7)  -- 2

SELECT FLOOR(252.7)    -- 252
SELECT FLOOR(252.1)    -- 252

--ROUND()

SELECT ROUND(36.2, 0)   -- 36
SELECT ROUND(36.5, 0)   -- 37
SELECT ROUND(36.8, 0)   -- 37
```

DCL - Data Control Language

GRANT

-- It is employeed to grant a privilege to a user
GRANT Command allows specified user to perform specified tasks.

Syntax :

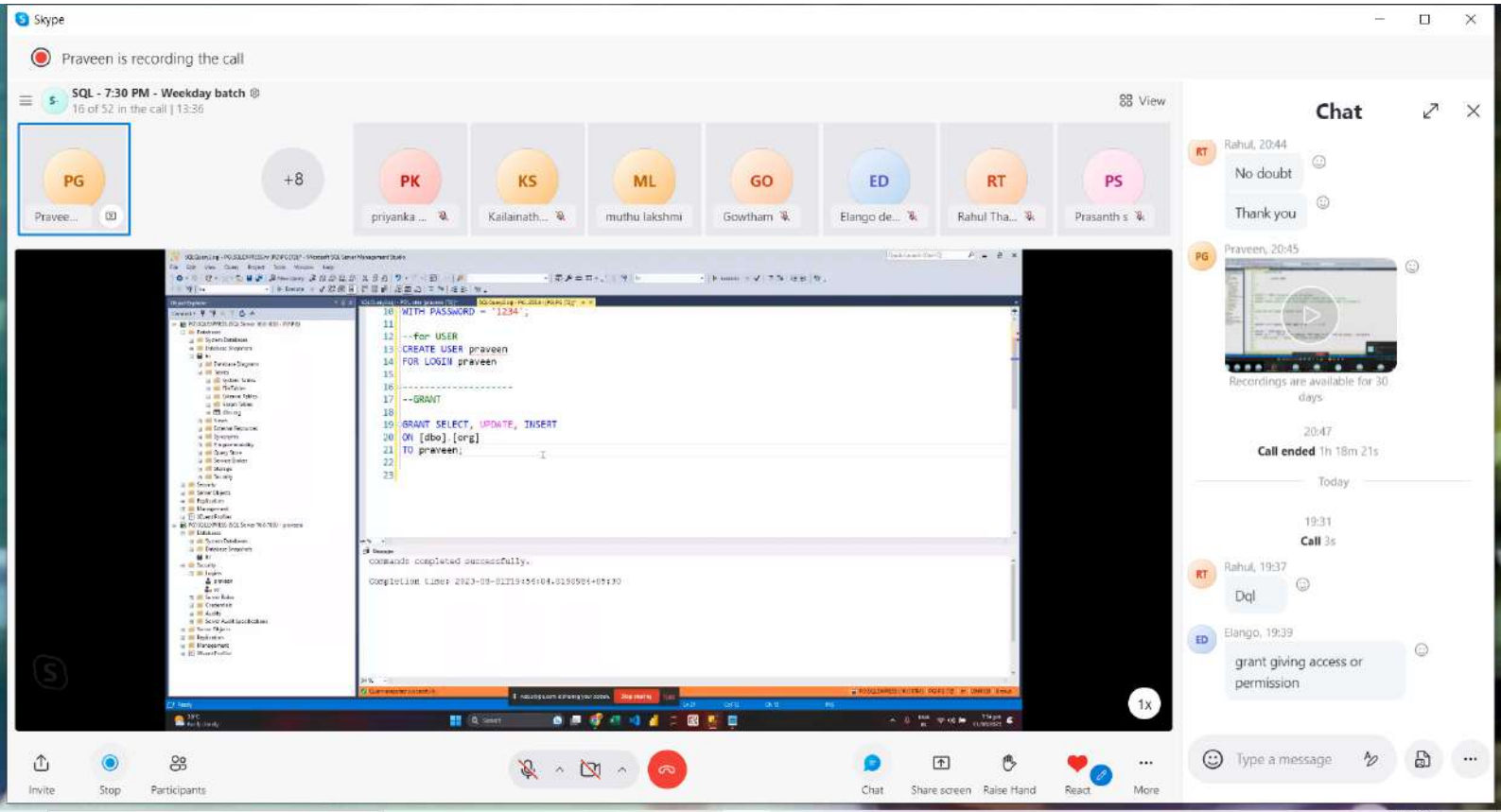
```
GRANT privilege_names
ON object_name
TO user
```

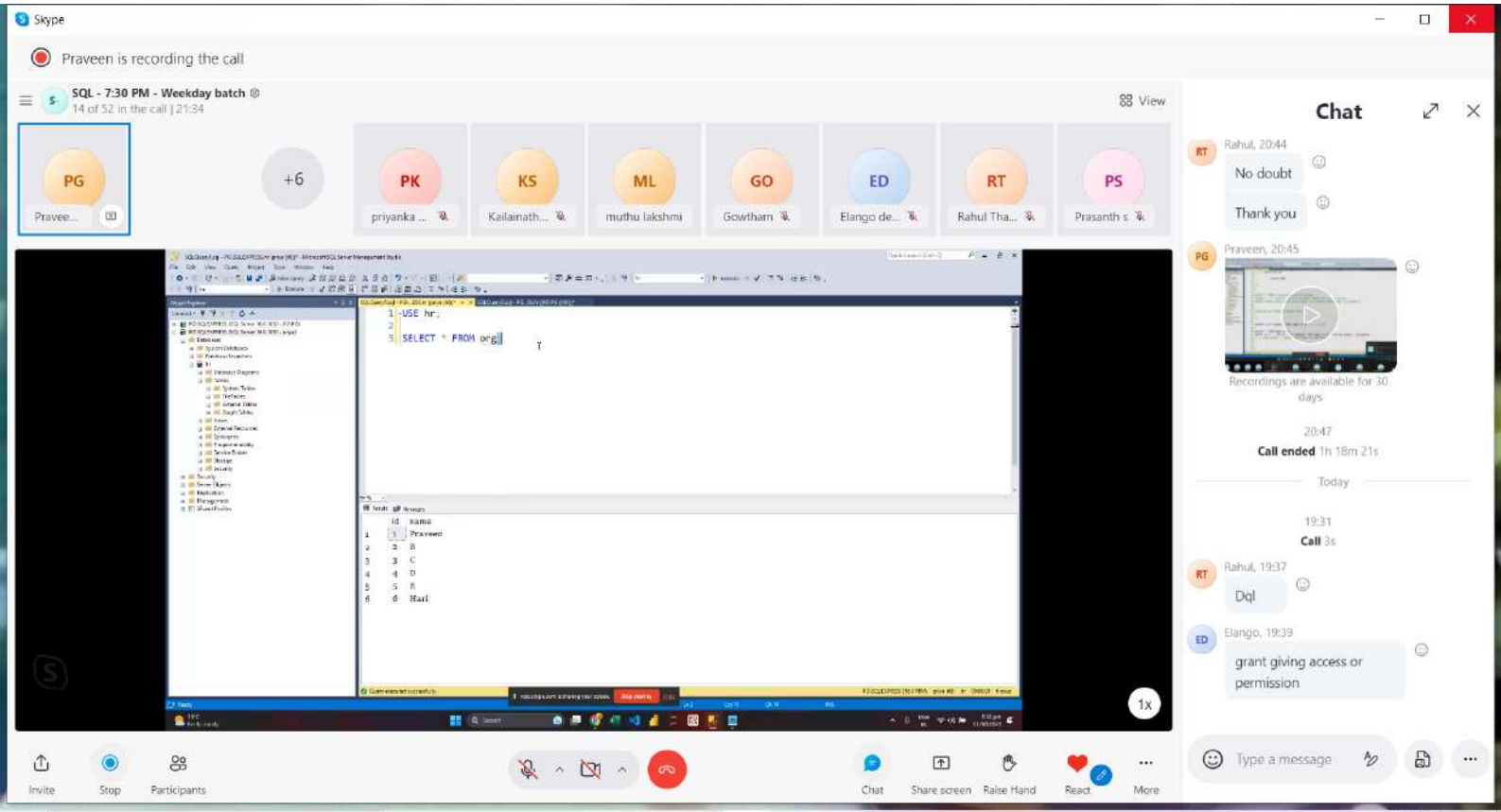
REVOKE

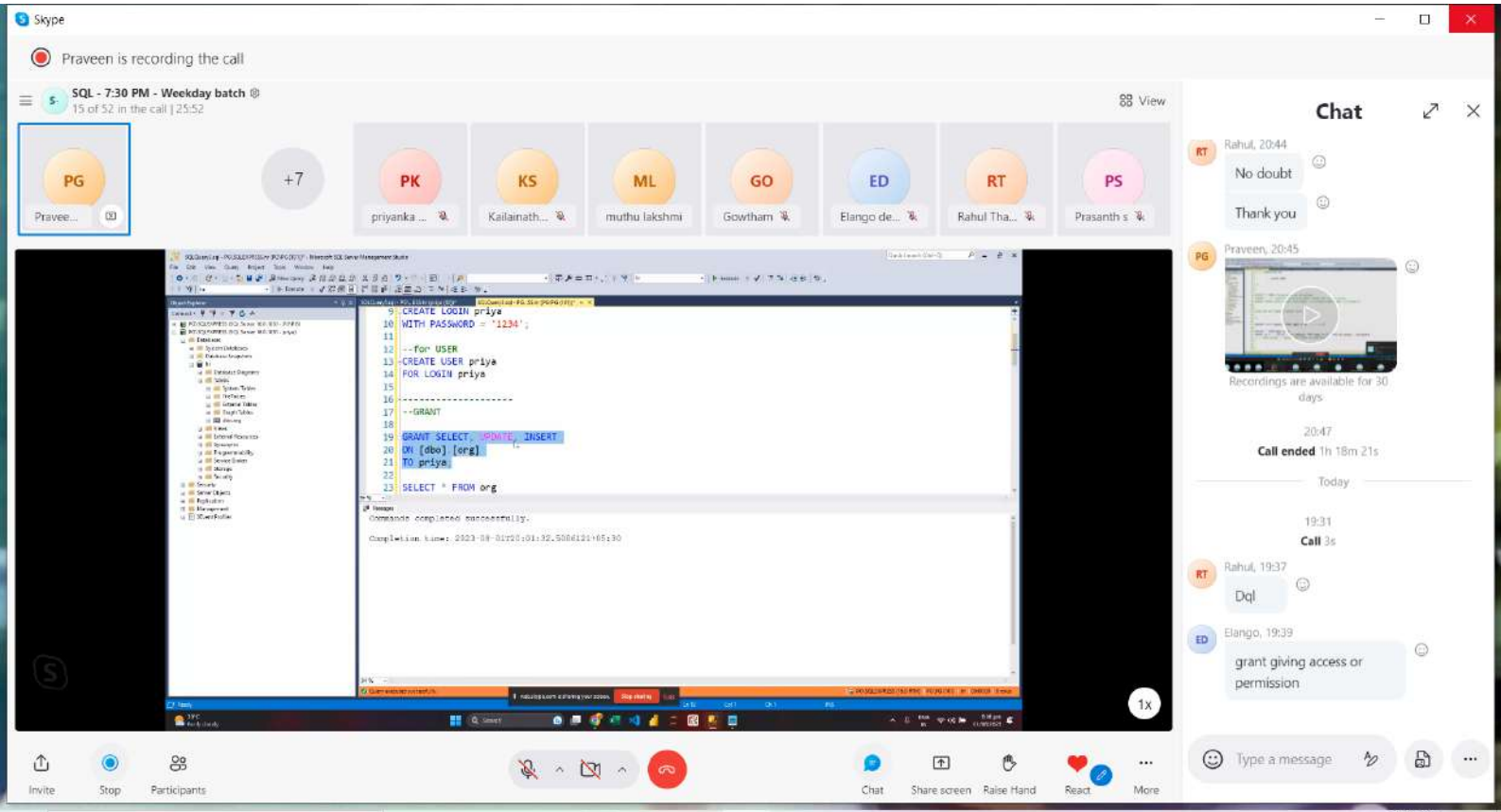
-- It is also employeed to remove a privilege from user.
REVOKE command helps the owner to cancel previously given/granted permissions

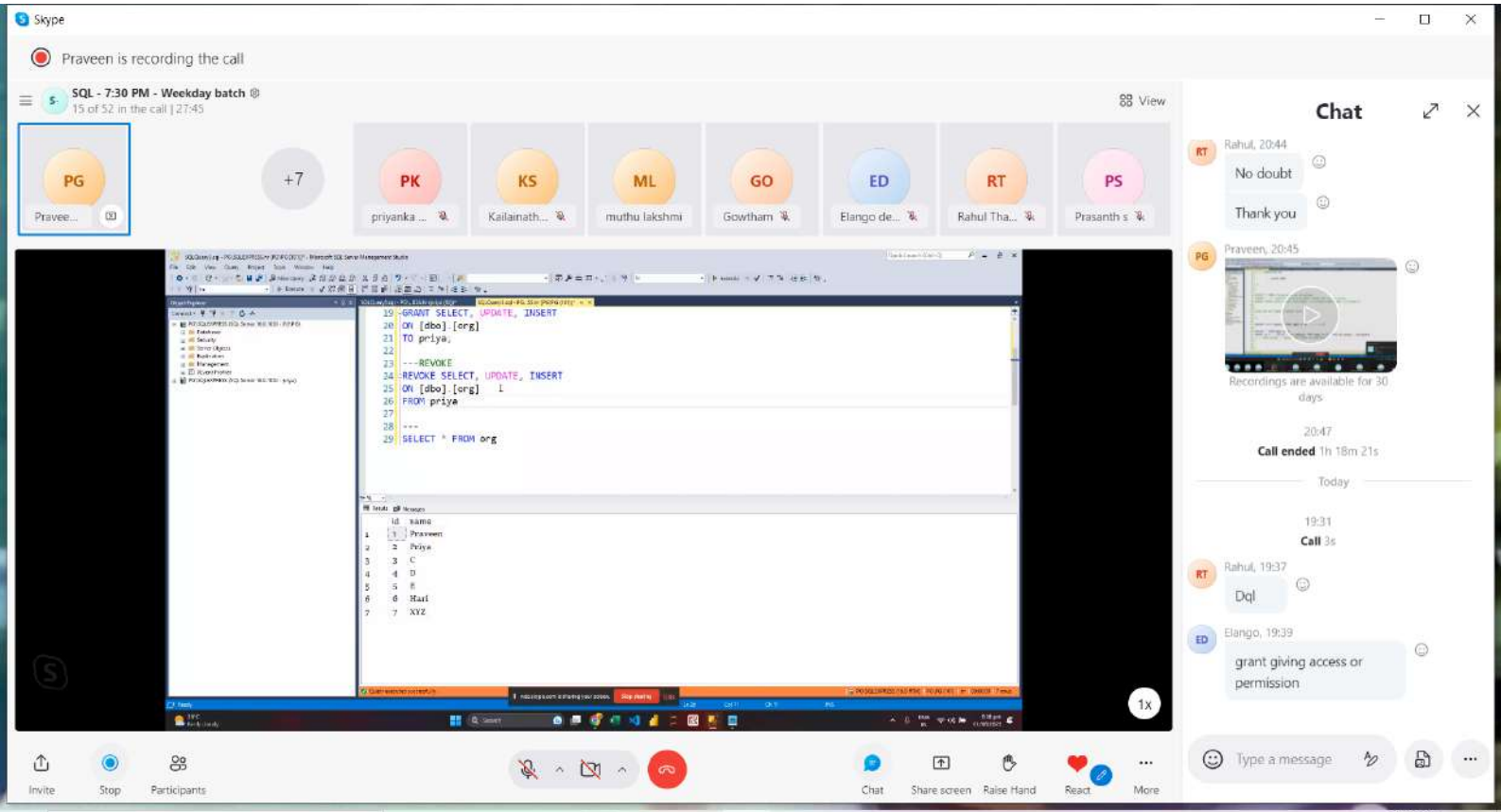
```
REVOKE privilege_names
ON object_name
FROM user
```

--privilege_names:
SELECT, UPDATE, DELETE, INSERT, ALTER, ALL
--Object
TABLE
--USER -- to whom access was granted


Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
16 of 52 in the call | 13:36
View
PG
Pravee...
+8
PK
priyanka ...
KS
Kailainath...
ML
muthu lakshmi
GO
Gowtham
ED
Elango de...
RT
Rahul Tha...
PS
Prasanth s
WITH PASSWORD = '1234';
--for USER
CREATE USER praveen
FOR LOGIN praveen
--GRANT
GRANT SELECT, UPDATE, INSERT
ON [dbo].[org]
TO praveen;
Commands completed successfully.
Completion time: 2023-08-01T19:56:04.0196984+05:30
1x
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Chat
Rahul, 20:44
No doubt
Thank you
Praveen, 20:45
Recordings are available for 30 days
20:47
Call ended 1h 18m 21s
Today
19:31
Call 3s
Rahul, 19:37
Dql
Elango, 19:39
grant giving access or permission
Type a message
Type here to search
ENG
07:54 PM
01-08-2023


Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
14 of 52 in the call | 21:34
View
PG
Pravee...
+6
PK
priyanka ...
KS
Kailainath...
ML
muthu lakshmi
GO
Gowtham
ED
Elango de...
RT
Rahul Tha...
PS
Prasanth s
-USE hr;
SELECT * FROM org;
id name
1 Praveen
2 B
3 C
4 D
5 E
6 Hari
1x
Chat
Rahul, 20:44
No doubt
Thank you
Praveen, 20:45
Recordings are available for 30 days
20:47
Call ended 1h 18m 21s
Today
19:31
Call 3s
Rahul, 19:37
Dql
Elango, 19:39
grant giving access or permission
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:02 PM
01-08-2023


Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
15 of 52 in the call | 25:52
View
PG
Pravee...
+7
PK
priyanka ...
KS
Kailainath...
ML
muthu lakshmi
GO
Gowtham
ED
Elango de...
RT
Rahul Tha...
PS
Prasanth s
CREATE LOGIN priya
WITH PASSWORD = '1234';
--for USER
CREATE USER priya
FOR LOGIN priya
--GRANT
GRANT SELECT, UPDATE, INSERT
ON [dbo] [org]
TO priya
SELECT * FROM org
Commands completed successfully.
Completion time: 2023-08-01T20:01:32.5006121+05:30
1x
Chat
Rahul, 20:44
No doubt
Thank you
Praveen, 20:45
Recordings are available for 30 days
20:47
Call ended 1h 18m 21s
Today
19:31
Call 3s
Rahul, 19:37
Dql
Elango, 19:39
grant giving access or permission
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:06 PM
01-08-2023


Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
15 of 52 in the call | 27:45
View
PG
Pravee...
+7
PK
priyanka ...
KS
Kailainath...
ML
muthu lakshmi
GO
Gowtham
ED
Elango de...
RT
Rahul Tha...
PS
Prasanth s
-GRANT SELECT, UPDATE, INSERT
ON [dbo].[org]
TO priya;
---REVOKE
REVOKE SELECT, UPDATE, INSERT
ON [dbo].[org]
FROM priya
SELECT * FROM org
id name
1 Praveen
2 Priya
3 C
4 D
5 E
6 Hari
7 XYZ
1x
Chat
Rahul, 20:44
No doubt
Thank you
Praveen, 20:45
Recordings are available for 30 days
20:47
Call ended 1h 18m 21s
Today
19:31
Call 3s
Rahul, 19:37
Dql
Elango, 19:39
grant giving access or permission
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:08 PM
01-08-2023

SCHEMA

-- Schema is a logical collection of database objects such as tables, views, stored procedures, indexes etc...
-- It is also known as Container, created by user
-- The database user who creates a schema is schema owners
-- Built-in-Schema ==> Dbo, guests, sys etcs.
-- dho is default schema for the newly created database

CREATE SCHEMA <SCHEMA NAME>

DROP SCHEMA <SCHEMA NAME>

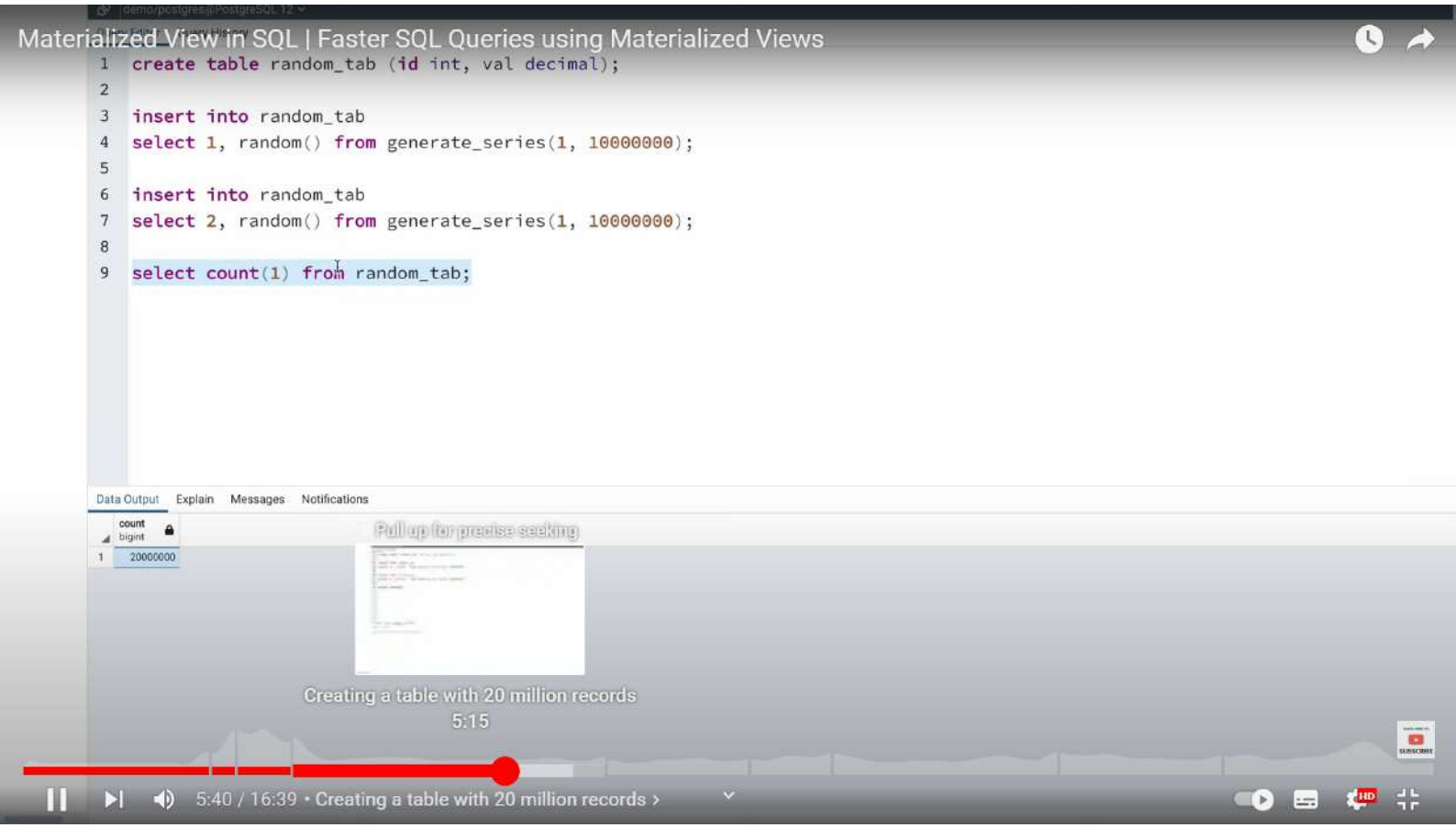
VIEWS in SQL

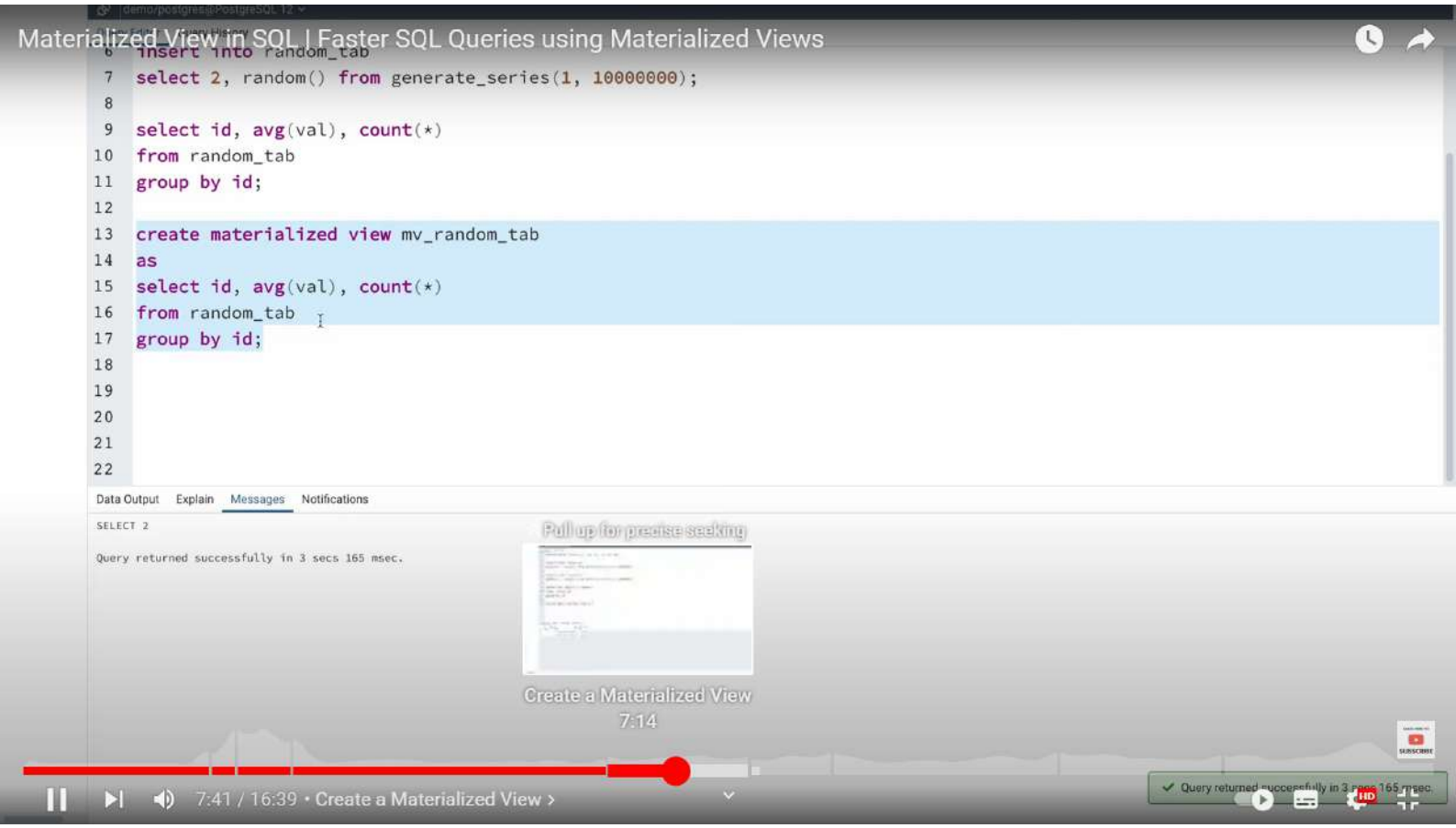
--Creating a view is simply creating a VIRTUAL TABLE using a query.
--A view is a SQL statement that is stored in the database with an associated name
--It will not store data but select query statement is only stored

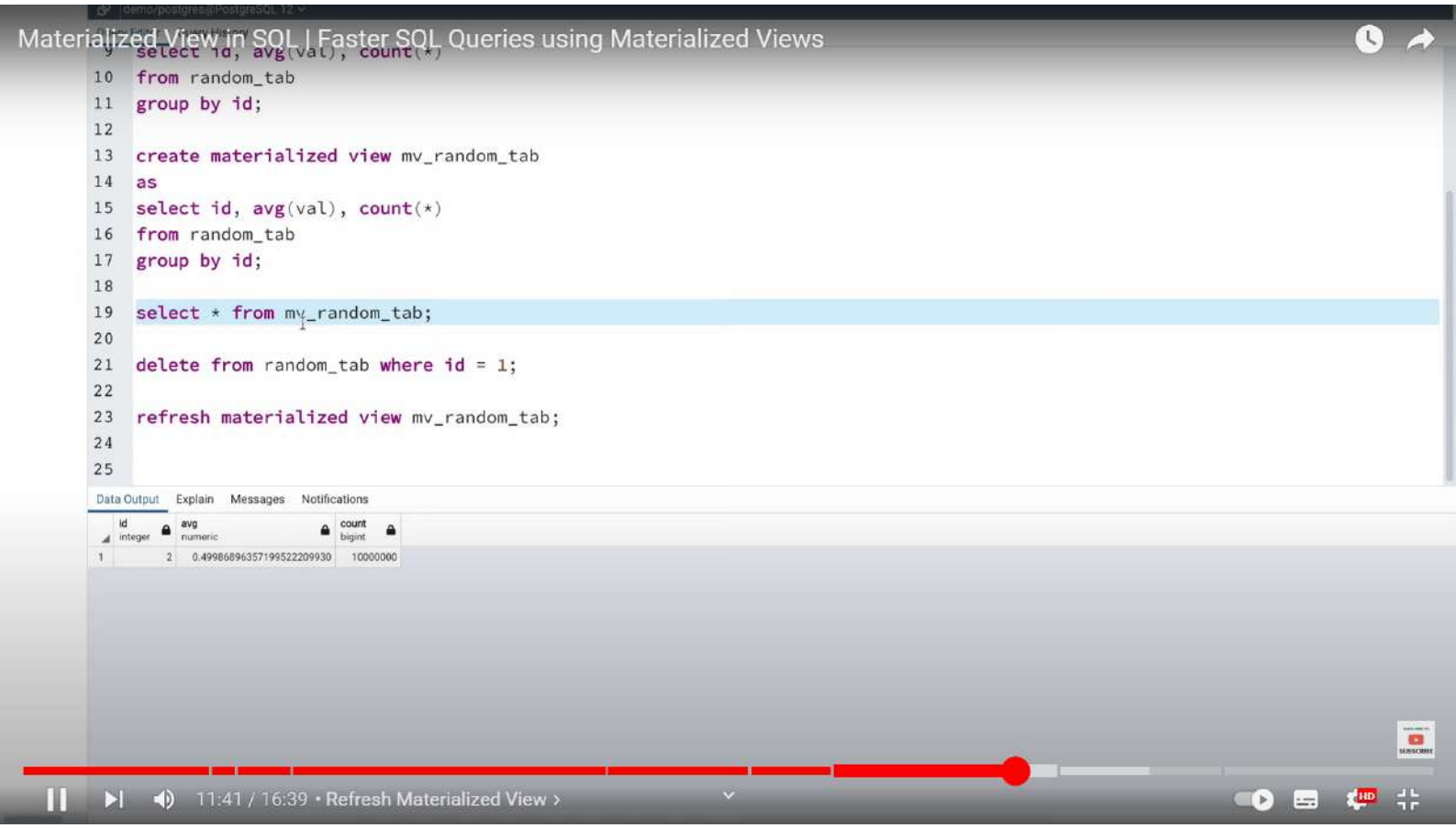
SYNTAX :

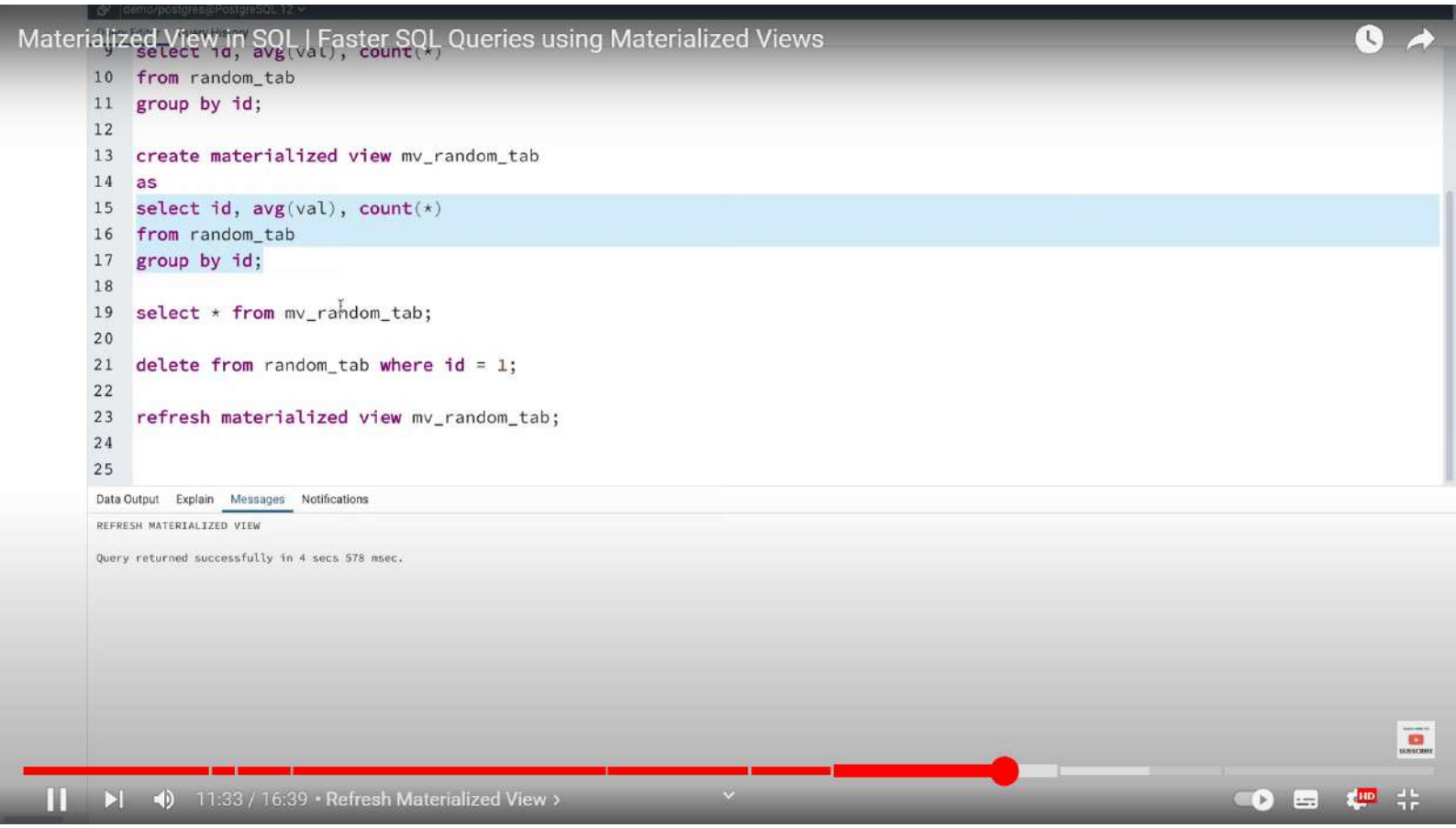
```
CREATE VIEW <view_name>
AS
SELECT col_1, Col_2......
FROM <table_name>
[WHERE condition]
```

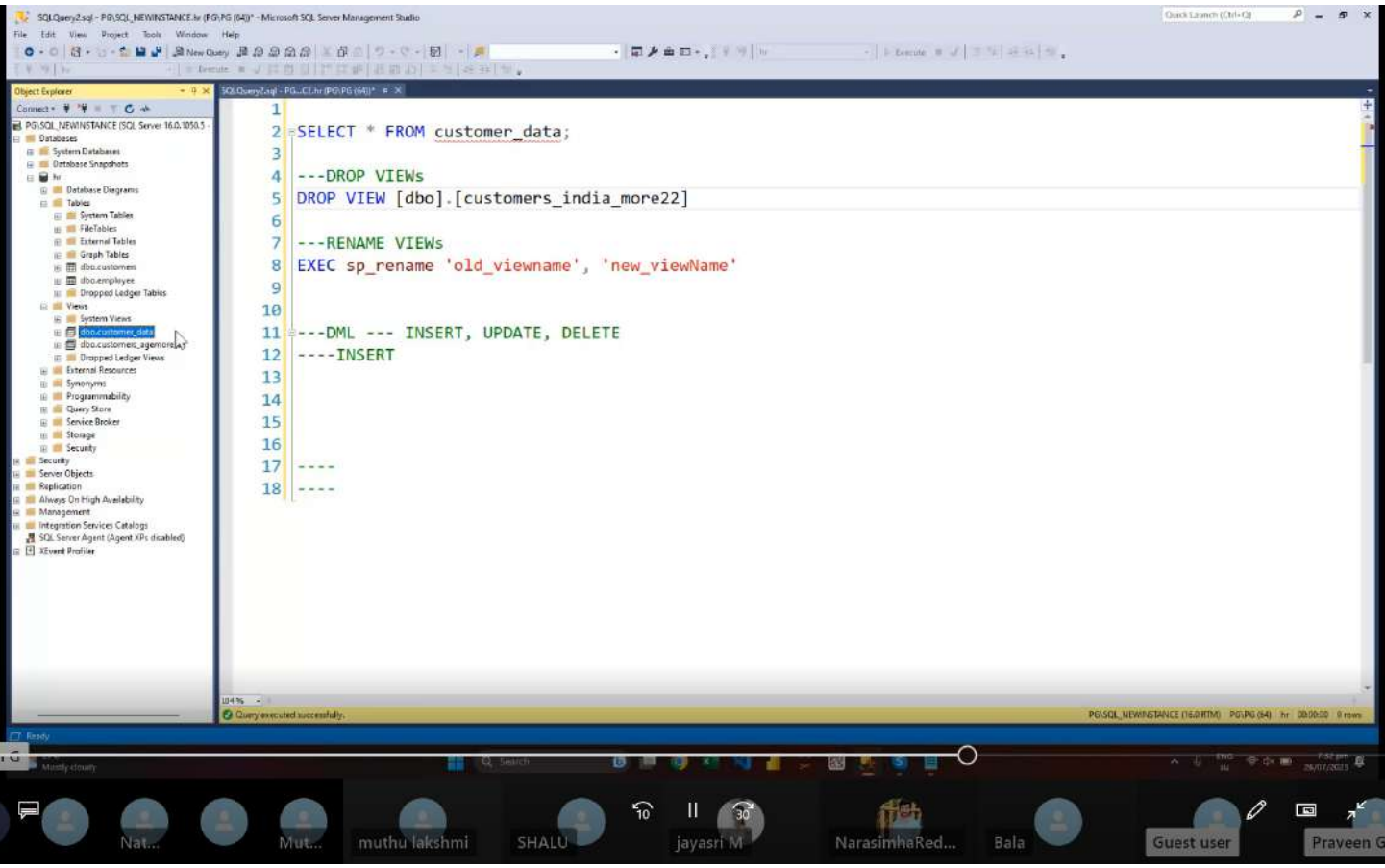
1. SIMPLE VIEW -- working with single table -- we can perform DML operations
2. COMPLEX VIEW -- Working with multiple tables -- We cannot perform DML operations
---JOINS, SET OPERATORS....

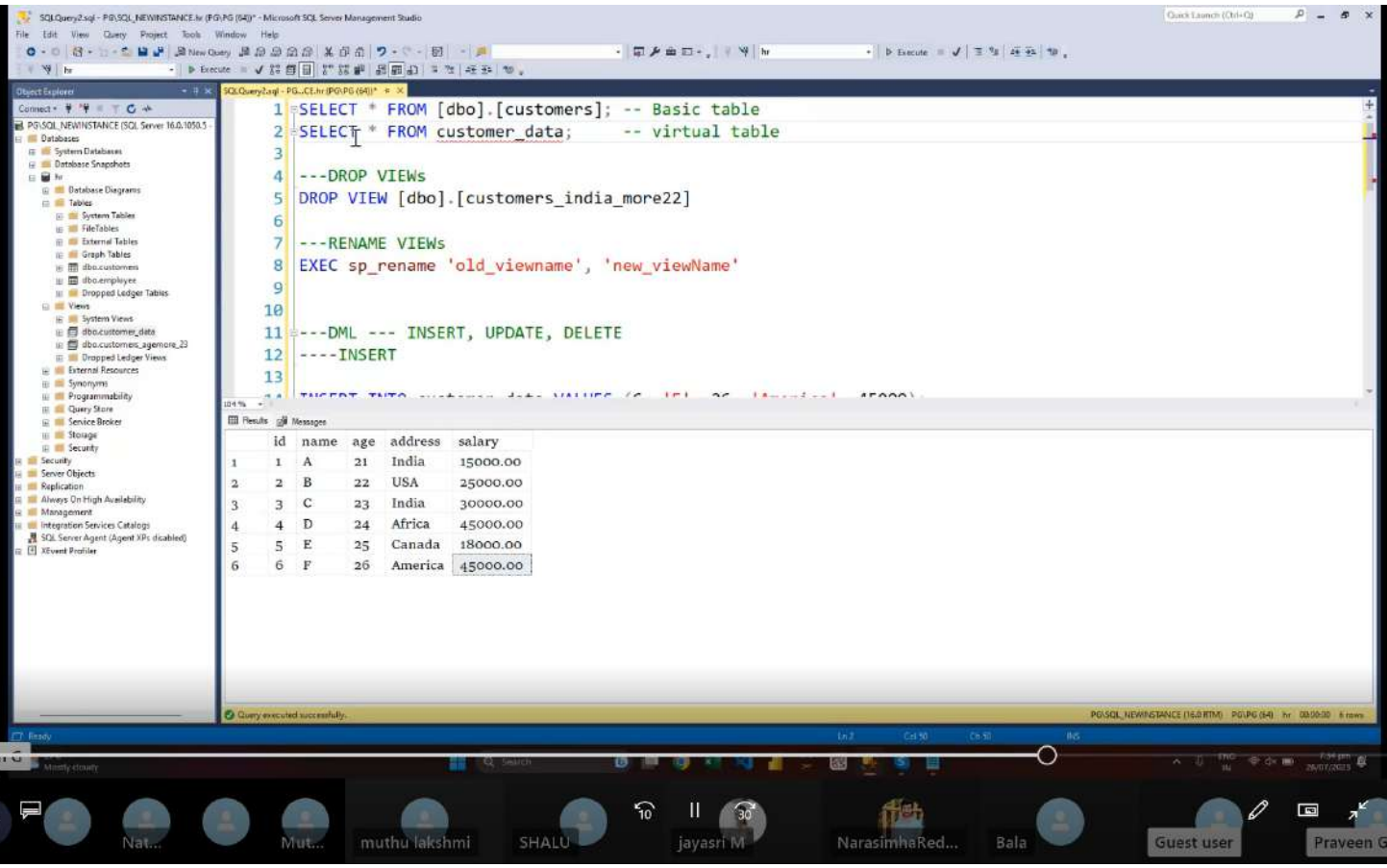

Materialized View in SQL | Faster SQL Queries using Materialized Views
create table random_tab (id int, val decimal);
insert into random_tab
select 1, random() from generate_series(1, 10000000);
insert into random_tab
select 2, random() from generate_series(1, 10000000);
select count(1) from random_tab;
Data Output
Explain
Messages
Notifications
count
bigint
20000000
Pull up for precise seeking
Creating a table with 20 million records
5:15
5:40 / 16:39 • Creating a table with 20 million records ›


Materialized View in SQL | Faster SQL Queries using Materialized Views
insert into random_tab
select 2, random() from generate_series(1, 10000000);
select id, avg(val), count(*)
from random_tab
group by id;
create materialized view mv_random_tab
as
select id, avg(val), count(*)
from random_tab
group by id;
Data Output
Explain
Messages
Notifications
SELECT 2
Query returned successfully in 3 secs 165 msec.
Pull up for precise seeking
Create a Materialized View
7:14
7:41 / 16:39 • Create a Materialized View ›
Query returned successfully in 3 secs 165 msec.

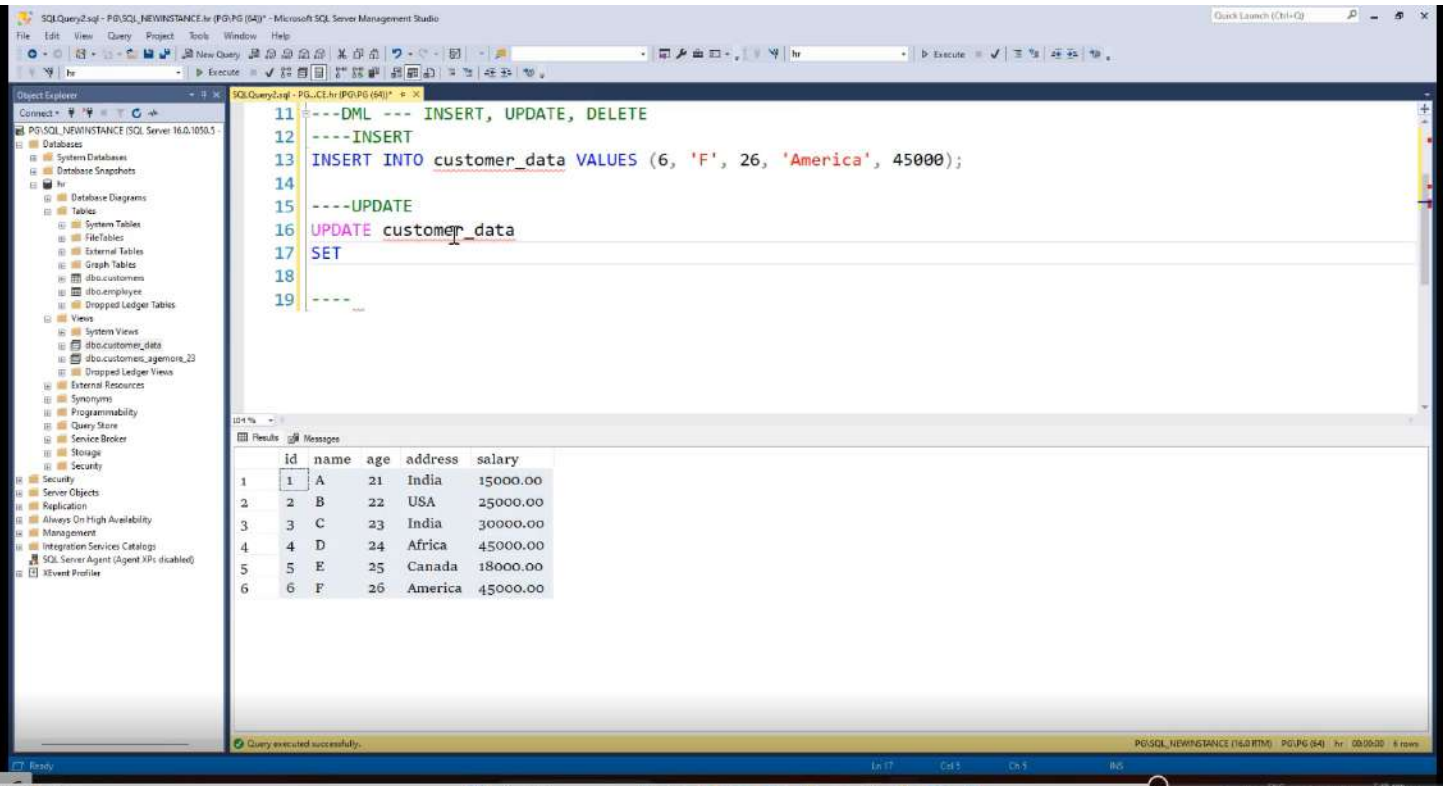

Materialized View in SQL | Faster SQL Queries using Materialized Views
select id, avg(val), count(*)
from random_tab
group by id;
create materialized view mv_random_tab
as
select id, avg(val), count(*)
from random_tab
group by id;
select * from mv_random_tab;
delete from random_tab where id = 1;
refresh materialized view mv_random_tab;
Data Output
Explain
Messages
Notifications
id integer
avg numeric
count bigint
1
2
0.49986896357199522209930
10000000
11:41 / 16:39 • Refresh Materialized View ›


Materialized View in SQL | Faster SQL Queries using Materialized Views
select id, avg(val), count(*)
from random_tab
group by id;
create materialized view mv_random_tab
as
select id, avg(val), count(*)
from random_tab
group by id;
select * from mv_random_tab;
delete from random_tab where id = 1;
refresh materialized view mv_random_tab;
Data Output
Explain
Messages
Notifications
REFRESH MATERIALIZED VIEW
Query returned successfully in 4 secs 578 msec.
11:33 / 16:39 • Refresh Materialized View ›

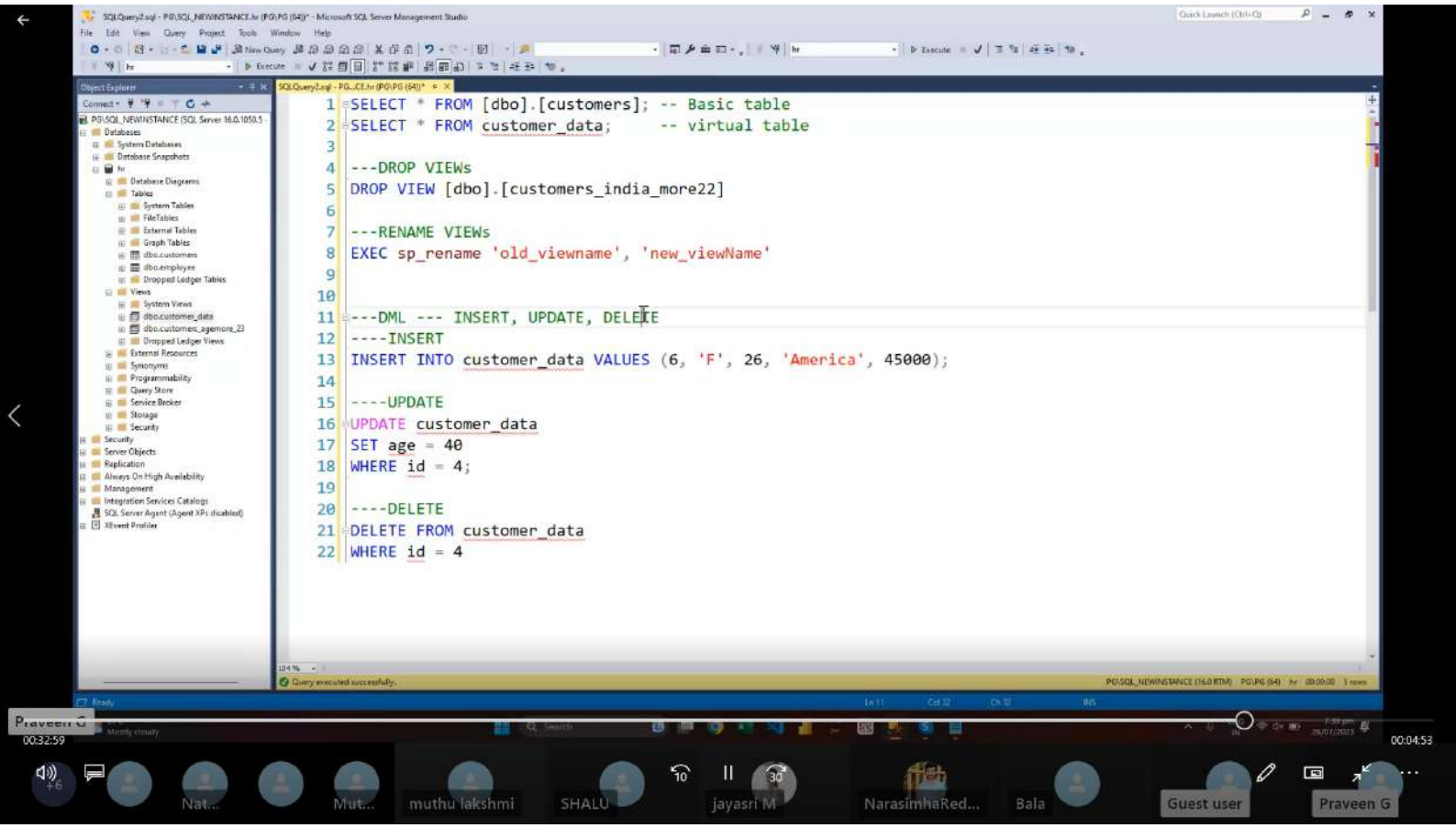

SQLQuery2.sql - PG\SQL_NEWINSTANCE.hr (PG\PG (64))* - Microsoft SQL Server Management Studio
Object Explorer
SELECT * FROM customer_data;
---DROP VIEWs
DROP VIEW [dbo].[customers_india_more22]
---RENAME VIEWs
EXEC sp_rename 'old_viewname', 'new_viewName'
---DML --- INSERT, UPDATE, DELETE
----INSERT
Query executed successfully.



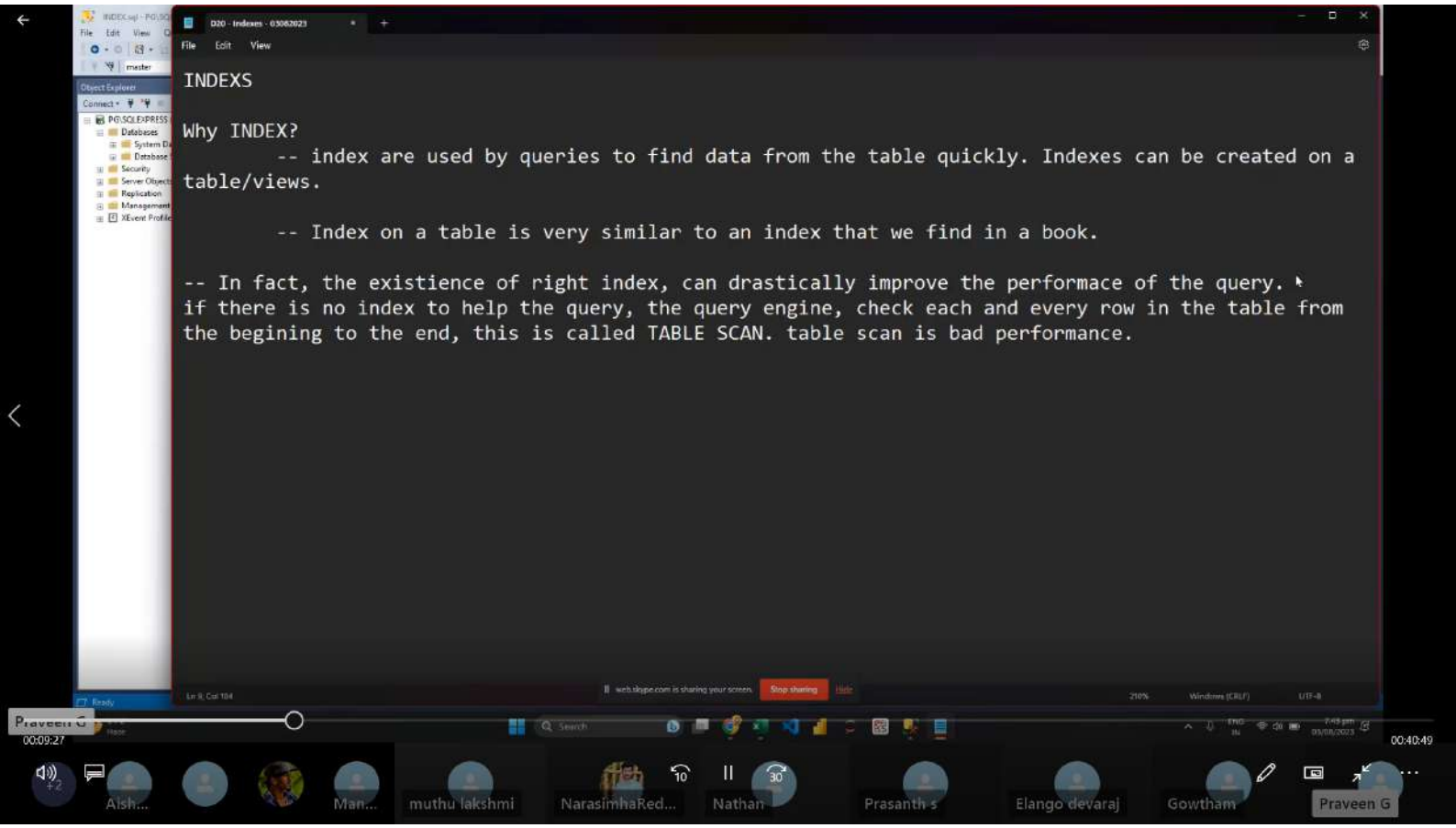
Praveen G
00:28:04
00:09:48
Nat...
Mut...
muthu lakshmi
SHALU
jayasri M
NarasimhaRed...
Bala
Guest user
Praveen G

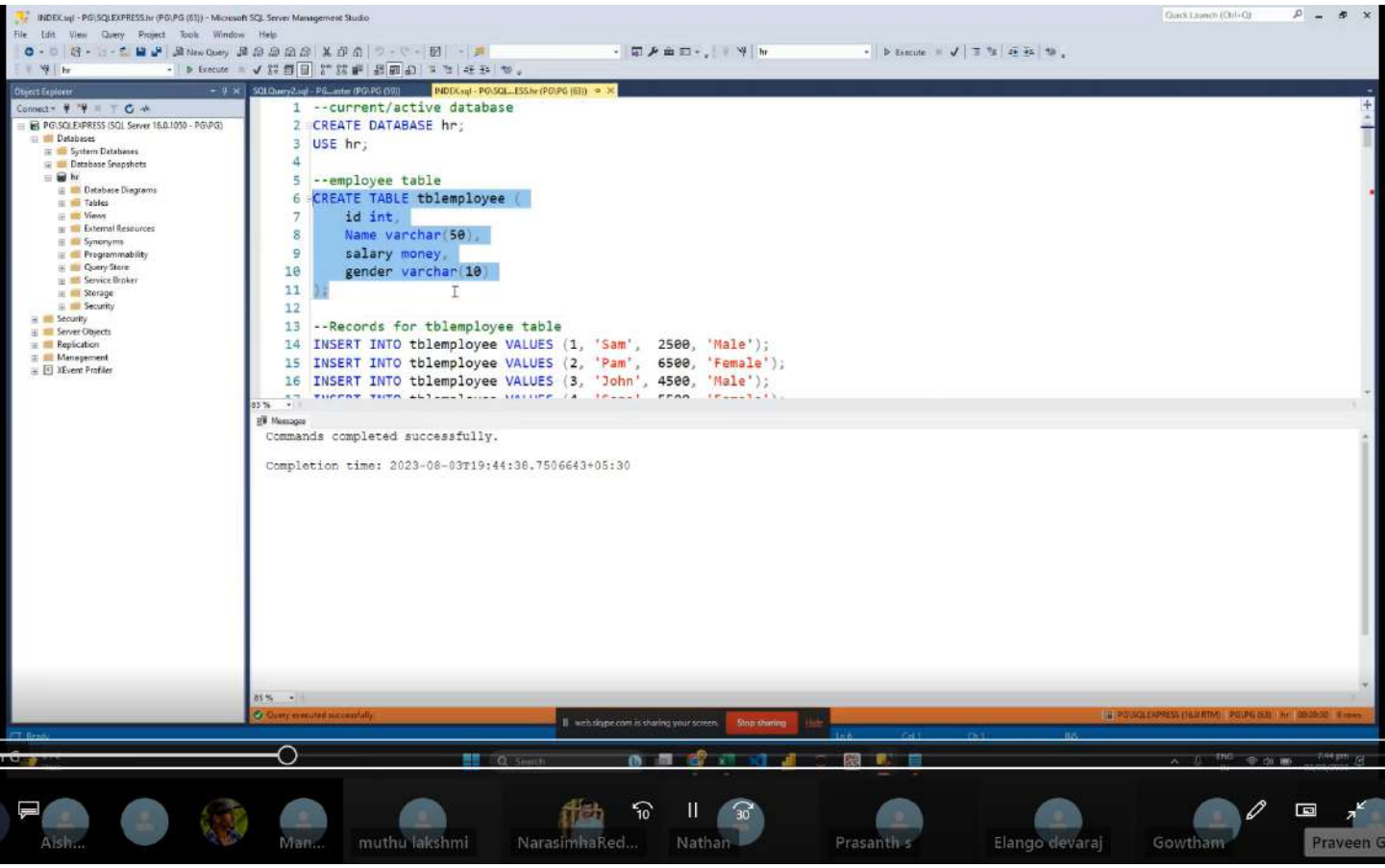


Praveen G
00:29:48
00:08:04
Nat...
Mut...
muthu lakshmi
SHALU
jayasri M
NarasimhaRed...
Bala
Guest user
Praveen G



Praveen G
00:32:59
00:04:53
Nat...
Mut...
muthu lakshmi
SHALU
jayasri M
NarasimhaRed...
Bala
Guest user
Praveen G

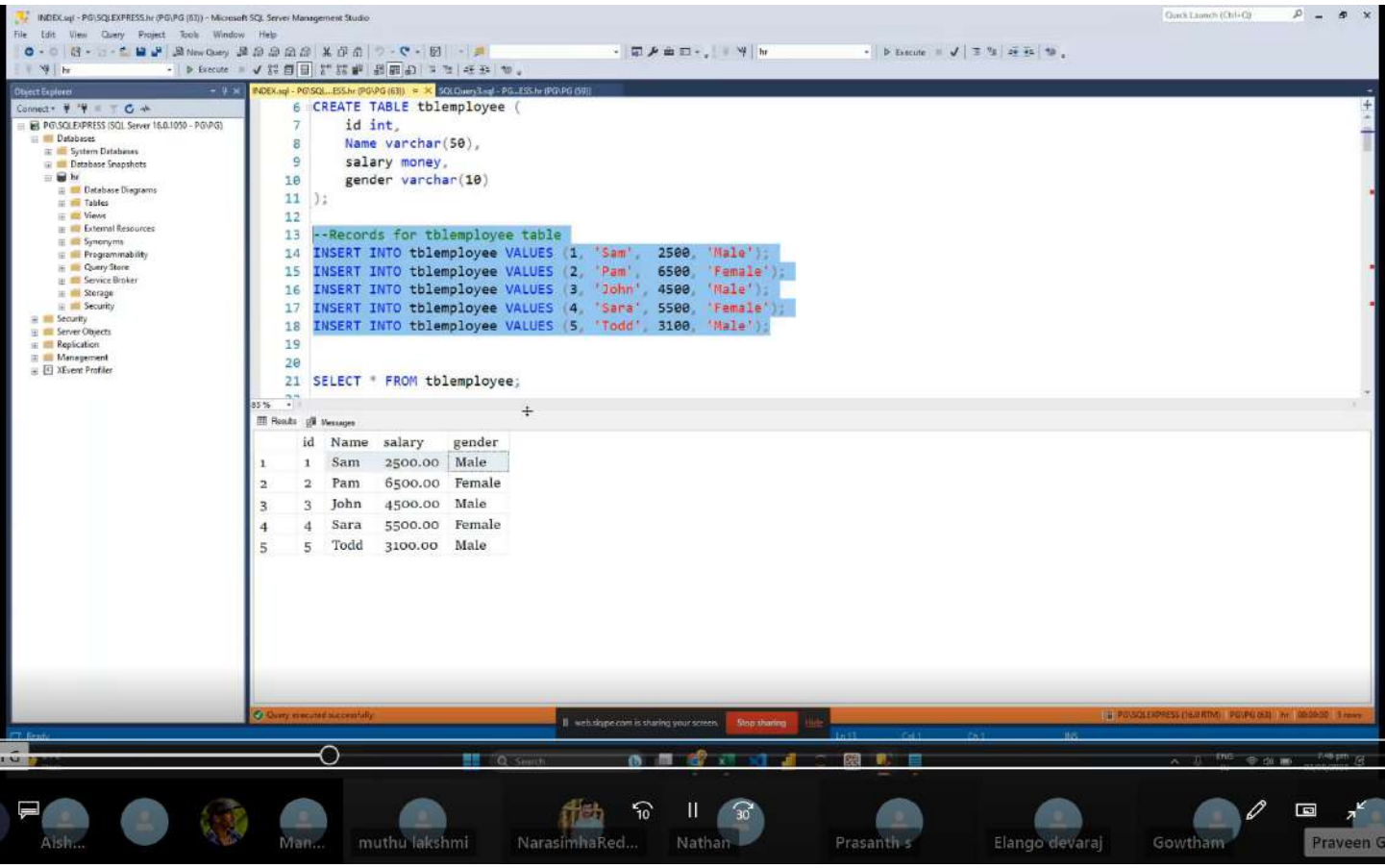

INDEXS
Why INDEX?
-- index are used by queries to find data from the table quickly. Indexes can be created on a table/views.
-- Index on a table is very similar to an index that we find in a book.
-- In fact, the existience of right index, can drastically improve the performace of the query.
if there is no index to help the query, the query engine, check each and every row in the table from the begining to the end, this is called TABLE SCAN. table scan is bad performance.
Praveen G
00:09:27
00:40:49
Aish...
Man...
muthu lakshmi
NarasimhaRed...
Nathan
Prasanth s
Elango devaraj
Gowtham
Praveen G

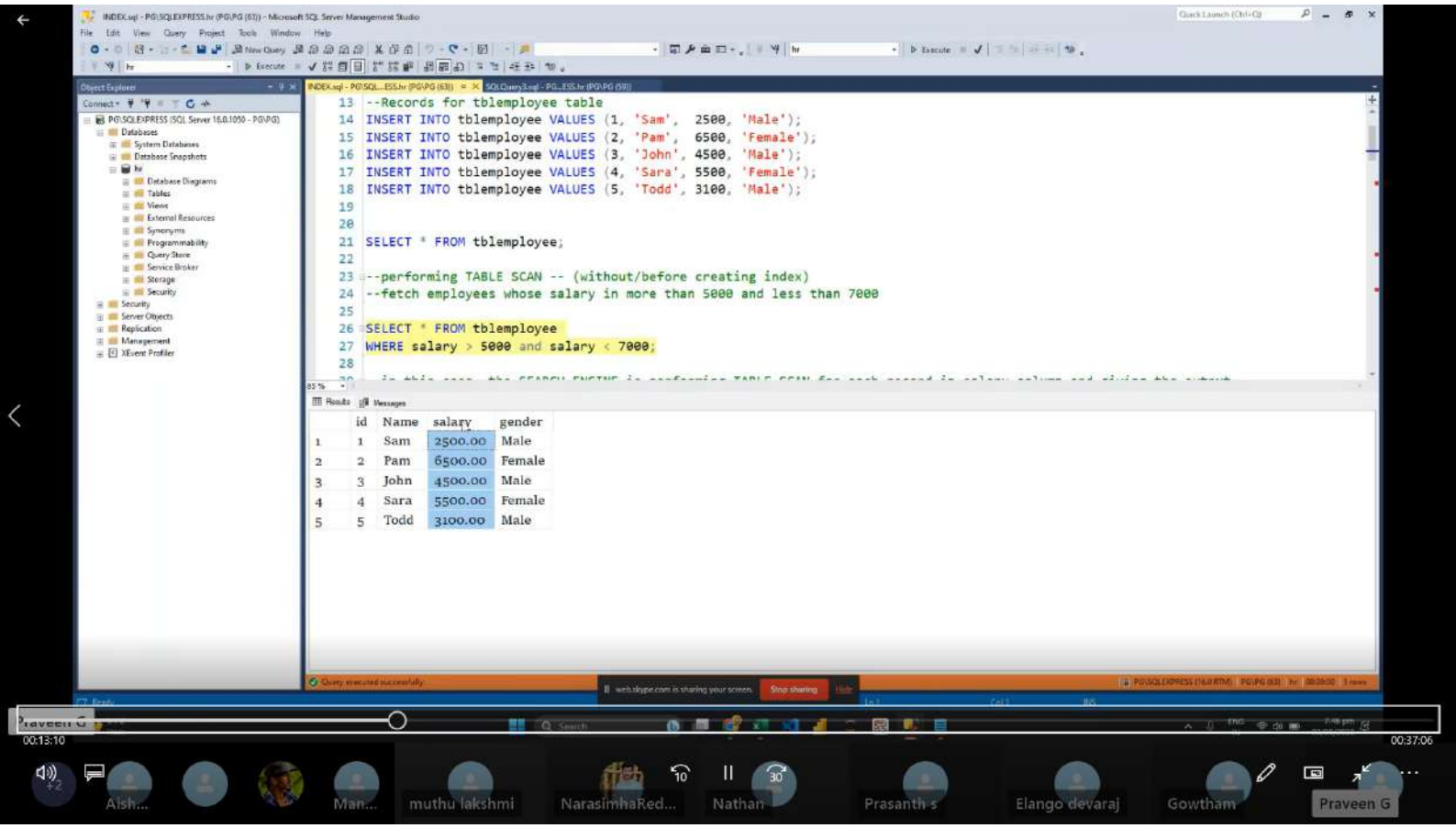


00:11:08

00:39:08

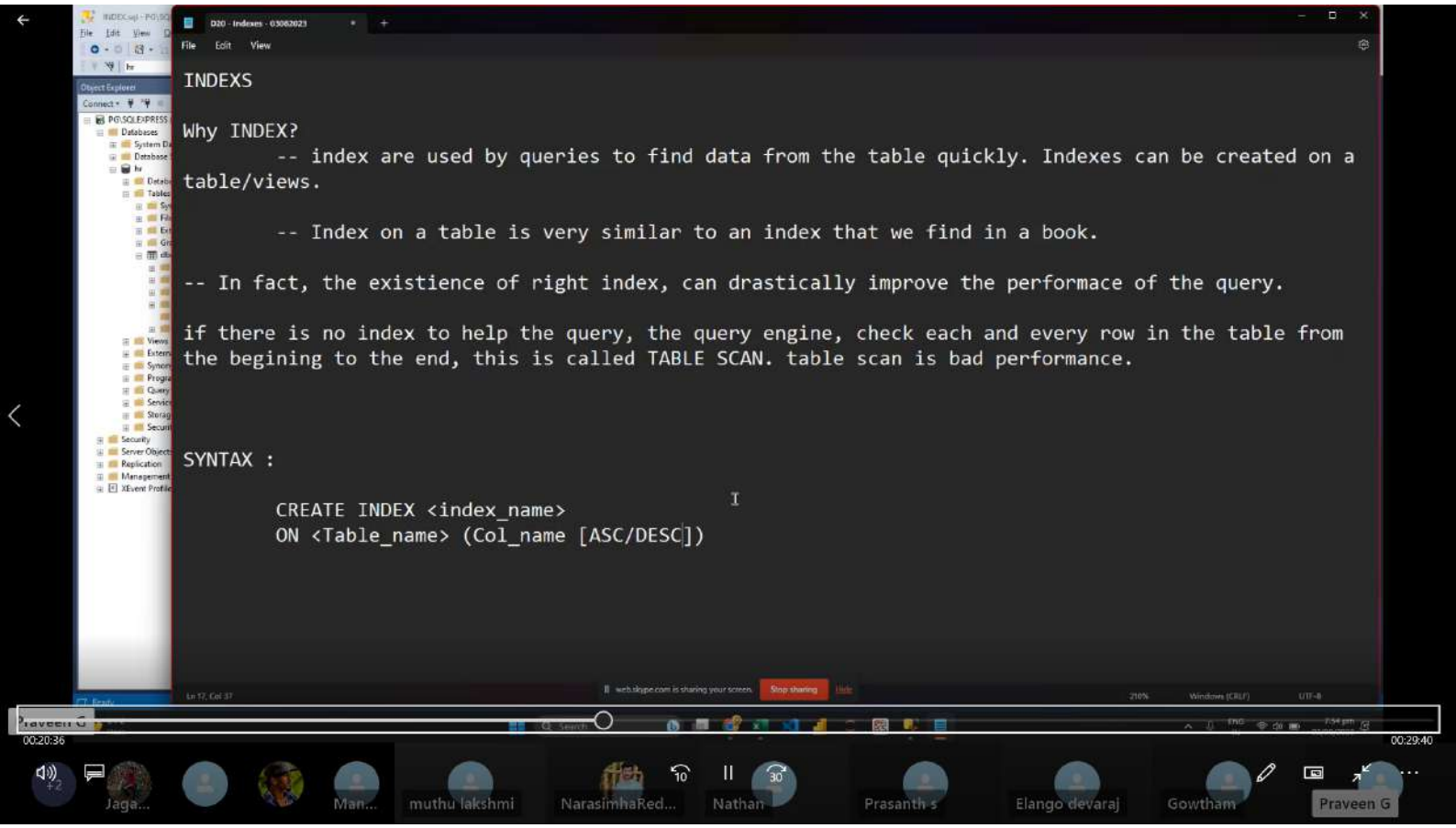
Aish... Man... muthu lakshmi NarasimhaRed... Nathan Prasanth s Elango devaraj Gowtham Praveen G


INDEX.sql - PG\SQLEXPRESS.hr (PG\PG (63)) - Microsoft SQL Server Management Studio
Object Explorer
Connect
PG\SQLEXPRESS (SQL Server 16.0.1050 - PG\PG)
Databases
System Databases
Database Snapshots
hr
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Query Store
Service Broker
Storage
Security
Security
Server Objects
Replication
Management
XEvent Profiler
INDEX.sql - PG\SQL...ESS.hr (PG\PG (63))
SQLQuery3.sql - PG...ESS.hr (PG\PG (59))
CREATE TABLE tblemployee (
id int,
Name varchar(50),
salary money,
gender varchar(10)
);
--Records for tblemployee table
INSERT INTO tblemployee VALUES (1, 'Sam', 2500, 'Male');
INSERT INTO tblemployee VALUES (2, 'Pam', 6500, 'Female');
INSERT INTO tblemployee VALUES (3, 'John', 4500, 'Male');
INSERT INTO tblemployee VALUES (4, 'Sara', 5500, 'Female');
INSERT INTO tblemployee VALUES (5, 'Todd', 3100, 'Male');
SELECT * FROM tblemployee;
Results
Messages
id Name salary gender
1 1 Sam 2500.00 Male
2 2 Pam 6500.00 Female
3 3 John 4500.00 Male
4 4 Sara 5500.00 Female
5 5 Todd 3100.00 Male
Query executed successfully.
web.skype.com is sharing your screen.
Stop sharing
Hide
Ready
Praveen G
00:12:35
00:37:41
Aish...
Man...
muthu lakshmi
NarasimhaRed...
Nathan
Prasanth s
Elango devaraj
Gowtham
Praveen G

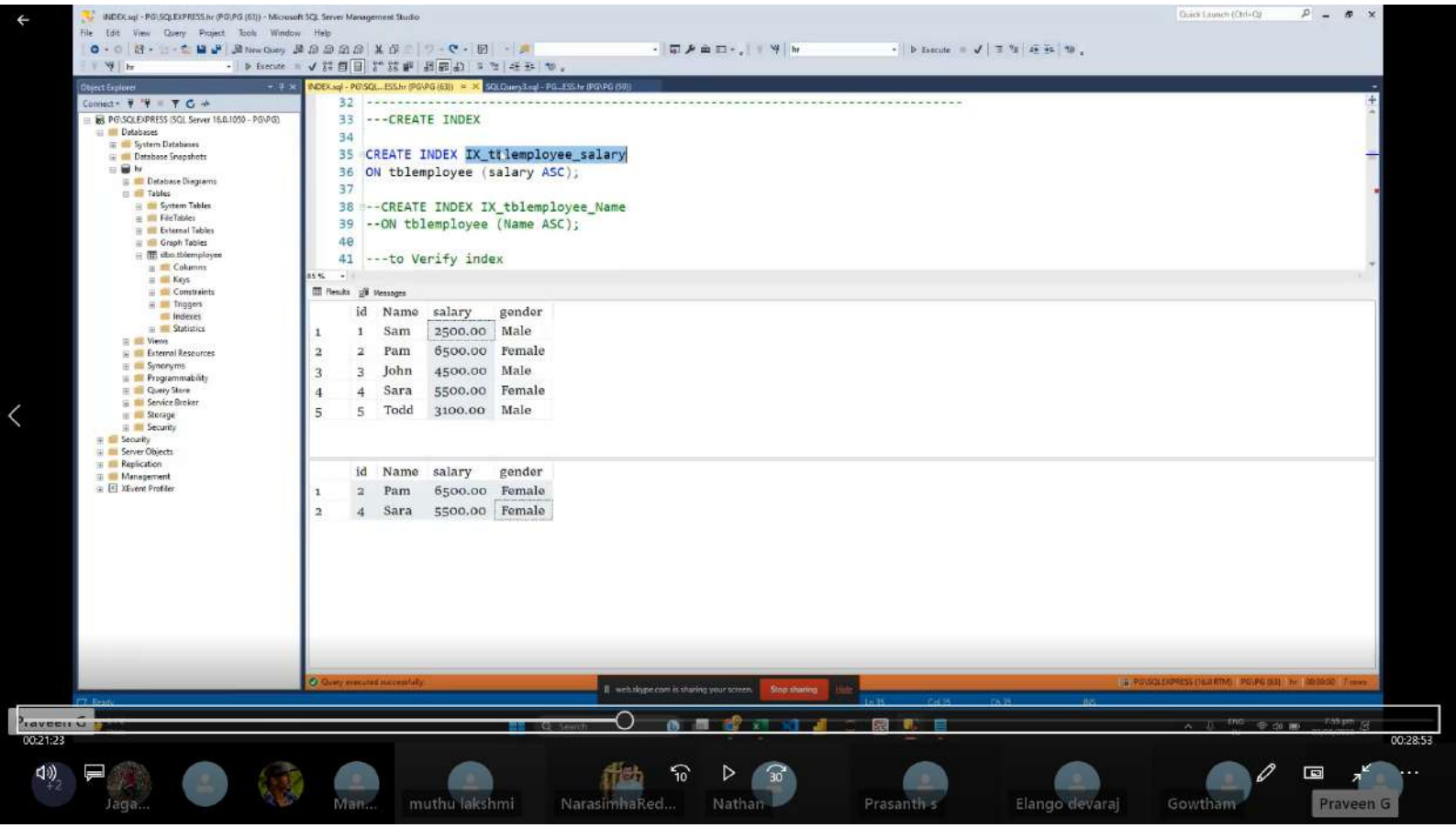


00:13:10 00:37:06

Aish... Man... muthu lakshmi NarasimhaRed... Nathan Prasanth s Elango devaraj Gowtham Praveen G

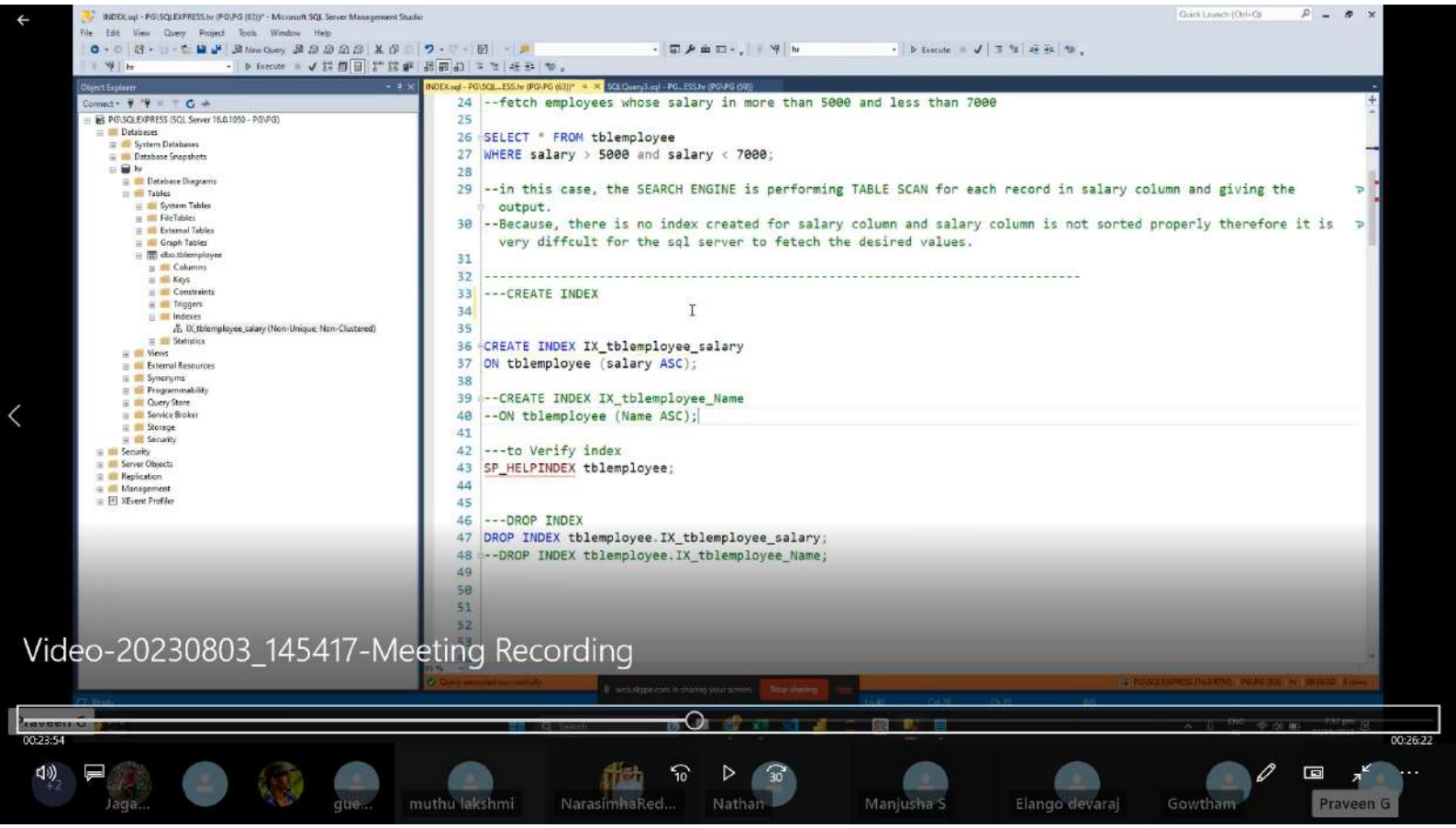


00:20:36
00:29:40
Jaga...
Man...
muthu lakshmi
NarasimhaRed...
Nathan
Prasanth s
Elango devaraj
Gowtham
Praveen G



00:21:23 00:28:53

Jaga... Man... muthu lakshmi NarasimhaRed... Nathan Prasanth s Elango devaraj Gowtham Praveen G



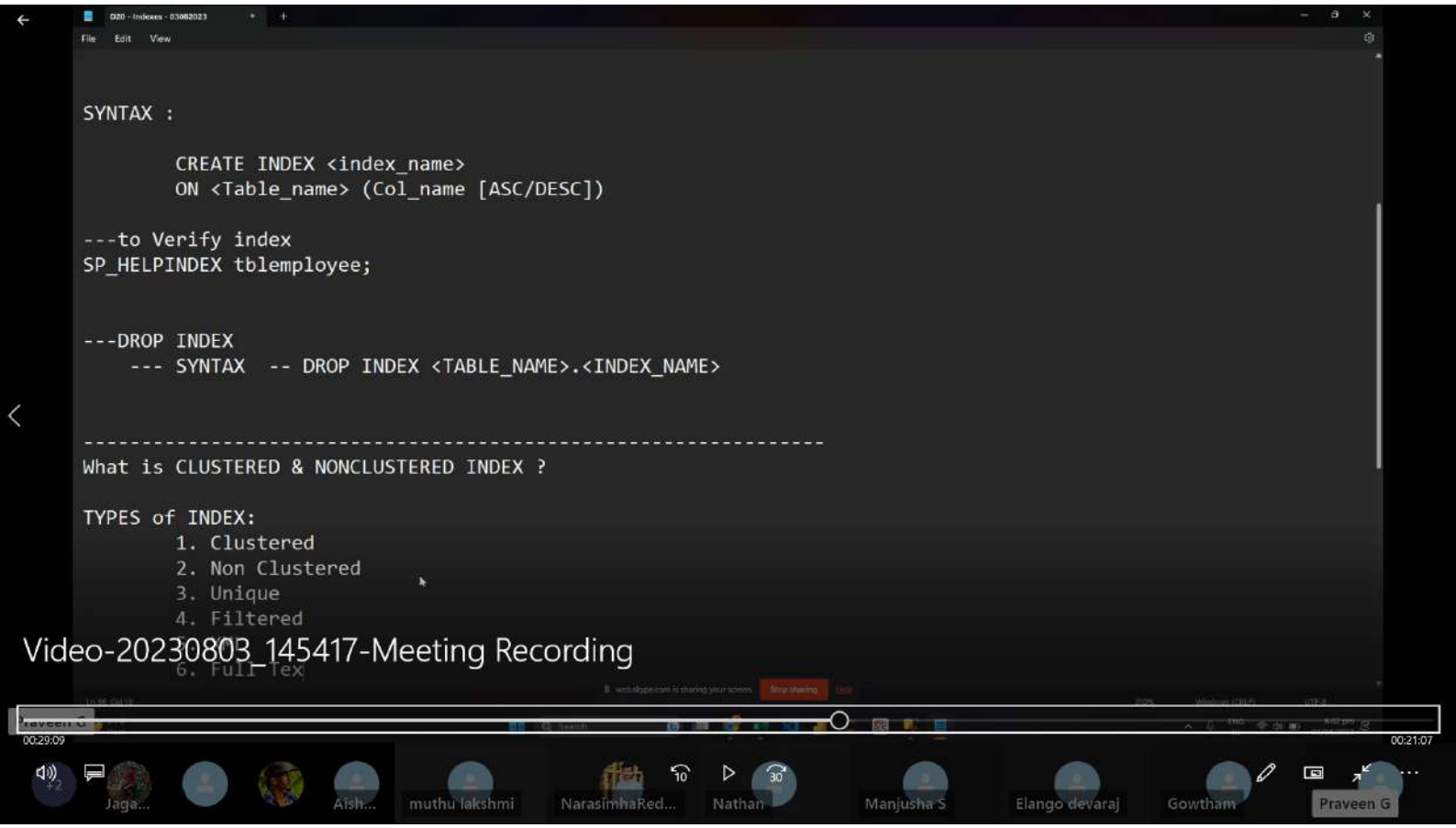
Video-20230803_145417-Meeting Recording

Praveen G

00:23:54

00:26:22

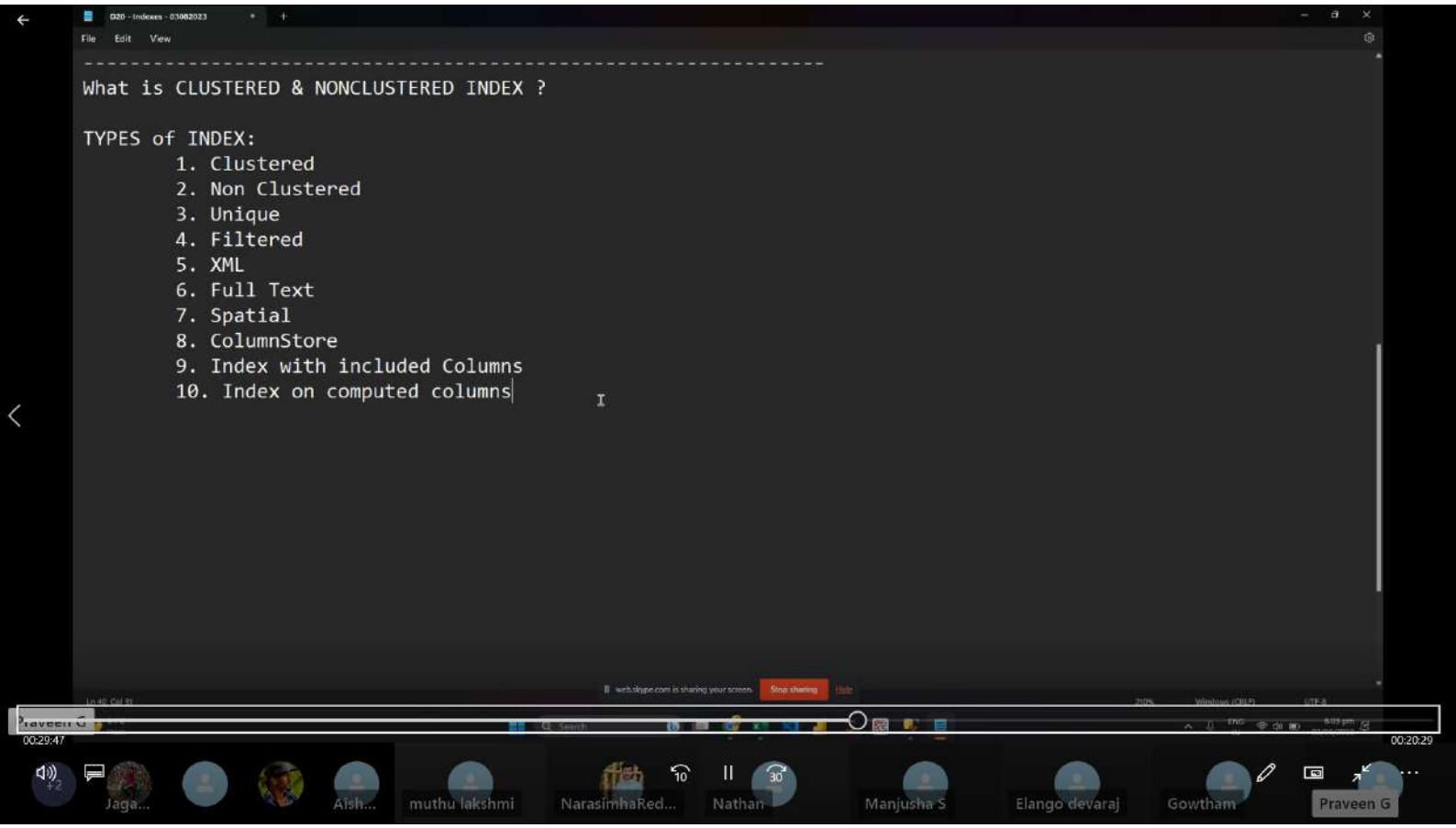
Jaga... gue... muthu lakshmi NarasimhaRed... Nathan Manjusha S Elango devaraj Gowtham Praveen G



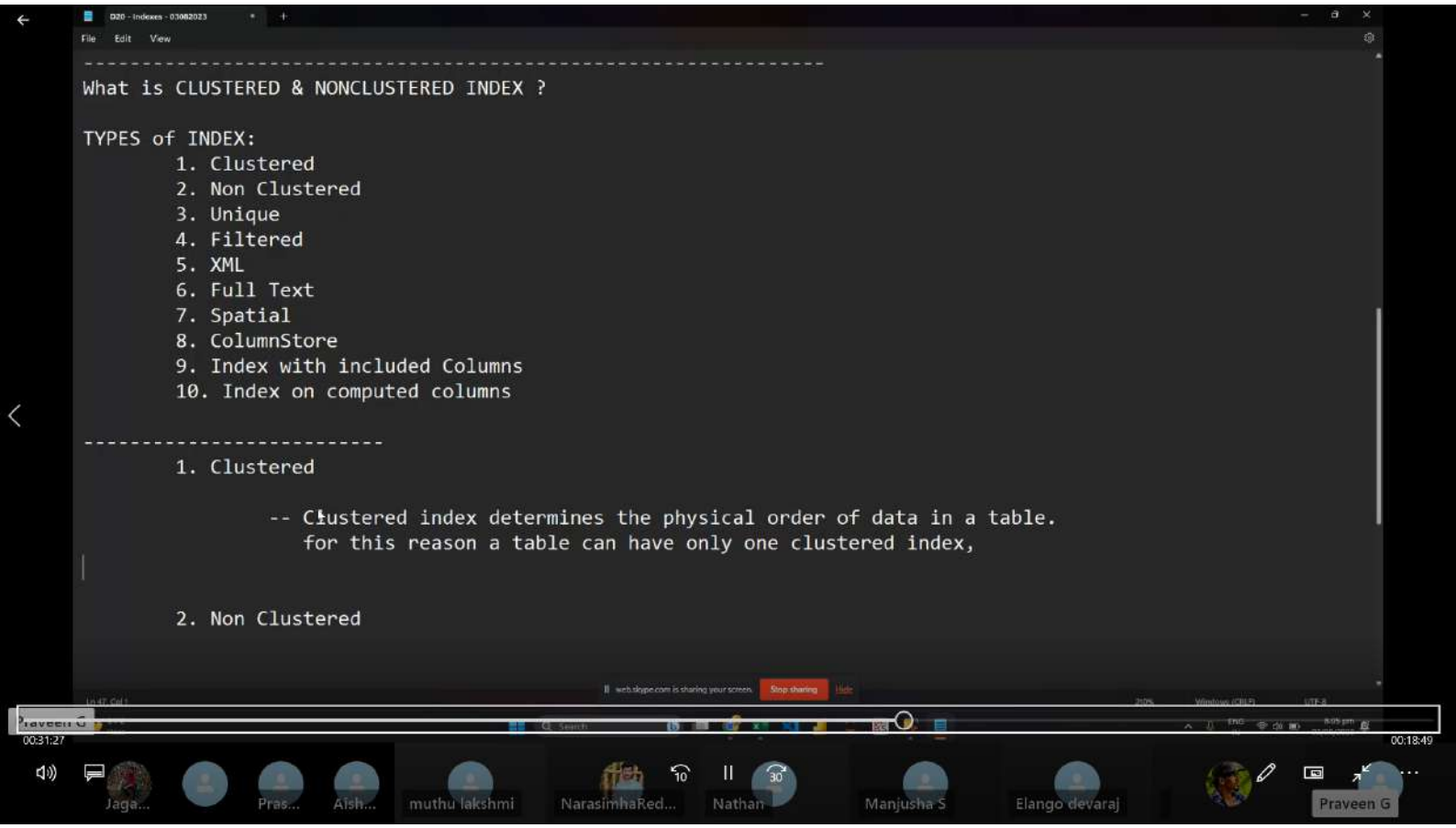
Video-20230803_145417-Meeting Recording

00:29:09 00:21:07

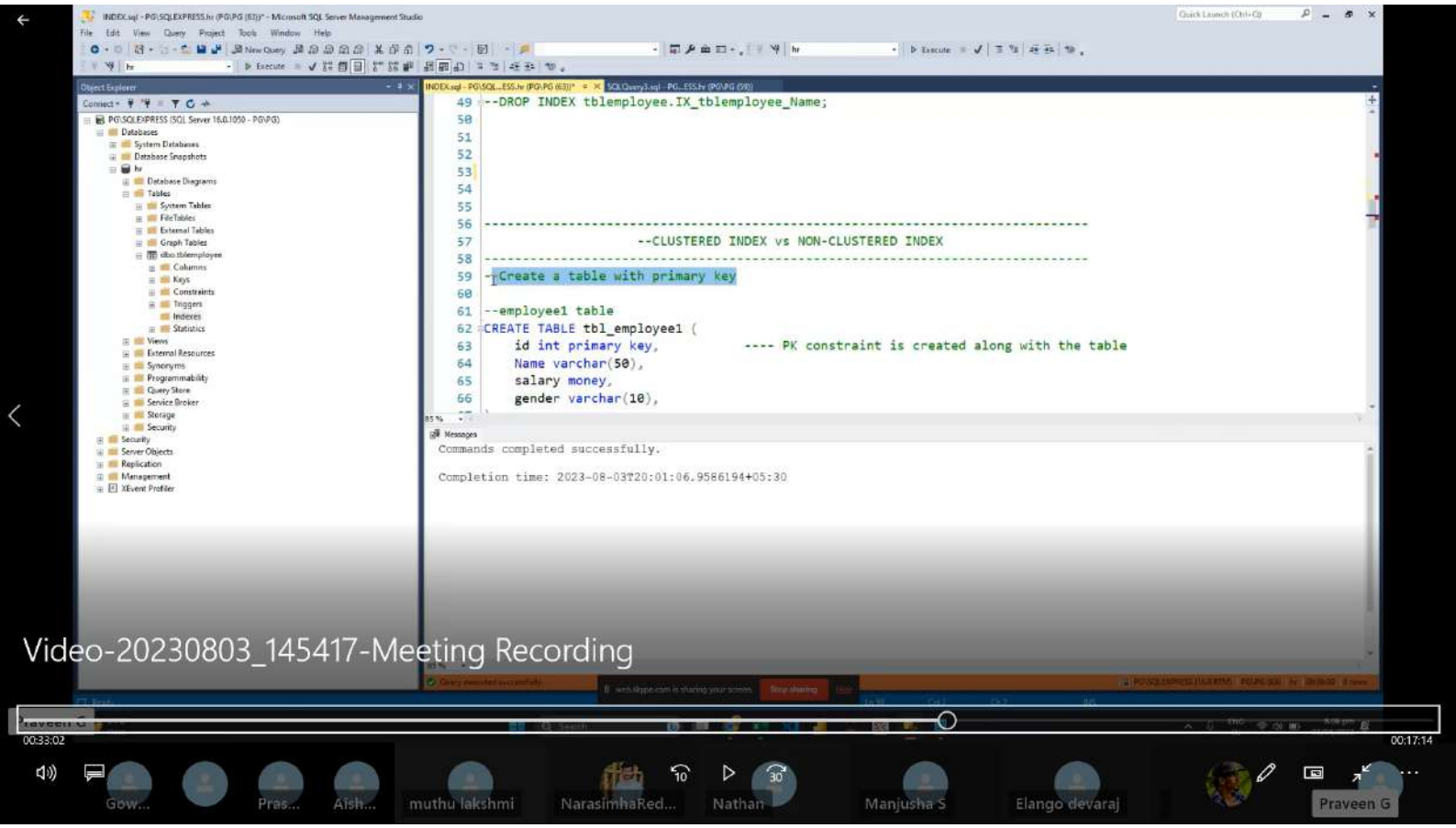
Jaga... Aish... muthu lakshmi NarasimhaRed... Nathan Manjusha S Elango devaraj Gowtham Praveen G



Praveen G
00:29:47
00:20:29
Jaga...
Aish...
muthu lakshmi
NarasimhaRed...
Nathan
Manjusha S
Elango devaraj
Gowtham
Praveen G



Praveen G
00:31:27
00:18:49
Jaga...
Pras...
Aish...
muthu lakshmi
NarasimhaRed...
Nathan
Manjusha S
Elango devaraj
Praveen G



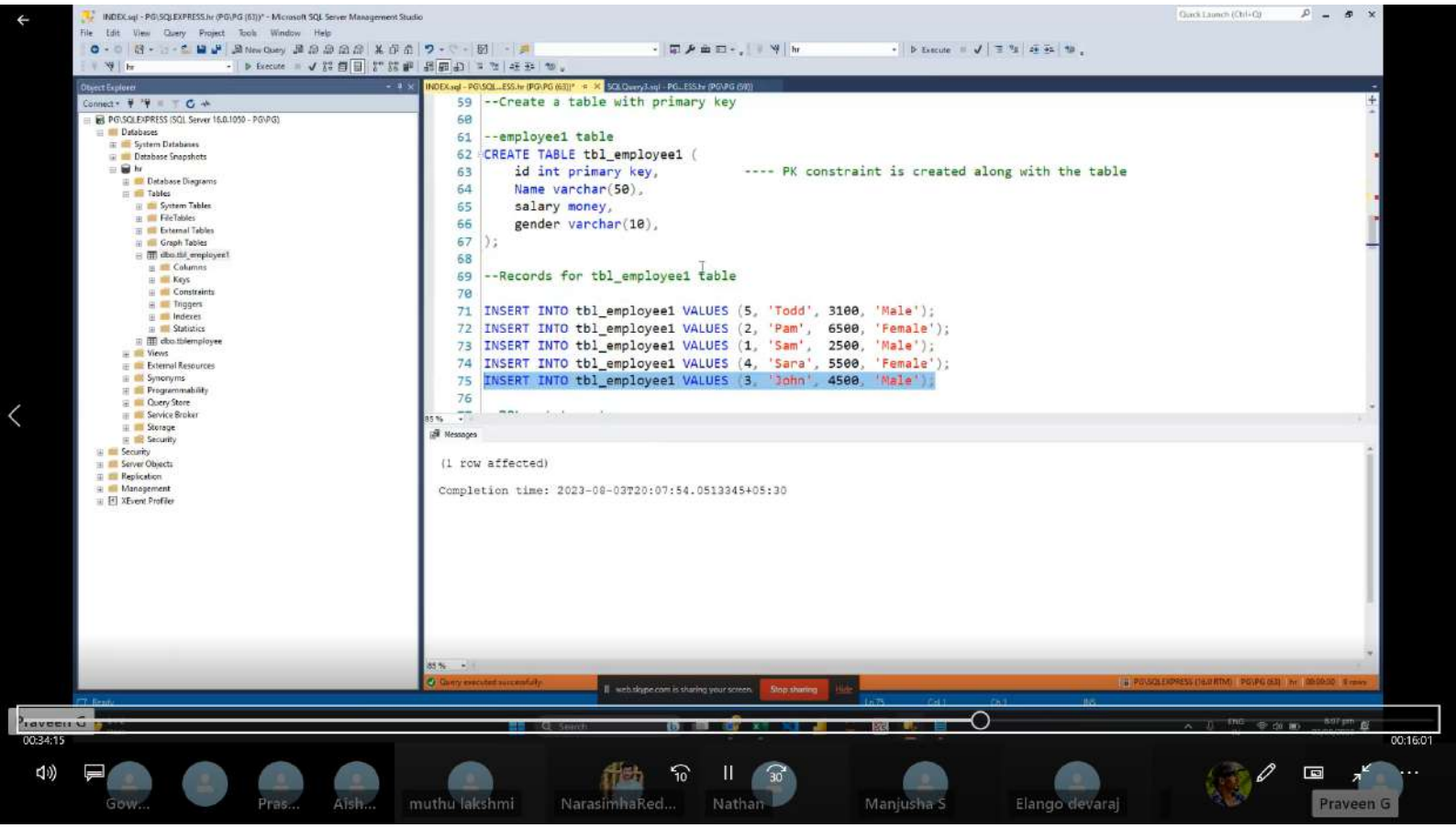
Video-20230803_145417-Meeting Recording

Praveen G

00:33:02

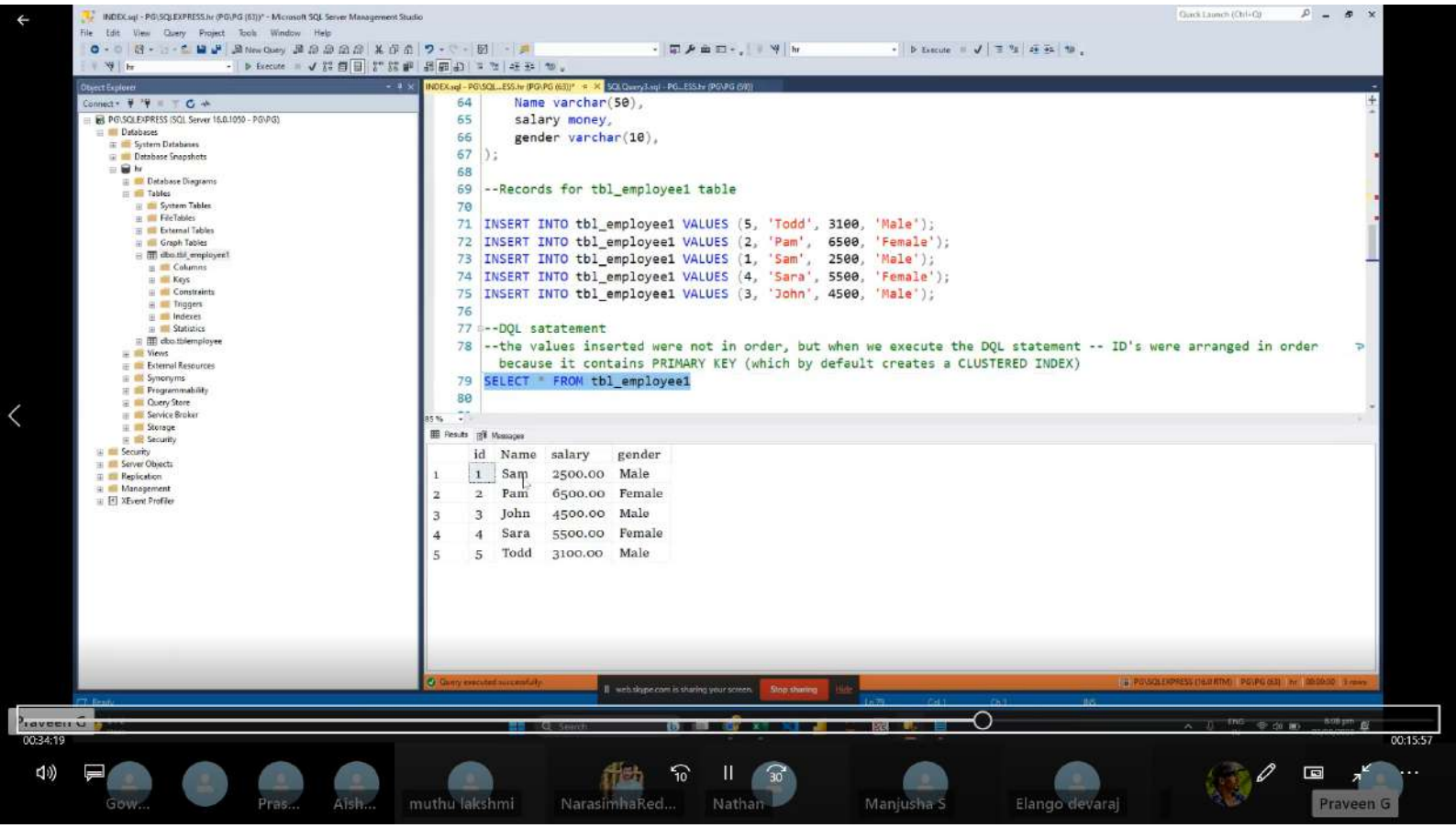
00:17:14

Gow... Pras... Aish... muthu lakshmi NarasimhaRed... Nathan Manjusha S Elango devaraj Praveen G

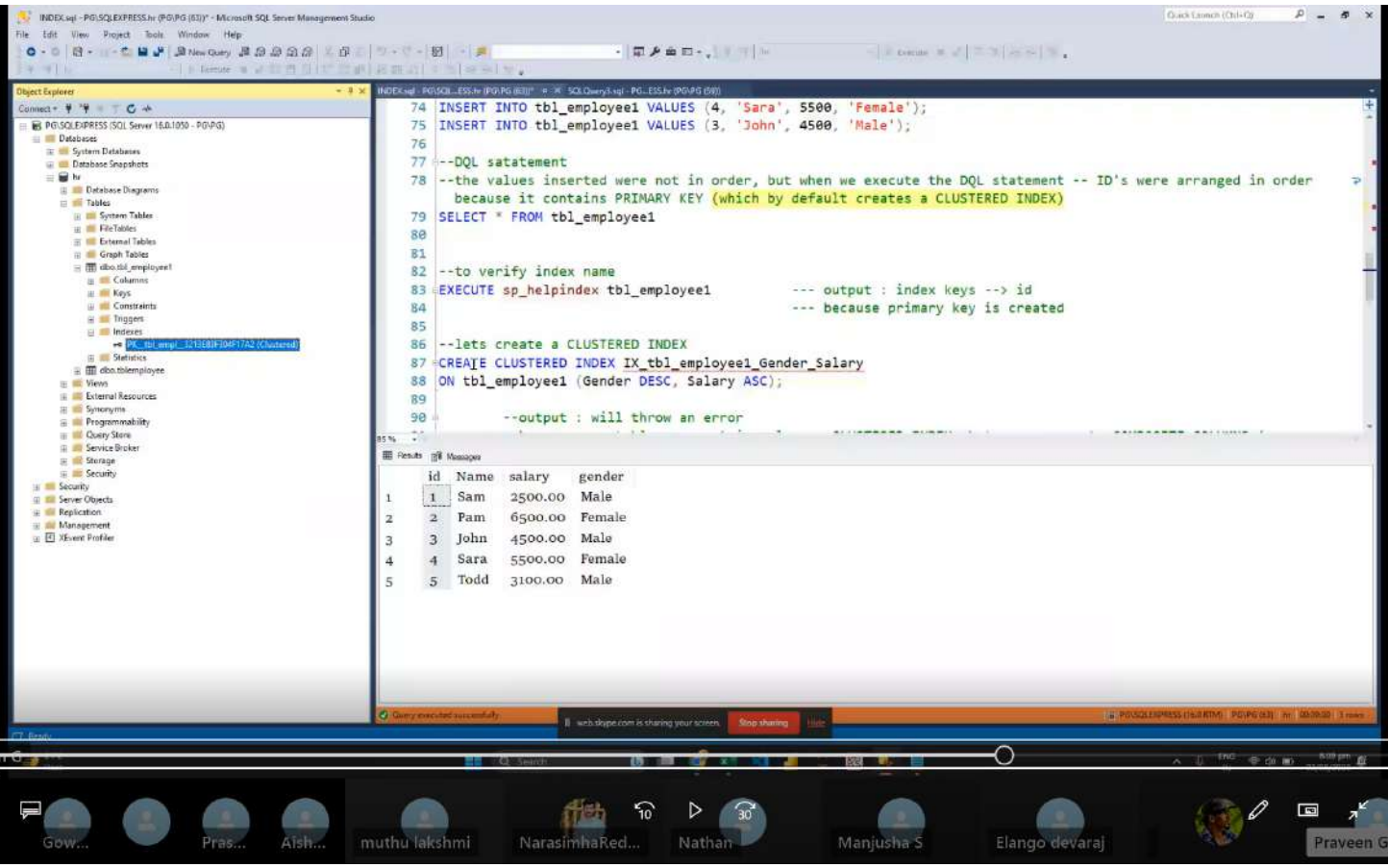


00:34:15 00:16:01

Gow... Pras... Aish... muthu lakshmi NarasimhaRed... Nathan Manjusha S Elango devaraj Praveen G



Praveen G
00:34:19
00:15:57
Gow... Pras... Aish... muthu lakshmi NarasimhaRed... Nathan Manjusha S Elango devaraj Praveen G

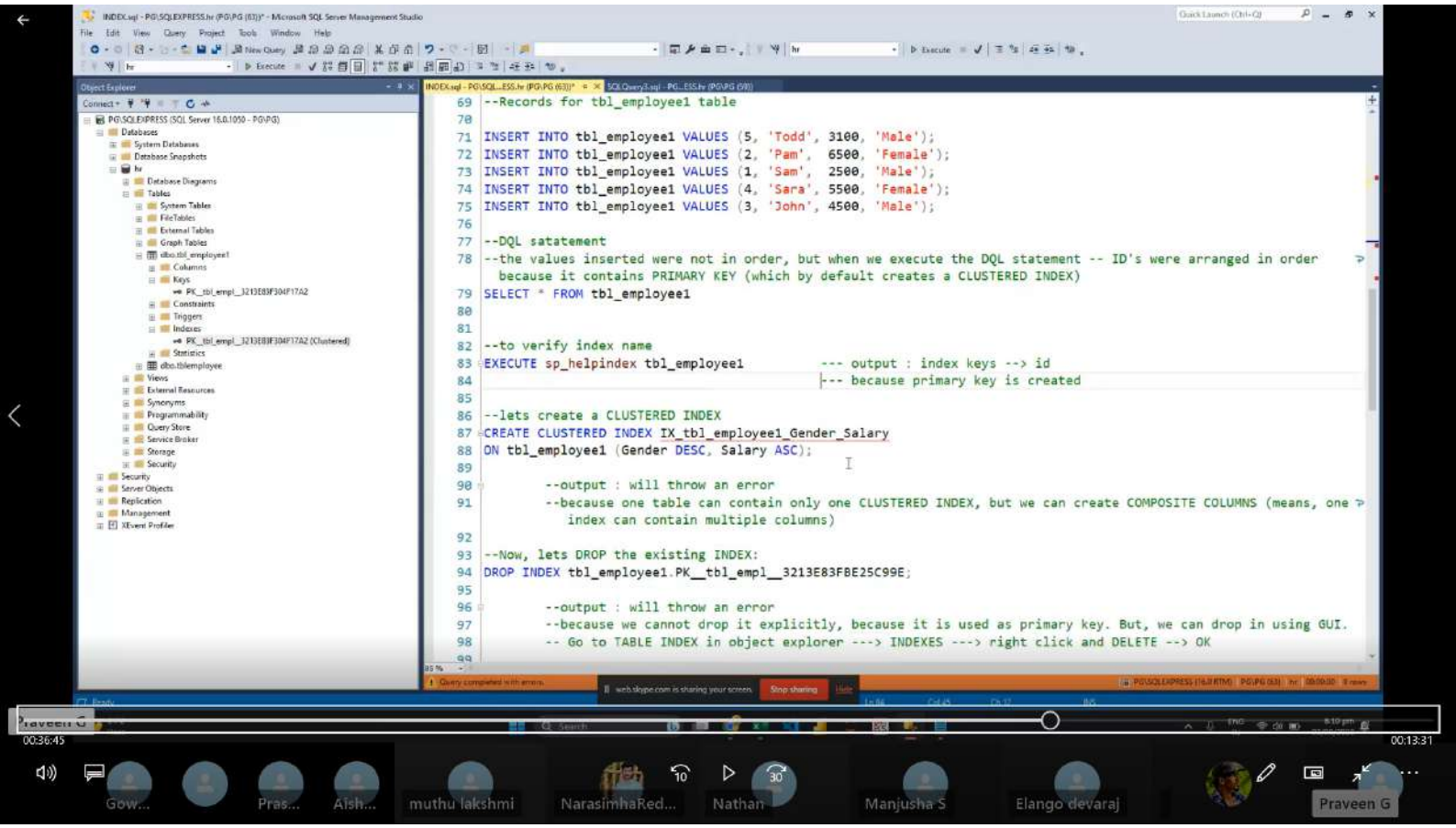


Praveen G

00:35:47

00:14:29

Gow... Pras... Aish... muthu lakshmi NarasimhaRed... Nathan Manjusha S Elango devaraj Praveen G

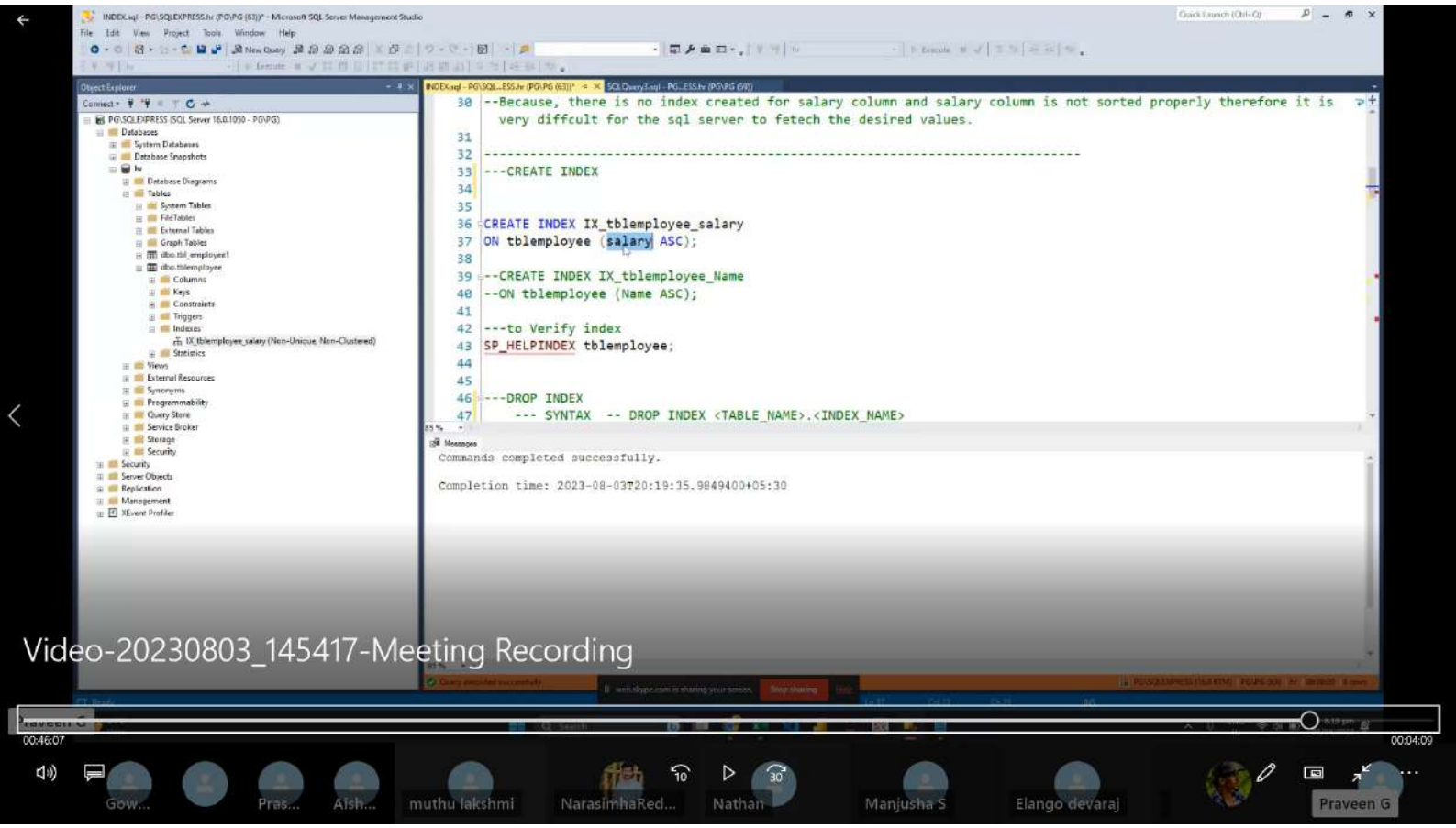


Praveen G
00:36:45
00:13:31
Gow...
Pras...
Aish...
muthu lakshmi
NarasimhaRed...
Nathan
Manjusha S
Elango devaraj
Praveen G

Video-20230803_145417-Meeting Recording

00:45:16 00:05:00

Gow... Pras... Aish... muthu lakshmi NarasimhaRed... Nathan Manjusha S Elango devaraj Praveen G



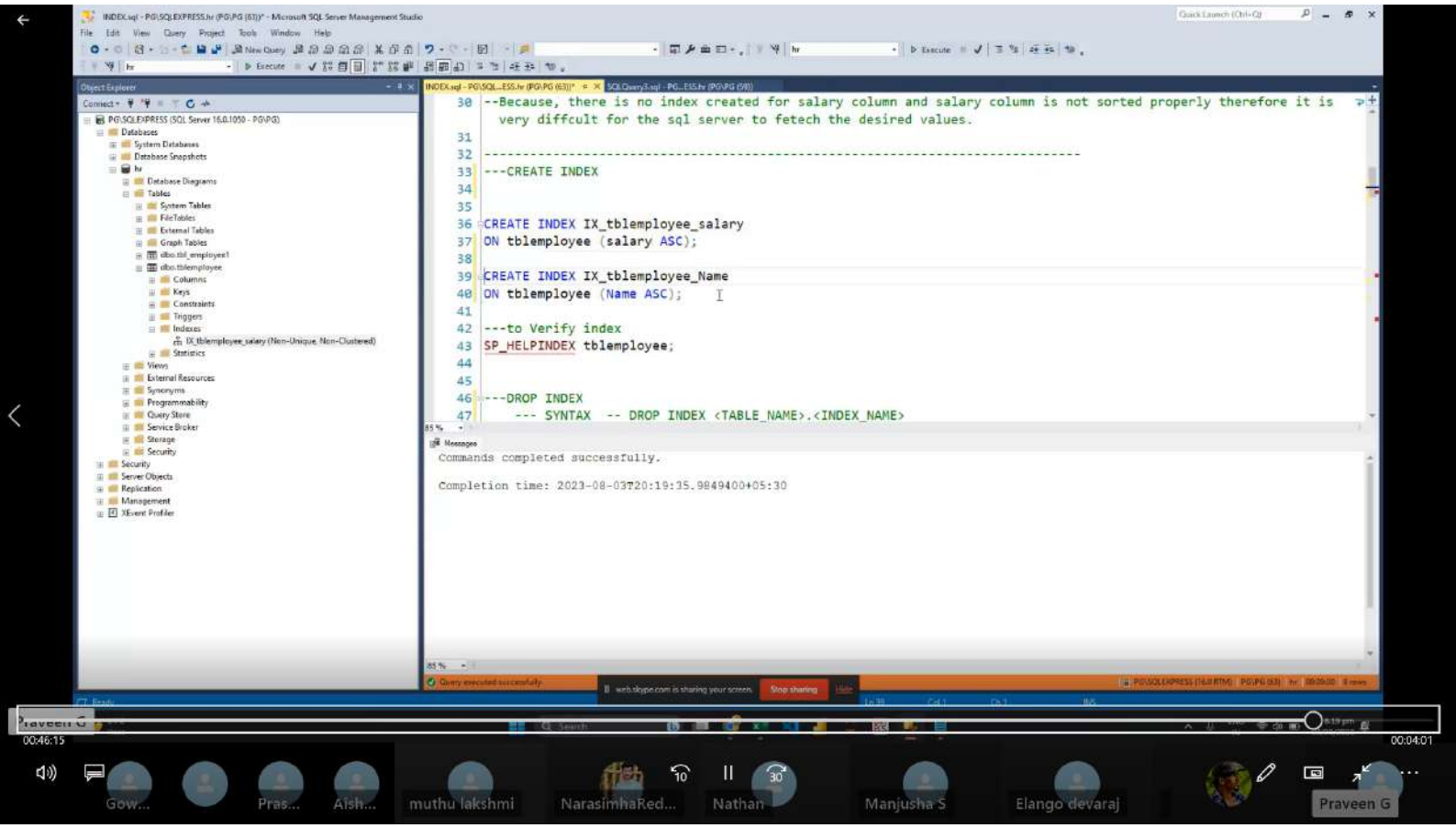
Video-20230803_145417-Meeting Recording

Praveen G

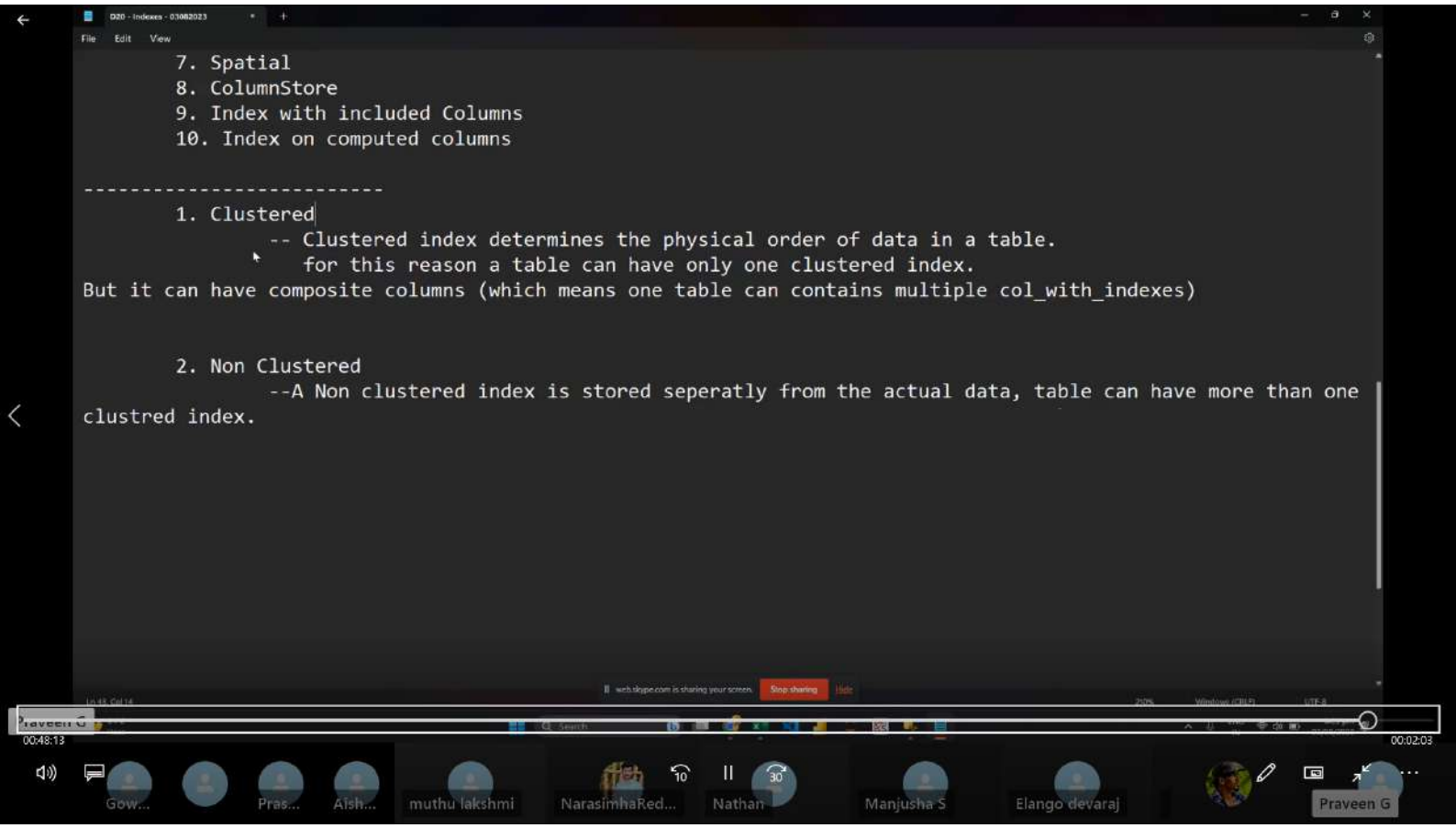
00:46:07

00:04:09

Gow... Pras... Aish... muthu lakshmi NarasimhaRed... Nathan Manjusha S Elango devaraj Praveen G



Praveen G
00:46:15
00:04:01
Gow...
Pras...
Aish...
muthu lakshmi
NarasimhaRed...
Nathan
Manjusha S
Elango devaraj
Praveen G


DZD - Indexes - 03082023
File Edit View
7. Spatial
8. ColumnStore
9. Index with included Columns
10. Index on computed columns
---
1. Clustered
-- Clustered index determines the physical order of data in a table.
for this reason a table can have only one clustered index.
But it can have composite columns (which means one table can contains multiple col_with_indexes)
2. Non Clustered
--A Non clustered index is stored seperatly from the actual data, table can have more than one clustred index.
web.skype.com is sharing your screen.
Stop sharing
Hide
Praveen G
00:48:13
00:02:03
Gow...
Pras...
Aish...
muthu lakshmi
NarasimhaRed...
Nathan
Manjusha S
Elango devaraj
Praveen G

*Untitled - Notepad

File Edit Format View Help

```
CREATE DATABASE hr;
USE hr;

--employee table
CREATE TABLE tblemployee (
        id int,
        Name varchar(50),
        salary money,
        gender varchar(10)
);

--Records for tblemployee table
INSERT INTO tblemployee VALUES (1, 'Sam',  2500, 'Male');
INSERT INTO tblemployee VALUES (2, 'Pam',  6500, 'Female');
INSERT INTO tblemployee VALUES (3, 'John', 4500, 'Male');
INSERT INTO tblemployee VALUES (4, 'Sara', 5500, 'Female');
INSERT INTO tblemployee VALUES (5, 'Todd', 3100, 'Male');


SELECT * FROM tblemployee;

--performing TABLE SCAN -- (without/before creating index)
--fetch employees whose salary in more than 5000 and less than 7000

SELECT * FROM tblemployee
WHERE salary > 5000 and salary < 7000;

--in this case, the SEARCH ENGINE is performing TABLE SCAN for each record in salary column and giving the output.
--Because, there is no index created for salary column and salary column is not sorted properly therefore it is very diffcult for the sql server to fetech the desired values.

-------------------------------------------------------------------------------
---CREATE INDEX


CREATE INDEX IX_tblemployee_salary
ON tblemployee (salary ASC);

CREATE INDEX IX_tblemployee_Name
ON tblemployee (Name ASC);

---to Verify index
SP_HELPINDEX tblemployee;
```

Ln 107, Col 28 | 100% | Windows (CRLF) | UTF-8

Type here to search

ENG 11:25 AM 04-08-2023

*Untitled - Notepad

File Edit Format View Help

```
---DROP INDEX
    --- SYNTAX  -- DROP INDEX <TABLE_NAME>.<INDEX_NAME>
DROP INDEX tblemployee.IX_tblemployee_salary;
--DROP INDEX tblemployee.IX_tblemployee_Name;




--------------------------------------------------------------------------------
                                    --CLUSTERED INDEX vs NON-CLUSTERED INDEX
--------------------------------------------------------------------------------
--Create a table with primary key

--employee1 table
CREATE TABLE tbl_employee1 (
        id int primary key,           ---- PK constraint is created along with the table
        Name varchar(50),
        salary money,
        gender varchar(10),
);

--Records for tbl_employee1 table

INSERT INTO tbl_employee1 VALUES (5, 'Todd', 3100, 'Male');
INSERT INTO tbl_employee1 VALUES (2, 'Pam',  6500, 'Female');
INSERT INTO tbl_employee1 VALUES (1, 'Sam',  2500, 'Male');
INSERT INTO tbl_employee1 VALUES (4, 'Sara', 5500, 'Female');
INSERT INTO tbl_employee1 VALUES (3, 'John', 4500, 'Male');

--DQL satatement
--the values inserted were not in order, but when we execute the DQL statement -- ID's were arranged in order because it contains PRIMARY KEY (which by default creates a CLUSTERED INDEX)
SELECT * FROM tbl_employee1


--to verify index name
EXECUTE sp_helpindex tbl_employee1          --- output : index keys --> id
                                                                    --- because primary key is created
```

Ln 107, Col 28 | 100% | Windows (CRLF) | UTF-8

Type here to search

ENG 11:25 AM 04-08-2023

*Untitled - Notepad

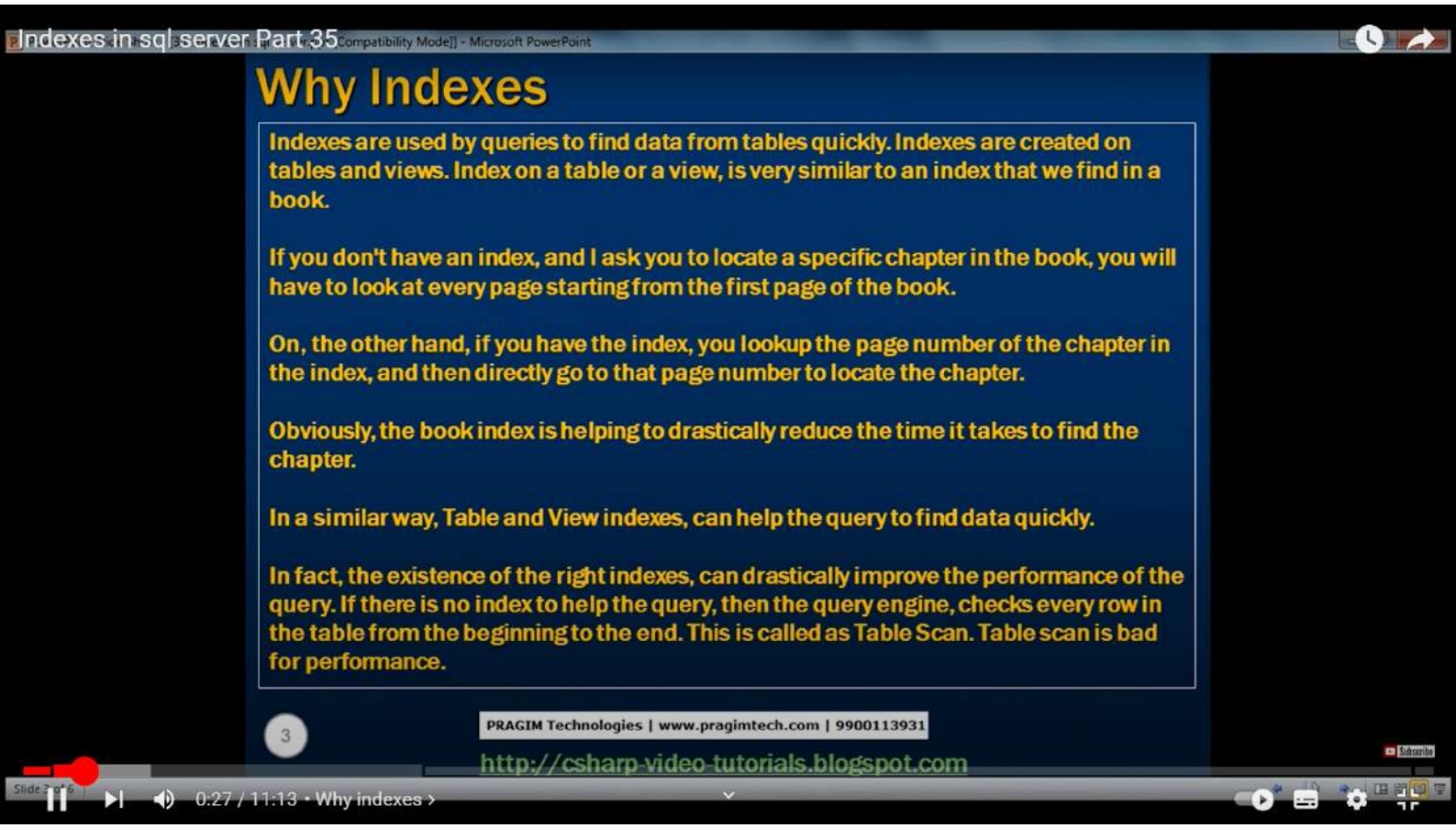
File Edit Format View Help

```
);

--Records for tbl_employee1 table

INSERT INTO tbl_employee1 VALUES (5, 'Todd', 3100, 'Male');
INSERT INTO tbl_employee1 VALUES (2, 'Pam',  6500, 'Female');
INSERT INTO tbl_employee1 VALUES (1, 'Sam',  2500, 'Male');
INSERT INTO tbl_employee1 VALUES (4, 'Sara', 5500, 'Female');
INSERT INTO tbl_employee1 VALUES (3, 'John', 4500, 'Male');

--DQL satatement
--the values inserted were not in order, but when we execute the DQL statement -- ID's were arranged in order because it contains PRIMARY KEY (which by default creates a CLUSTERED INDEX)
SELECT * FROM tbl_employee1


--to verify index name
EXECUTE sp_helpindex tbl_employee1          --- output : index keys --> id
                                                                                --- because primary key is created

--lets create a CLUSTERED INDEX
CREATE CLUSTERED INDEX IX_tbl_employee1_Gender_Salary
ON tbl_employee1 (Gender DESC, Salary ASC);

SELECT * FROM tbl_employee1
WHERE gender = 'Male'

                --output : will throw an error
                --because one table can contain only one CLUSTERED INDEX, but we can create COMPOSITE COLUMNS (means, one index can contain multiple columns)

--Now, lets DROP the existing INDEX:
DROP INDEX tbl_employee1.IX_tbl_employee1_Gender_Salary;

                --output : will throw an error
                --because we cannot drop it explicitly, because it is used as primary key. But, we can drop in using GUI.
                -- Go to TABLE INDEX in object explorer ---> INDEXES ---> right click and DELETE --> OK

--NOW, lets create a COMPOSITE COLUMNS -- CLUSTERED INDEX on multiple columns
CREATE CLUSTERED INDEX IX_tbl_employee1_Gender_Salary
ON tbl_employee1 (Gender DESC, Salary ASC);

--DQL statement
SELECT * FROM tbl_employee1
```

Ln 107, Col 28 | 100% | Windows (CRLF) | UTF-8

Type here to search

ENG 11:25 AM 04-08-2023


Indexes in sql server Part 35
Compatibility Mode]] - Microsoft PowerPoint
Why Indexes
Indexes are used by queries to find data from tables quickly. Indexes are created on tables and views. Index on a table or a view, is very similar to an index that we find in a book.
If you don't have an index, and I ask you to locate a specific chapter in the book, you will have to look at every page starting from the first page of the book.
On, the other hand, if you have the index, you lookup the page number of the chapter in the index, and then directly go to that page number to locate the chapter.
Obviously, the book index is helping to drastically reduce the time it takes to find the chapter.
In a similar way, Table and View indexes, can help the query to find data quickly.
In fact, the existence of the right indexes, can drastically improve the performance of the query. If there is no index to help the query, then the query engine, checks every row in the table from the beginning to the end. This is called as Table Scan. Table scan is bad for performance.
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:27 / 11:13 • Why indexes ›

Indexes in sql server Part 35

Creating an Index

```
CREATE Index IX_tblEmployee_Salary
ON tblEmployee (SALARY ASC)
```

The index stores salary of each employee, in the ascending order as shown below. The actual index may look slightly different.

Id	Name	Salary	Gender
1	Sam	2500	Male
2	Pam	6500	Female
3	John	4500	Male
4	Sara	5500	Female
5	Todd	3100	Male

Salary	RowAddress
2500	Row Address
3100	Row Address
4500	Row Address
5500	Row Address
6500	Row Address

Now, when the SQL server has to execute the same query, it has an index on the salary column to help this query. Salaries between the range of 5000 and 7000 are usually present at the bottom, since the salaries are arranged in an ascending order. SQL server picks up the row addresses from the index ords from the table, rather than scanning each row in the table. This is ca

Next Vide exes, Disadvantage etc...

Create Index

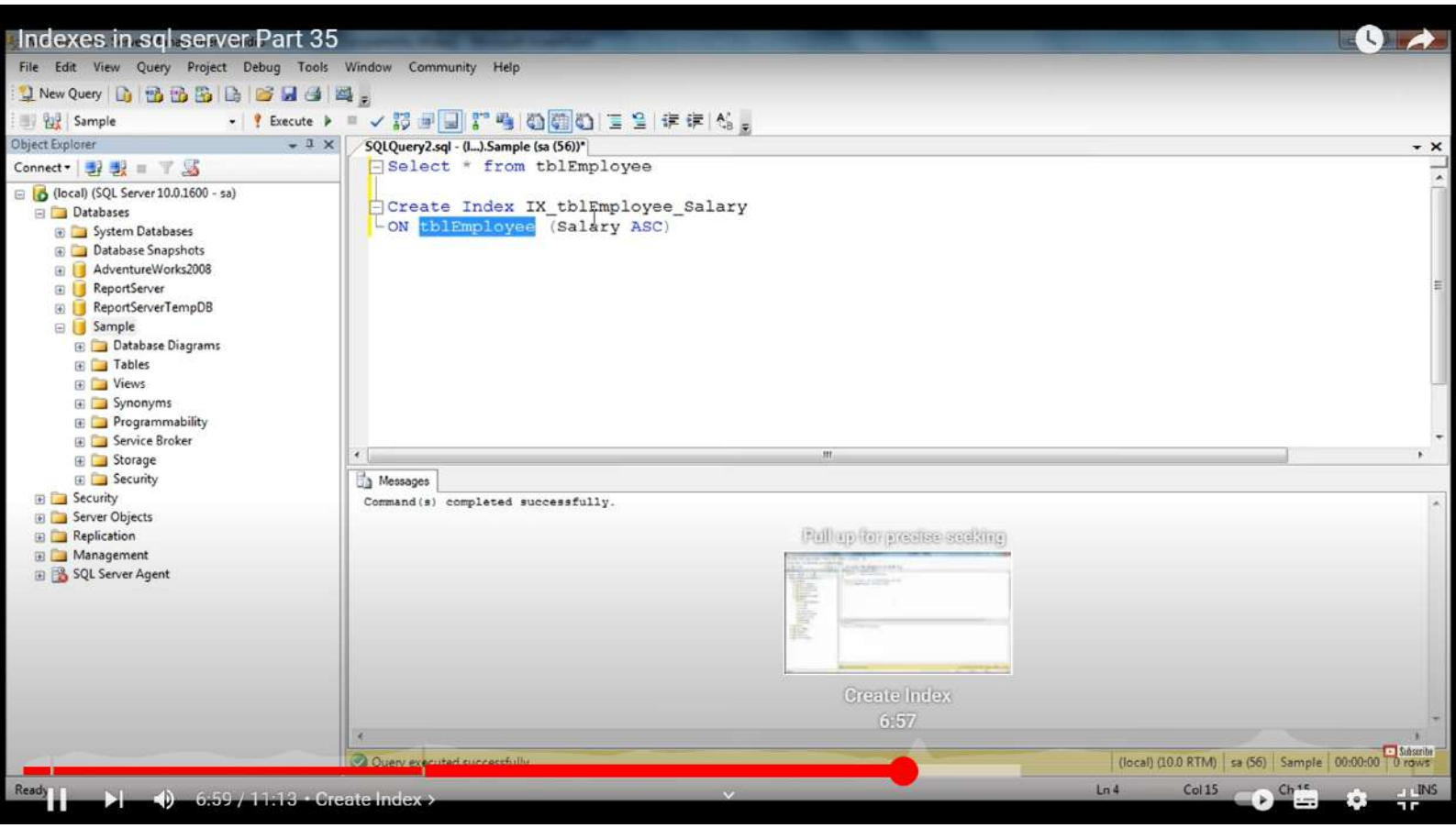
3:3

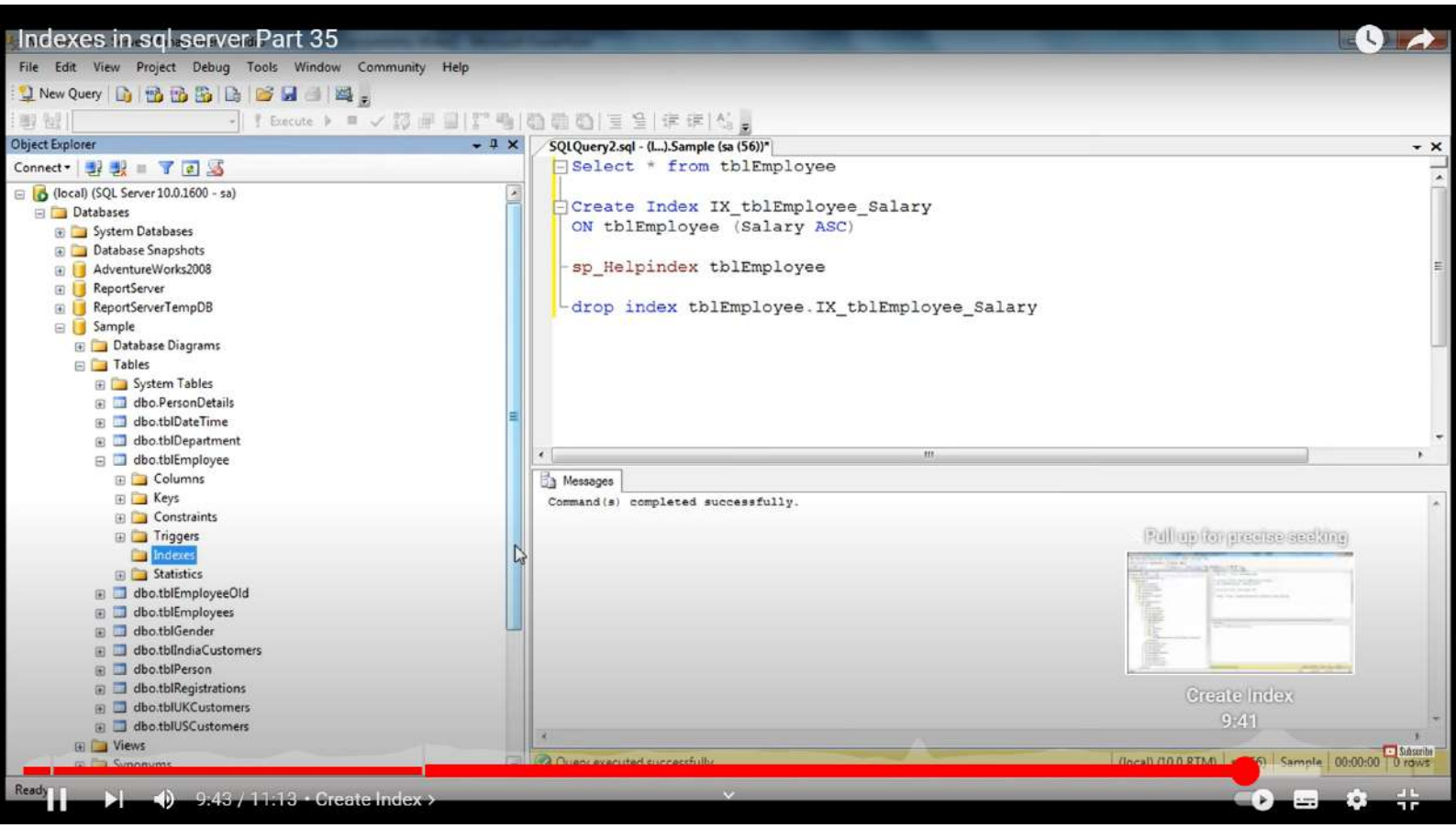
PRAGIM Technologies | www.pragimtech.com | 9900113931

5

http://csharp-video-tutorials.blogspot.com

3:19 / 11:13 • Create Index >


Indexes in sql server Part 35
File Edit View Query Project Debug Tools Window Community Help
New Query
Sample
Execute
Object Explorer
Connect
(local) (SQL Server 10.0.1600 - sa)
Databases
System Databases
Database Snapshots
AdventureWorks2008
ReportServer
ReportServerTempDB
Sample
Database Diagrams
Tables
Views
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
Management
SQL Server Agent
SQLQuery2.sql - (l...).Sample (sa (56))*
Select * from tblEmployee
Create Index IX_tblEmployee_Salary
ON tblEmployee (Salary ASC)
Messages
Command(s) completed successfully.
Pull up for precise seeking
Create Index
6:57
Query executed successfully.
(local) (10.0 RTM) sa (56) Sample 00:00:00 0 rows
Ready
Ln 4 Col 15
INS
6:59 / 11:13 • Create Index >


Indexes in sql server Part 35
File Edit View Project Debug Tools Window Community Help
New Query
Execute
Object Explorer
Connect
(local) (SQL Server 10.0.1600 - sa)
Databases
System Databases
Database Snapshots
AdventureWorks2008
ReportServer
ReportServerTempDB
Sample
Database Diagrams
Tables
System Tables
dbo.PersonDetails
dbo.tblDateTime
dbo.tblDepartment
dbo.tblEmployee
Columns
Keys
Constraints
Triggers
Indexes
Statistics
dbo.tblEmployeeOld
dbo.tblEmployees
dbo.tblGender
dbo.tblIndiaCustomers
dbo.tblPerson
dbo.tblRegistrations
dbo.tblUKCustomers
dbo.tblUSCustomers
Views
SQLQuery2.sql - (l...).Sample (sa (56))*
Select * from tblEmployee
Create Index IX_tblEmployee_Salary
ON tblEmployee (Salary ASC)
sp_Helpindex tblEmployee
drop index tblEmployee.IX_tblEmployee_Salary
Messages
Command(s) completed successfully.
Pull up for precise seeking
Create Index
9:41
Ready
9:43 / 11:13 • Create Index ›

Clustered and nonclustered indexes in sql server Part 36

Clustered Index

A clustered index determines the physical order of data in a table. For this reason, a table can have only one clustered index.

```
CREATE TABLE [tblEmployee]
(
  [Id] int Primary Key,
  [Name] nvarchar(50),
  [Salary] int,
  [Gender] nvarchar(10),
  [City] nvarchar(50)
)
```

Note that Id column is marked as primary key. Primary key, constraint create clustered indexes automatically if no clustered index already exists on the table.

To confirm: `Execute sp_helpindex tblEmployee`

Note that, the values for Id column are not in a sequential order

```
Insert into tblEmployee Values(3,'John',4500,'Male','New York')
Insert into tblEmployee Values(1,'Sam',2500,'Male','London')
Insert into tblEmployee Values(4,'Sara',5500,'Female','Tokyo')
Insert into tblEmployee Values(5,'Todd',3100,'Male','Toronto')
Insert into tblEmployee Values(2,'Pam',6500,'Female','Sydney')
```

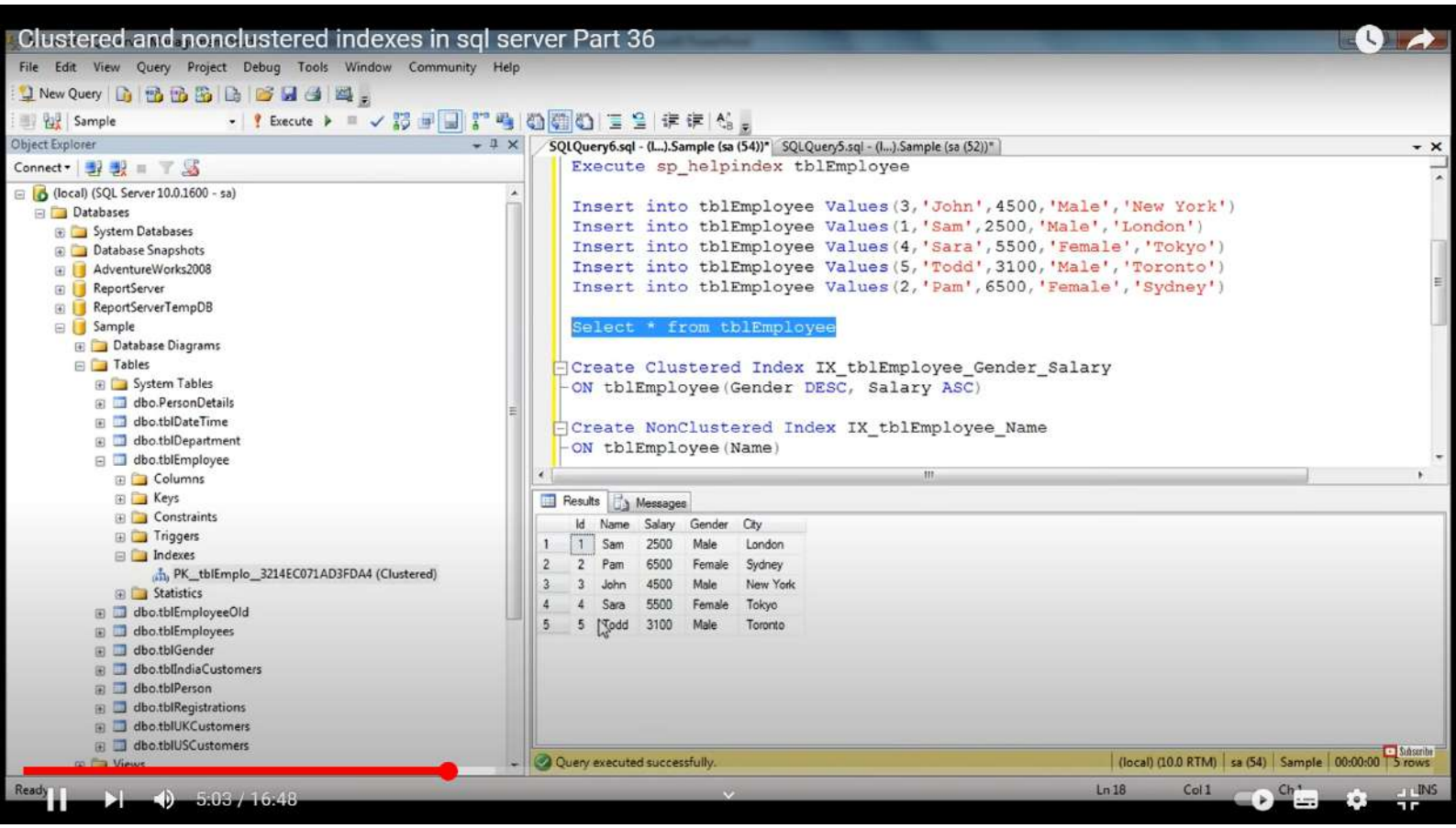
```
Select * from tblEmployee
```

Id	Name	Salary	Gender	City
1	Sam	2500	Male	London
2	Pam	6500	Female	Sydney
3	John	4500	Male	New York
4	Sara	5500	Female	Tokyo
5	Todd	3100	Male	Toronto

PRAGIM Technologies | www.pragimtech.com | 9900113931

http://csharp-video-tutorials.blogspot.com

1:12 / 16:48


Clustered and nonclustered indexes in sql server Part 36
File Edit View Query Project Debug Tools Window Community Help
New Query
Sample
Execute
Object Explorer
Connect
(local) (SQL Server 10.0.1600 - sa)
Databases
System Databases
Database Snapshots
AdventureWorks2008
ReportServer
ReportServerTempDB
Sample
Database Diagrams
Tables
System Tables
dbo.PersonDetails
dbo.tblDateTime
dbo.tblDepartment
dbo.tblEmployee
Columns
Keys
Constraints
Triggers
Indexes
PK__tblEmplo__3214EC071AD3FDA4 (Clustered)
Statistics
dbo.tblEmployeeOld
dbo.tblEmployees
dbo.tblGender
dbo.tblIndiaCustomers
dbo.tblPerson
dbo.tblRegistrations
dbo.tblUKCustomers
dbo.tblUSCustomers
SQLQuery6.sql - (l...).Sample (sa (54))*
SQLQuery5.sql - (l...).Sample (sa (52))*
Execute sp_helpindex tblEmployee
Insert into tblEmployee Values(3,'John',4500,'Male','New York')
Insert into tblEmployee Values(1,'Sam',2500,'Male','London')
Insert into tblEmployee Values(4,'Sara',5500,'Female','Tokyo')
Insert into tblEmployee Values(5,'Todd',3100,'Male','Toronto')
Insert into tblEmployee Values(2,'Pam',6500,'Female','Sydney')
Select * from tblEmployee
Create Clustered Index IX_tblEmployee_Gender_Salary
ON tblEmployee(Gender DESC, Salary ASC)
Create NonClustered Index IX_tblEmployee_Name
ON tblEmployee(Name)
Results
Messages
Id Name Salary Gender City
1 1 Sam 2500 Male London
2 2 Pam 6500 Female Sydney
3 3 John 4500 Male New York
4 4 Sara 5500 Female Tokyo
5 5 Todd 3100 Male Toronto
Query executed successfully.
(local) (10.0 RTM) sa (54) Sample 00:00:00 5 rows
Ready
Ln 18 Col 1 Ch 1 INS
5:03 / 16:48

Clustered and nonclustered indexes in sql server Part 36

NonClustered Index

```
Create NonClustered Index IX_tblEmployee_Name
ON tblEmployee(Name)
```

Id	Name	Salary	Gender	City
1	Sam	2500	Male	London
5	Todd	3100	Male	Toronto
3	John	4500	Male	New York
4	Sara	5500	Female	Tokyo
2	Pam	6500	Female	Sydney

Name	Row Address
John	Row Address
Pam	Row Address
Sam	Row Address
Sara	Row Address
Todd	Row Address

A nonclustered index is analogous to an index in a textbook. The data is stored in one place, the index in another place. The index will have pointers to the storage location of the data.

Since, the nonclustered index is stored separately from the actual data, a table can have more than one non clustered index, just like how a book can have an index by Chapters at the beginning and another index by common terms at the end.

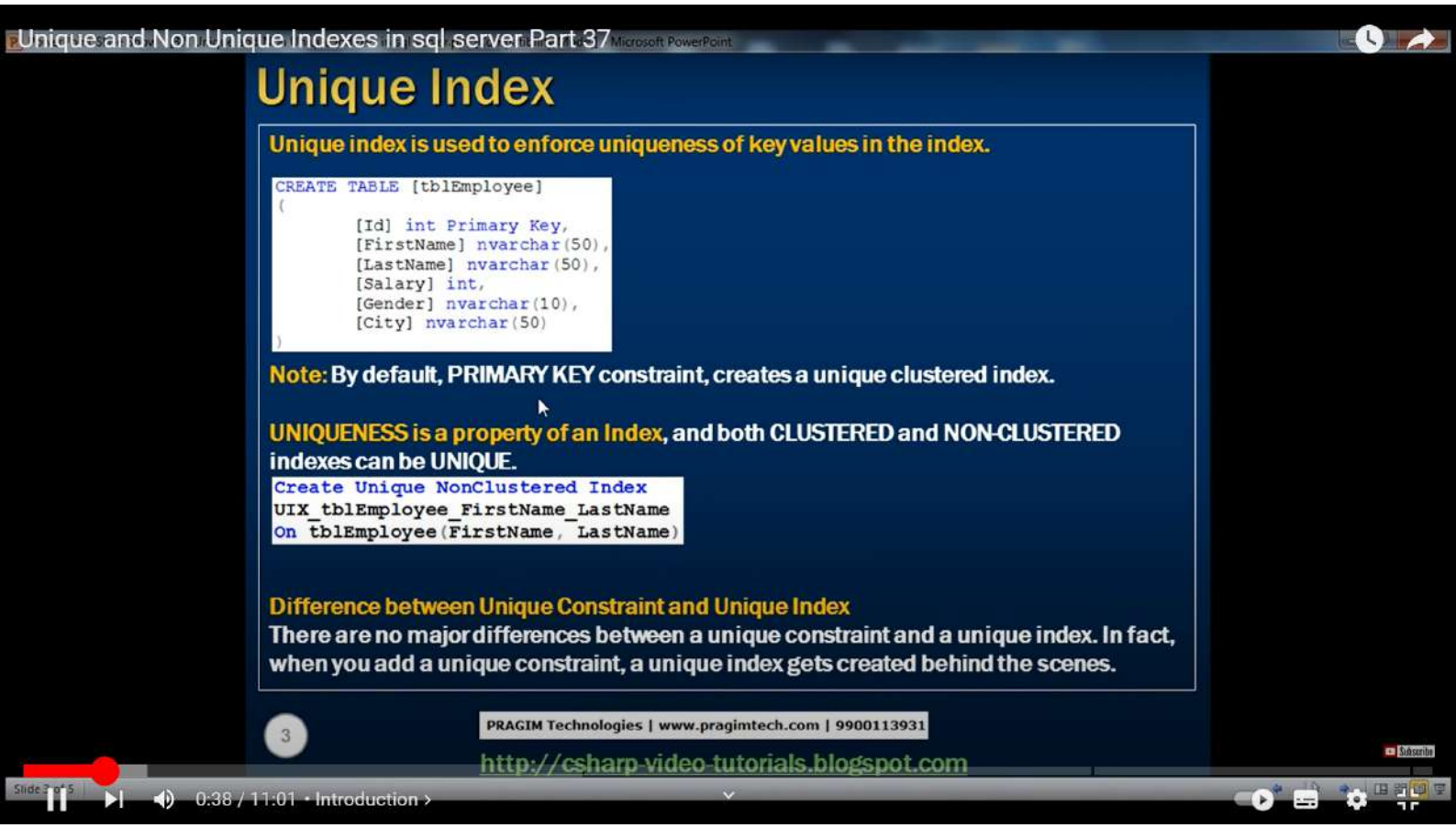
In the index itself, the data is stored in an ascending or descending order of the index key, which doesn't in any way influence the storage of data in t

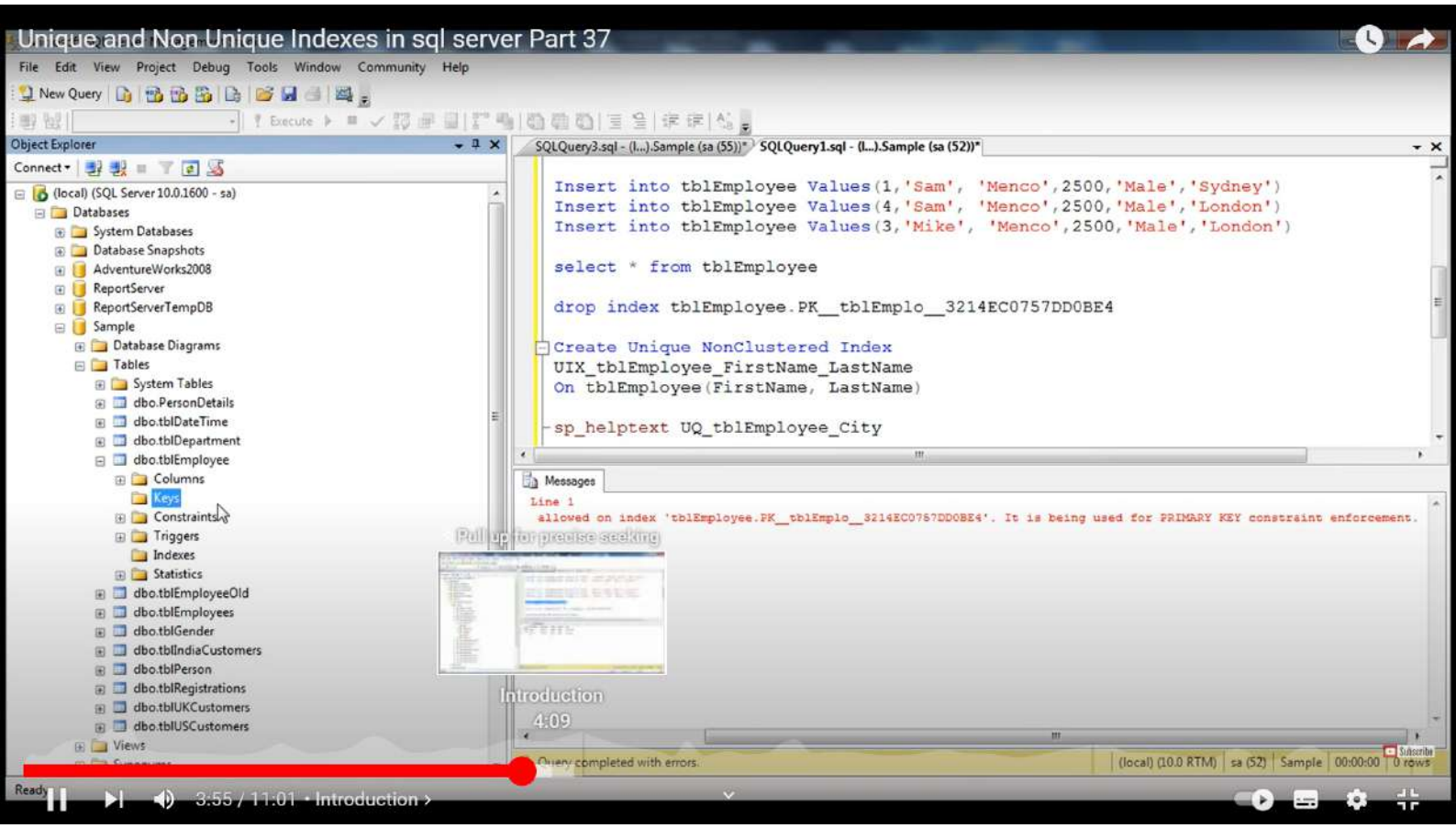
PRAGIM Technologies | www.pragimtech.com | 9246113931

6

http://csharp-video-tutorials.blogspot.com

10:55 / 16:48


Unique and Non Unique Indexes in sql server Part 37
Microsoft PowerPoint
Unique Index
Unique index is used to enforce uniqueness of key values in the index.
CREATE TABLE [tblEmployee]
(
[Id] int Primary Key,
[FirstName] nvarchar(50),
[LastName] nvarchar(50),
[Salary] int,
[Gender] nvarchar(10),
[City] nvarchar(50)
)
Note: By default, PRIMARY KEY constraint, creates a unique clustered index.
UNIQUENESS is a property of an Index, and both CLUSTERED and NON-CLUSTERED indexes can be UNIQUE.
Create Unique NonClustered Index
UIX_tblEmployee_FirstName_LastName
On tblEmployee(FirstName, LastName)
Difference between Unique Constraint and Unique Index
There are no major differences between a unique constraint and a unique index. In fact, when you add a unique constraint, a unique index gets created behind the scenes.
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:38 / 11:01 • Introduction ›


Unique and Non Unique Indexes in sql server Part 37
File Edit View Project Debug Tools Window Community Help
New Query
Object Explorer
Connect
(local) (SQL Server 10.0.1600 - sa)
Databases
System Databases
Database Snapshots
AdventureWorks2008
ReportServer
ReportServerTempDB
Sample
Database Diagrams
Tables
System Tables
dbo.PersonDetails
dbo.tblDateTime
dbo.tblDepartment
dbo.tblEmployee
Columns
Keys
Constraints
Triggers
Indexes
Statistics
dbo.tblEmployeeOld
dbo.tblEmployees
dbo.tblGender
dbo.tblIndiaCustomers
dbo.tblPerson
dbo.tblRegistrations
dbo.tblUKCustomers
dbo.tblUSCustomers
Views
SQLQuery3.sql - (l...).Sample (sa (55))*
SQLQuery1.sql - (l...).Sample (sa (52))*
Insert into tblEmployee Values(1,'Sam', 'Menco',2500,'Male','Sydney')
Insert into tblEmployee Values(4,'Sam', 'Menco',2500,'Male','London')
Insert into tblEmployee Values(3,'Mike', 'Menco',2500,'Male','London')
select * from tblEmployee
drop index tblEmployee.PK__tblEmplo__3214EC0757DD0BE4
Create Unique NonClustered Index
UIX_tblEmployee_FirstName_LastName
On tblEmployee(FirstName, LastName)
sp_helptext UQ_tblEmployee_City
Messages
Line 1
allowed on index 'tblEmployee.PK__tblEmplo__3214EC0757DD0BE4'. It is being used for PRIMARY KEY constraint enforcement.
Pull up for precise seeking
Introduction
4:09
Query completed with errors.
(local) (10.0 RTM) sa (52) Sample 00:00:00 0 rows
Ready
3:55 / 11:01 • Introduction ›

Unique Index

When should you be creating a Unique constraint over a unique index?
To make our intentions clear, create a unique constraint, when data integrity is the objective. This makes the objective of the index very clear. In either cases, data is validated in the same manner, and the query optimizer does not differentiate between a unique index created by a unique constraint or manually created.

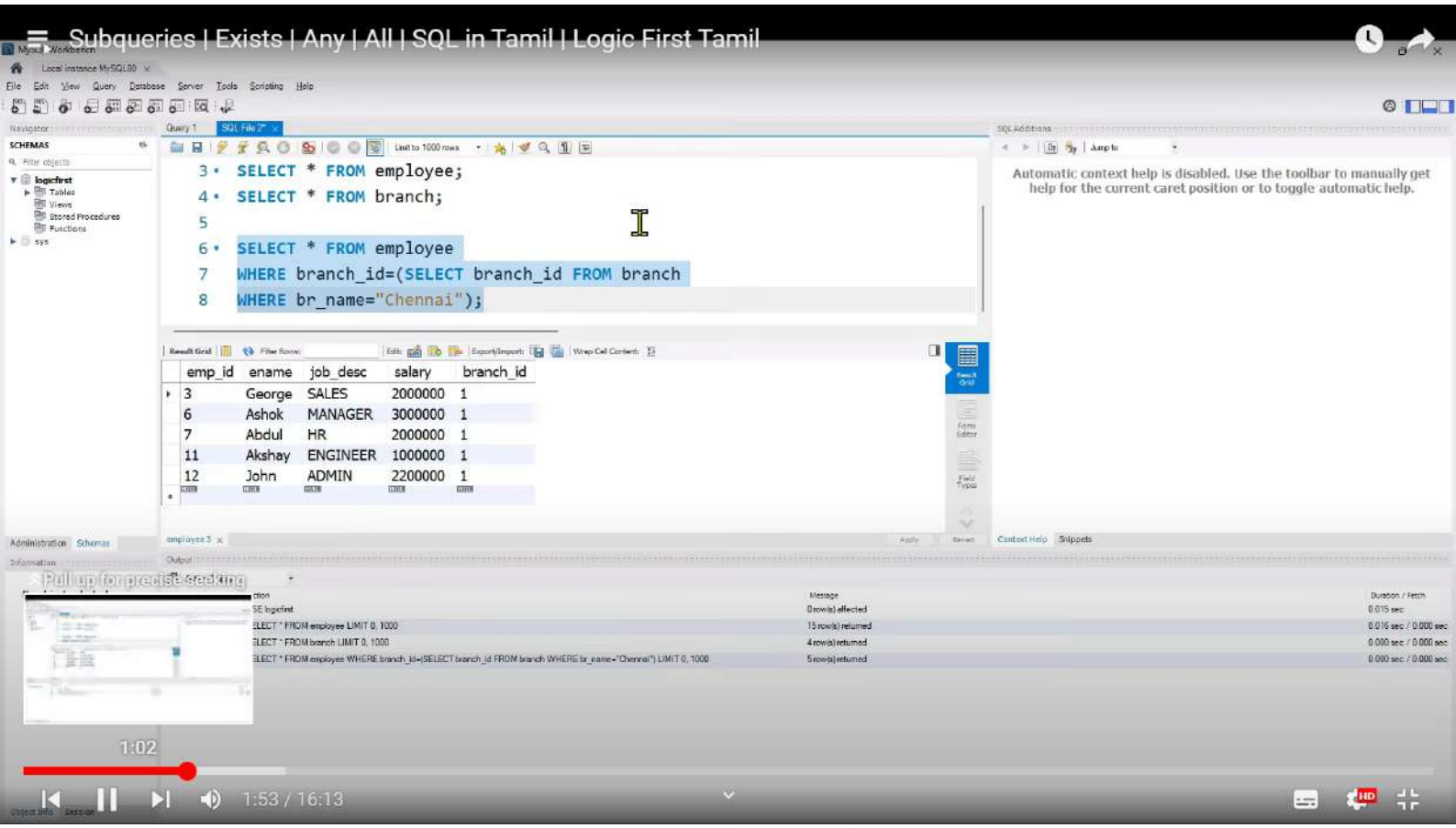
Useful points to remember:
1. By default, a PRIMARY KEY constraint, creates a unique clustered index, where as a UNIQUE constraint creates a unique Non-Clustered index. These defaults can be changed if you wish to.

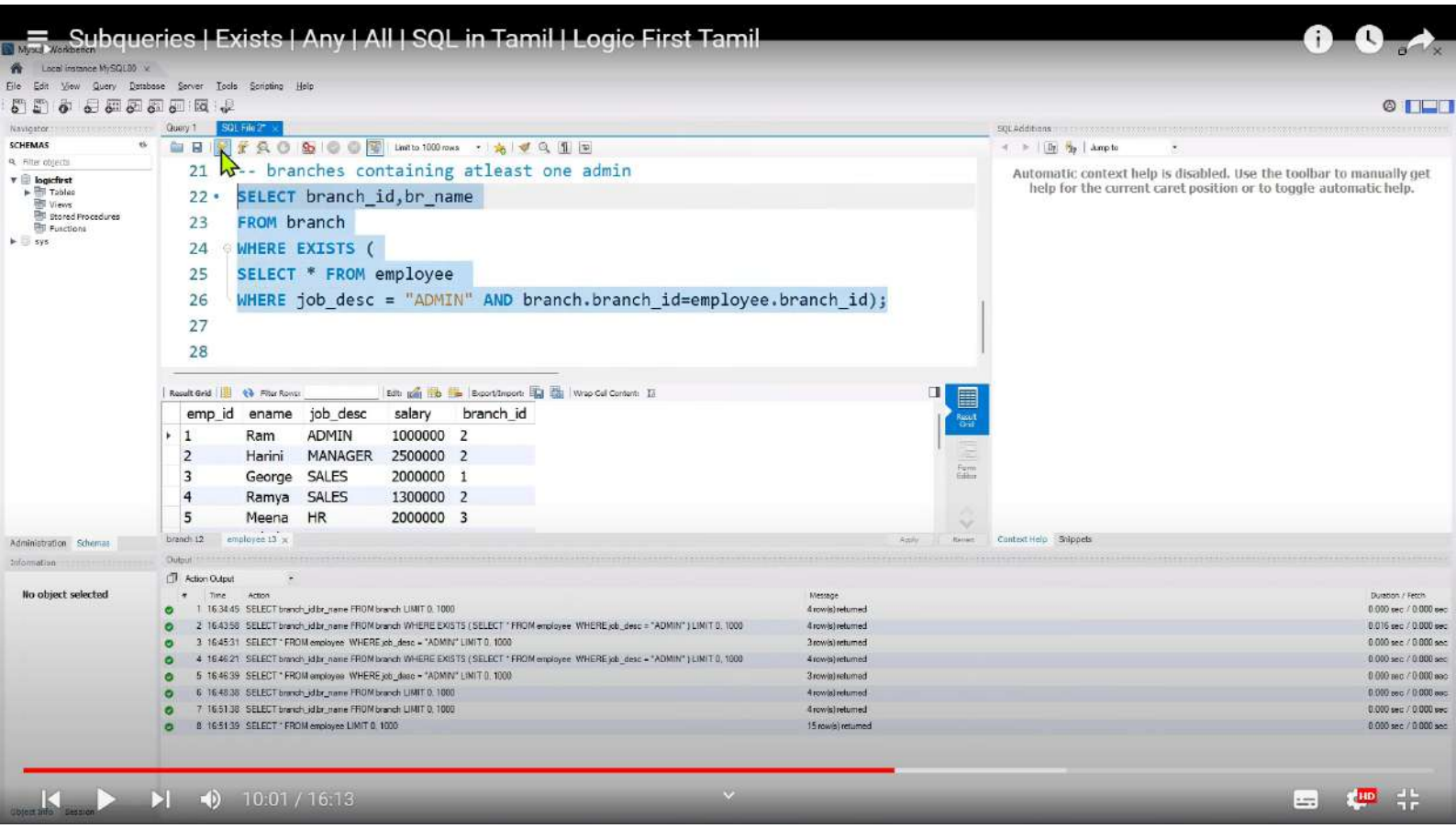
2. A UNIQUE constraint or a UNIQUE index cannot be created on an existing table, if the table contains duplicate values in the key columns. Obviously, to solve this, remove the key columns from the index definition or delete or update the duplicate values.

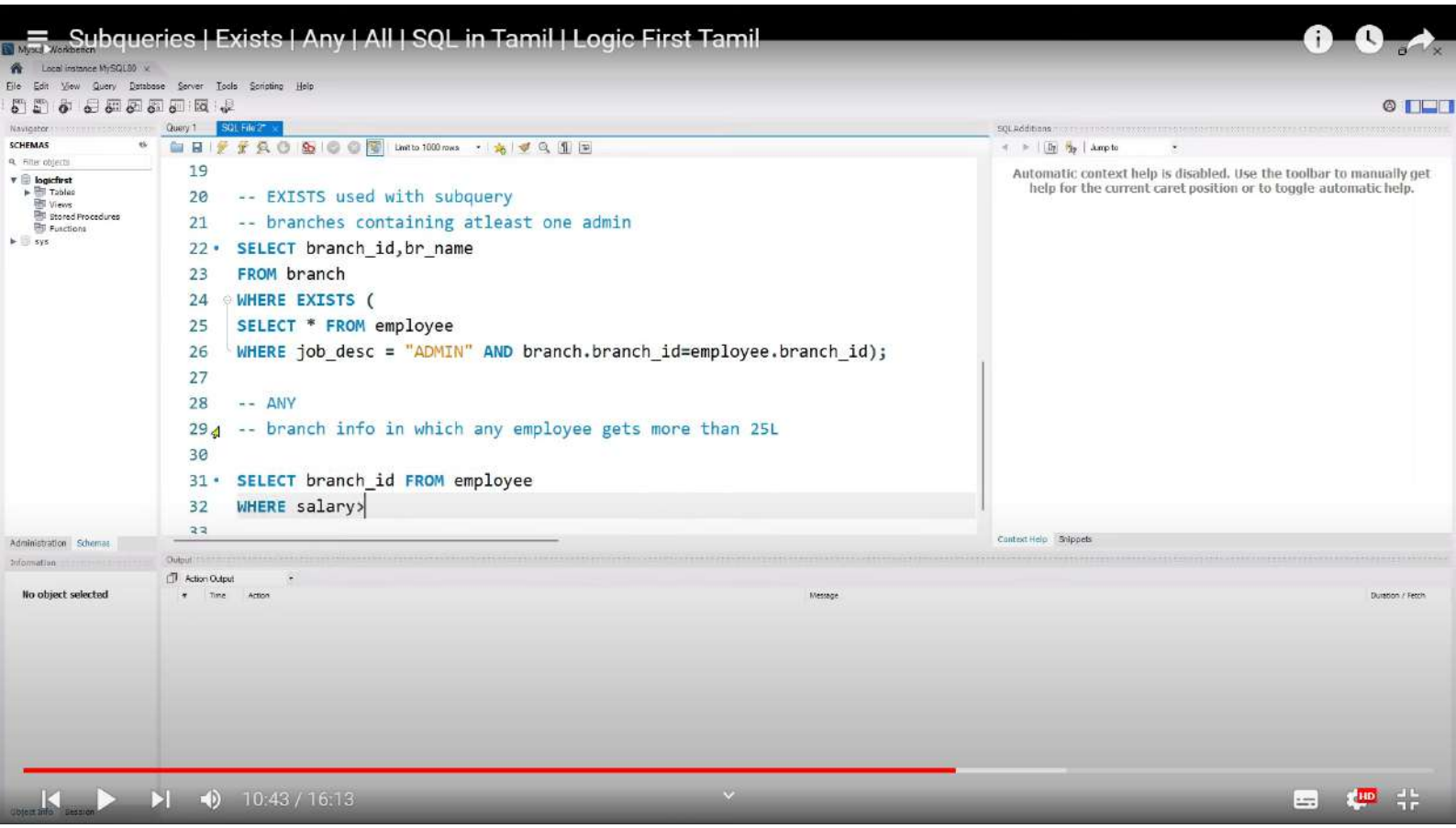
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

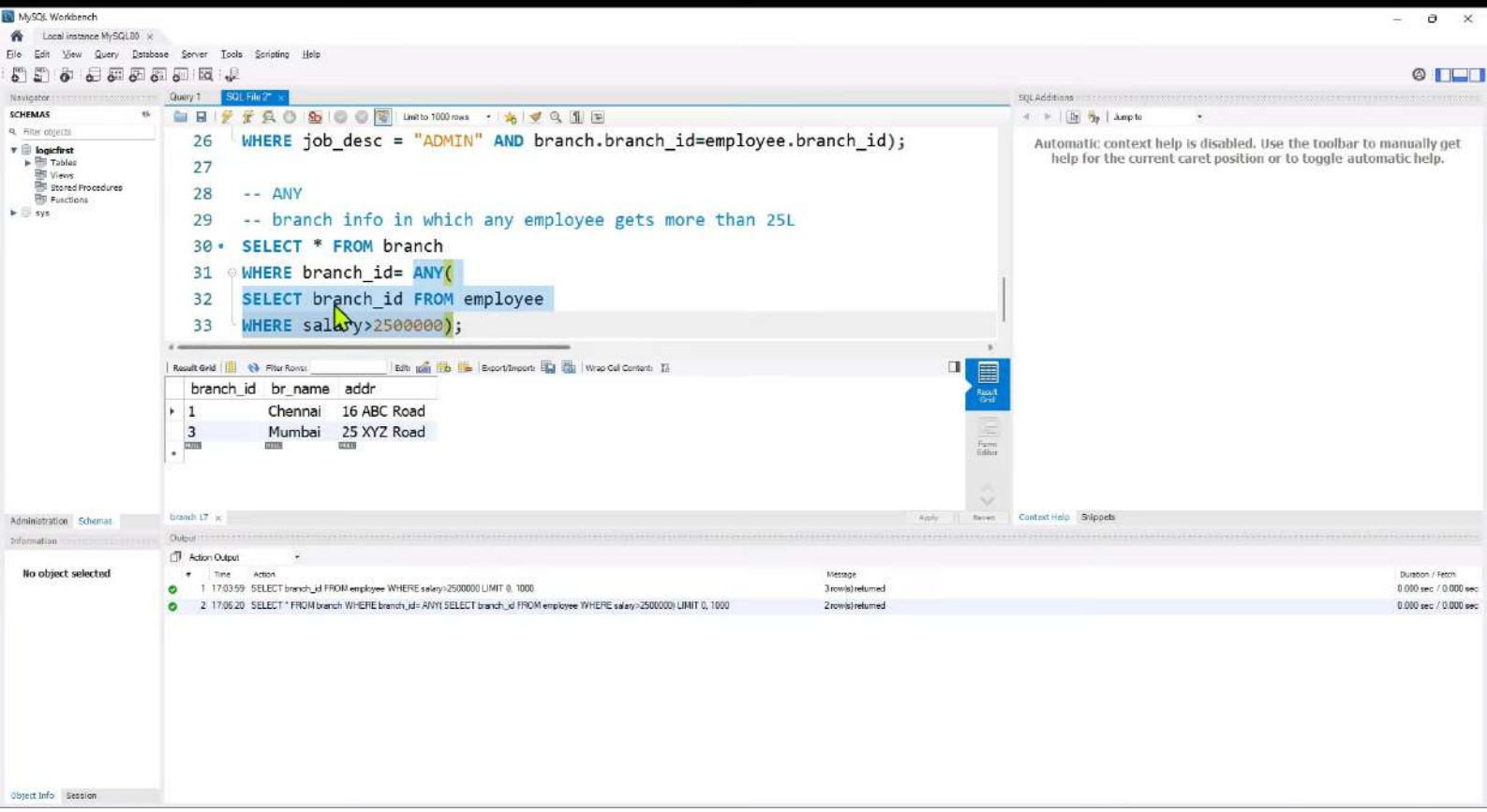
```
SUBQUERY
---------

-- A query inside an another query is called as Sub/Nested Query.

Synx :

      SELECT * FROM <Table_name> [WHERE condition] (SELECT * FROM.....(SELECT *
FROM.....))

> as per the execution process of sub query it is classified into two types :
      1. Non - Corelated Sub Query (NCSQ)
            -- First inner query is executed and later outer query is executed

            1.1 Simple/Single row sub query
            -----------------------------------
                  -- when a sub query returns a single value is called
simple/Single row sub query

---------------------------------------
SUBQUERY = OUTER + INNER
---------------------------------------
Step 1 : inner query

      SELECT max(empsalary) from emp -------> 75000

Step 2 : outer query

      SELECT * FROM emp
      WHERE empsalary = ???

----correct method
SELECT max(empsalary) from emp -------> 75000

SELECT * FROM emp
      WHERE empsalary = --???

SELECT * FROM emp WHERE empsalary = max(empsalary); --- this will not work
SELECT * FROM emp WHERE empsalary = (INNER QUERY)

--non corelated sq
SELECT * FROM emp WHERE empsalary = (SELECT max(empsalary) from emp)


            1.2 Multiple row sub query
            -----------------------------------
                  -- when a sub query returns a multiples value is called Multiple
row sub query

-- to display details from the table whose dept is same as the dept of the employee

-- SCOTT and SMITH

--STEP 1 : INNER QUERY
SELECT dpt_name FROM emp2
WHERE ename = 'Smith' or ename = 'Scott';

-- now, we found the value of dept of SCOTT and SMITH
```

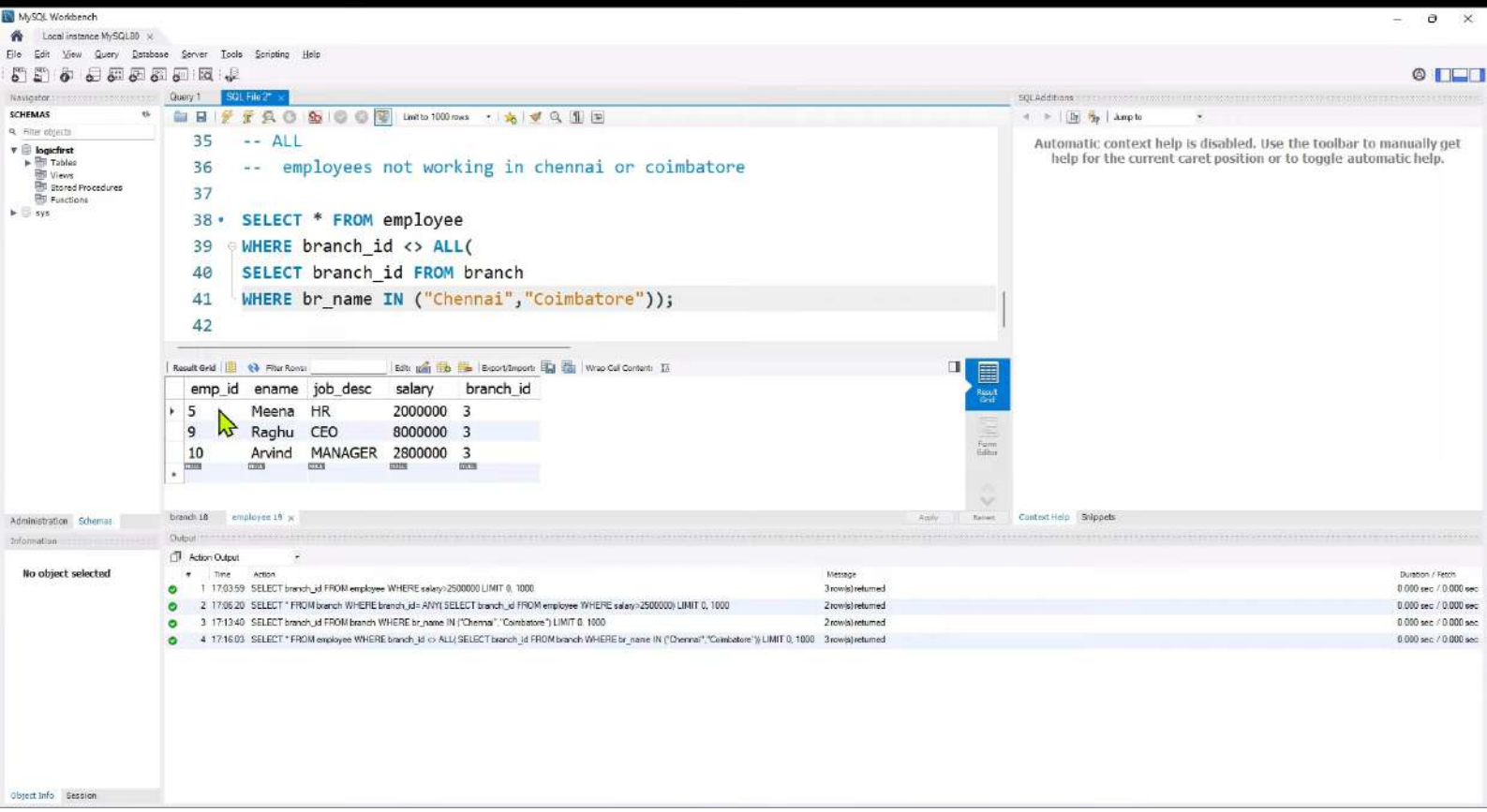
```
--STEP 2 : OUTER QUERY
SELECT * FROM emp2
WHERE dpt_name = ??? --- (INNER QUERY)

--STEP 3 = step 2 + Step 1


SELECT * FROM emp2
WHERE dpt_name IN (SELECT dpt_name FROM emp2 WHERE ename = 'Smith' or ename =
'Scott');

SELECT * FROM emp2
WHERE dpt_name IN (SELECT dpt_name FROM emp2 WHERE ename IN ('Scott', 'Smith'));


================================================================================
============================================================


2. Corelated SubQuery
            -- First outer query is executed and later inner query is executed.




----- CORELATED SQ

            -- ALIAS NAME

/*


SELECT * FROM <Table_name> as <tn_alias>
WHERE N - 1 = (SELECT count(<col_name>) FROM <TN> as <tn_alias>
WHERE <table alias name2>.<col_name> (</>)<table alias name>.<col_name>)


Find the 1st highest salary [n=1]

*/

SELECT count(salary) FROM emp2 as E2  -------> 8

SELECT * FROM emp2 as E1
WHERE (n-1) = (SELECT count(salary) FROM emp2 as E2 WHERE e2.salary > e1.salary)
------Nth salary : replace the value by subtracting
```


Subqueries | Exists | Any | All | SQL in Tamil | Logic First Tamil
SELECT * FROM employee;
SELECT * FROM branch;
SELECT * FROM employee
WHERE branch_id=(SELECT branch_id FROM branch
WHERE br_name="Chennai");
emp_id ename job_desc salary branch_id
3 George SALES 2000000 1
6 Ashok MANAGER 3000000 1
7 Abdul HR 2000000 1
11 Akshay ENGINEER 1000000 1
12 John ADMIN 2200000 1
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Pull up for precise seeking
1:02
1:53 / 16:13


Subqueries | Exists | Any | All | SQL in Tamil | Logic First Tamil
-- branches containing atleast one admin
SELECT branch_id,br_name
FROM branch
WHERE EXISTS (
SELECT * FROM employee
WHERE job_desc = "ADMIN" AND branch.branch_id=employee.branch_id);
emp_id ename job_desc salary branch_id
1 Ram ADMIN 1000000 2
2 Harini MANAGER 2500000 2
3 George SALES 2000000 1
4 Ramya SALES 1300000 2
5 Meena HR 2000000 3
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
No object selected
10:01 / 16:13


Subqueries | Exists | Any | All | SQL in Tamil | Logic First Tamil
-- EXISTS used with subquery
-- branches containing atleast one admin
SELECT branch_id,br_name
FROM branch
WHERE EXISTS (
SELECT * FROM employee
WHERE job_desc = "ADMIN" AND branch.branch_id=employee.branch_id);
-- ANY
-- branch info in which any employee gets more than 25L
SELECT branch_id FROM employee
WHERE salary>
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
No object selected
10:43 / 16:13


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
Functions
sys
Query 1
SQL File 2*
Limit to 1000 rows
26 WHERE job_desc = "ADMIN" AND branch.branch_id=employee.branch_id);
27
28 -- ANY
29 -- branch info in which any employee gets more than 25L
30 SELECT * FROM branch
31 WHERE branch_id= ANY(
32 SELECT branch_id FROM employee
33 WHERE salary>2500000);
Result Grid
branch_id br_name addr
1 Chennai 16 ABC Road
3 Mumbai 25 XYZ Road
SQL Additions
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Administration Schemas
Information
No object selected
Object Info Session
Output
Action Output
Time Action Message Duration / Fetch
1 17:03:59 SELECT branch_id FROM employee WHERE salary>2500000 LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
2 17:06:20 SELECT * FROM branch WHERE branch_id= ANY(SELECT branch_id FROM employee WHERE salary>2500000) LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
Functions
sys
Query 1
SQL File 2*
Limit to 1000 rows
35 -- ALL
36 -- employees not working in chennai or coimbatore
37
38 • SELECT * FROM employee
39 WHERE branch_id <> ALL(
40 SELECT branch_id FROM branch
41 WHERE br_name IN ("Chennai","Coimbatore"));
42
Result Grid
emp_id ename job_desc salary branch_id
5 Meena HR 2000000 3
9 Raghu CEO 8000000 3
10 Arvind MANAGER 2800000 3
SQL Additions
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Administration Schemas
Information
No object selected
Object Info Session
Output
Action Output
1 17:03:59 SELECT branch_id FROM employee WHERE salary>2500000 LIMIT 0, 1000 3 row(s) returned
2 17:06:20 SELECT * FROM branch WHERE branch_id= ANY(SELECT branch_id FROM employee WHERE salary>2500000) LIMIT 0, 1000 2 row(s) returned
3 17:13:40 SELECT branch_id FROM branch WHERE br_name IN ("Chennai", "Coimbatore") LIMIT 0, 1000 2 row(s) returned
4 17:16:03 SELECT * FROM employee WHERE branch_id <> ALL(SELECT branch_id FROM branch WHERE br_name IN ("Chennai","Coimbatore")) LIMIT 0, 1000 3 row(s) returned

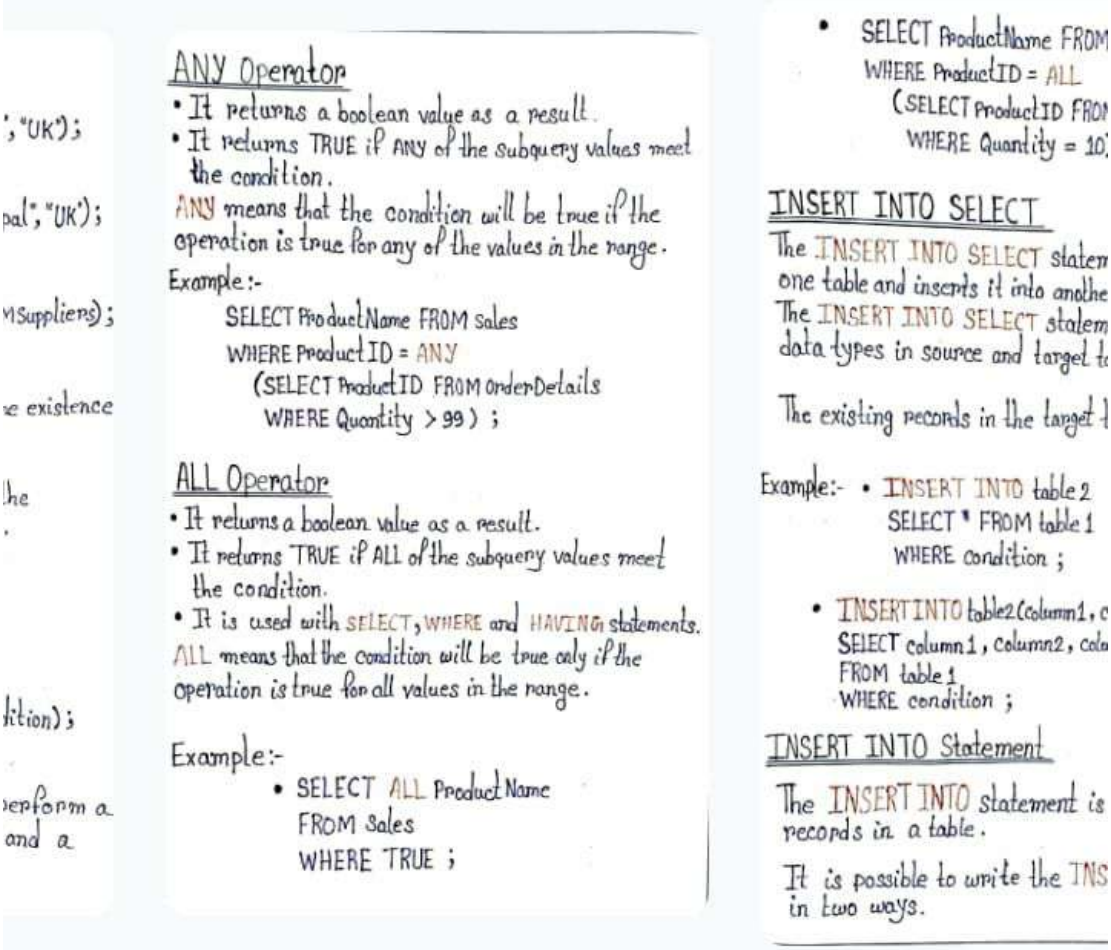
Muthukarthiga G likes this

Yuvraj Solanki · 3rd+
1M+ Impressions 🚀 | CSE Undergrad |...
18h · 🌐

+ Follow

Structured query language (SQL) is a programming language for storing and processing information in a relational database. ... see more

🎯**SQL Notes** · 24 pages



';"UK");
pal","UK");
vSuppliers);
ce existence
the
hition);
perform a
and a

ANY Operator

- It returns a boolean value as a result.
- It returns TRUE if ANY of the subquery values meet the condition.

ANY means that the condition will be true if the operation is true for any of the values in the range.

Example:-

```
SELECT ProductName FROM Sales
WHERE ProductID = ANY
  (SELECT ProductID FROM OrderDetails
   WHERE Quantity > 99) ;
```

ALL Operator

- It returns a boolean value as a result.
- It returns TRUE if ALL of the subquery values meet the condition.
- It is used with SELECT, WHERE and HAVING statements.

ALL means that the condition will be true only if the operation is true for all values in the range.

Example:-

```
• SELECT ALL ProductName
  FROM Sales
  WHERE TRUE ;
```

```
• SELECT ProductName FROM
  WHERE ProductID = ALL
    (SELECT ProductID FRO
     WHERE Quantity = 10
```

INSERT INTO SELECT

The INSERT INTO SELECT statem
one table and inserts it into anothe
The INSERT INTO SELECT statem
data types in source and target t

The existing records in the target t

Example:-

```
• INSERT INTO table2
  SELECT * FROM table1
  WHERE condition ;

• INSERT INTO table2 (column1, c
  SELECT column1, column2, colu
  FROM table1
  WHERE condition ;
```

INSERT INTO Statement

The INSERT INTO statement is
records in a table.

It is possible to write the INS
in two ways.

251 64 comments • 14 reposts

Like Comment Repost Send

LinkedIn
Promoted

Update your job preferences to let recruiters know what roles you're interested in. **https://lnkd.in/ewS3knh**

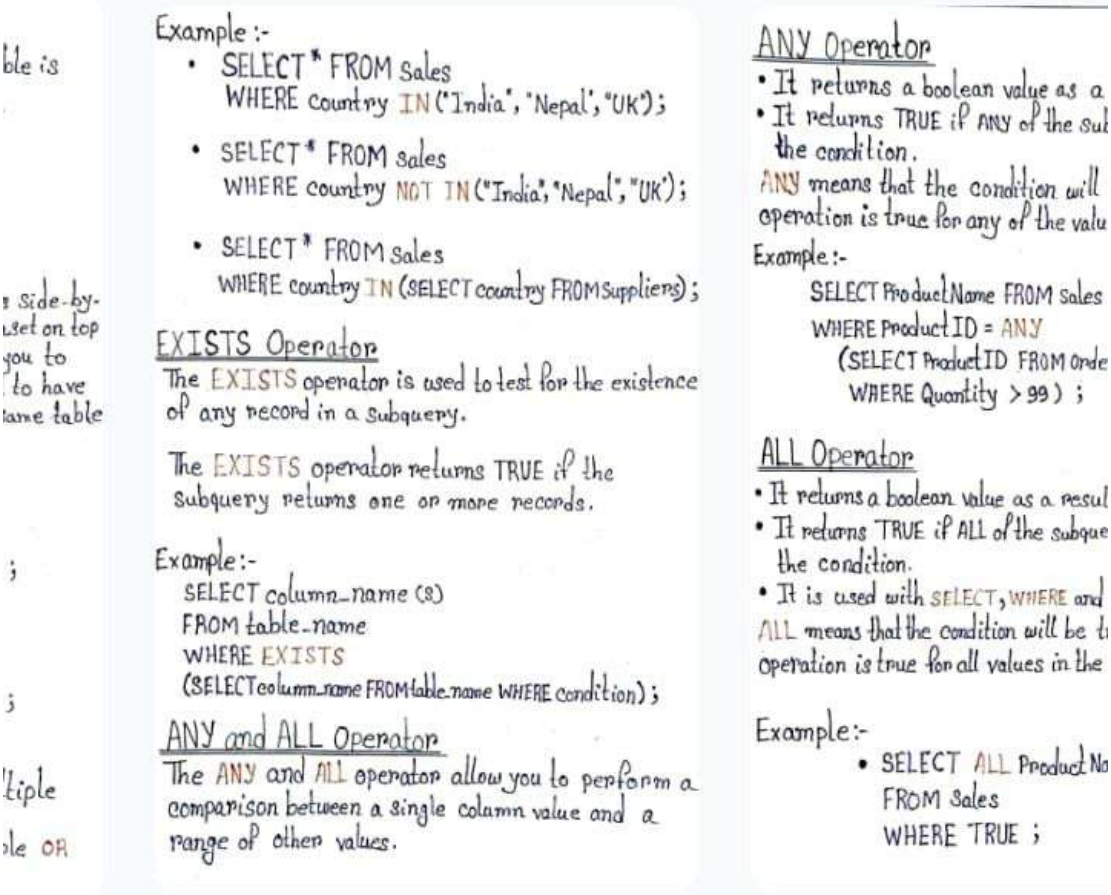
Muthukarthiga G likes this

Yuvraj Solanki · 3rd+
1M+ Impressions 🚀 | CSE Undergrad |...
18h · 🌐

+ Follow

Structured query language (SQL) is a programming language for storing and processing information in a relational database. ... see more

🎯**SQL Notes** · 24 pages



Example :-
- SELECT * FROM Sales
  WHERE country IN ("India", "Nepal", "UK");
- SELECT * FROM sales
  WHERE country NOT IN ("India", "Nepal", "UK");
- SELECT * FROM Sales
  WHERE country IN (SELECT country FROM Suppliers);

EXISTS Operator
The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns TRUE if the subquery returns one or more records.

Example:-
SELECT column_name (s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);

ANY and ALL Operator
The ANY and ALL operator allow you to perform a comparison between a single column value and a range of other values.

ANY Operator
- It returns a boolean value as a
- It returns TRUE if ANY of the sub
the condition.

ANY means that the condition will
operation is true for any of the valu

Example:-
SELECT ProductName FROM Sales
WHERE ProductID = ANY
(SELECT ProductID FROM Orde
WHERE Quantity > 99) ;

ALL Operator
- It returns a boolean value as a resul
- It returns TRUE if ALL of the subque
the condition.
- It is used with SELECT, WHERE and

ALL means that the condition will be t
operation is true for all values in the

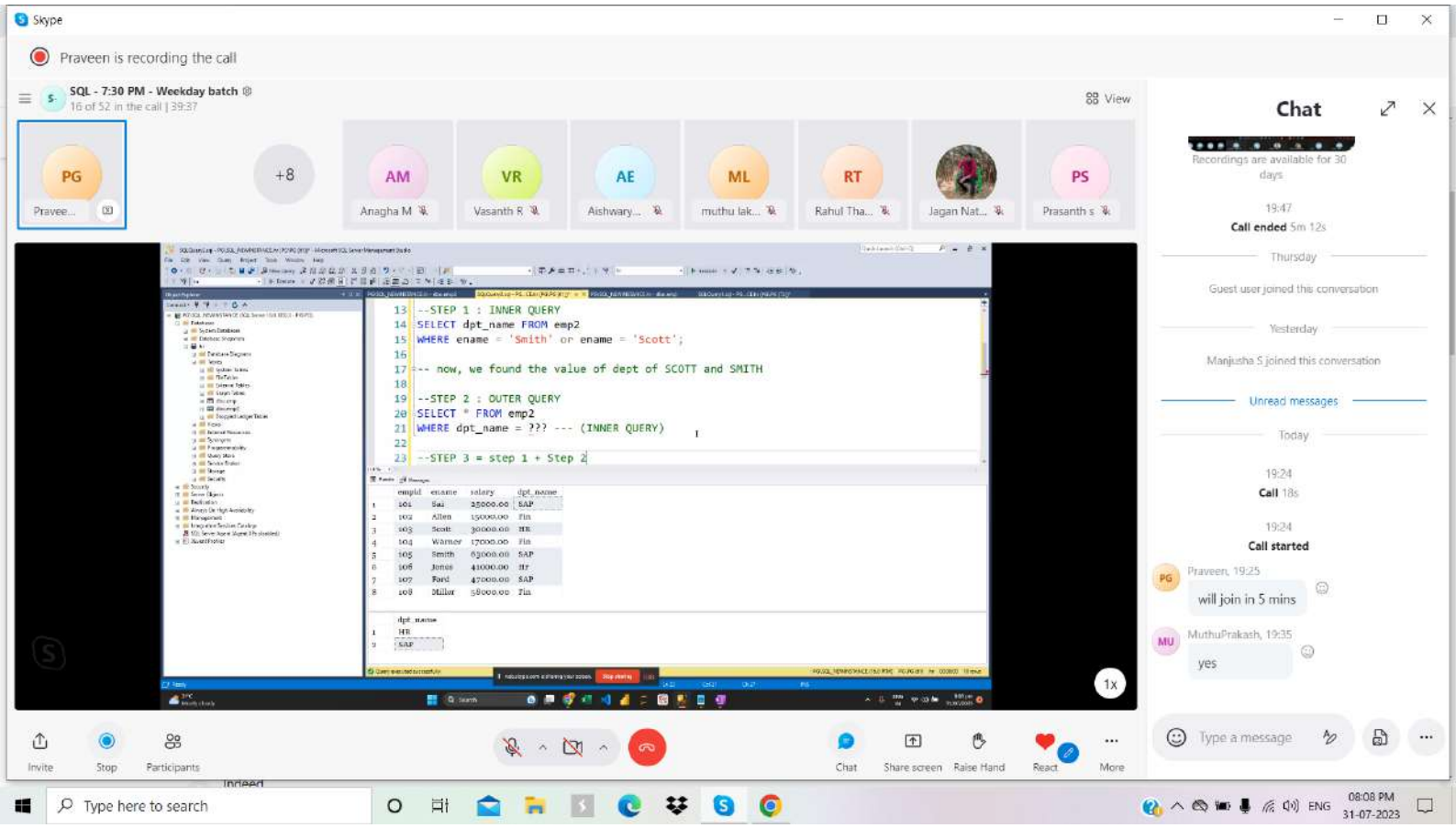
Example:-
- SELECT ALL ProductNa
FROM Sales
WHERE TRUE ;

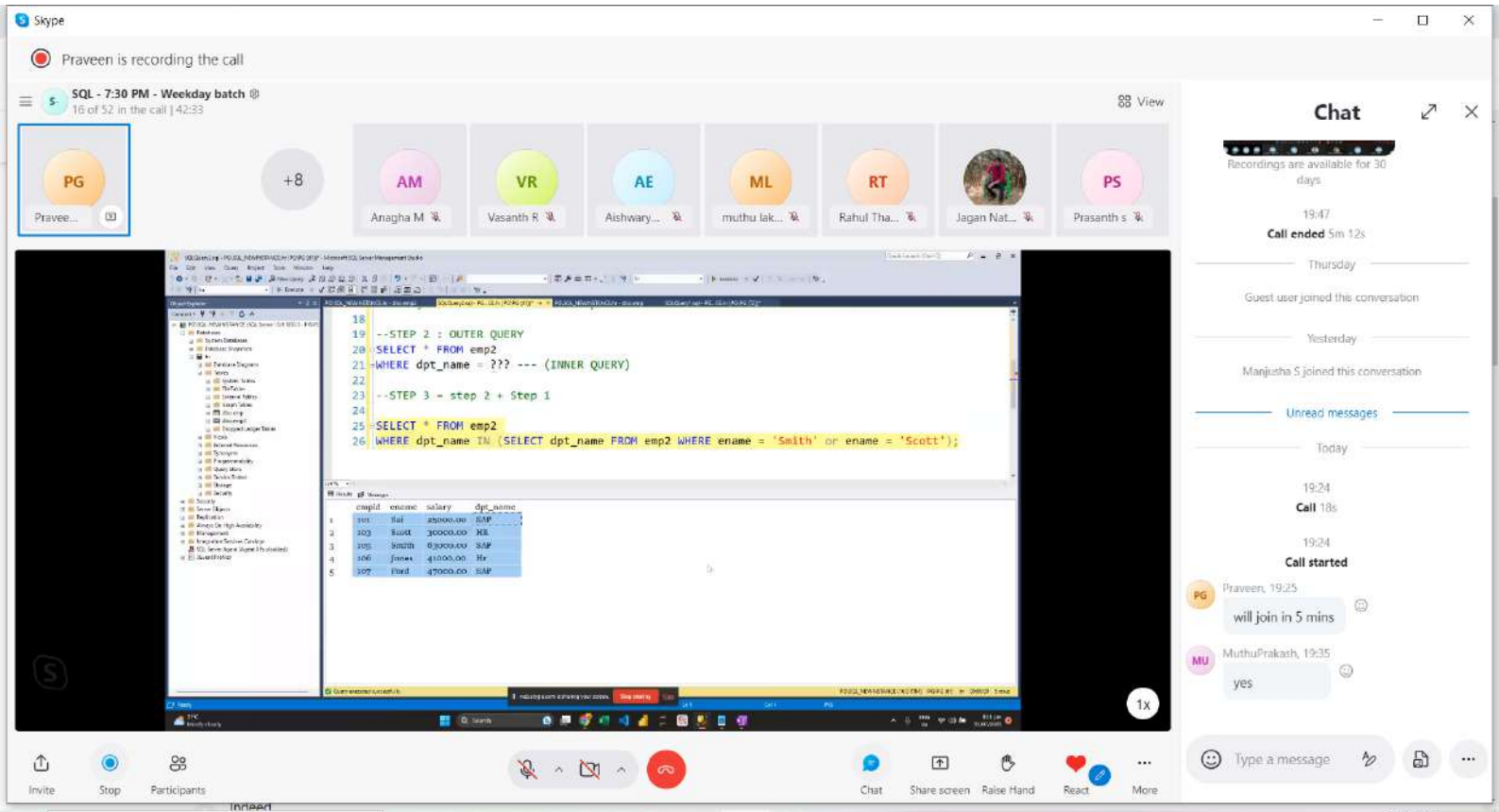
251 64 comments · 14 reposts

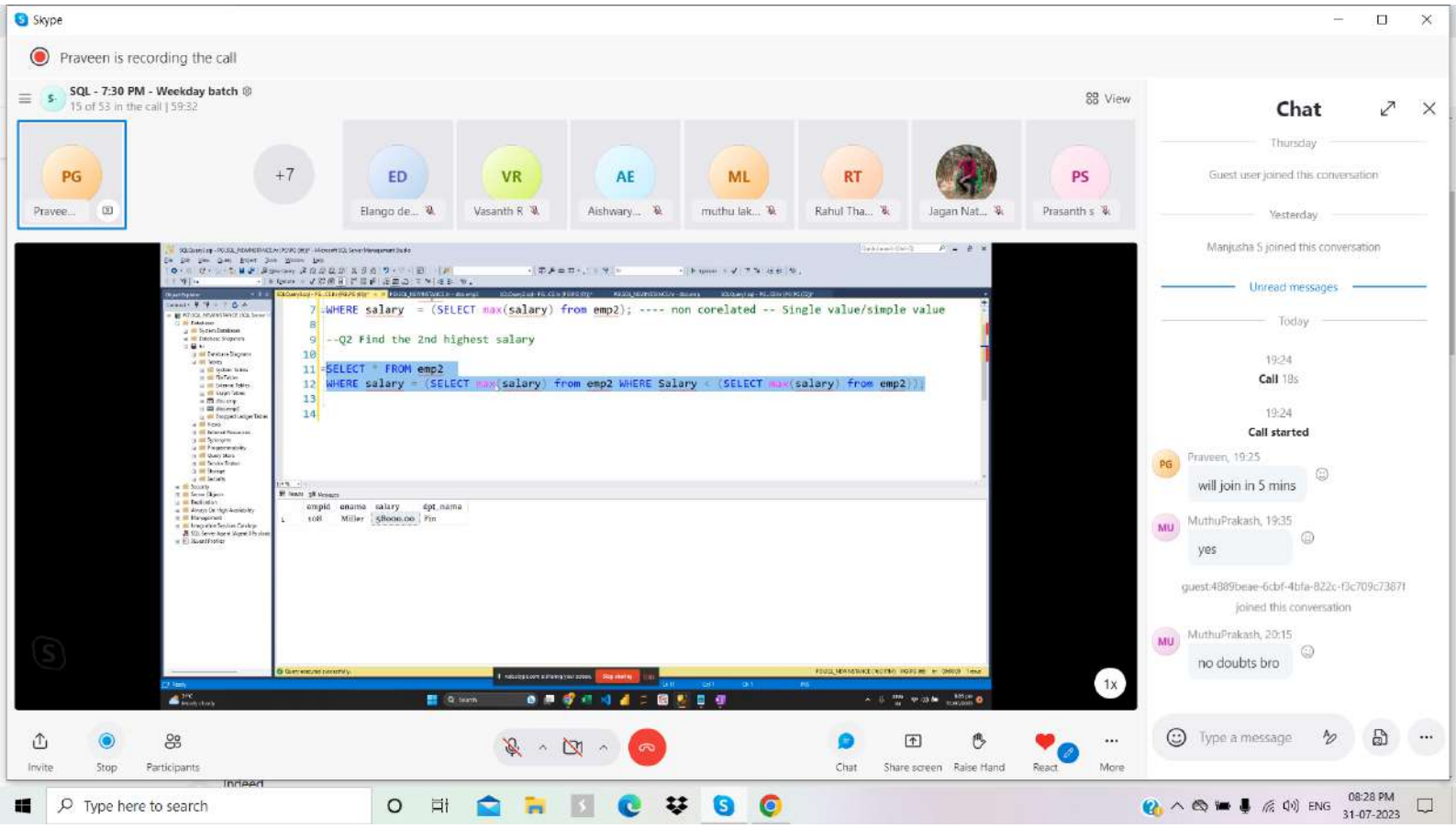
Like Comment Repost Send

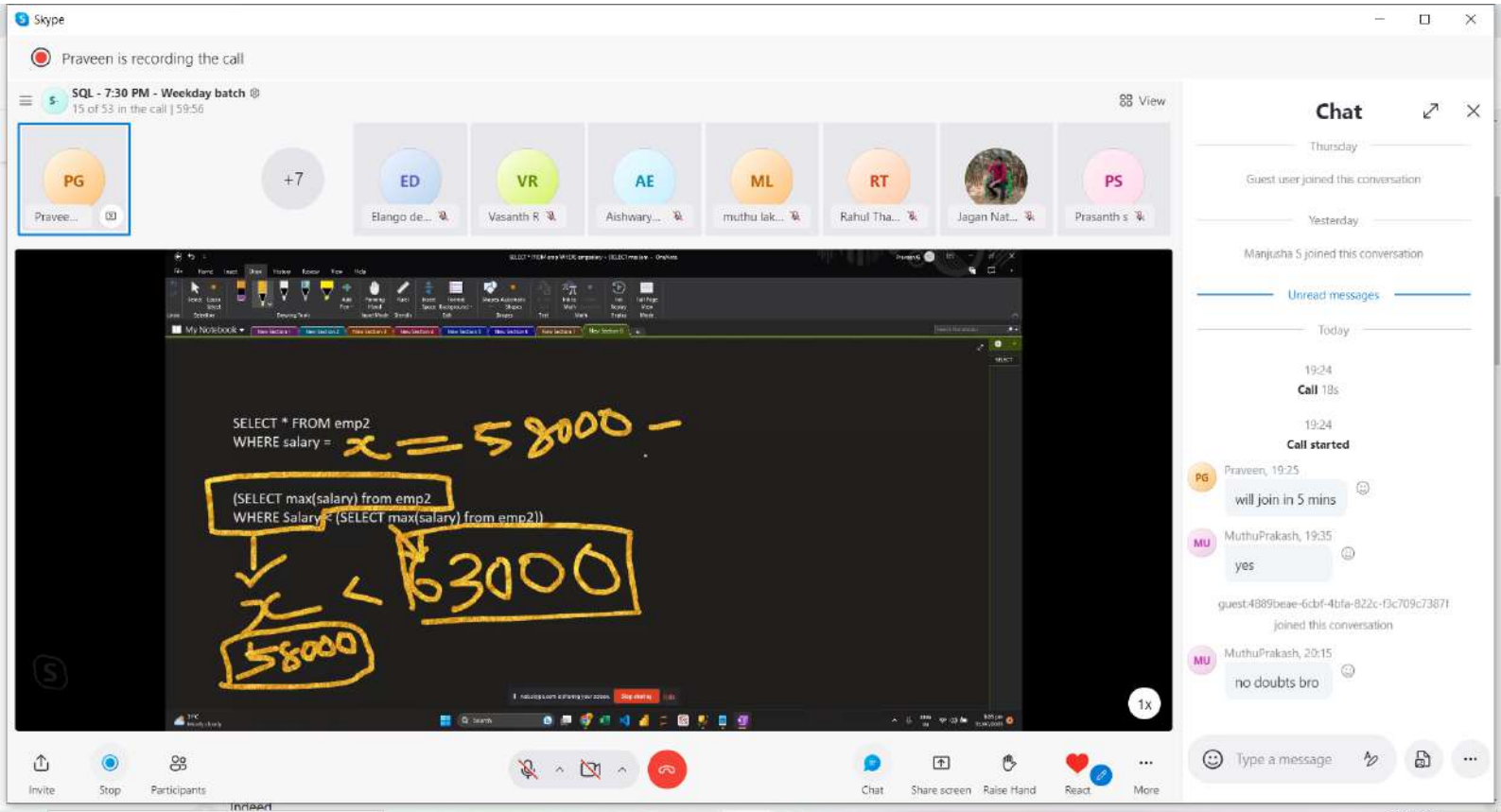
LinkedIn
Promoted

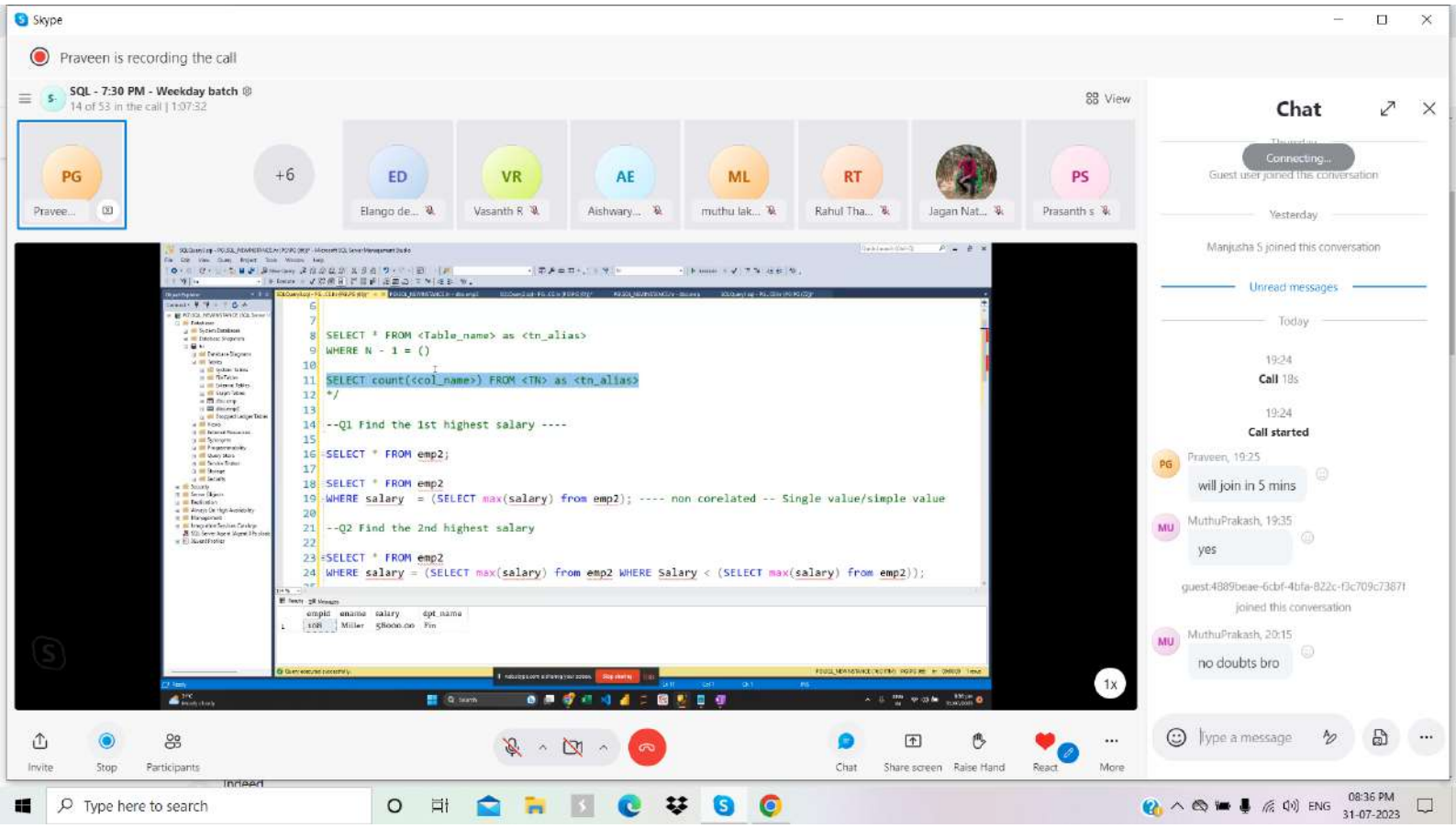
Update your job preferences to let recruiters know what roles you're interested in. **https://lnkd.in/ewS3knh**

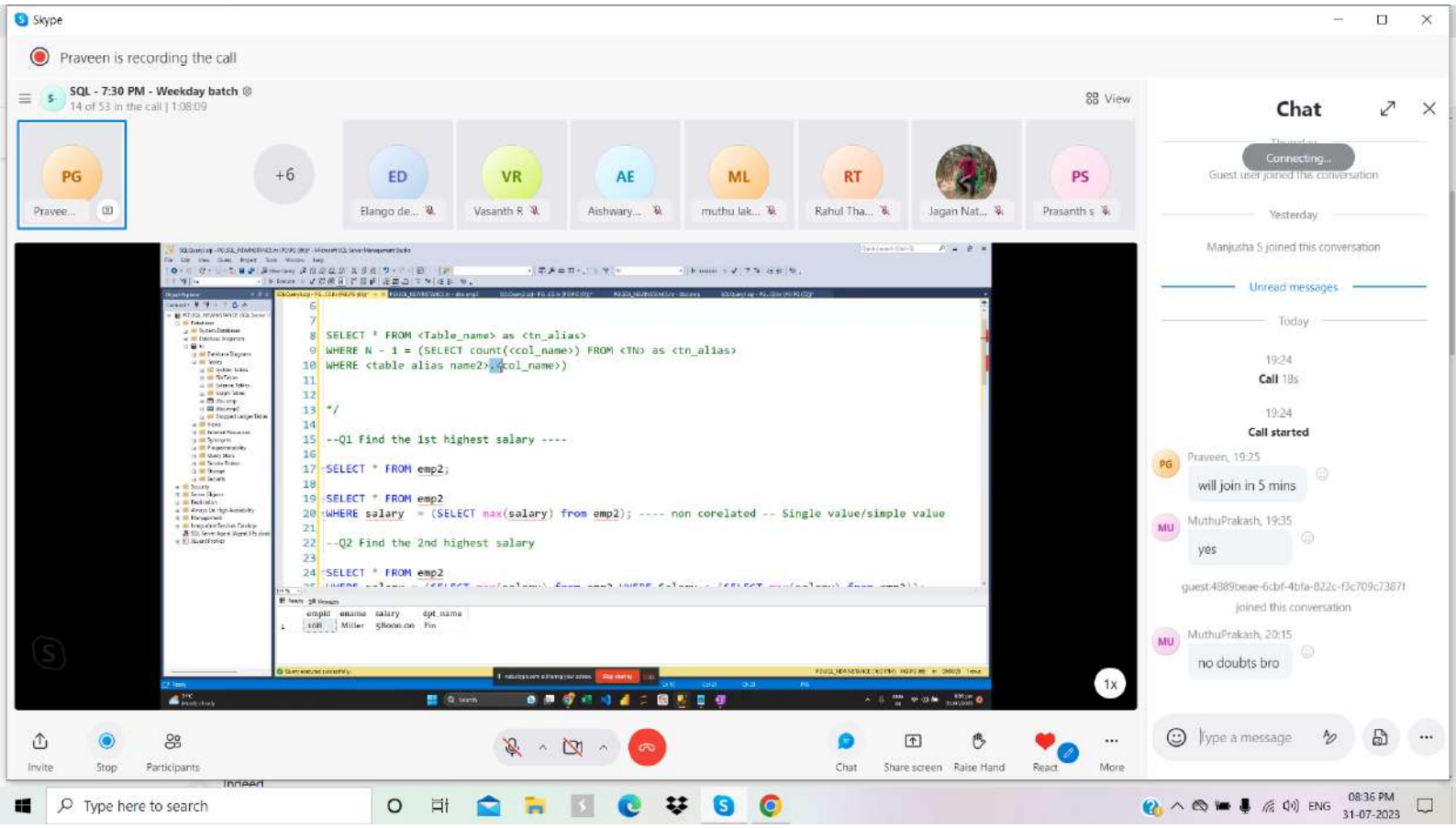

Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
16 of 52 in the call | 39:37
View
PG
Pravee...
+8
AM
Anagha M
VR
Vasanth R
AE
Aishwary...
ML
muthu lak...
RT
Rahul Tha...
Jagan Nat...
PS
Prasanth s
--STEP 1 : INNER QUERY
SELECT dpt_name FROM emp2
WHERE ename = 'Smith' or ename = 'Scott';
-- now, we found the value of dept of SCOTT and SMITH
--STEP 2 : OUTER QUERY
SELECT * FROM emp2
WHERE dpt_name = ??? --- (INNER QUERY)
--STEP 3 = step 1 + Step 2
empId ename salary dpt_name
1 101 Sai 25000.00 SAP
2 102 Allen 15000.00 Fin
3 103 Scott 30000.00 HR
4 104 Warner 17000.00 Fin
5 105 Smith 63000.00 SAP
6 106 Jones 41000.00 HR
7 107 Ford 47000.00 SAP
8 108 Dillor 58000.00 Fin
dpt_name
1 HR
2 SAP
1x
Chat
Recordings are available for 30 days
19:47
Call ended 5m 12s
Thursday
Guest user joined this conversation
Yesterday
Manjusha S joined this conversation
Unread messages
Today
19:24
Call 18s
19:24
Call started
Praveen, 19:25
will join in 5 mins
MuthuPrakash, 19:35
yes
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:08 PM
31-07-2023

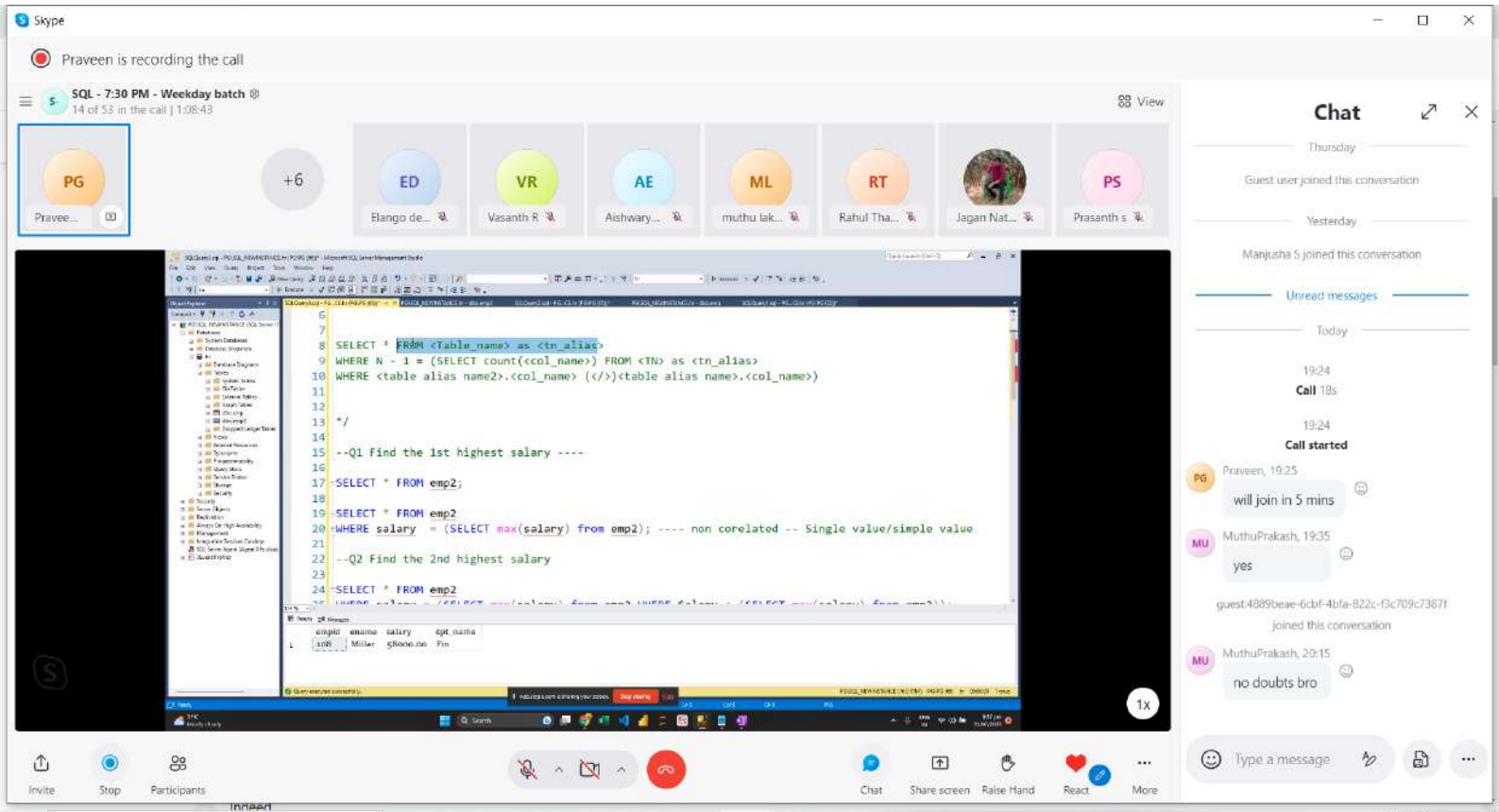

Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
16 of 52 in the call | 42:33
View
PG
Pravee...
+8
AM
Anagha M
VR
Vasanth R
AE
Aishwary...
ML
muthu lak...
RT
Rahul Tha...
Jagan Nat...
PS
Prasanth s
--STEP 2 : OUTER QUERY
SELECT * FROM emp2
WHERE dpt_name = ??? --- (INNER QUERY)
--STEP 3 - step 2 + Step 1
SELECT * FROM emp2
WHERE dpt_name IN (SELECT dpt_name FROM emp2 WHERE ename = 'Smith' or ename = 'Scott');
empid ename salary dpt_name
101 Rai 25000.00 SAP
103 Scott 30000.00 HR
105 Smith 63000.00 SAP
106 James 41000.00 Hr
107 Ford 47000.00 SAP
1x
Chat
Recordings are available for 30 days
19:47
Call ended 5m 12s
Thursday
Guest user joined this conversation
Yesterday
Manjusha S joined this conversation
Unread messages
Today
19:24
Call 18s
19:24
Call started
Praveen, 19:25
will join in 5 mins
MuthuPrakash, 19:35
yes
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:11 PM
31-07-2023

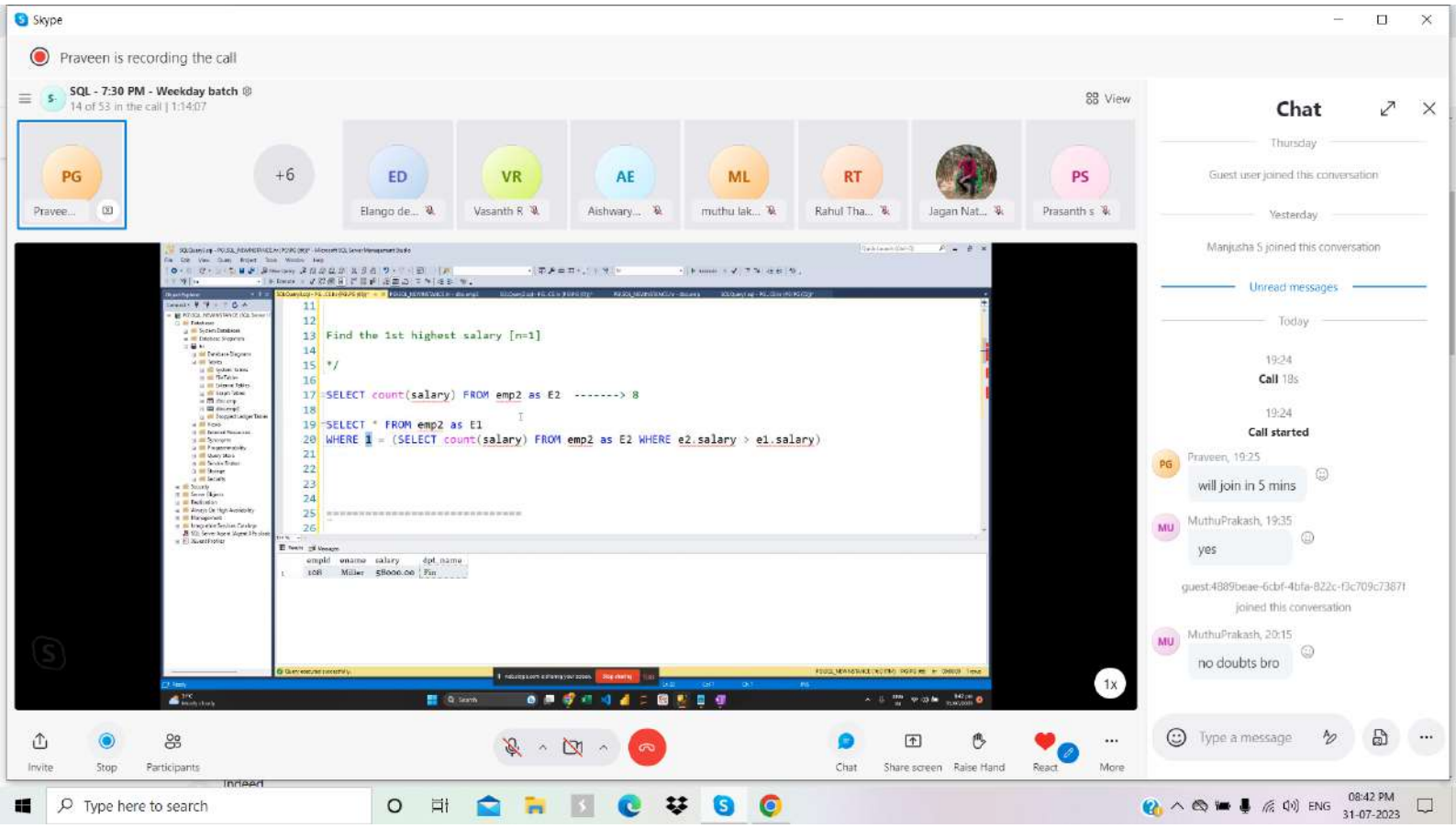

Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
15 of 53 in the call | 59:32
View
PG
Pravee...
+7
ED
Elango de...
VR
Vasanth R
AE
Aishwary...
ML
muthu lak...
RT
Rahul Tha...
Jagan Nat...
PS
Prasanth s
-WHERE salary = (SELECT max(salary) from emp2); ---- non corelated -- Single value/simple value
--Q2 Find the 2nd highest salary
SELECT * FROM emp2
WHERE salary = (SELECT max(salary) from emp2 WHERE Salary < (SELECT max(salary) from emp2));
empid ename salary dpt_name
108 Miller 58000.00 Fin
1x
Chat
Thursday
Guest user joined this conversation
Yesterday
Manjusha S joined this conversation
Unread messages
Today
19:24
Call 19s
19:24
Call started
Praveen, 19:25
will join in 5 mins
MuthuPrakash, 19:35
yes
guest:4889beae-6cbf-4bfa-822c-f3c709c7387f
joined this conversation
MuthuPrakash, 20:15
no doubts bro
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:28 PM
31-07-2023


Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
15 of 53 in the call | 59:56
View
PG
Pravee...
+7
ED
Elango de...
VR
Vasanth R
AE
Aishwary...
ML
muthu lak...
RT
Rahul Tha...
Jagan Nat...
PS
Prasanth s
SELECT * FROM emp2
WHERE salary = x = 58000 -
(SELECT max(salary) from emp2
WHERE Salary < (SELECT max(salary) from emp2))
x < 63000
58000
1x
Chat
Thursday
Guest user joined this conversation
Yesterday
Manjusha S joined this conversation
Unread messages
Today
19:24
Call 19s
19:24
Call started
Praveen, 19:25
will join in 5 mins
MuthuPrakash, 19:35
yes
guest:4889beae-6cbf-4bfa-822c-f3c709c7387f
joined this conversation
MuthuPrakash, 20:15
no doubts bro
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:28 PM
31-07-2023


Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
14 of 53 in the call | 1:07:32
View
PG
Pravee...
+6
ED
Elango de...
VR
Vasanth R
AE
Aishwary...
ML
muthu lak...
RT
Rahul Tha...
Jagan Nat...
PS
Prasanth s
SELECT * FROM <Table_name> as <tn_alias>
WHERE N - 1 = ()
SELECT count(<col_name>) FROM <TN> as <tn_alias>
*/
--Q1 Find the 1st highest salary ----
SELECT * FROM emp2;
SELECT * FROM emp2
WHERE salary = (SELECT max(salary) from emp2); ---- non corelated -- Single value/simple value
--Q2 Find the 2nd highest salary
SELECT * FROM emp2
WHERE salary = (SELECT max(salary) from emp2 WHERE Salary < (SELECT max(salary) from emp2));
empid ename salary dpt_name
108 Miller 58000.00 Fin
1x
Chat
Connecting...
Guest user joined this conversation
Yesterday
Manjusha S joined this conversation
Unread messages
Today
19:24
Call 19s
19:24
Call started
Praveen, 19:25
will join in 5 mins
MuthuPrakash, 19:35
yes
guest:4889beae-6cbf-4bfa-822c-f3c709c7387f
joined this conversation
MuthuPrakash, 20:15
no doubts bro
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:36 PM
31-07-2023


Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
14 of 53 in the call | 1:08:09
View
PG
Pravee...
+6
ED
Elango de...
VR
Vasanth R
AE
Aishwary...
ML
muthu lak...
RT
Rahul Tha...
Jagan Nat...
PS
Prasanth s
Chat
Connecting...
Guest user joined this conversation
Yesterday
Manjusha S joined this conversation
Unread messages
Today
19:24
Call 19s
19:24
Call started
Praveen, 19:25
will join in 5 mins
MuthuPrakash, 19:35
yes
guest:4889beae-6cbf-4bfa-822c-f3c709c7387f
joined this conversation
MuthuPrakash, 20:15
no doubts bro
Type a message
SELECT * FROM <Table_name> as <tn_alias>
WHERE N - 1 = (SELECT count(<col_name>) FROM <TN> as <tn_alias>
WHERE <table alias name2>.<col_name>)
*/
--Q1 Find the 1st highest salary ----
SELECT * FROM emp2;
SELECT * FROM emp2
WHERE salary = (SELECT max(salary) from emp2); ---- non corelated -- Single value/simple value
--Q2 Find the 2nd highest salary
SELECT * FROM emp2
empid ename salary dpt_name
108 Miller 58000.00 Fin
1x
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:36 PM
31-07-2023


Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
14 of 53 in the call | 1:08:43
View
PG
Pravee...
+6
ED
Elango de...
VR
Vasanth R
AE
Aishwary...
ML
muthu lak...
RT
Rahul Tha...
Jagan Nat...
PS
Prasanth s
SELECT * FROM <Table_name> as <tn_alias>
WHERE N - 1 = (SELECT count(<col_name>) FROM <TN> as <tn_alias>
WHERE <table alias name2>.<col_name> (</>)<table alias name>.<col_name>)
*/
--Q1 Find the 1st highest salary ----
SELECT * FROM emp2;
SELECT * FROM emp2
WHERE salary = (SELECT max(salary) from emp2); ---- non corelated -- Single value/simple value
--Q2 Find the 2nd highest salary
SELECT * FROM emp2
empid ename salary dpt_name
1 108 Miller 58000.00 Fin
1x
Chat
Thursday
Guest user joined this conversation
Yesterday
Manjusha S joined this conversation
Unread messages
Today
19:24
Call 19s
19:24
Call started
Praveen, 19:25
will join in 5 mins
MuthuPrakash, 19:35
yes
guest:4889beae-6cbf-4bfa-822c-f3c709c7387f
joined this conversation
MuthuPrakash, 20:15
no doubts bro
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:37 PM
31-07-2023


Skype
Praveen is recording the call
SQL - 7:30 PM - Weekday batch
14 of 53 in the call | 1:14:07
View
PG
Pravee...
+6
ED
Elango de...
VR
Vasanth R
AE
Aishwary...
ML
muthu lak...
RT
Rahul Tha...
Jagan Nat...
PS
Prasanth s
Find the 1st highest salary [n=1]
*/
SELECT count(salary) FROM emp2 as E2 -------> 8
SELECT * FROM emp2 as E1
WHERE 1 = (SELECT count(salary) FROM emp2 as E2 WHERE e2.salary > e1.salary)
empId ename salary dpt_name
108 Miller $8000.00 Fin
1x
Chat
Thursday
Guest user joined this conversation
Yesterday
Manjusha S joined this conversation
Unread messages
Today
19:24
Call 19s
19:24
Call started
Praveen, 19:25
will join in 5 mins
MuthuPrakash, 19:35
yes
guest:4889beae-6cbf-4bfa-822c-f3c709c7387f
joined this conversation
MuthuPrakash, 20:15
no doubts bro
Type a message
Invite
Stop
Participants
Chat
Share screen
Raise Hand
React
More
Type here to search
ENG
08:42 PM
31-07-2023

Introduction to MySQL cursor

To handle a result set inside a stored procedure, you use a cursor. A cursor allows you to iterate a set of rows returned by a query and process each row individually.

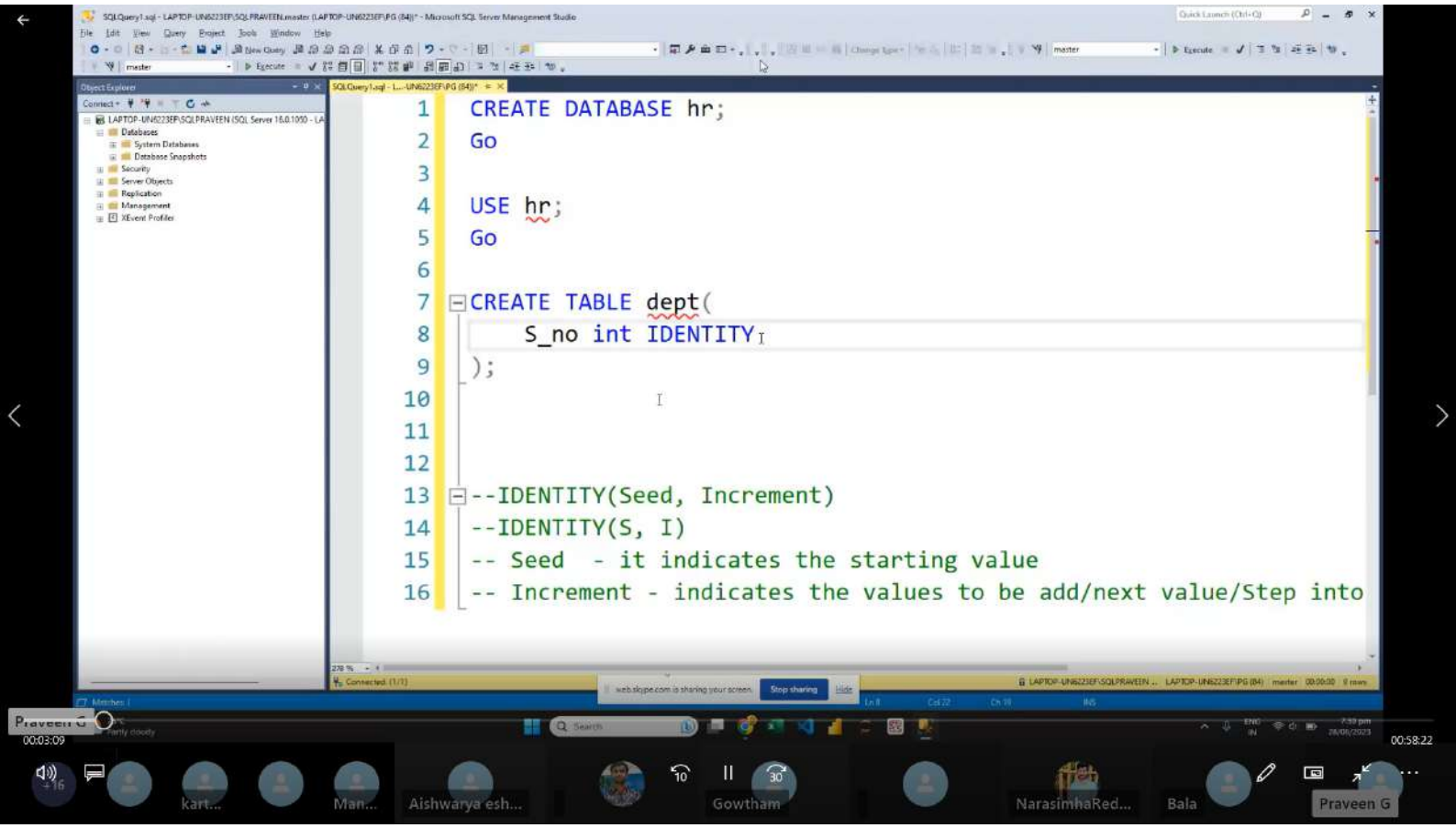
MySQL cursor is read-only, non-scrollable and asensitive.

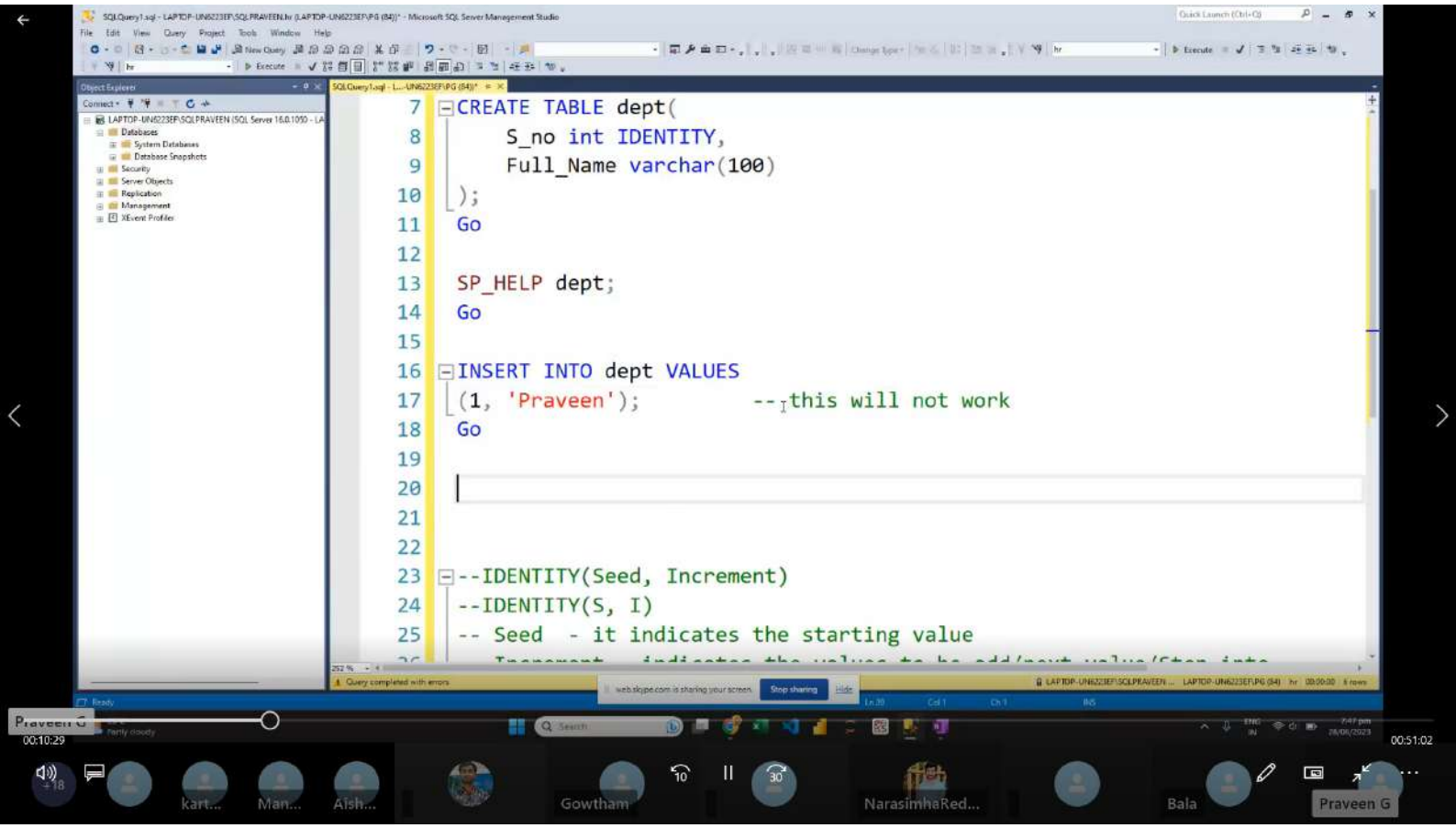
- **Read-only**: you cannot update data in the underlying table through the cursor.
- **Non-scrollable**: you can only fetch rows in the order determined by the `SELECT` statement. You cannot fetch rows in the reversed order. In addition, you cannot skip rows or jump to a specific row in the result set.
- **Asensitive**: there are two kinds of cursors: asensitive cursor and insensitive cursor. An asensitive cursor points to the actual data, whereas an insensitive cursor uses a temporary copy of the data. An asensitive cursor performs faster than an insensitive cursor because it does not have to make a temporary copy of data. However, any change that made to the data from other connections will affect the data that is being used by an asensitive cursor, therefore, it is safer if you do not update the data that is being used by an asensitive cursor. MySQL cursor is asensitive.

You can use MySQL cursors in stored procedures, stored functions, and triggers.

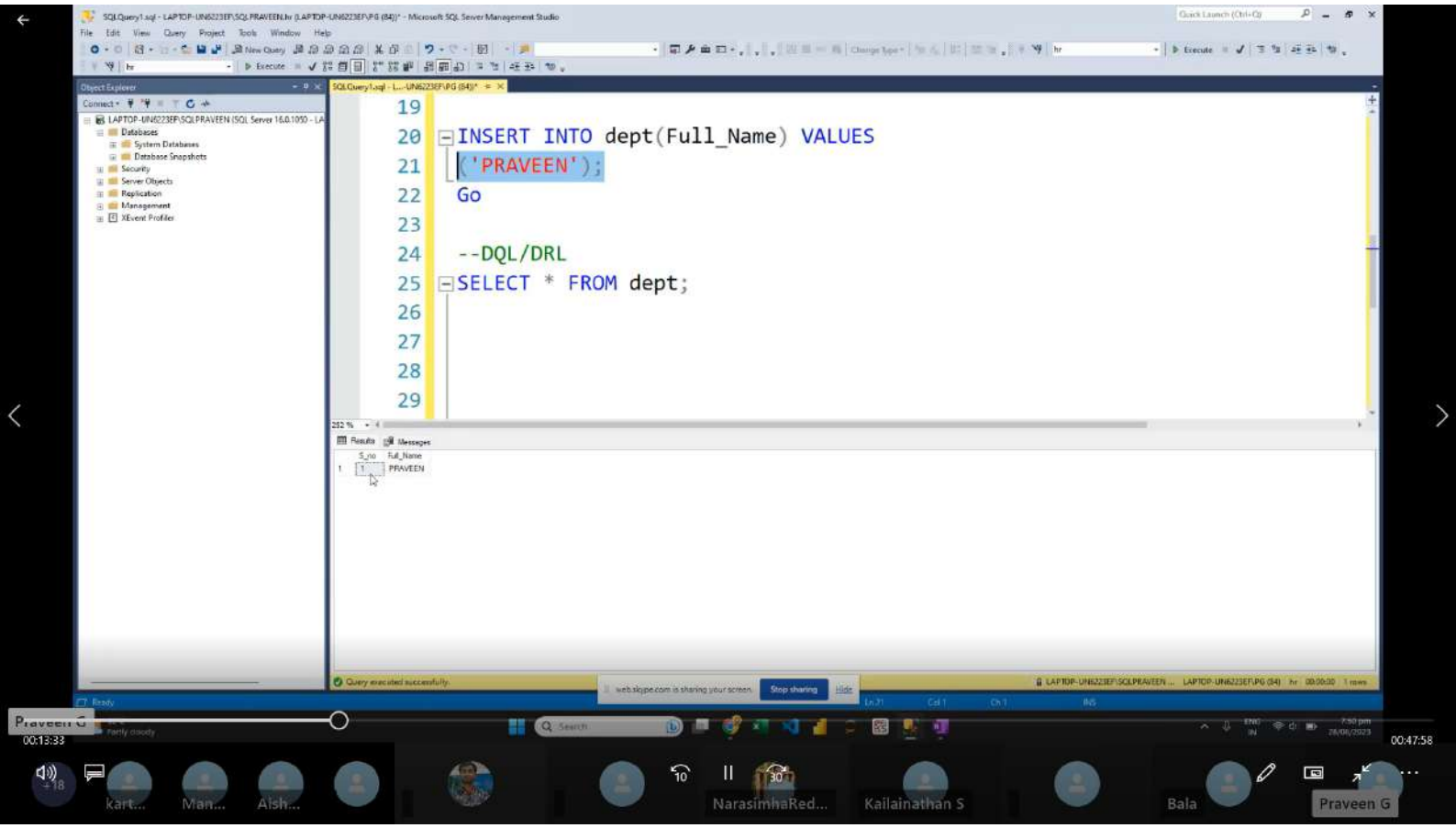
Working with MySQL cursor

Changing MySQL Delimiter
Creating Stored Procedures
Removing Stored Procedures
Modifying Stored Procedures
Listing Stored Procedures
Variables
Parameters
IF THEN
CASE Statement
LOOP
WHILE
REPEAT
LEAVE
Cursors
Handling Errors
Raising Errors
Stored Procedures that Return Multiple Values

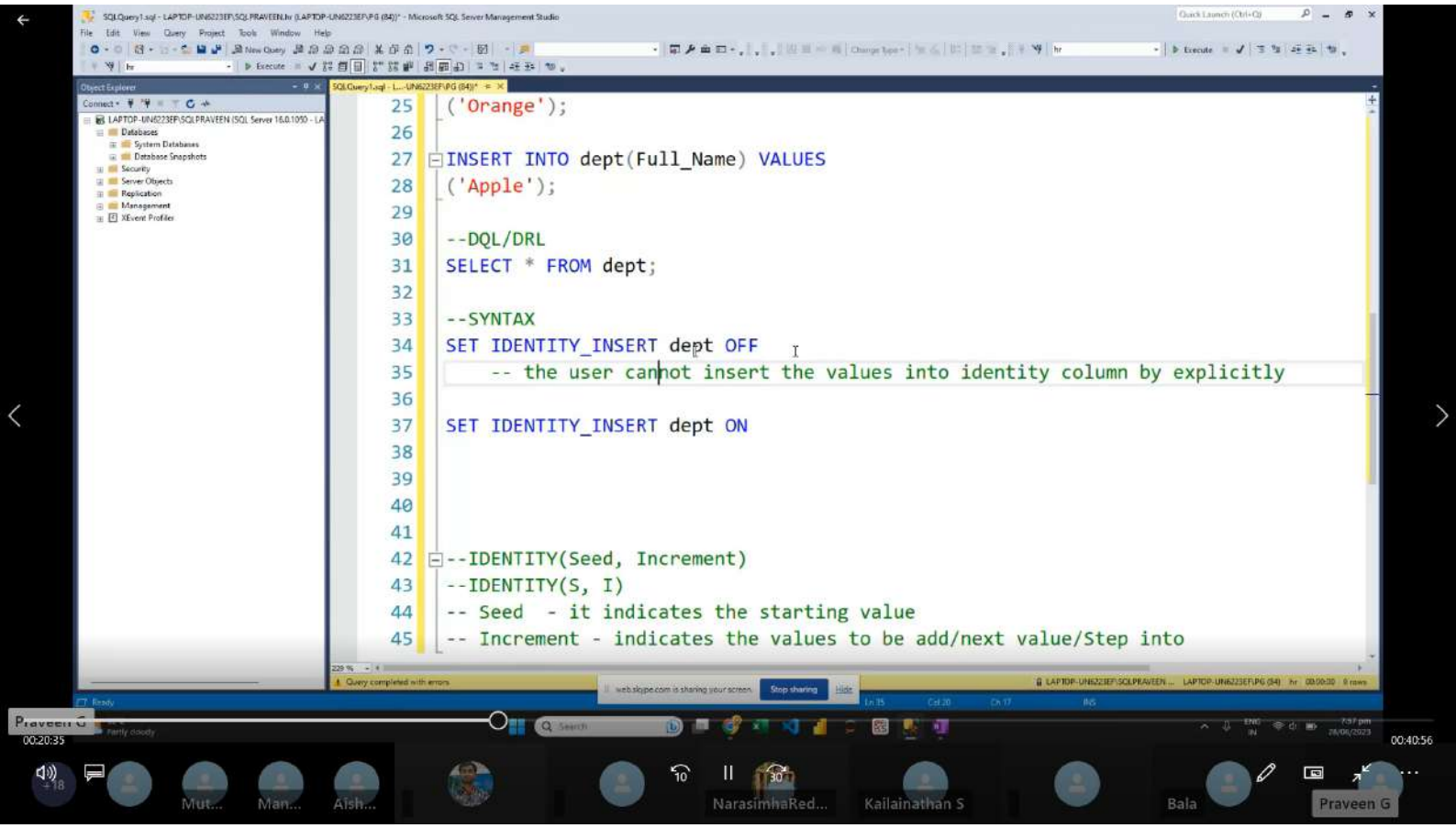

CREATE DATABASE hr;
Go
USE hr;
Go
CREATE TABLE dept(
S_no int IDENTITY
);
--IDENTITY(Seed, Increment)
--IDENTITY(S, I)
-- Seed - it indicates the starting value
-- Increment - indicates the values to be add/next value/Step into

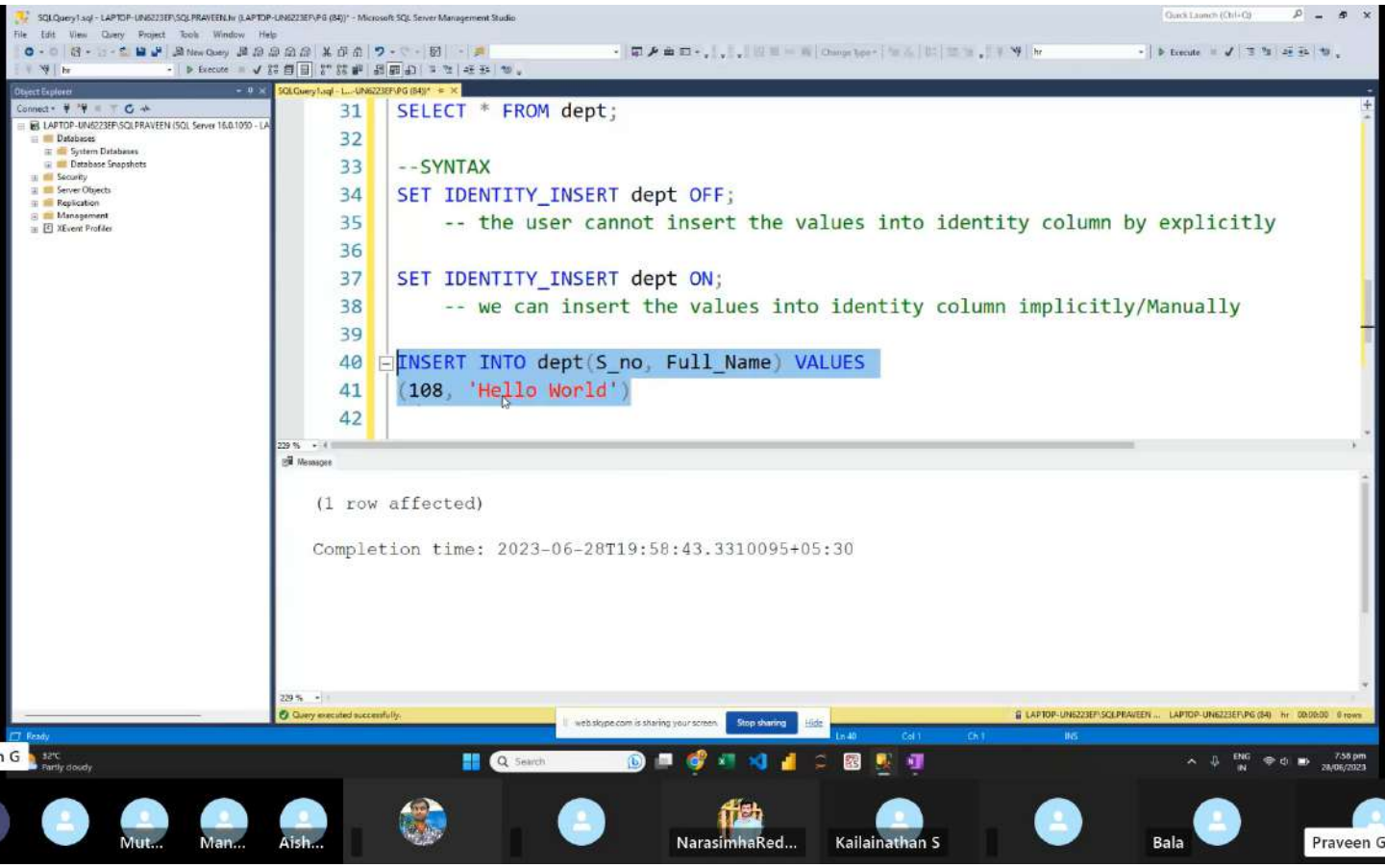


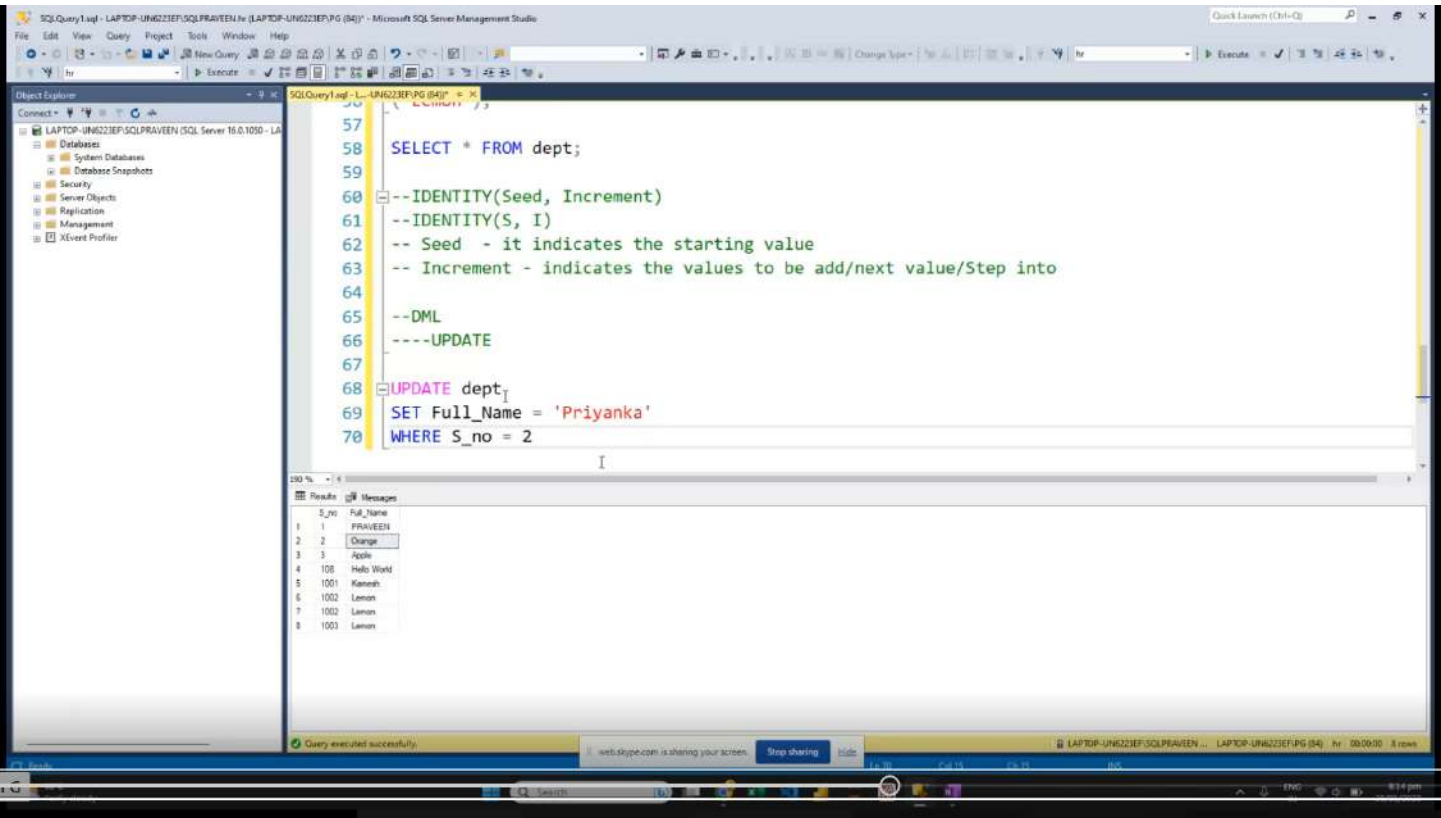
Praveen G
00:10:29
00:51:02
kart...
Man...
Aish...
Gowtham
NarasimhaRed...
Bala
Praveen G



Praveen G
00:13:33
00:47:58
kart...
Man...
Aish...
NarasimhaRed...
Kailainathan S
Bala
Praveen G

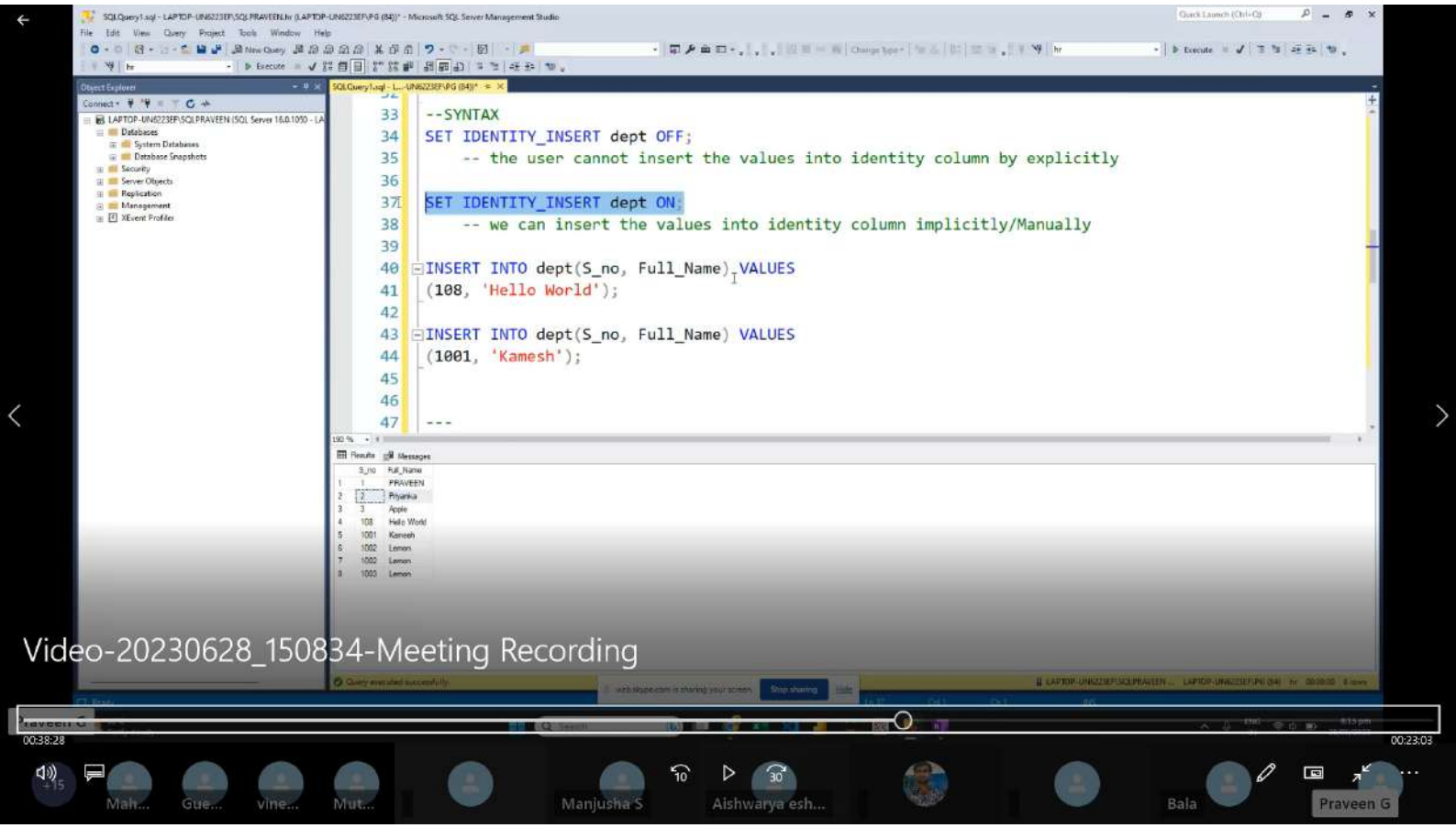

('Orange');
INSERT INTO dept(Full_Name) VALUES
('Apple');
--DQL/DRL
SELECT * FROM dept;
--SYNTAX
SET IDENTITY_INSERT dept OFF
-- the user cannot insert the values into identity column by explicitly
SET IDENTITY_INSERT dept ON
--IDENTITY(Seed, Increment)
--IDENTITY(S, I)
-- Seed - it indicates the starting value
-- Increment - indicates the values to be add/next value/Step into


SELECT * FROM dept;
--SYNTAX
SET IDENTITY_INSERT dept OFF;
-- the user cannot insert the values into identity column by explicitly
SET IDENTITY_INSERT dept ON;
-- we can insert the values into identity column implicitly/Manually
INSERT INTO dept(S_no, Full_Name) VALUES
(108, 'Hello World')
(1 row affected)
Completion time: 2023-06-28T19:58:43.3310095+05:30
Query executed successfully.
web.skype.com is sharing your screen.
Stop sharing
Hide
Praveen G
+16
Mut...
Man...
Aish...
NarasimhaRed...
Kailainathan S
Bala
Praveen G



00:37:36 00:23:55

Mah... Gue... vine... Mut... Manjusha S Aishwarya esh... Bala Praveen G

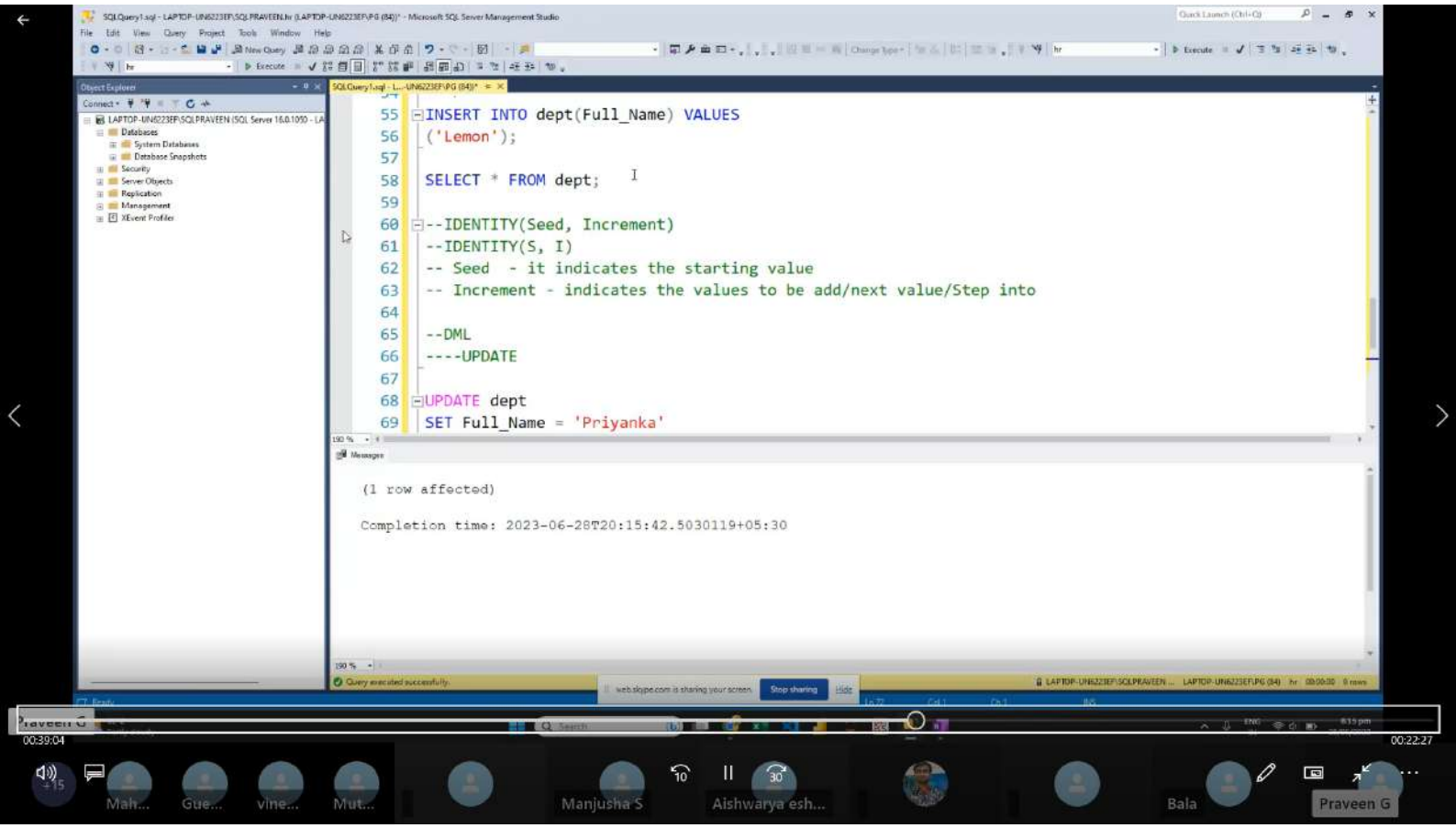


Video-20230628_150834-Meeting Recording

00:38:28

00:23:03

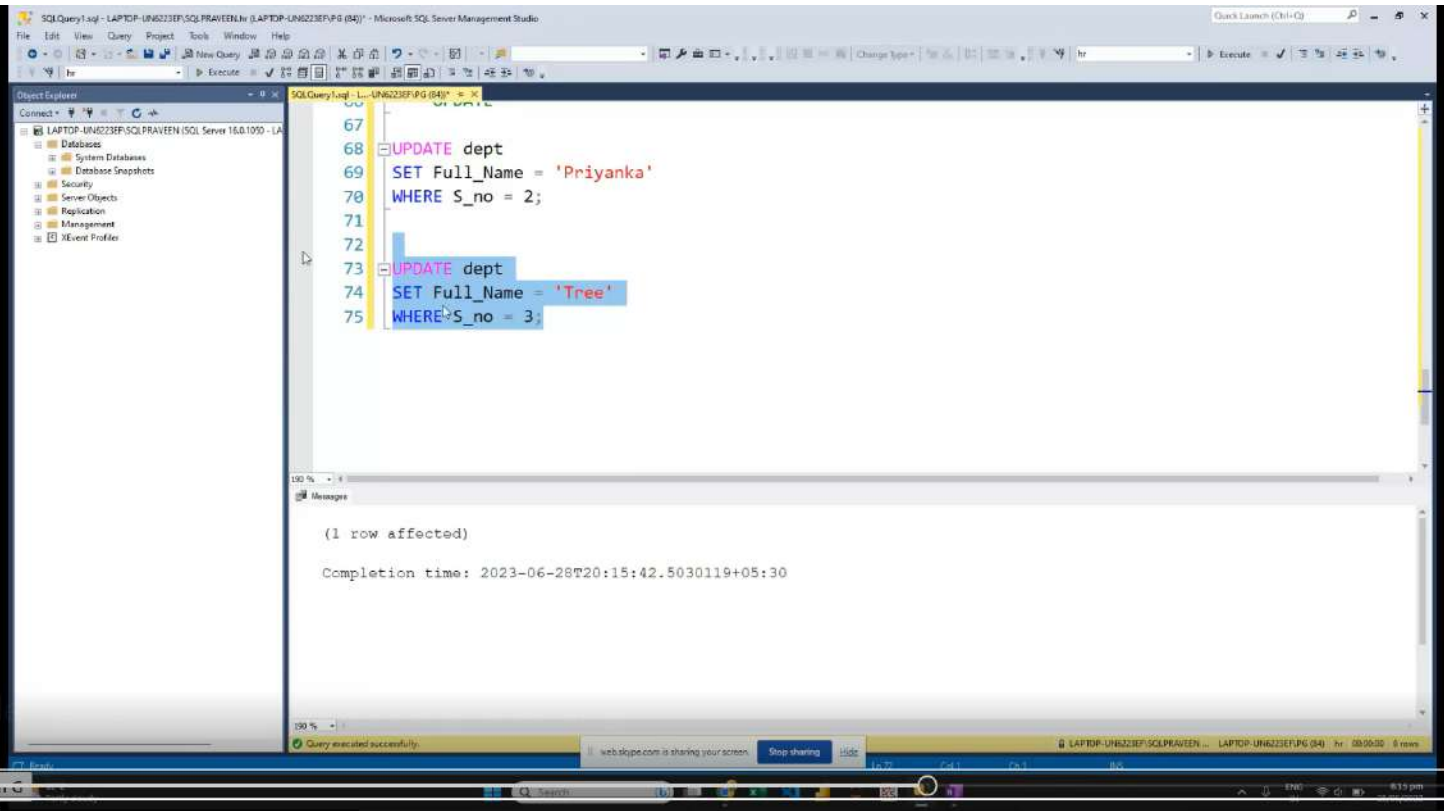
Mah... Gue... vine... Mut... Manjusha S Aishwarya esh... Bala Praveen G



00:39:04

00:22:27

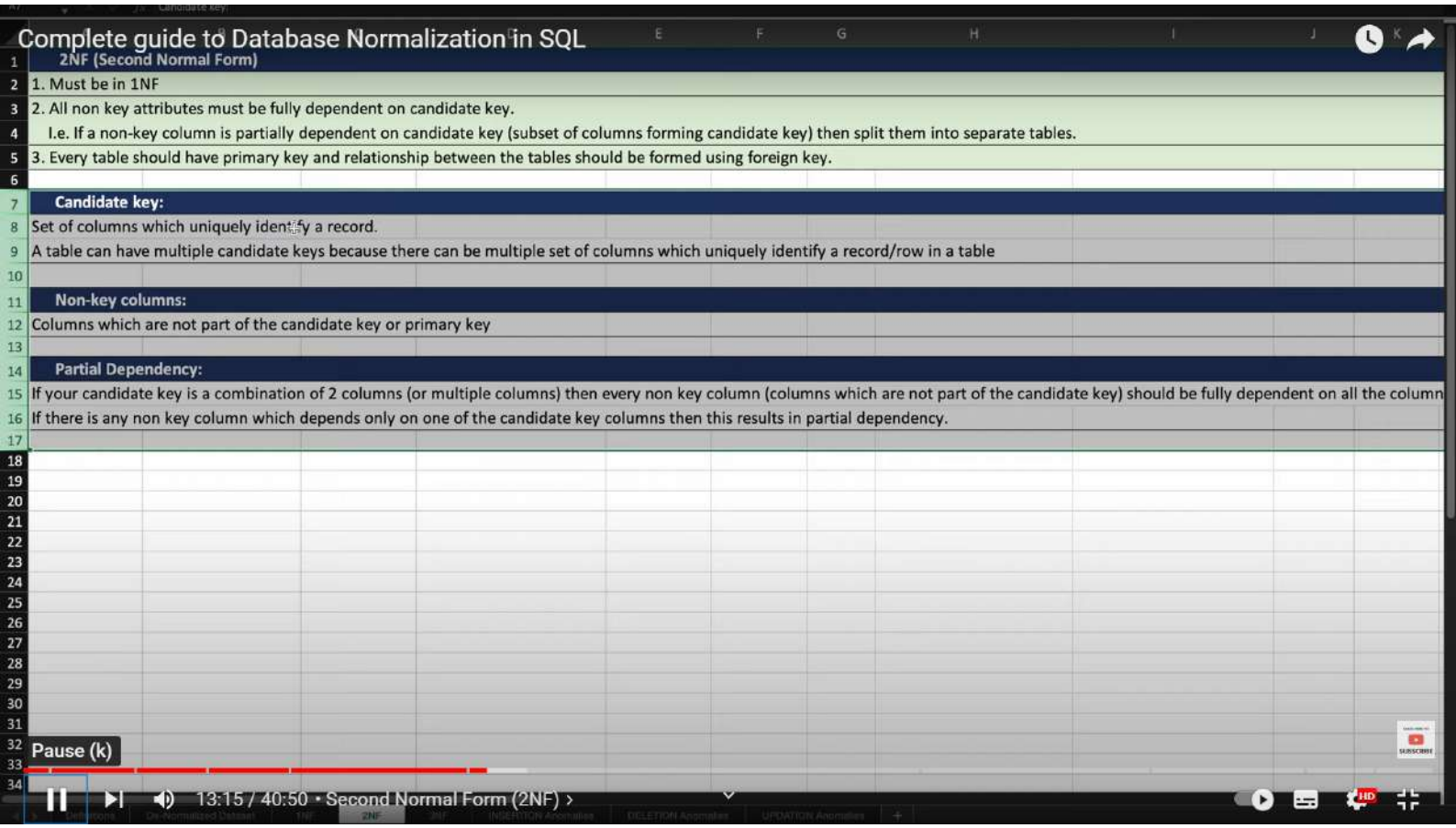
Mah... Gue... vine... Mut... Manjusha S Aishwarya esh... Bala Praveen G

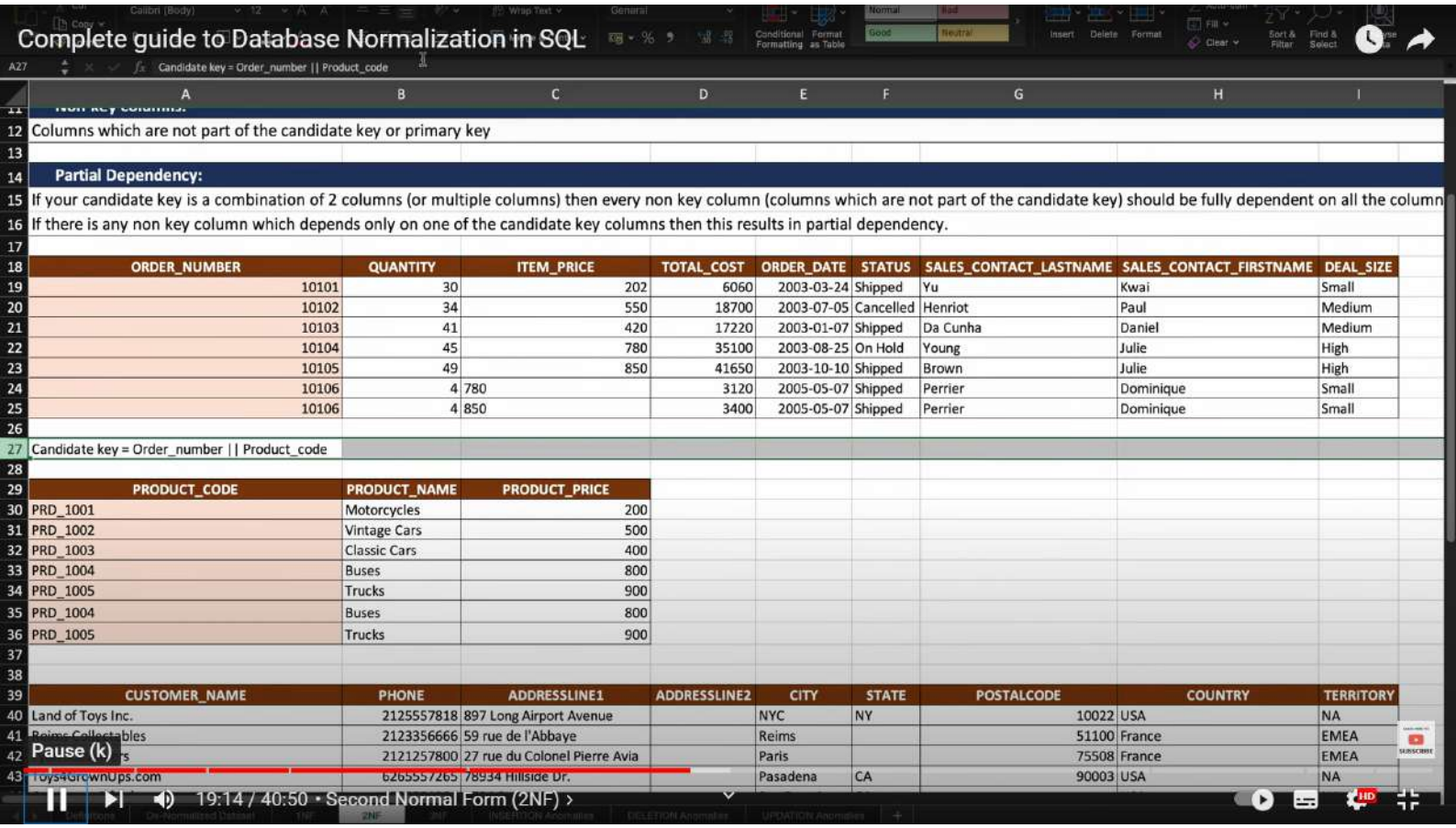


00:39:03

00:22:28

Mah... Gue... vine... Mut... Manjusha S Aishwarya esh... Bala Praveen G

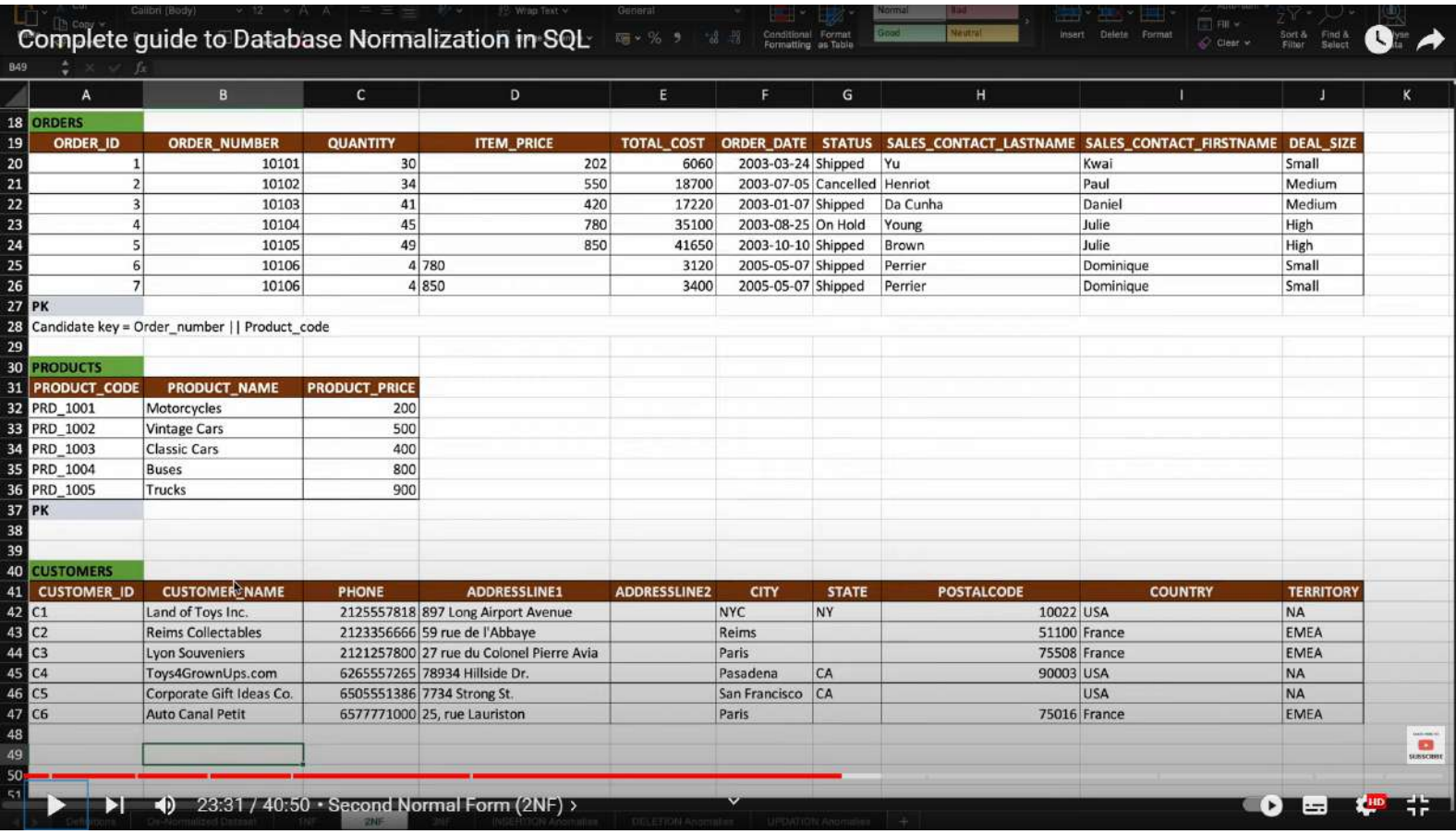

Complete guide to Database Normalization in SQL
2NF (Second Normal Form)
1. Must be in 1NF
2. All non key attributes must be fully dependent on candidate key.
I.e. If a non-key column is partially dependent on candidate key (subset of columns forming candidate key) then split them into separate tables.
3. Every table should have primary key and relationship between the tables should be formed using foreign key.
Candidate key:
Set of columns which uniquely identify a record.
A table can have multiple candidate keys because there can be multiple set of columns which uniquely identify a record/row in a table
Non-key columns:
Columns which are not part of the candidate key or primary key
Partial Dependency:
If your candidate key is a combination of 2 columns (or multiple columns) then every non key column (columns which are not part of the candidate key) should be fully dependent on all the column
If there is any non key column which depends only on one of the candidate key columns then this results in partial dependency.
Pause (k)
13:15 / 40:50 • Second Normal Form (2NF) ›



ORDER_NUMBER	QUANTITY	ITEM_PRICE	TOTAL_COST	ORDER_DATE	STATUS	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME	DEAL_SIZE
10101	30	202	6060	2003-03-24	Shipped	Yu	Kwai	Small
10102	34	550	18700	2003-07-05	Cancelled	Henriot	Paul	Medium
10103	41	420	17220	2003-01-07	Shipped	Da Cunha	Daniel	Medium
10104	45	780	35100	2003-08-25	On Hold	Young	Julie	High
10105	49	850	41650	2003-10-10	Shipped	Brown	Julie	High
10106	4	780	3120	2005-05-07	Shipped	Perrier	Dominique	Small
10106	4	850	3400	2005-05-07	Shipped	Perrier	Dominique	Small

PRODUCT_CODE	PRODUCT_NAME	PRODUCT_PRICE
PRD_1001	Motorcycles	200
PRD_1002	Vintage Cars	500
PRD_1003	Classic Cars	400
PRD_1004	Buses	800
PRD_1005	Trucks	900
PRD_1004	Buses	800
PRD_1005	Trucks	900

CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY
Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA
Reims Collectables	2123356666	59 rue de l'Abbaye		Reims		51100	France	EMEA
Paris	2121257800	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA
Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA



ORDERS

ORDER_ID	ORDER_NUMBER	QUANTITY	ITEM_PRICE	TOTAL_COST	ORDER_DATE	STATUS	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME	DEAL_SIZE
1	10101	30	202	6060	2003-03-24	Shipped	Yu	Kwai	Small
2	10102	34	550	18700	2003-07-05	Cancelled	Henriot	Paul	Medium
3	10103	41	420	17220	2003-01-07	Shipped	Da Cunha	Daniel	Medium
4	10104	45	780	35100	2003-08-25	On Hold	Young	Julie	High
5	10105	49	850	41650	2003-10-10	Shipped	Brown	Julie	High
6	10106	4	780	3120	2005-05-07	Shipped	Perrier	Dominique	Small
7	10106	4	850	3400	2005-05-07	Shipped	Perrier	Dominique	Small

PK

Candidate key = Order_number || Product_code

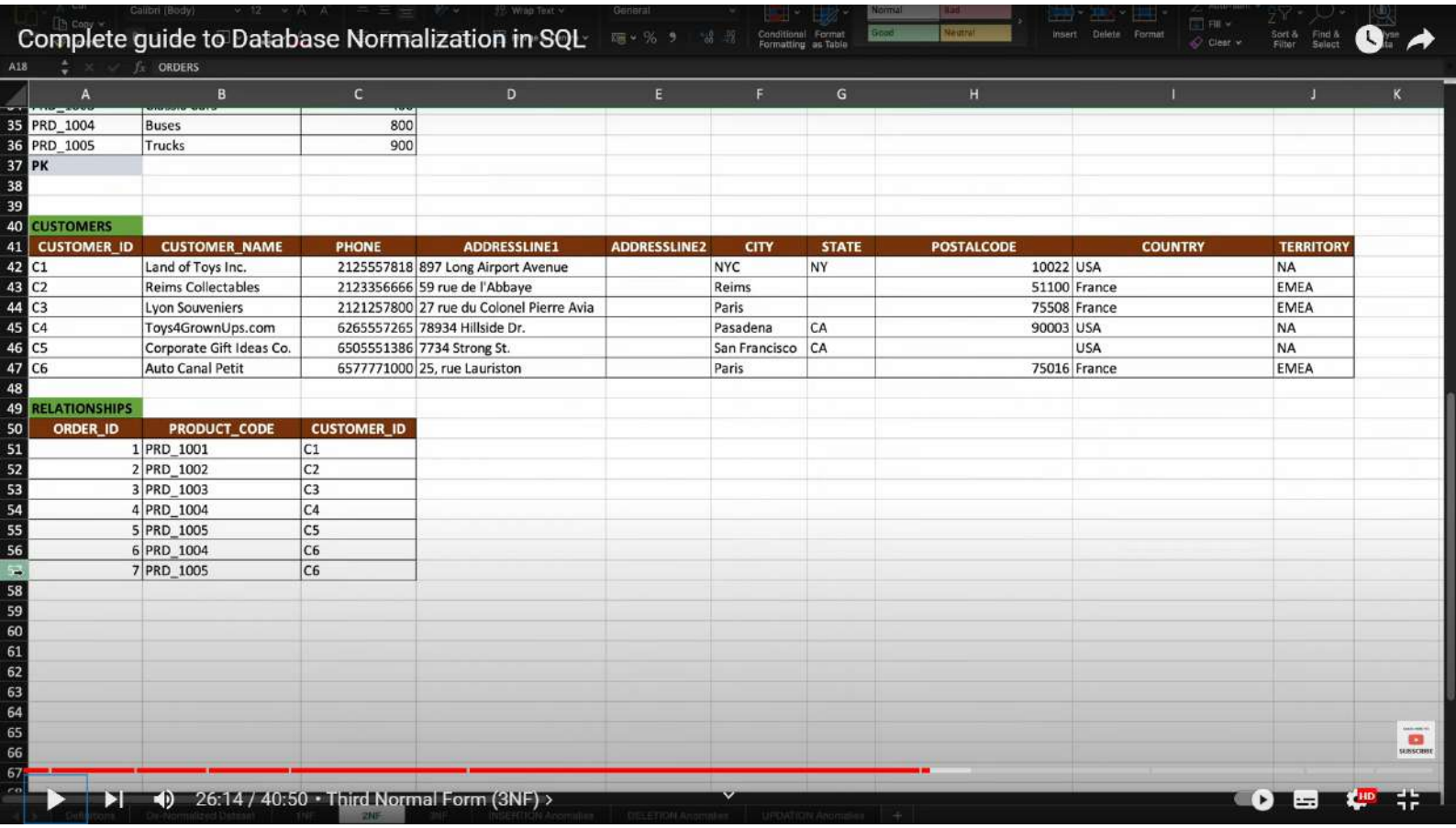
PRODUCTS

PRODUCT_CODE	PRODUCT_NAME	PRODUCT_PRICE
PRD_1001	Motorcycles	200
PRD_1002	Vintage Cars	500
PRD_1003	Classic Cars	400
PRD_1004	Buses	800
PRD_1005	Trucks	900

PK

CUSTOMERS

CUSTOMER_ID	CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY
C1	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA
C2	Reims Collectables	2123356666	59 rue de l'Abbaye		Reims		51100	France	EMEA
C3	Lyon Souveniers	2121257800	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA
C4	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA
C5	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisco	CA		USA	NA
C6	Auto Canal Petit	6577771000	25, rue Lauriston		Paris		75016	France	EMEA



PRD_1004	Buses	800
PRD_1005	Trucks	900

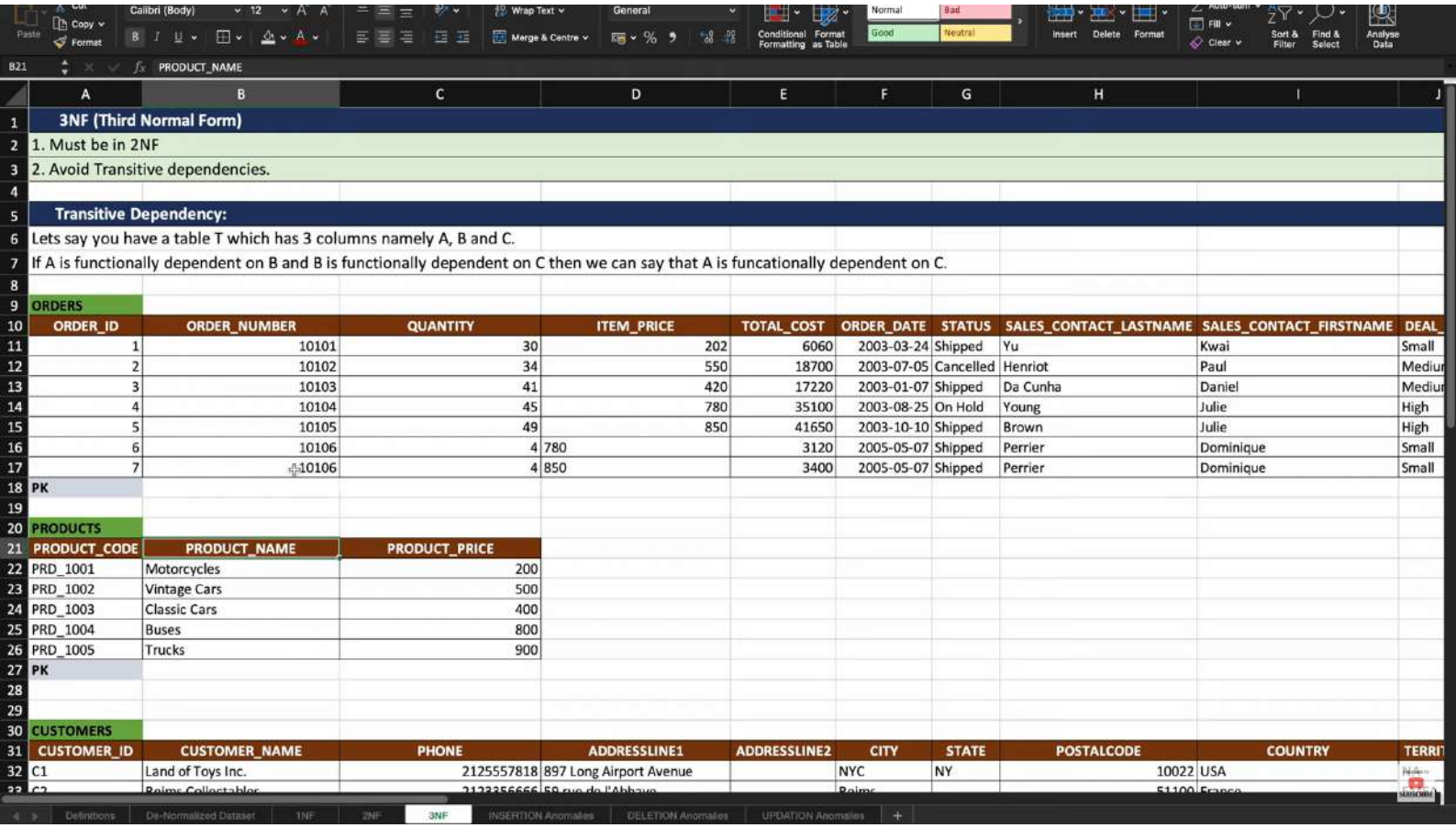
PK

CUSTOMERS

CUSTOMER_ID	CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY
C1	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA
C2	Reims Collectables	2123356666	59 rue de l'Abbaye		Reims		51100	France	EMEA
C3	Lyon Souveniers	2121257800	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA
C4	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA
C5	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisco	CA		USA	NA
C6	Auto Canal Petit	6577771000	25, rue Lauriston		Paris		75016	France	EMEA

RELATIONSHIPS

ORDER_ID	PRODUCT_CODE	CUSTOMER_ID
1	PRD_1001	C1
2	PRD_1002	C2
3	PRD_1003	C3
4	PRD_1004	C4
5	PRD_1005	C5
6	PRD_1004	C6
7	PRD_1005	C6



3NF (Third Normal Form)

1. Must be in 2NF
2. Avoid Transitive dependencies.

Transitive Dependency:

Lets say you have a table T which has 3 columns namely A, B and C.
If A is functionally dependent on B and B is functionally dependent on C then we can say that A is funcationally dependent on C.

ORDERS

ORDER_ID	ORDER_NUMBER	QUANTITY	ITEM_PRICE	TOTAL_COST	ORDER_DATE	STATUS	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME	DEAL_
1	10101	30	202	6060	2003-03-24	Shipped	Yu	Kwai	Small
2	10102	34	550	18700	2003-07-05	Cancelled	Henriot	Paul	Mediur
3	10103	41	420	17220	2003-01-07	Shipped	Da Cunha	Daniel	Mediur
4	10104	45	780	35100	2003-08-25	On Hold	Young	Julie	High
5	10105	49	850	41650	2003-10-10	Shipped	Brown	Julie	High
6	10106	4	780	3120	2005-05-07	Shipped	Perrier	Dominique	Small
7	10106	4	850	3400	2005-05-07	Shipped	Perrier	Dominique	Small

PK

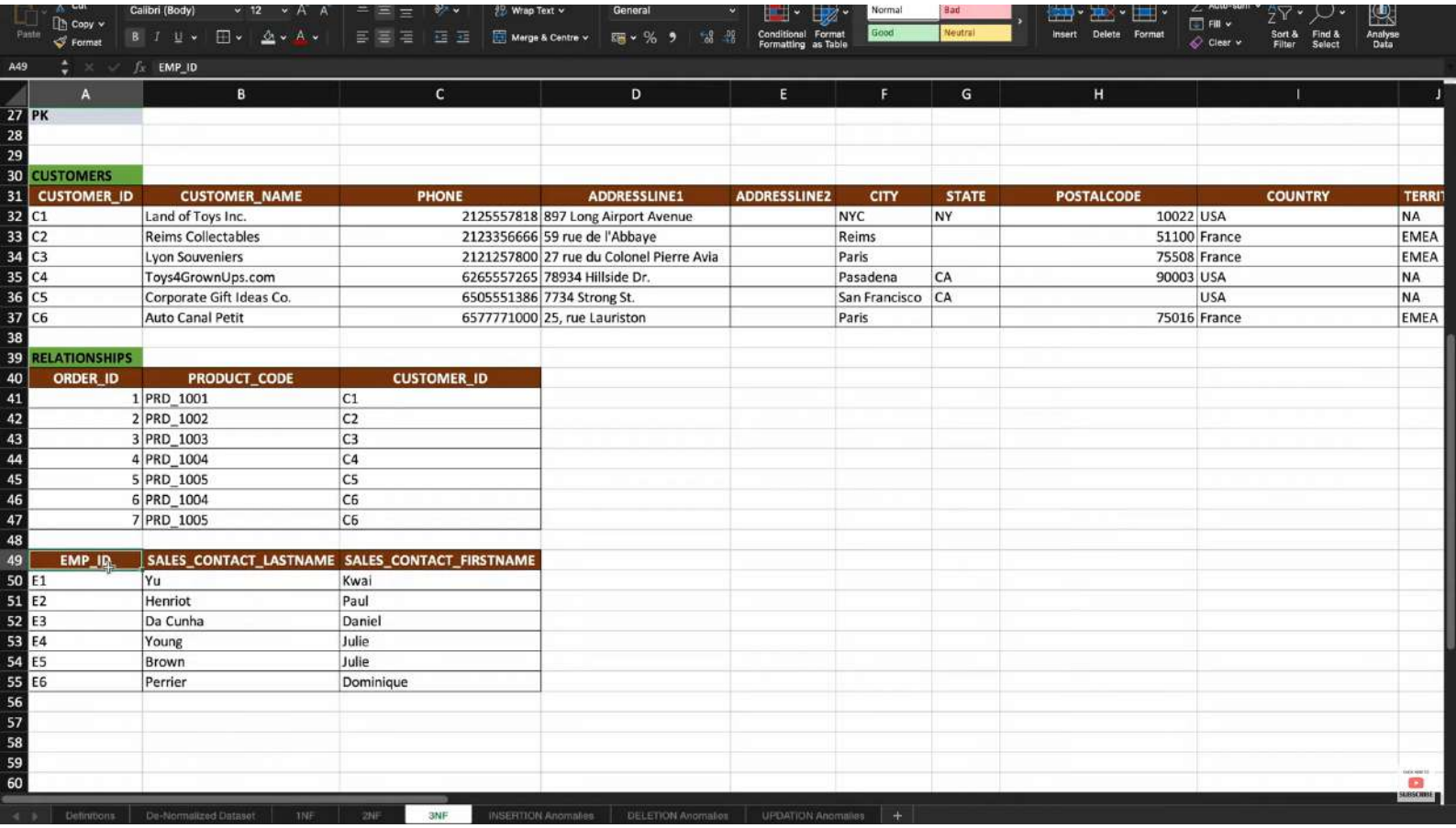
PRODUCTS

PRODUCT_CODE	PRODUCT_NAME	PRODUCT_PRICE
PRD_1001	Motorcycles	200
PRD_1002	Vintage Cars	500
PRD_1003	Classic Cars	400
PRD_1004	Buses	800
PRD_1005	Trucks	900

PK

CUSTOMERS

CUSTOMER_ID	CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRI
C1	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	
C2	Reims Collectables	2123356666	59 rue de l'Abbaye		Reims		51100	France	



PK

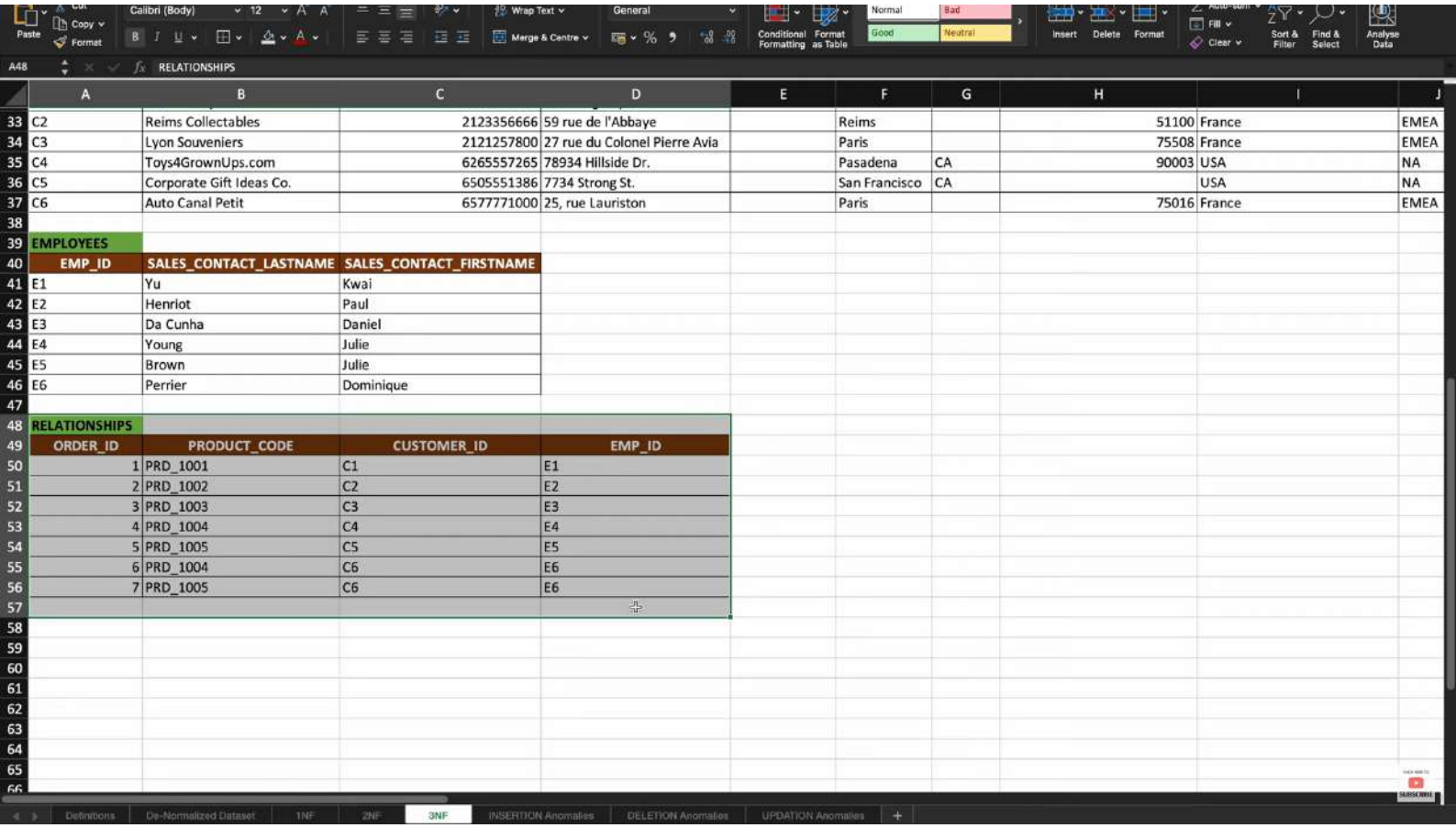
CUSTOMERS

CUSTOMER_ID	CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRI
C1	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA
C2	Reims Collectables	2123356666	59 rue de l'Abbaye		Reims		51100	France	EMEA
C3	Lyon Souveniers	2121257800	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA
C4	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA
C5	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisco	CA		USA	NA
C6	Auto Canal Petit	6577771000	25, rue Lauriston		Paris		75016	France	EMEA

RELATIONSHIPS

ORDER_ID	PRODUCT_CODE	CUSTOMER_ID
1	PRD_1001	C1
2	PRD_1002	C2
3	PRD_1003	C3
4	PRD_1004	C4
5	PRD_1005	C5
6	PRD_1004	C6
7	PRD_1005	C6

EMP_ID	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME
E1	Yu	Kwai
E2	Henriot	Paul
E3	Da Cunha	Daniel
E4	Young	Julie
E5	Brown	Julie
E6	Perrier	Dominique



C2	Reims Collectables	2123356666	59 rue de l'Abbaye		Reims		51100	France	EMEA
C3	Lyon Souveniers	2121257800	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA
C4	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA
C5	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisco	CA		USA	NA
C6	Auto Canal Petit	6577771000	25, rue Lauriston		Paris		75016	France	EMEA

EMPLOYEES

EMP_ID	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME
E1	Yu	Kwai
E2	Henriot	Paul
E3	Da Cunha	Daniel
E4	Young	Julie
E5	Brown	Julie
E6	Perrier	Dominique

RELATIONSHIPS

ORDER_ID	PRODUCT_CODE	CUSTOMER_ID	EMP_ID
1	PRD_1001	C1	E1
2	PRD_1002	C2	E2
3	PRD_1003	C3	E3
4	PRD_1004	C4	E4
5	PRD_1005	C5	E5
6	PRD_1004	C6	E6
7	PRD_1005	C6	E6

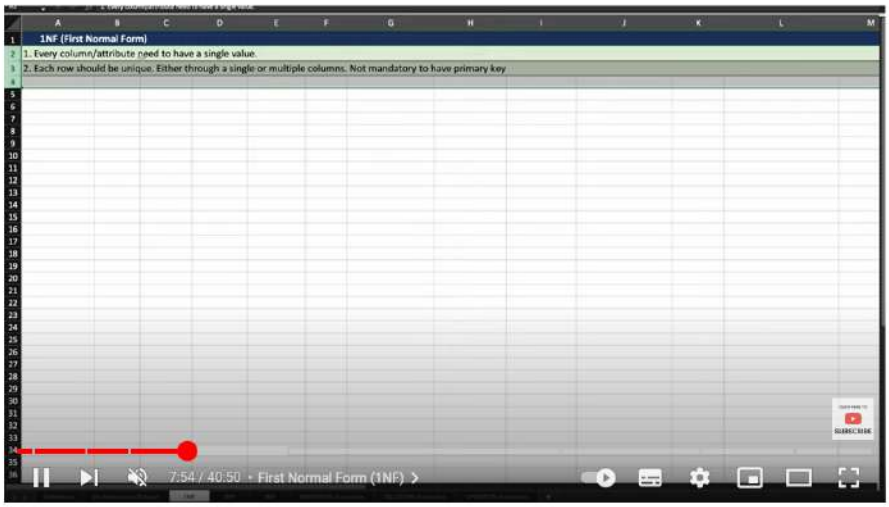
(1) WhatsApp | Meet – pax-moph-dkr | (8941) Complete guide to Datab

youtube.com/watch?v=r8PQ5fg_kiY

Gmail | Print Certificate | YouTube | Maps | Job Profiles | Inmakes Learning H...

YouTube IN

Search



Complete guide to Database Normalization in SQL

techTFQ
220K subscribers

Subscribed | 2.1K | Share | Download

Show chat replay

HPE GreenLake
Ad • hpe.com/in/en/greenlake/edge
Learn more

All | From techTFQ | SQL | Database normal

Become an Automation QA Tester
Project-led & real work-ex based course to become a QA Automation Testing expert.
Ad • Crio.Do
Apply now

Subquery in SQL | Correlated Subquery + Complete SQL...
techTFQ
661K views • 1 year ago

SQL Views Tutorial | VIEWS in

Screenshot (548).pdf | D18 - Sub Query -....pdf | example.pdf | D18 - Sub Query -....txt | Show all

Type here to search

ENG 08:19 AM 01-08-2023

*Untitled - Notepad

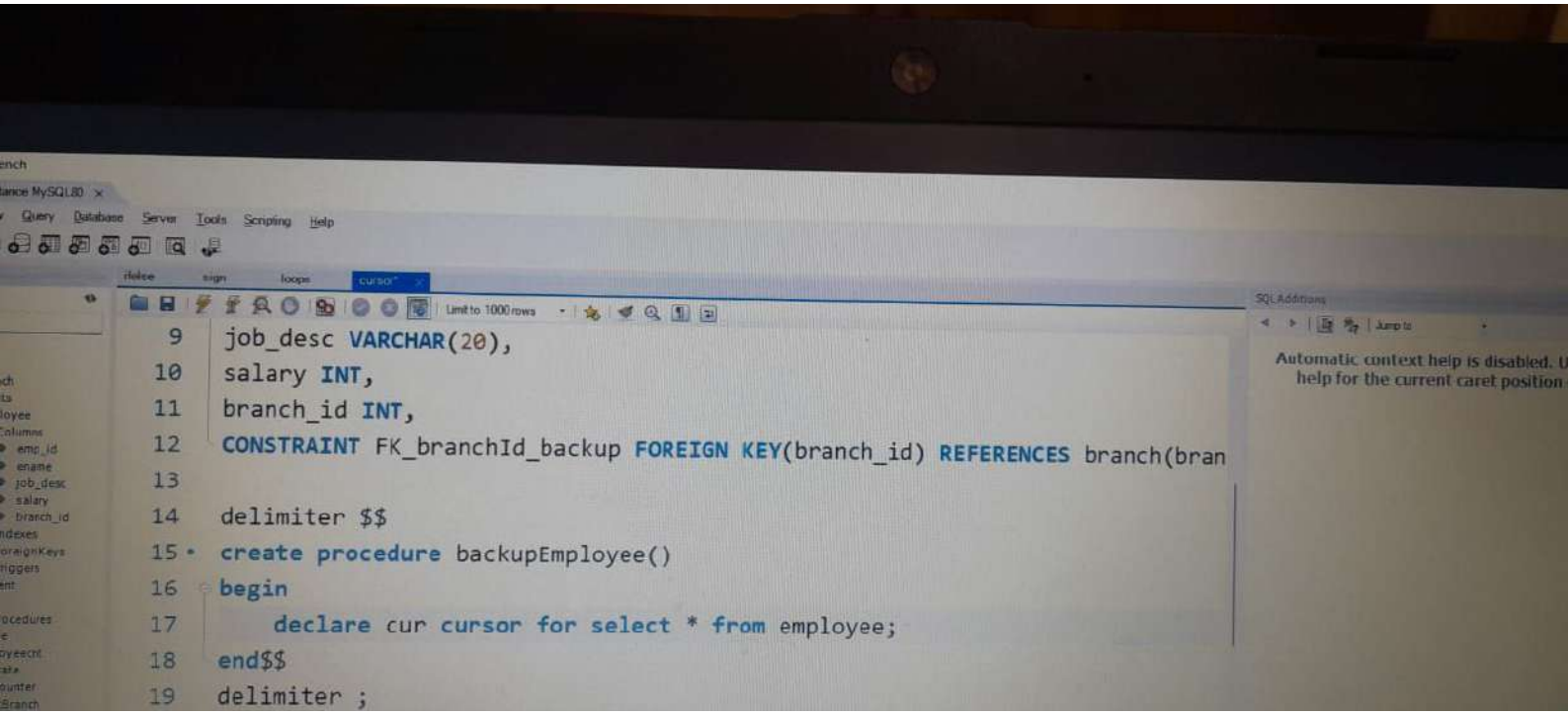
File Edit Format View Help

RECURSIVE sql query is a query that refes to its own output in order to perform additional operation it is commonly used for hierarchical ot tree-like data structure.
cusor is temporary memory or temporary work station .it allocated by database sever at the time of performing dml operation on the table by the user
cusor are used to store database table.
 normalization is a process of desing
 the database effectively such that we can avoid data redunancy, recongized the data cross the multiple tables inorder to reduc the data redunency.

Ln 5, Col 149 | 100% | Windows (CRLF) | UTF-8

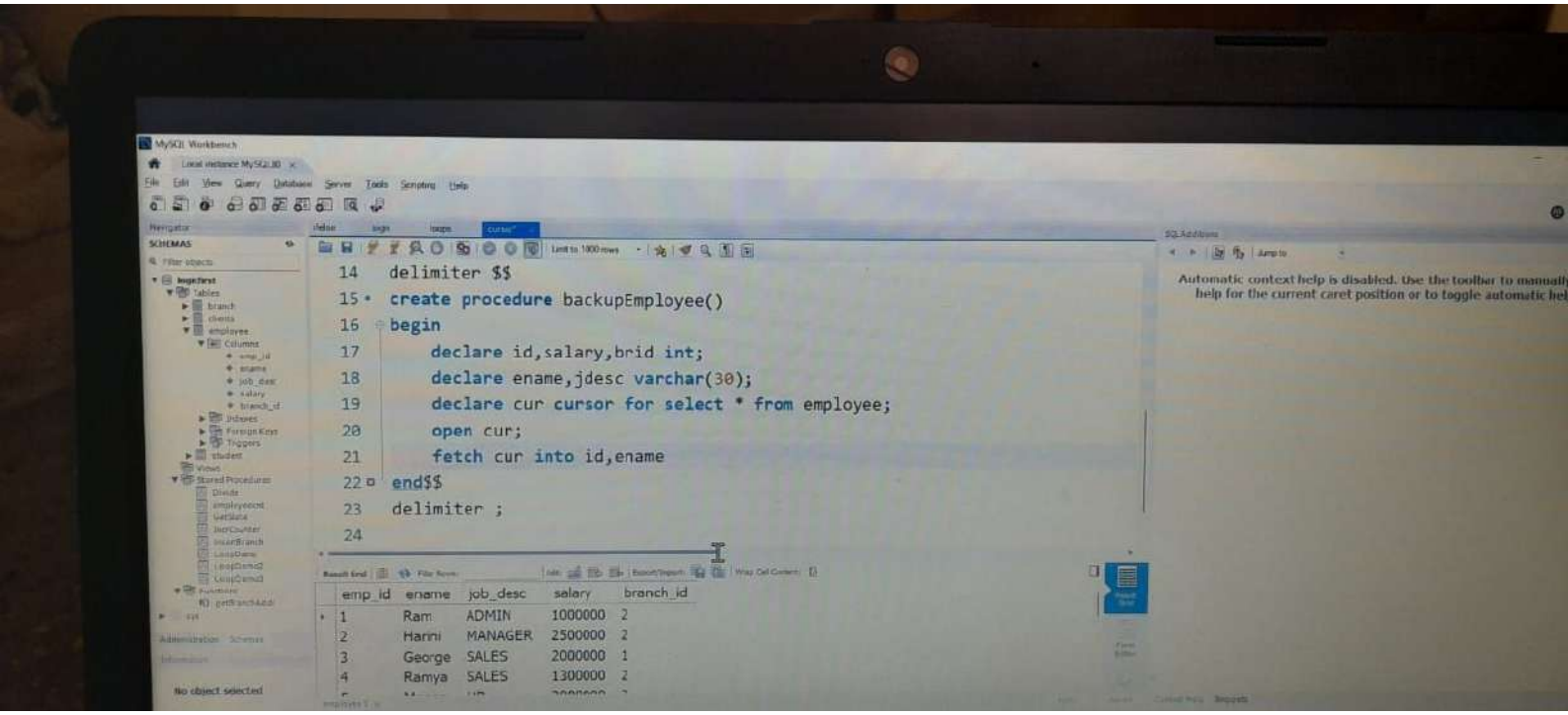
Type here to search

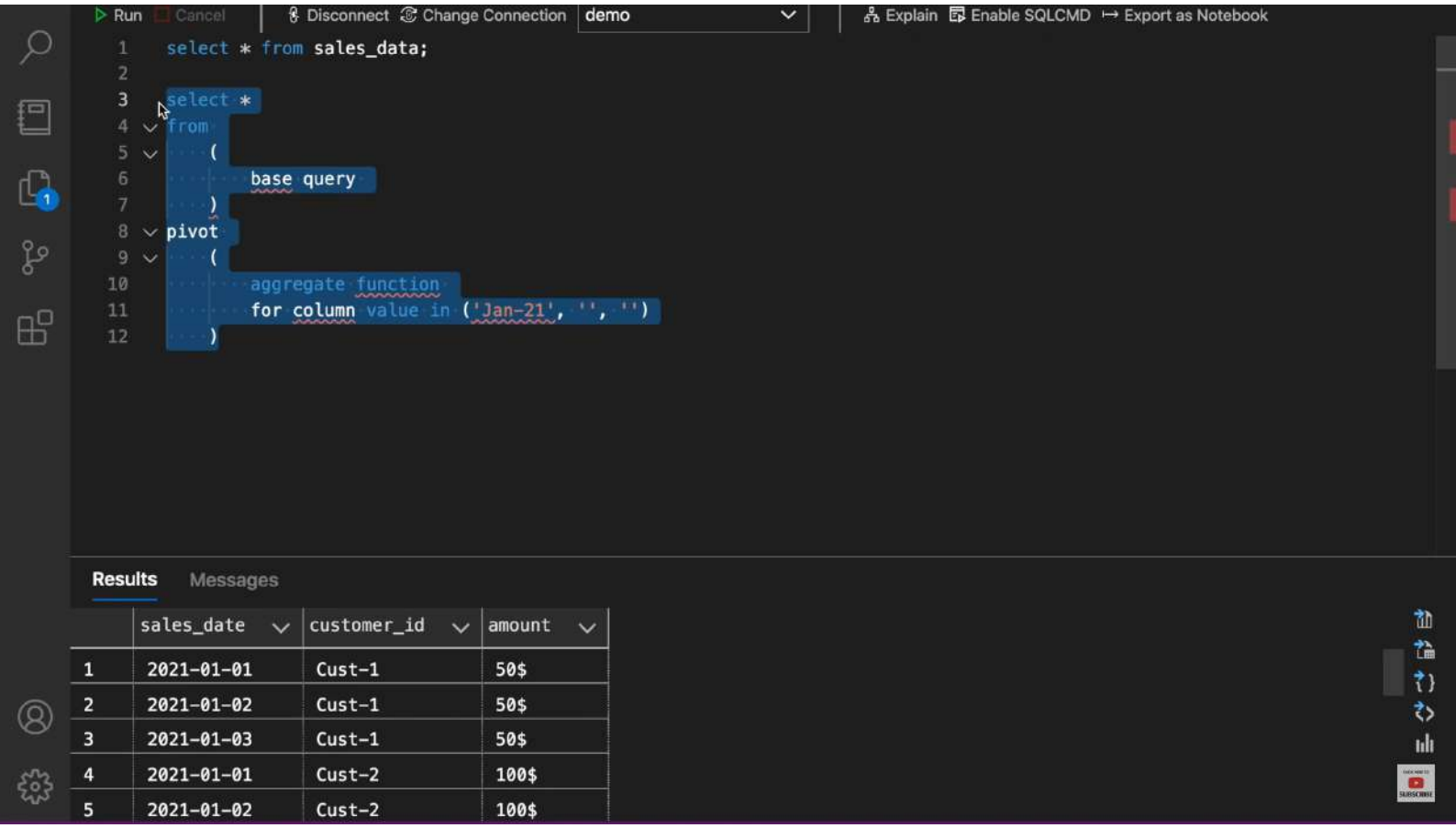
ENG 10:17 AM 01-08-2023

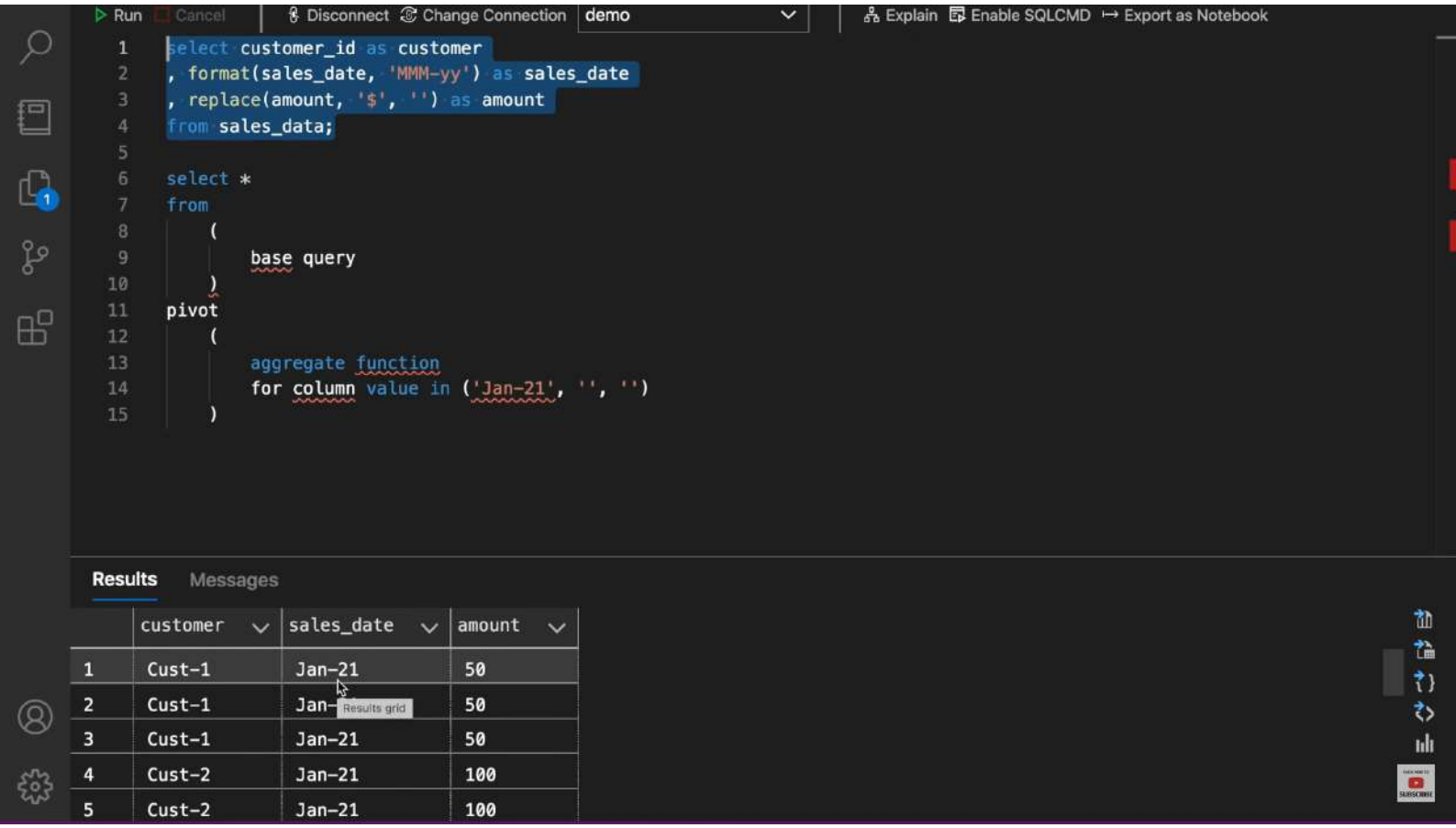

Query Database Server Tools Scripting Help
cursor*
Limit to 1000 rows
job_desc VARCHAR(20),
salary INT,
branch_id INT,
CONSTRAINT FK_branchId_backup FOREIGN KEY(branch_id) REFERENCES branch(bran
delimiter $$
create procedure backupEmployee()
begin
declare cur cursor for select * from employee;
end$$
delimiter ;
SQL Additions
Automatic context help is disabled.
help for the current caret position

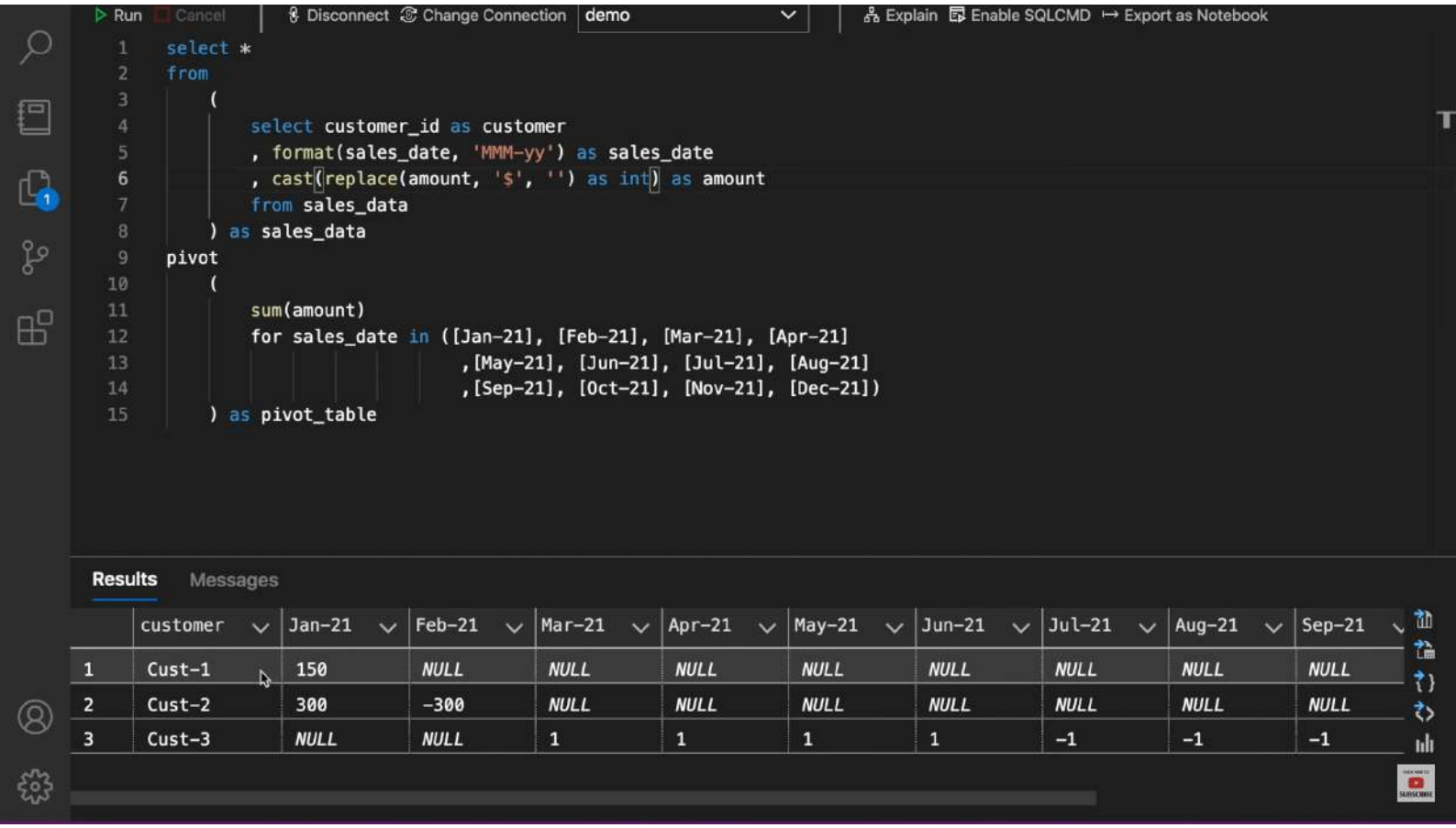
```
11  branch_id INT,
12  CONSTRAINT FK_branchId_backup FOREIGN KEY(branch_id) REFERENCES branch(bran
13
14  delimiter $$
15  create procedure backupEmployee()
16  begin
17      declare cur cursor for select * from employee;
18      open cur;
19
20  end$$
21  delimiter ;
```

emp_id	ename	job_desc	salary	branch_id
13	Abinaya	ENGINEER	2100000	2
14	Vidya	ADMIN	2200000	
15	Ranjani	ENGINEER	2100000	

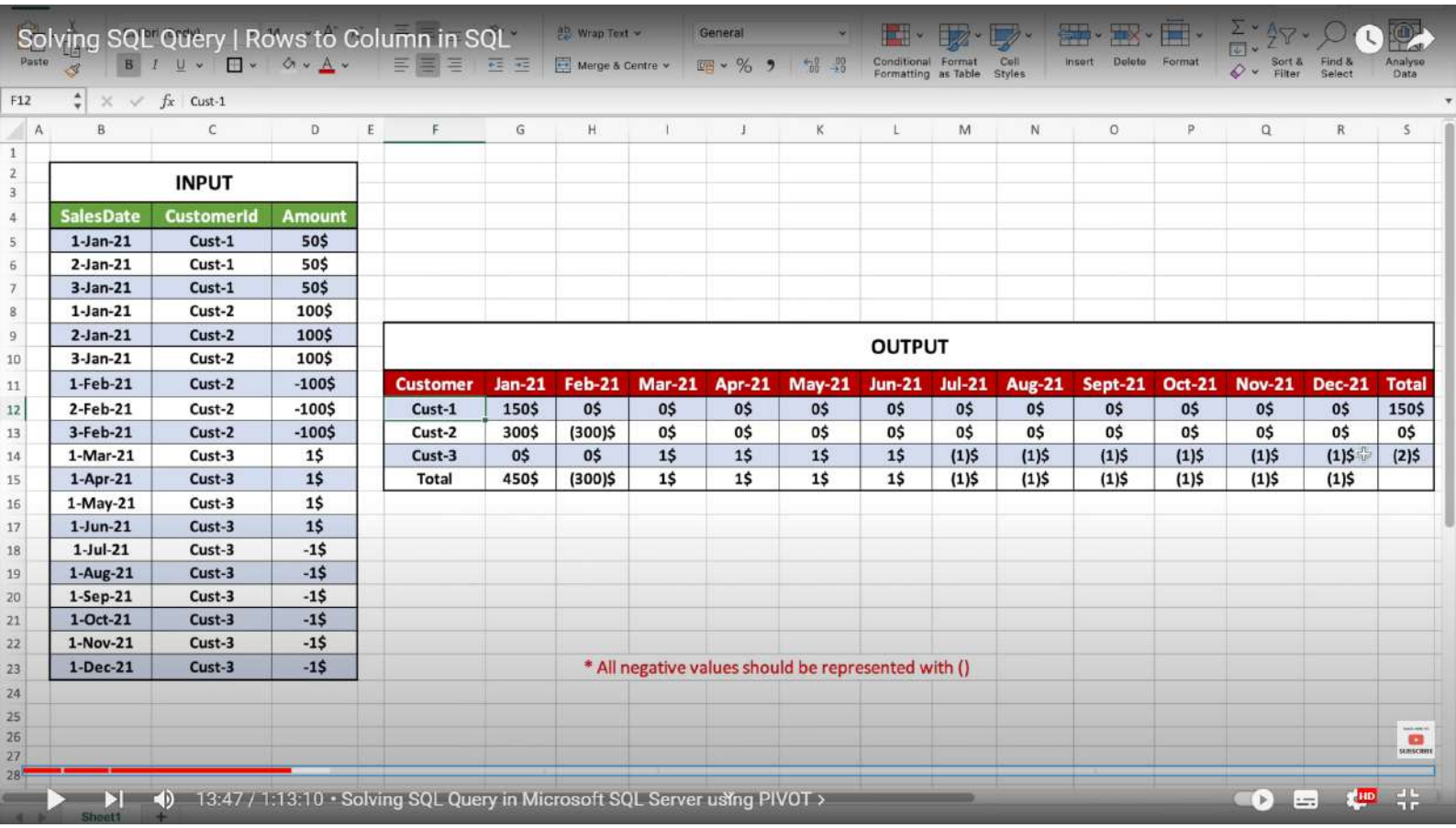

MySQL Workbench
14 delimiter $$
15 create procedure backupEmployee()
16 begin
17 declare id,salary,brid int;
18 declare ename,jdesc varchar(30);
19 declare cur cursor for select * from employee;
20 open cur;
21 fetch cur into id,ename
22 end$$
23 delimiter ;
24
emp_id ename job_desc salary branch_id
1 Ram ADMIN 1000000 2
2 Harini MANAGER 2500000 2
3 George SALES 2000000 1
4 Ramya SALES 1300000 2


Run
Cancel
Disconnect
Change Connection
demo
Explain
Enable SQLCMD
Export as Notebook
select * from sales_data;
select *
from
(
base query
)
pivot
(
aggregate function
for column value in ('Jan-21', '', '')
)
Results
Messages
sales_date
customer_id
amount
1 2021-01-01 Cust-1 50$
2 2021-01-02 Cust-1 50$
3 2021-01-03 Cust-1 50$
4 2021-01-01 Cust-2 100$
5 2021-01-02 Cust-2 100$


Run
Cancel
Disconnect
Change Connection
demo
Explain
Enable SQLCMD
Export as Notebook
select customer_id as customer
, format(sales_date, 'MMM-yy') as sales_date
, replace(amount, '$', '') as amount
from sales_data;
select *
from
(
base query
)
pivot
(
aggregate function
for column value in ('Jan-21', '', '')
)
Results
Messages
customer
sales_date
amount
1 Cust-1 Jan-21 50
2 Cust-1 Jan- 50
3 Cust-1 Jan-21 50
4 Cust-2 Jan-21 100
5 Cust-2 Jan-21 100
Results grid



	customer	Jan-21	Feb-21	Mar-21	Apr-21	May-21	Jun-21	Jul-21	Aug-21	Sep-21
1	Cust-1	150	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*
2	Cust-2	300	-300	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*
3	Cust-3	*NULL*	*NULL*	1	1	1	1	-1	-1	-1



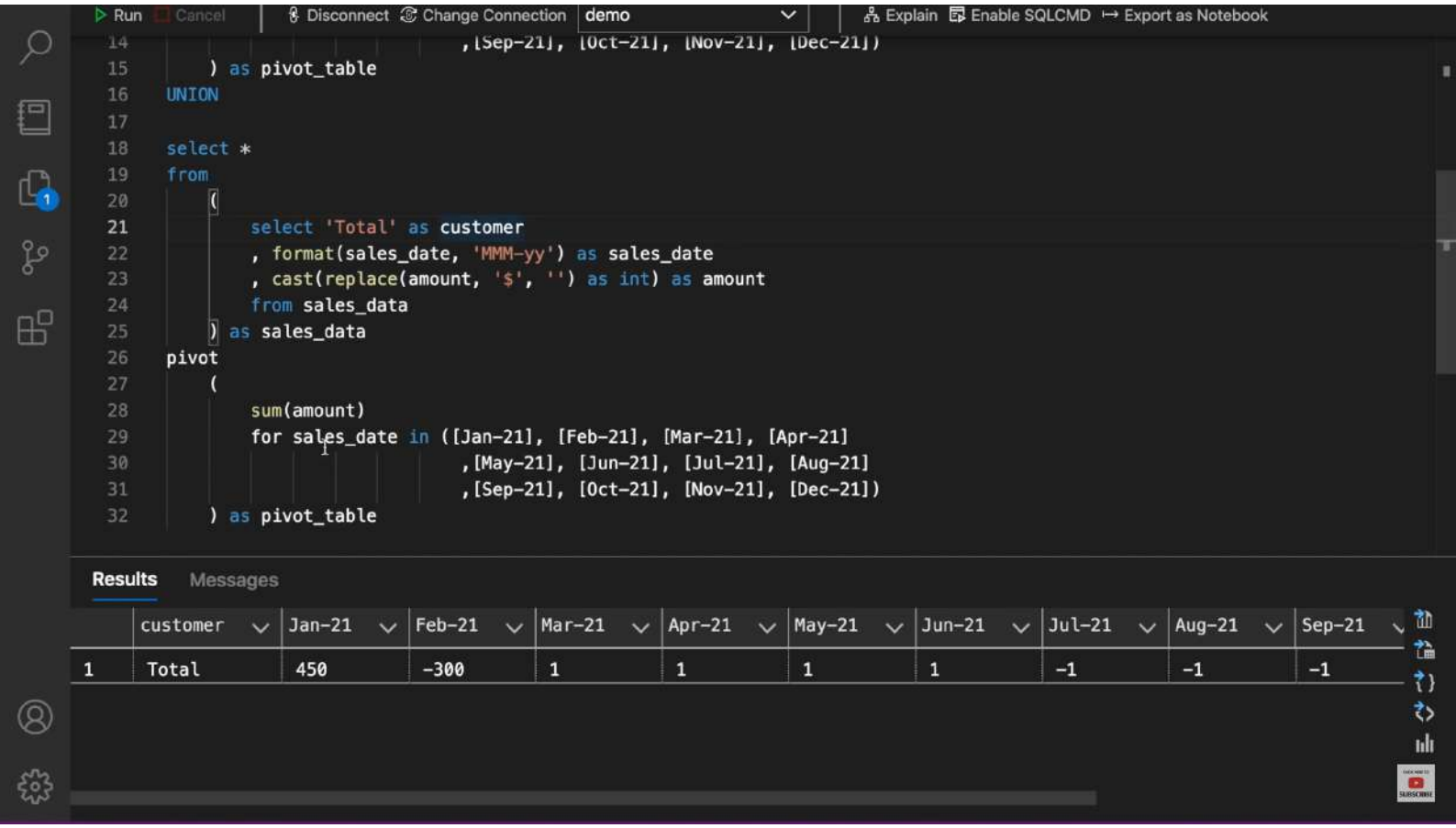
INPUT

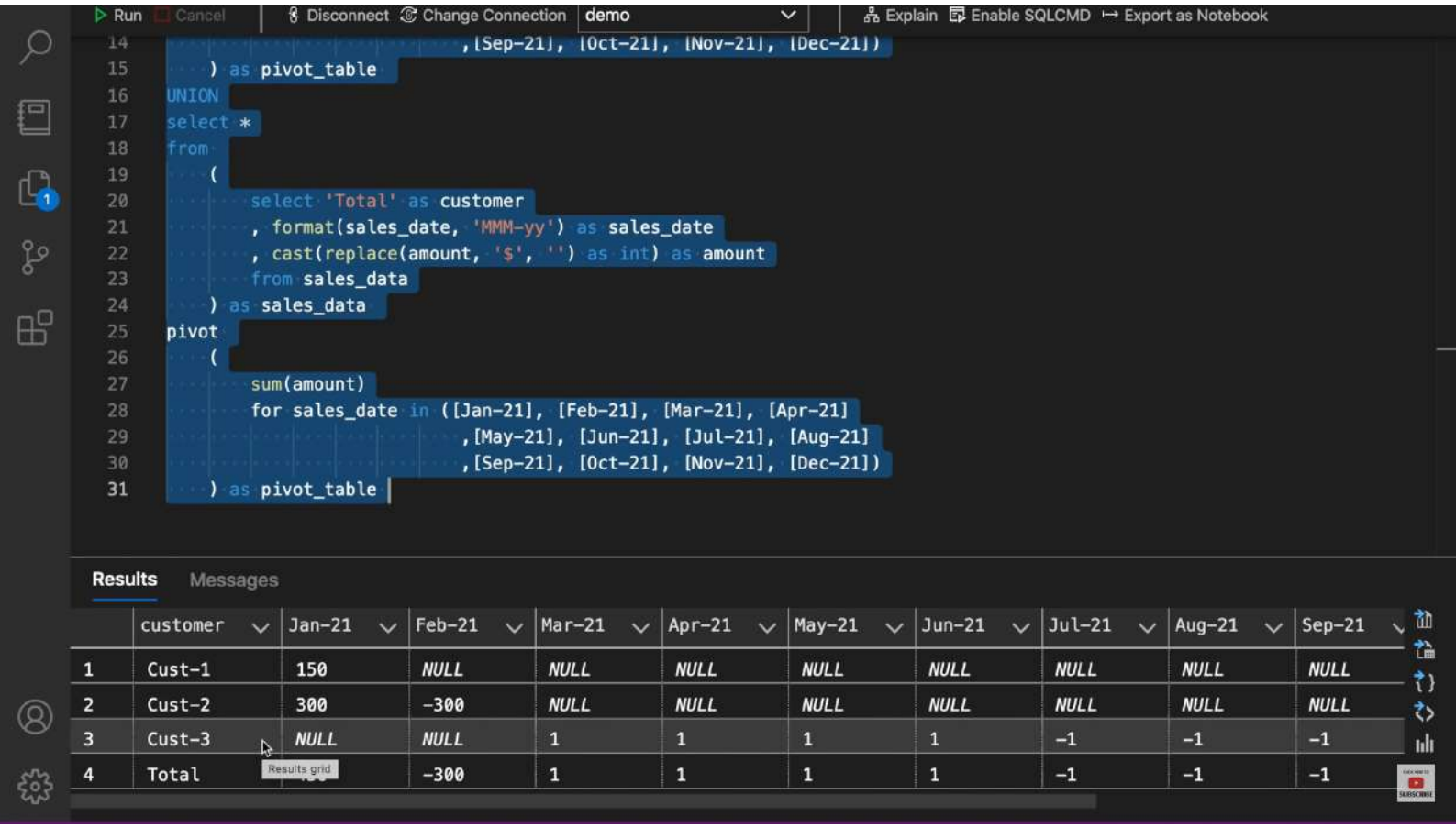
SalesDate	CustomerId	Amount
1-Jan-21	Cust-1	50$
2-Jan-21	Cust-1	50$
3-Jan-21	Cust-1	50$
1-Jan-21	Cust-2	100$
2-Jan-21	Cust-2	100$
3-Jan-21	Cust-2	100$
1-Feb-21	Cust-2	-100$
2-Feb-21	Cust-2	-100$
3-Feb-21	Cust-2	-100$
1-Mar-21	Cust-3	1$
1-Apr-21	Cust-3	1$
1-May-21	Cust-3	1$
1-Jun-21	Cust-3	1$
1-Jul-21	Cust-3	-1$
1-Aug-21	Cust-3	-1$
1-Sep-21	Cust-3	-1$
1-Oct-21	Cust-3	-1$
1-Nov-21	Cust-3	-1$
1-Dec-21	Cust-3	-1$

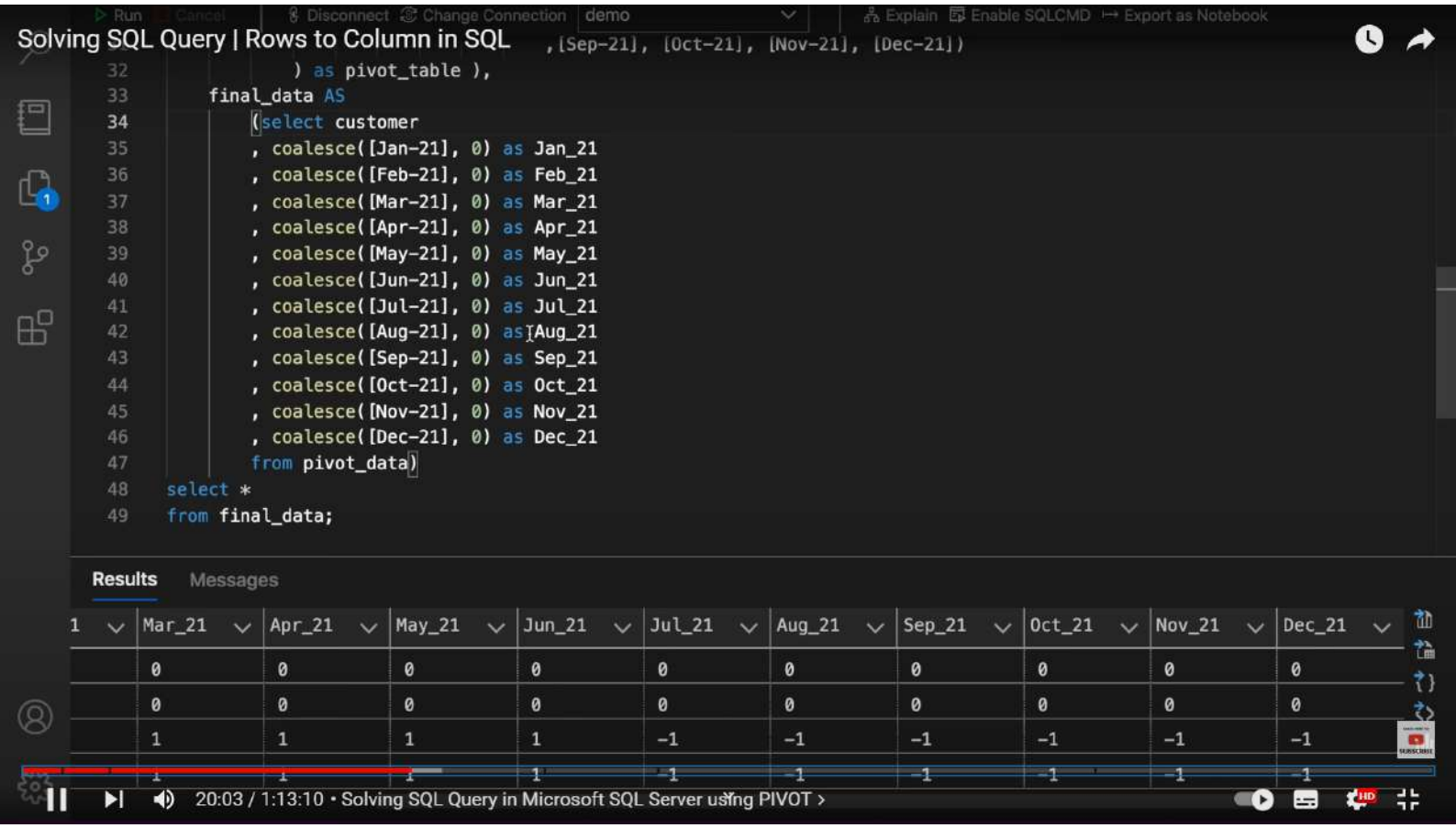
OUTPUT

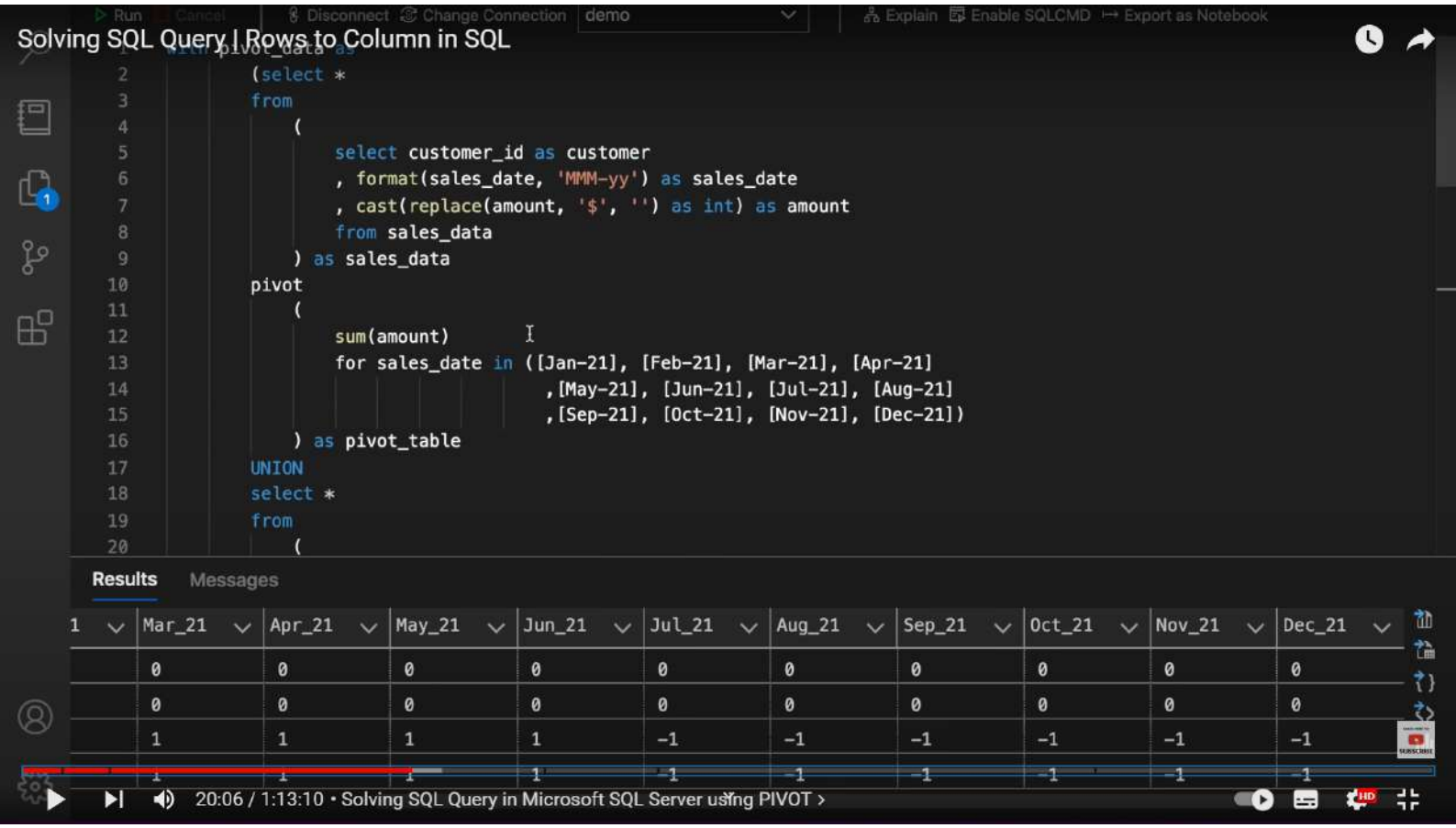
Customer	Jan-21	Feb-21	Mar-21	Apr-21	May-21	Jun-21	Jul-21	Aug-21	Sept-21	Oct-21	Nov-21	Dec-21	Total
Cust-1	150$	0$	0$	0$	0$	0$	0$	0$	0$	0$	0$	0$	150$
Cust-2	300$	(300)$	0$	0$	0$	0$	0$	0$	0$	0$	0$	0$	0$
Cust-3	0$	0$	1$	1$	1$	1$	(1)$	(1)$	(1)$	(1)$	(1)$	(1)$	(2)$
Total	450$	(300)$	1$	1$	1$	1$	(1)$	(1)$	(1)$	(1)$	(1)$	(1)$	

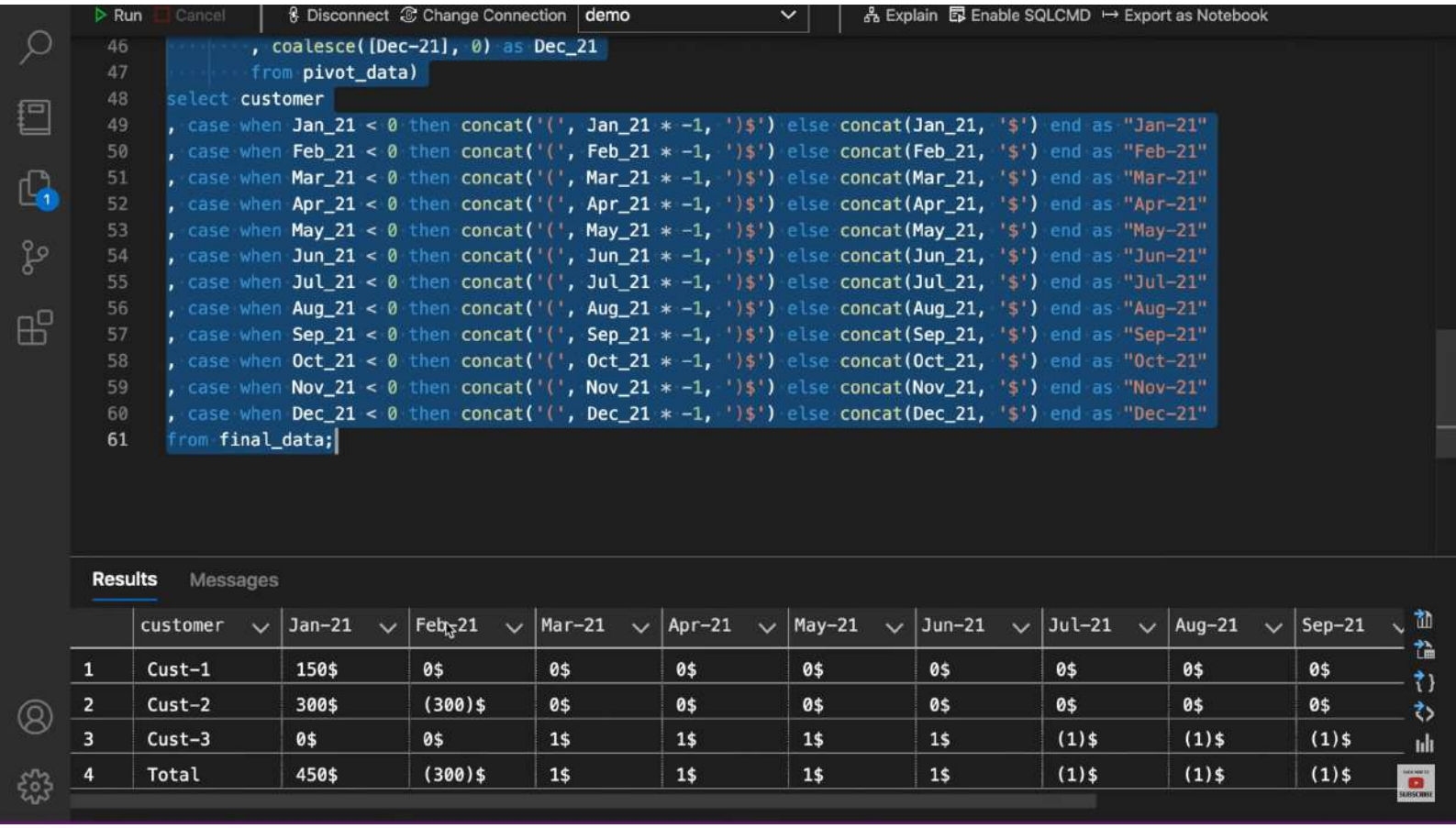
* All negative values should be represented with ()


Run
Cancel
Disconnect
Change Connection
demo
Explain
Enable SQLCMD
Export as Notebook
,[Sep-21], [Oct-21], [Nov-21], [Dec-21])
) as pivot_table
UNION
select *
from
(
select 'Total' as customer
, format(sales_date, 'MMM-yy') as sales_date
, cast(replace(amount, '$', '') as int) as amount
from sales_data
) as sales_data
pivot
(
sum(amount)
for sales_date in ([Jan-21], [Feb-21], [Mar-21], [Apr-21]
,[May-21], [Jun-21], [Jul-21], [Aug-21]
,[Sep-21], [Oct-21], [Nov-21], [Dec-21])
) as pivot_table
Results
Messages
customer
Jan-21
Feb-21
Mar-21
Apr-21
May-21
Jun-21
Jul-21
Aug-21
Sep-21
1
Total
450
-300
1
1
1
1
-1
-1
-1

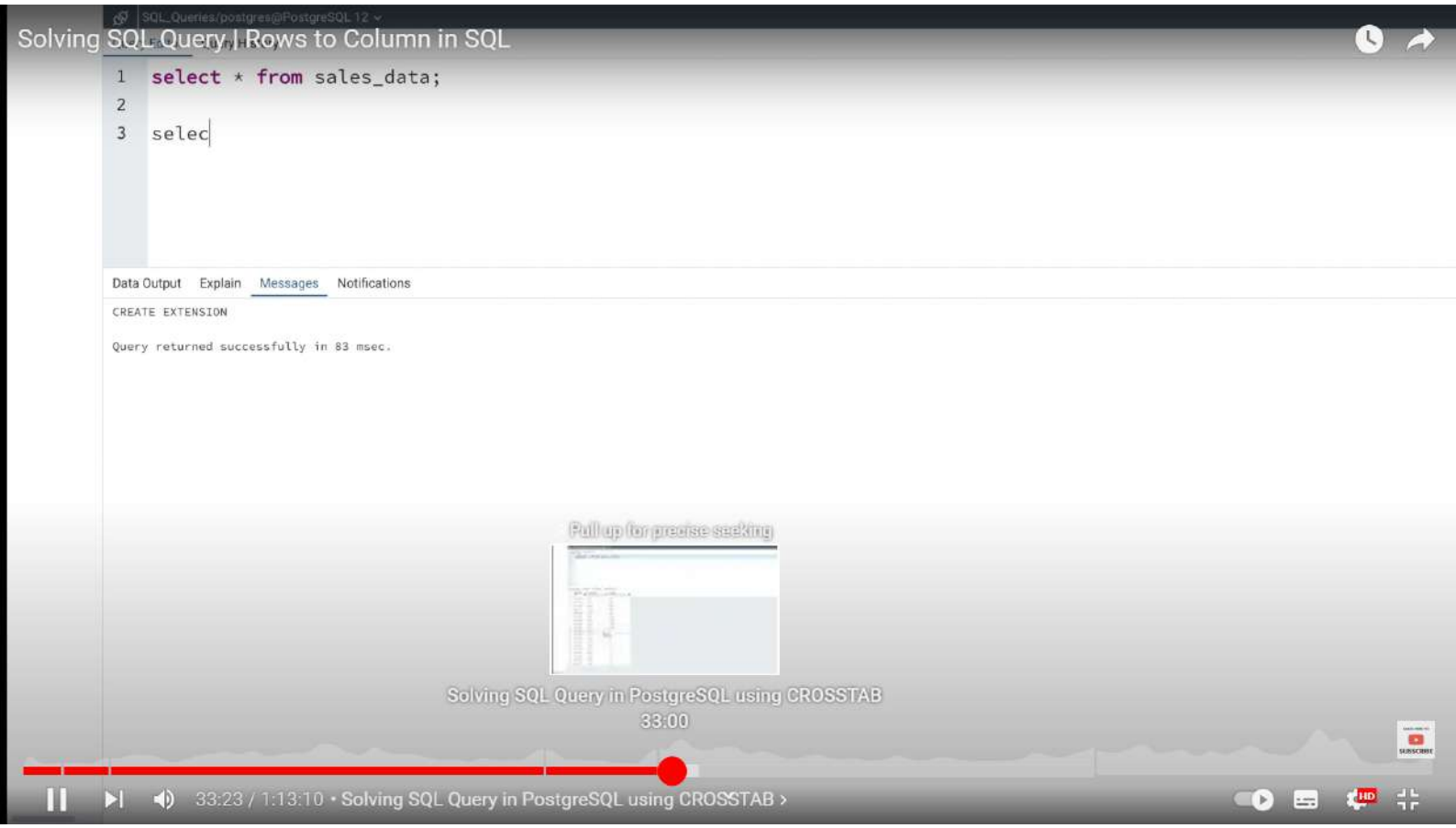

Run
Cancel
Disconnect
Change Connection
demo
Explain
Enable SQLCMD
Export as Notebook
,[Sep-21], [Oct-21], [Nov-21], [Dec-21])
) as pivot_table
UNION
select *
from
(
select 'Total' as customer
, format(sales_date, 'MMM-yy') as sales_date
, cast(replace(amount, '$', '') as int) as amount
from sales_data
) as sales_data
pivot
(
sum(amount)
for sales_date in ([Jan-21], [Feb-21], [Mar-21], [Apr-21]
,[May-21], [Jun-21], [Jul-21], [Aug-21]
,[Sep-21], [Oct-21], [Nov-21], [Dec-21])
) as pivot_table
Results
Messages
customer
Jan-21
Feb-21
Mar-21
Apr-21
May-21
Jun-21
Jul-21
Aug-21
Sep-21
1 Cust-1 150 NULL NULL NULL NULL NULL NULL NULL NULL
2 Cust-2 300 -300 NULL NULL NULL NULL NULL NULL NULL
3 Cust-3 NULL NULL 1 1 1 1 -1 -1 -1
4 Total -300 1 1 1 1 -1 -1 -1
Results grid

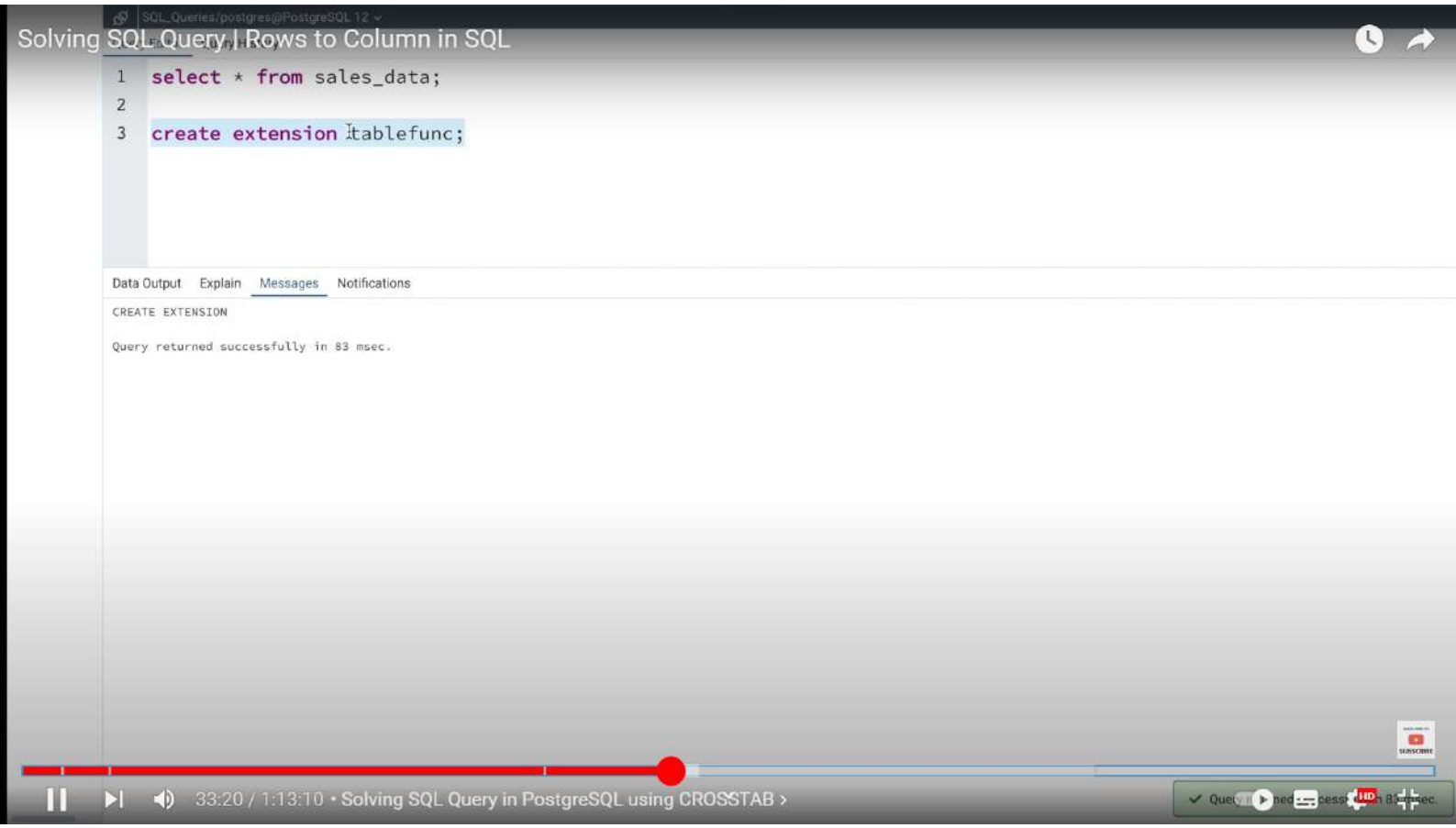

Solving SQL Query | Rows to Column in SQL
Run
Cancel
Disconnect
Change Connection
demo
Explain
Enable SQLCMD
Export as Notebook
,[Sep-21], [Oct-21], [Nov-21], [Dec-21])
) as pivot_table),
final_data AS
(select customer
, coalesce([Jan-21], 0) as Jan_21
, coalesce([Feb-21], 0) as Feb_21
, coalesce([Mar-21], 0) as Mar_21
, coalesce([Apr-21], 0) as Apr_21
, coalesce([May-21], 0) as May_21
, coalesce([Jun-21], 0) as Jun_21
, coalesce([Jul-21], 0) as Jul_21
, coalesce([Aug-21], 0) as Aug_21
, coalesce([Sep-21], 0) as Sep_21
, coalesce([Oct-21], 0) as Oct_21
, coalesce([Nov-21], 0) as Nov_21
, coalesce([Dec-21], 0) as Dec_21
from pivot_data)
select *
from final_data;
Results
Messages
Mar_21
Apr_21
May_21
Jun_21
Jul_21
Aug_21
Sep_21
Oct_21
Nov_21
Dec_21
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
1 1 1 1 -1 -1 -1 -1 -1 -1
1 1 1 1 -1 -1 -1 -1 -1 -1
20:03 / 1:13:10 • Solving SQL Query in Microsoft SQL Server using PIVOT ›

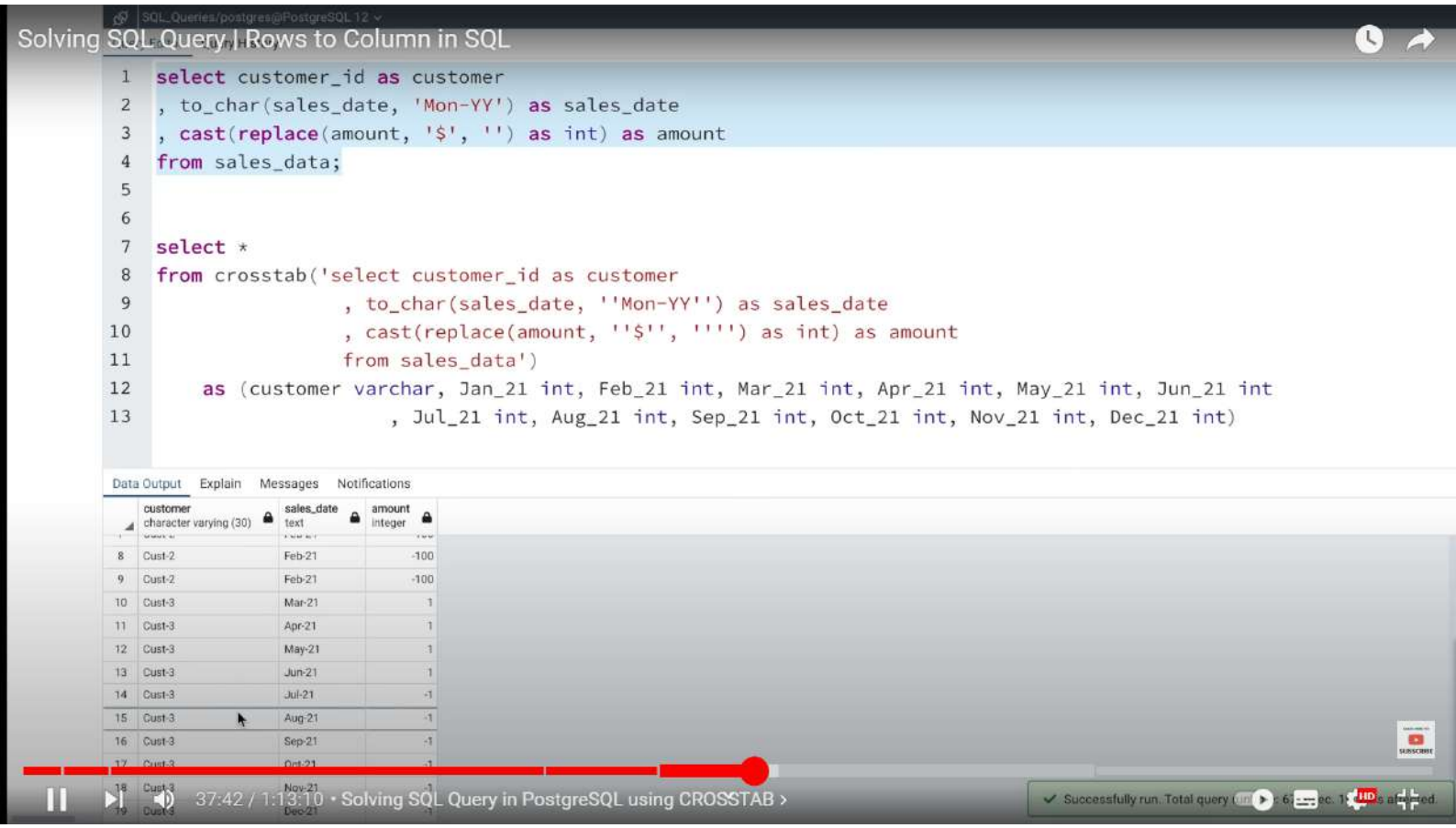

Run
Cancel
Disconnect
Change Connection
demo
Explain
Enable SQLCMD
Export as Notebook
Solving SQL Query | Rows to Column in SQL
with pivot_data as
(select *
from
(
select customer_id as customer
, format(sales_date, 'MMM-yy') as sales_date
, cast(replace(amount, '$', '') as int) as amount
from sales_data
) as sales_data
pivot
(
sum(amount)
for sales_date in ([Jan-21], [Feb-21], [Mar-21], [Apr-21]
,[May-21], [Jun-21], [Jul-21], [Aug-21]
,[Sep-21], [Oct-21], [Nov-21], [Dec-21])
) as pivot_table
UNION
select *
from
(
Results
Messages
Mar_21 Apr_21 May_21 Jun_21 Jul_21 Aug_21 Sep_21 Oct_21 Nov_21 Dec_21
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 1 1 1 -1 -1 -1 -1 -1 -1
1 1 1 1 -1 -1 -1 -1 -1 -1
20:06 / 1:13:10 • Solving SQL Query in Microsoft SQL Server using PIVOT ›



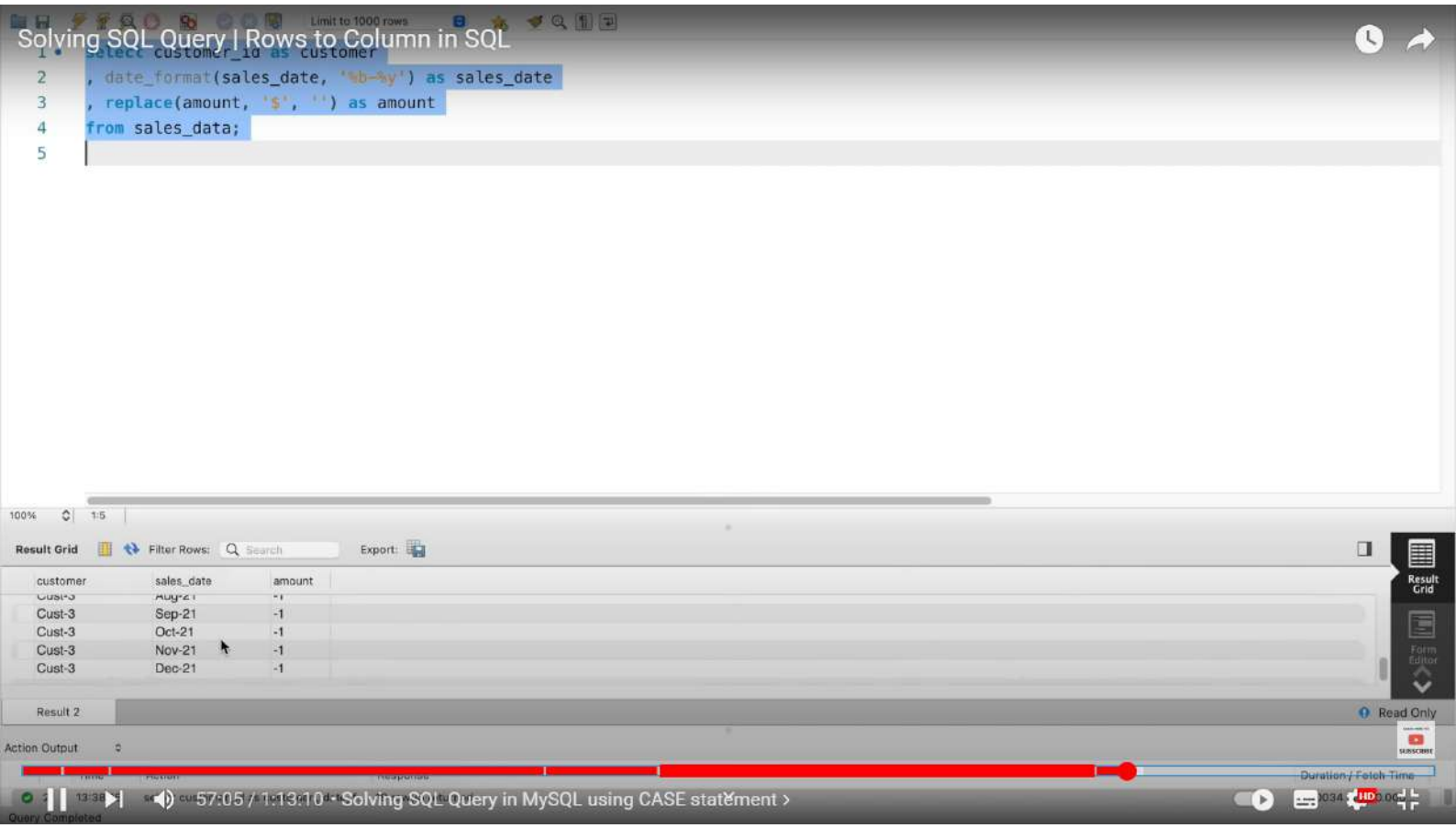
	customer	Jan-21	Feb-21	Mar-21	Apr-21	May-21	Jun-21	Jul-21	Aug-21	Sep-21
1	Cust-1	150$	0$	0$	0$	0$	0$	0$	0$	0$
2	Cust-2	300$	(300)$	0$	0$	0$	0$	0$	0$	0$
3	Cust-3	0$	0$	1$	1$	1$	1$	(1)$	(1)$	(1)$
4	Total	450$	(300)$	1$	1$	1$	1$	(1)$	(1)$	(1)$

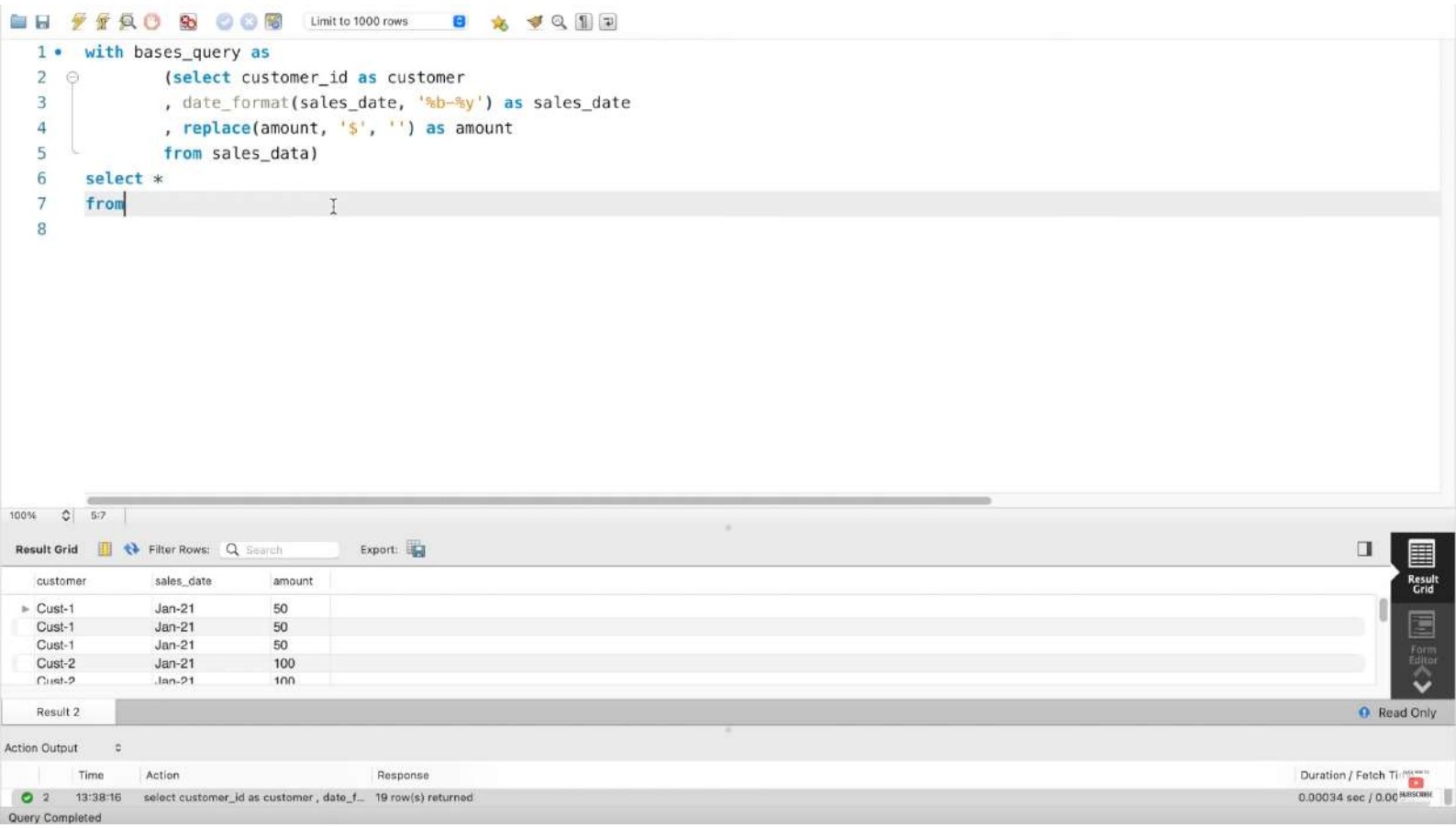

Solving SQL Query | Rows to Column in SQL
SQL_Queries/postgres@PostgreSQL 12
Query History
1 select * from sales_data;
2
3 selec
Data Output
Explain
Messages
Notifications
CREATE EXTENSION
Query returned successfully in 83 msec.
Pull up for precise seeking
Solving SQL Query in PostgreSQL using CROSSTAB
33:00
33:23 / 1:13:10 • Solving SQL Query in PostgreSQL using CROSSTAB ›

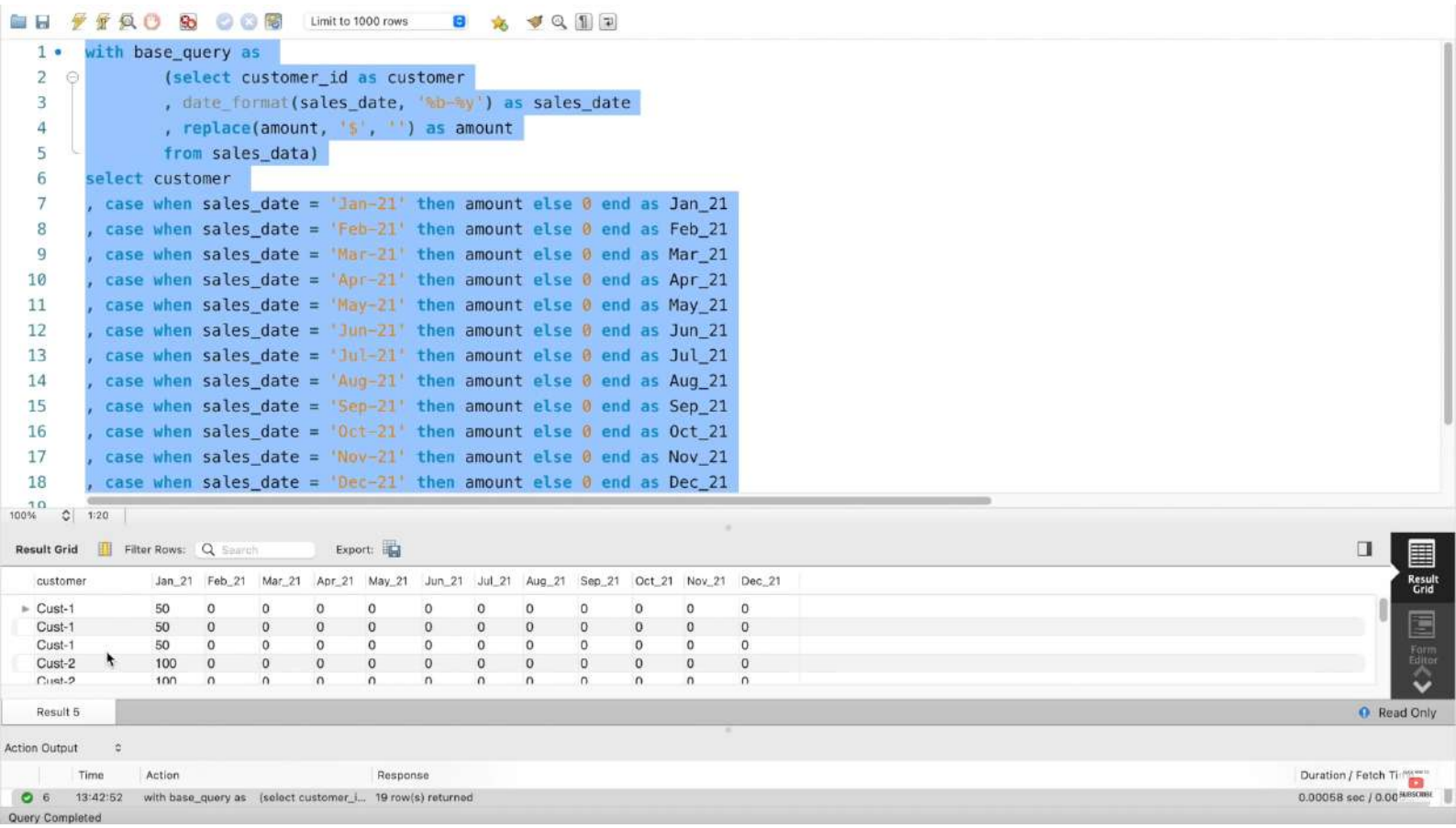

Solving SQL Query | Rows to Column in SQL
SQL_Queries/postgres@PostgreSQL 12
1 select * from sales_data;
2
3 create extension tablefunc;
Data Output
Explain
Messages
Notifications
CREATE EXTENSION
Query returned successfully in 83 msec.
33:20 / 1:13:10 • Solving SQL Query in PostgreSQL using CROSSTAB ›

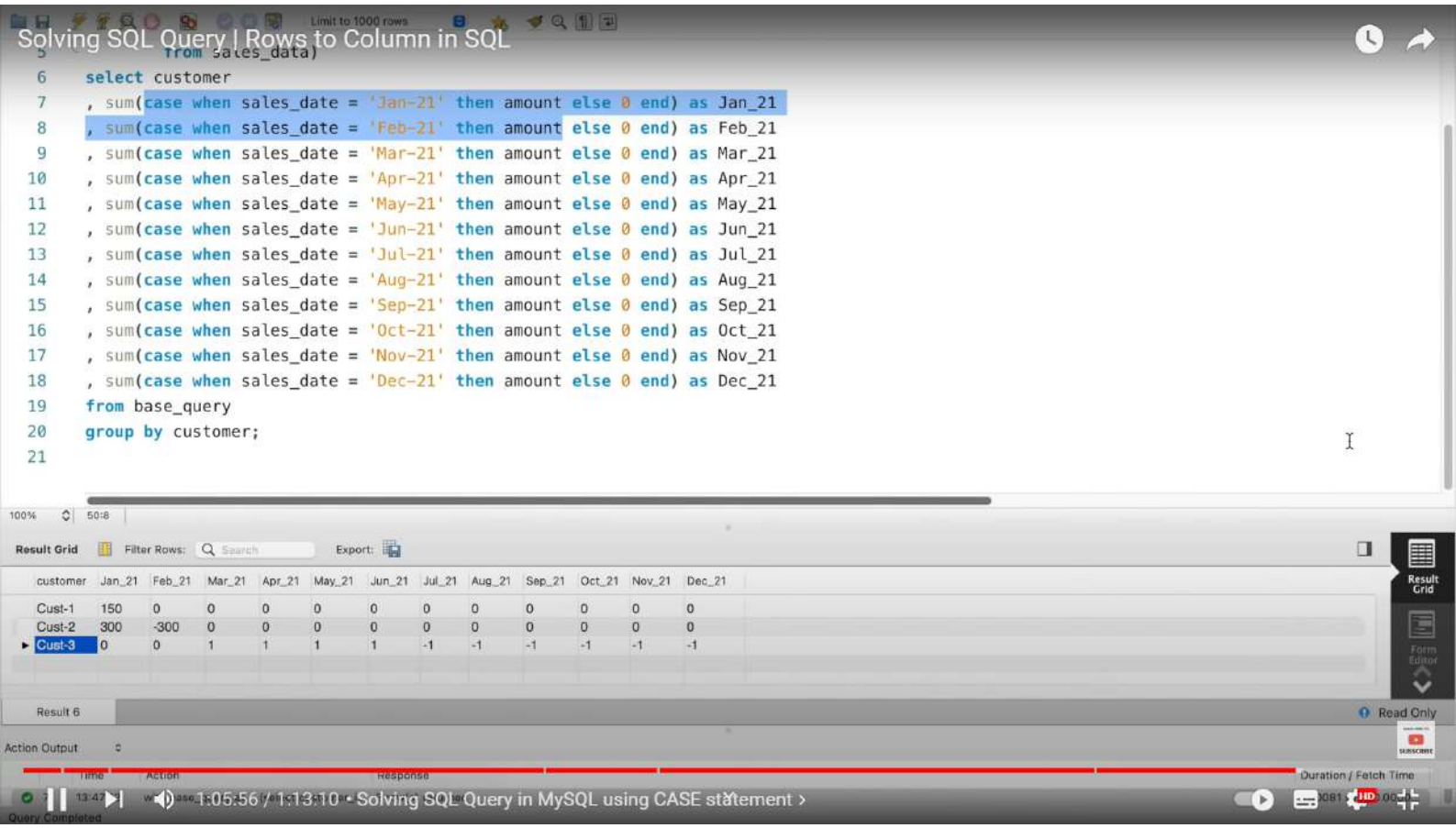

Solving SQL Query | Rows to Column in SQL
SQL_Queries/postgres@PostgreSQL 12
select customer_id as customer
, to_char(sales_date, 'Mon-YY') as sales_date
, cast(replace(amount, '$', '') as int) as amount
from sales_data;
select *
from crosstab('select customer_id as customer
, to_char(sales_date, ''Mon-YY'') as sales_date
, cast(replace(amount, ''$'', '''') as int) as amount
from sales_data')
as (customer varchar, Jan_21 int, Feb_21 int, Mar_21 int, Apr_21 int, May_21 int, Jun_21 int
, Jul_21 int, Aug_21 int, Sep_21 int, Oct_21 int, Nov_21 int, Dec_21 int)
Data Output
Explain
Messages
Notifications
customer character varying (30)
sales_date text
amount integer
8 Cust-2 Feb-21 -100
9 Cust-2 Feb-21 -100
10 Cust-3 Mar-21 1
11 Cust-3 Apr-21 1
12 Cust-3 May-21 1
13 Cust-3 Jun-21 1
14 Cust-3 Jul-21 -1
15 Cust-3 Aug-21 -1
16 Cust-3 Sep-21 -1
37:42 / 1:13:10 • Solving SQL Query in PostgreSQL using CROSSTAB ›
Successfully run. Total query

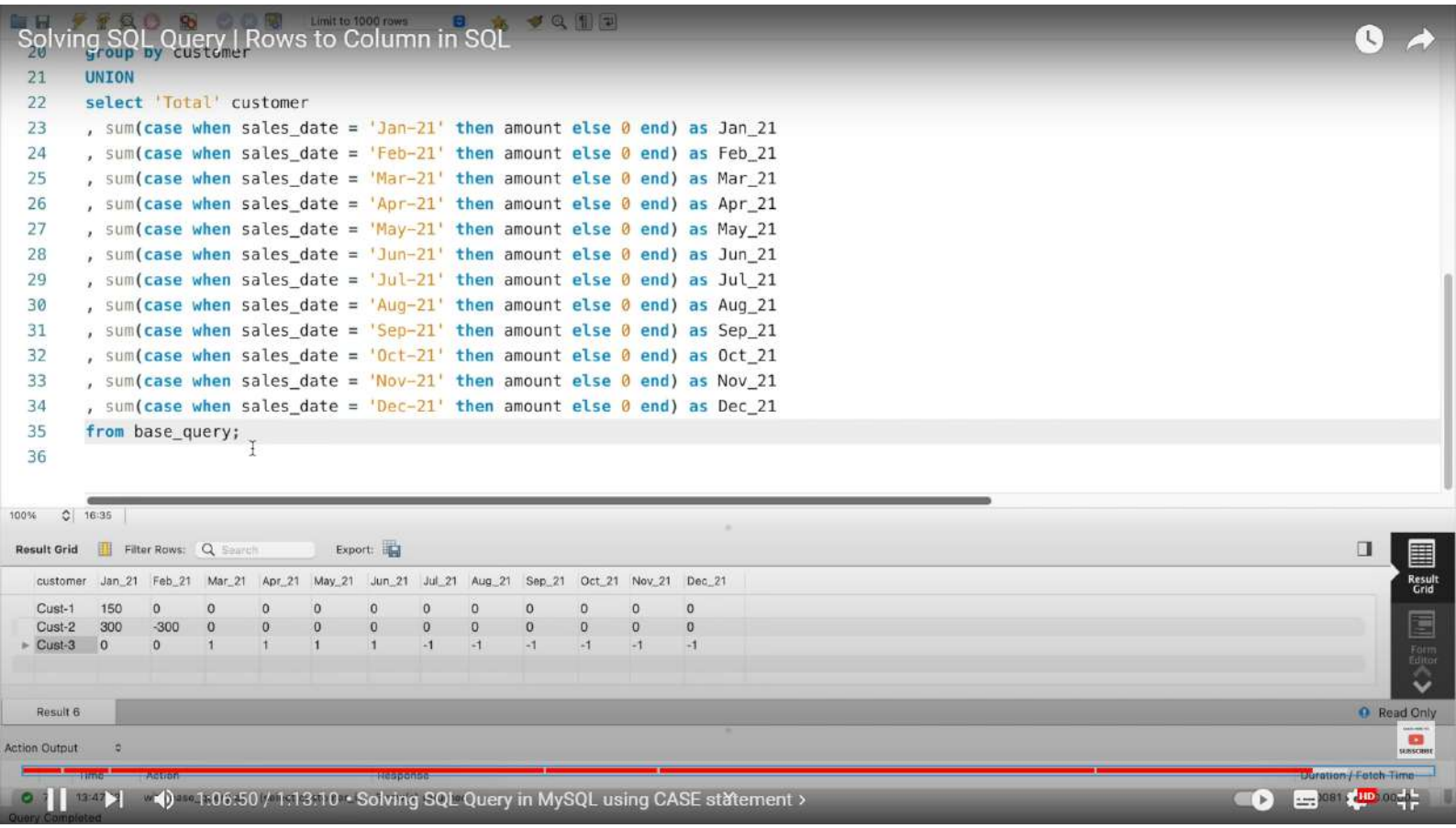

Solving SQL Query | Rows to Column in SQL
SQL_Queries/postgres@PostgreSQL 12
12
13
14 select *
15 from crosstab('select customer_id as customer
16 , to_char(sales_date, ''Mon-YY'') as sales_date
17 , sum(cast(replace(amount, ''$'', '''') as int)) as amount
18 from sales_data
19 group by customer_id, to_char(sales_date, ''Mon-YY'')
20 order by 1')
21 as (customer varchar, Jan_21 bigint, Feb_21 bigint, Mar_21 bigint, Apr_21 bigint, May_21 bigint, Jun_21 big
22 , Jul_21 bigint, Aug_21 bigint, Sep_21 bigint, Oct_21 bigint, Nov_21 bigint, Dec_21 bigint)
23
24
25
Data Output
Explain
Messages
Notifications
customer character varying
jan_21 bigint
feb_21 bigint
mar_21 bigint
apr_21 bigint
may_21 bigint
jun_21 bigint
jul_21 bigint
aug_21 bigint
sep_21 bigint
oct_21 bigint
nov_21 bigint
dec_21 bigint
1 Cust-1 150 [null] [null] [null] [null] [null] [null]
2 Cust-2 300 -300 [null] [null] [null] [null] [null]
3 Cust-3 -1 -1 1 1 1 -1 -1 [null] [null]
Pull up to precise seeking
Solving SQL Query in PostgreSQL using CROSSTAB
42:45
42:50 / 1:13:10 • Solving SQL Query in PostgreSQL using CROSSTAB ›

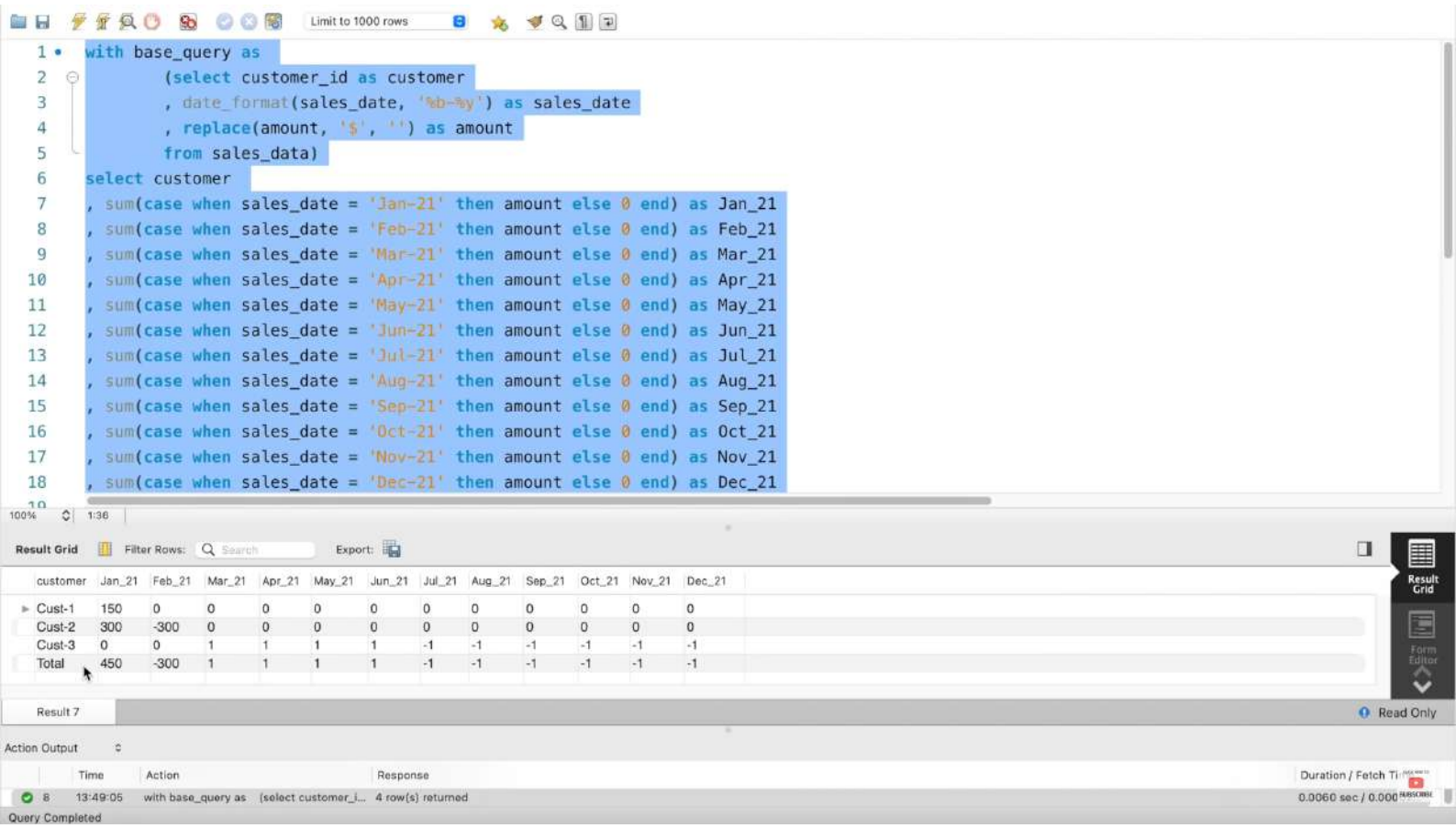

Solving SQL Query | Rows to Column in SQL
Limit to 1000 rows
select customer_id as customer
, date_format(sales_date, '%b-%y') as sales_date
, replace(amount, '$', '') as amount
from sales_data;
100%
1:5
Result Grid
Filter Rows:
Search
Export:
customer
sales_date
amount
Cust-3
Sep-21
-1
Cust-3
Oct-21
-1
Cust-3
Nov-21
-1
Cust-3
Dec-21
-1
Result 2
Read Only
Action Output
Duration / Fetch Time
Query Completed
57:05 / 1:13:10 • Solving SQL Query in MySQL using CASE statement ›


Limit to 1000 rows
1 • with bases_query as
2 (select customer_id as customer
3 , date_format(sales_date, '%b-%y') as sales_date
4 , replace(amount, '$', '') as amount
5 from sales_data)
6 select *
7 from
8
100%
5:7
Result Grid
Filter Rows:
Search
Export:
customer
sales_date
amount
Cust-1
Jan-21
50
Cust-1
Jan-21
50
Cust-1
Jan-21
50
Cust-2
Jan-21
100
Result 2
Read Only
Action Output
Time
Action
Response
Duration / Fetch Ti
2
13:38:16
select customer_id as customer , date_f...
19 row(s) returned
0.00034 sec / 0.00
Query Completed


Limit to 1000 rows
with base_query as
(select customer_id as customer
, date_format(sales_date, '%b-%y') as sales_date
, replace(amount, '$', '') as amount
from sales_data)
select customer
, case when sales_date = 'Jan-21' then amount else 0 end as Jan_21
, case when sales_date = 'Feb-21' then amount else 0 end as Feb_21
, case when sales_date = 'Mar-21' then amount else 0 end as Mar_21
, case when sales_date = 'Apr-21' then amount else 0 end as Apr_21
, case when sales_date = 'May-21' then amount else 0 end as May_21
, case when sales_date = 'Jun-21' then amount else 0 end as Jun_21
, case when sales_date = 'Jul-21' then amount else 0 end as Jul_21
, case when sales_date = 'Aug-21' then amount else 0 end as Aug_21
, case when sales_date = 'Sep-21' then amount else 0 end as Sep_21
, case when sales_date = 'Oct-21' then amount else 0 end as Oct_21
, case when sales_date = 'Nov-21' then amount else 0 end as Nov_21
, case when sales_date = 'Dec-21' then amount else 0 end as Dec_21
100%
1:20
Result Grid
Filter Rows:
Search
Export:
customer Jan_21 Feb_21 Mar_21 Apr_21 May_21 Jun_21 Jul_21 Aug_21 Sep_21 Oct_21 Nov_21 Dec_21
Cust-1 50 0 0 0 0 0 0 0 0 0 0 0
Cust-1 50 0 0 0 0 0 0 0 0 0 0 0
Cust-1 50 0 0 0 0 0 0 0 0 0 0 0
Cust-2 100 0 0 0 0 0 0 0 0 0 0 0
Result 5
Read Only
Action Output
Time Action Response Duration / Fetch Ti
6 13:42:52 with base_query as (select customer_i... 19 row(s) returned 0.00058 sec / 0.00
Query Completed


Solving SQL Query | Rows to Column in SQL
from sales_data)
select customer
, sum(case when sales_date = 'Jan-21' then amount else 0 end) as Jan_21
, sum(case when sales_date = 'Feb-21' then amount else 0 end) as Feb_21
, sum(case when sales_date = 'Mar-21' then amount else 0 end) as Mar_21
, sum(case when sales_date = 'Apr-21' then amount else 0 end) as Apr_21
, sum(case when sales_date = 'May-21' then amount else 0 end) as May_21
, sum(case when sales_date = 'Jun-21' then amount else 0 end) as Jun_21
, sum(case when sales_date = 'Jul-21' then amount else 0 end) as Jul_21
, sum(case when sales_date = 'Aug-21' then amount else 0 end) as Aug_21
, sum(case when sales_date = 'Sep-21' then amount else 0 end) as Sep_21
, sum(case when sales_date = 'Oct-21' then amount else 0 end) as Oct_21
, sum(case when sales_date = 'Nov-21' then amount else 0 end) as Nov_21
, sum(case when sales_date = 'Dec-21' then amount else 0 end) as Dec_21
from base_query
group by customer;
Result Grid
customer Jan_21 Feb_21 Mar_21 Apr_21 May_21 Jun_21 Jul_21 Aug_21 Sep_21 Oct_21 Nov_21 Dec_21
Cust-1 150 0 0 0 0 0 0 0 0 0 0 0
Cust-2 300 -300 0 0 0 0 0 0 0 0 0 0
Cust-3 0 0 1 1 1 1 -1 -1 -1 -1 -1 -1
Result 6
Read Only
Action Output
1:05:56 / 1:13:10 • Solving SQL Query in MySQL using CASE statement ›


Solving SQL Query | Rows to Column in SQL
group by customer
UNION
select 'Total' customer
, sum(case when sales_date = 'Jan-21' then amount else 0 end) as Jan_21
, sum(case when sales_date = 'Feb-21' then amount else 0 end) as Feb_21
, sum(case when sales_date = 'Mar-21' then amount else 0 end) as Mar_21
, sum(case when sales_date = 'Apr-21' then amount else 0 end) as Apr_21
, sum(case when sales_date = 'May-21' then amount else 0 end) as May_21
, sum(case when sales_date = 'Jun-21' then amount else 0 end) as Jun_21
, sum(case when sales_date = 'Jul-21' then amount else 0 end) as Jul_21
, sum(case when sales_date = 'Aug-21' then amount else 0 end) as Aug_21
, sum(case when sales_date = 'Sep-21' then amount else 0 end) as Sep_21
, sum(case when sales_date = 'Oct-21' then amount else 0 end) as Oct_21
, sum(case when sales_date = 'Nov-21' then amount else 0 end) as Nov_21
, sum(case when sales_date = 'Dec-21' then amount else 0 end) as Dec_21
from base_query;
Result Grid
Filter Rows:
Export:
customer Jan_21 Feb_21 Mar_21 Apr_21 May_21 Jun_21 Jul_21 Aug_21 Sep_21 Oct_21 Nov_21 Dec_21
Cust-1 150 0 0 0 0 0 0 0 0 0 0 0
Cust-2 300 -300 0 0 0 0 0 0 0 0 0 0
Cust-3 0 0 1 1 1 1 -1 -1 -1 -1 -1 -1
Result 6
Read Only
Action Output
Query Completed
1:06:50 / 1:13:10 • Solving SQL Query in MySQL using CASE statement ›



```
with base_query as
       (select customer_id as customer
       , date_format(sales_date, '%b-%y') as sales_date
       , replace(amount, '$', '') as amount
       from sales_data)
select customer
, sum(case when sales_date = 'Jan-21' then amount else 0 end) as Jan_21
, sum(case when sales_date = 'Feb-21' then amount else 0 end) as Feb_21
, sum(case when sales_date = 'Mar-21' then amount else 0 end) as Mar_21
, sum(case when sales_date = 'Apr-21' then amount else 0 end) as Apr_21
, sum(case when sales_date = 'May-21' then amount else 0 end) as May_21
, sum(case when sales_date = 'Jun-21' then amount else 0 end) as Jun_21
, sum(case when sales_date = 'Jul-21' then amount else 0 end) as Jul_21
, sum(case when sales_date = 'Aug-21' then amount else 0 end) as Aug_21
, sum(case when sales_date = 'Sep-21' then amount else 0 end) as Sep_21
, sum(case when sales_date = 'Oct-21' then amount else 0 end) as Oct_21
, sum(case when sales_date = 'Nov-21' then amount else 0 end) as Nov_21
, sum(case when sales_date = 'Dec-21' then amount else 0 end) as Dec_21
```

Result Grid

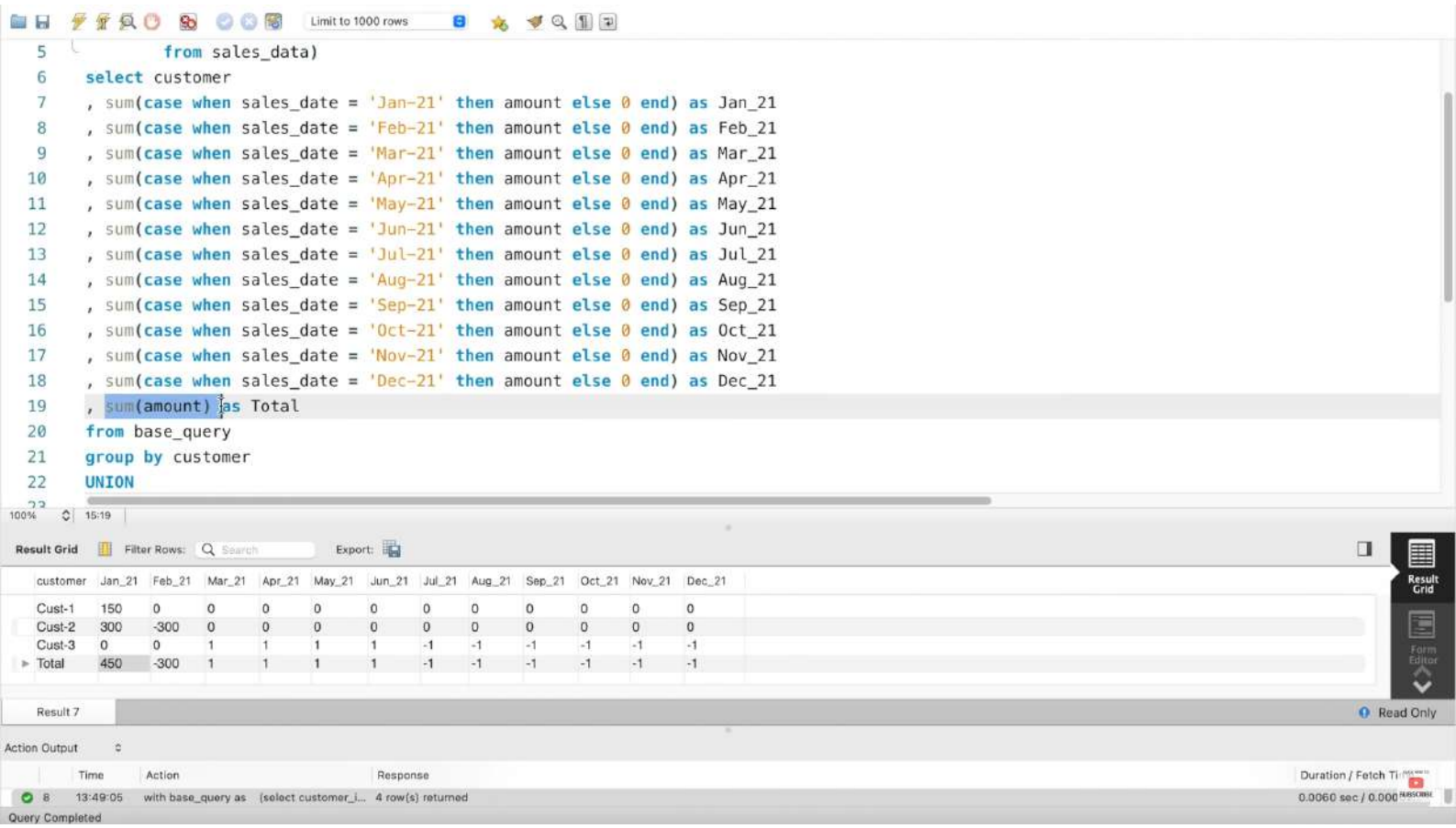
customer	Jan_21	Feb_21	Mar_21	Apr_21	May_21	Jun_21	Jul_21	Aug_21	Sep_21	Oct_21	Nov_21	Dec_21
Cust-1	150	0	0	0	0	0	0	0	0	0	0	0
Cust-2	300	-300	0	0	0	0	0	0	0	0	0	0
Cust-3	0	0	1	1	1	1	-1	-1	-1	-1	-1	-1
Total	450	-300	1	1	1	1	-1	-1	-1	-1	-1	-1

Result 7 — Read Only

Action Output

	Time	Action	Response	Duration / Fetch Ti
8	13:49:05	with base_query as (select customer_i...	4 row(s) returned	0.0060 sec / 0.000

Query Completed



```
        from sales_data)
select customer
, sum(case when sales_date = 'Jan-21' then amount else 0 end) as Jan_21
, sum(case when sales_date = 'Feb-21' then amount else 0 end) as Feb_21
, sum(case when sales_date = 'Mar-21' then amount else 0 end) as Mar_21
, sum(case when sales_date = 'Apr-21' then amount else 0 end) as Apr_21
, sum(case when sales_date = 'May-21' then amount else 0 end) as May_21
, sum(case when sales_date = 'Jun-21' then amount else 0 end) as Jun_21
, sum(case when sales_date = 'Jul-21' then amount else 0 end) as Jul_21
, sum(case when sales_date = 'Aug-21' then amount else 0 end) as Aug_21
, sum(case when sales_date = 'Sep-21' then amount else 0 end) as Sep_21
, sum(case when sales_date = 'Oct-21' then amount else 0 end) as Oct_21
, sum(case when sales_date = 'Nov-21' then amount else 0 end) as Nov_21
, sum(case when sales_date = 'Dec-21' then amount else 0 end) as Dec_21
, sum(amount) as Total
from base_query
group by customer
UNION
```

Result Grid

customer	Jan_21	Feb_21	Mar_21	Apr_21	May_21	Jun_21	Jul_21	Aug_21	Sep_21	Oct_21	Nov_21	Dec_21
Cust-1	150	0	0	0	0	0	0	0	0	0	0	0
Cust-2	300	-300	0	0	0	0	0	0	0	0	0	0
Cust-3	0	0	1	1	1	1	-1	-1	-1	-1	-1	-1
Total	450	-300	1	1	1	1	-1	-1	-1	-1	-1	-1

Result 7 — Read Only

Action Output

	Time	Action	Response	Duration / Fetch Ti...
8	13:49:05	with base_query as (select customer_i...	4 row(s) returned	0.0060 sec / 0.000

Query Completed

Introduction

Definition

A Trigger is a *special type* of **stored procedure** that is invoked *automatically* in response to an **event**.

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Introduction

Definition

A Trigger is a *special type* of **stored procedure** that is invoked *automatically* in response to an **event**.

It is a database object that reside in the **system catalog** that fires automatically when a data modification is made against a table.

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Stored Procedure vs Trigger

IntelliPaat

Stored Procedure

- Set of SQL statements, which has to be *explicitly called* by the programmer
- It is done so by the statement:
  exec [procedure_name];
- Can accept parameters
- Can return a value

Trigger

- Set of SQL statements, which is *automatically called* when an event occurs
- Fires when a data modification is done against a table
- Cannot accept parameters
- Cannot return a value

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Stored Procedure vs Trigger

IntelliPaat

Stored Procedure

- Set of SQL statements, which has to be *explicitly called* by the programmer
- It is done so by the statement:
  exec [procedure_name];
- Can accept parameters
- Can return a value
- Can use transaction statements

Trigger

- Set of SQL statements, which is *automatically called* when an event occurs
- Fires when a data modification is done against a table
- Cannot accept parameters
- Cannot return a value
- Cannot use transaction statements

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Types of Trigger

Generally there are **2 types** of Triggers according to SQL standards

Row-level Triggers

It fires **for each row** that is inserted, updated or deleted

Statement-level Triggers

It fires **only once** for each event regardless of how many rows are affected

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Types of Trigger

IntelliPaat

6 types of Triggers in MySQL

Defined on DML statements like **insert**, **update**, **delete**

2 variations for each – **before** and **after**

- **before insert**
- **after insert**
- **before update**
- **after update**
- **before delete**
- **after delete**

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Syntax

```
CREATE TRIGGER [trigger_name]
[trigger_time] (BEFORE | AFTER)
[trigger_event] (INSERT | UPDATE | DELETE)
ON [table_name]
FOR each row
BEGIN
  --variable declarations
  --trigger code
END
```

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Syntax

NEW keyword

NEW is a pseudo-record name that refers to the new table row for insert and update operations in row-level triggers

OLD keyword

OLD is a pseudo-record name that refers to the old table row for update and delete operations in row-level triggers

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Example

```
CREATE TRIGGER adjust_rating
BEFORE
INSERT
ON customer
FOR each row
BEGIN
  IF NEW.rating < 0 THEN SET NEW.rating = 0;
  END IF;
END
```

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Stored Procedure in sql

IntelliPaat

STORED PROCEDURE is a prepared sql code which can be saved and reused

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Stored Procedure without parameter Syntax


```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

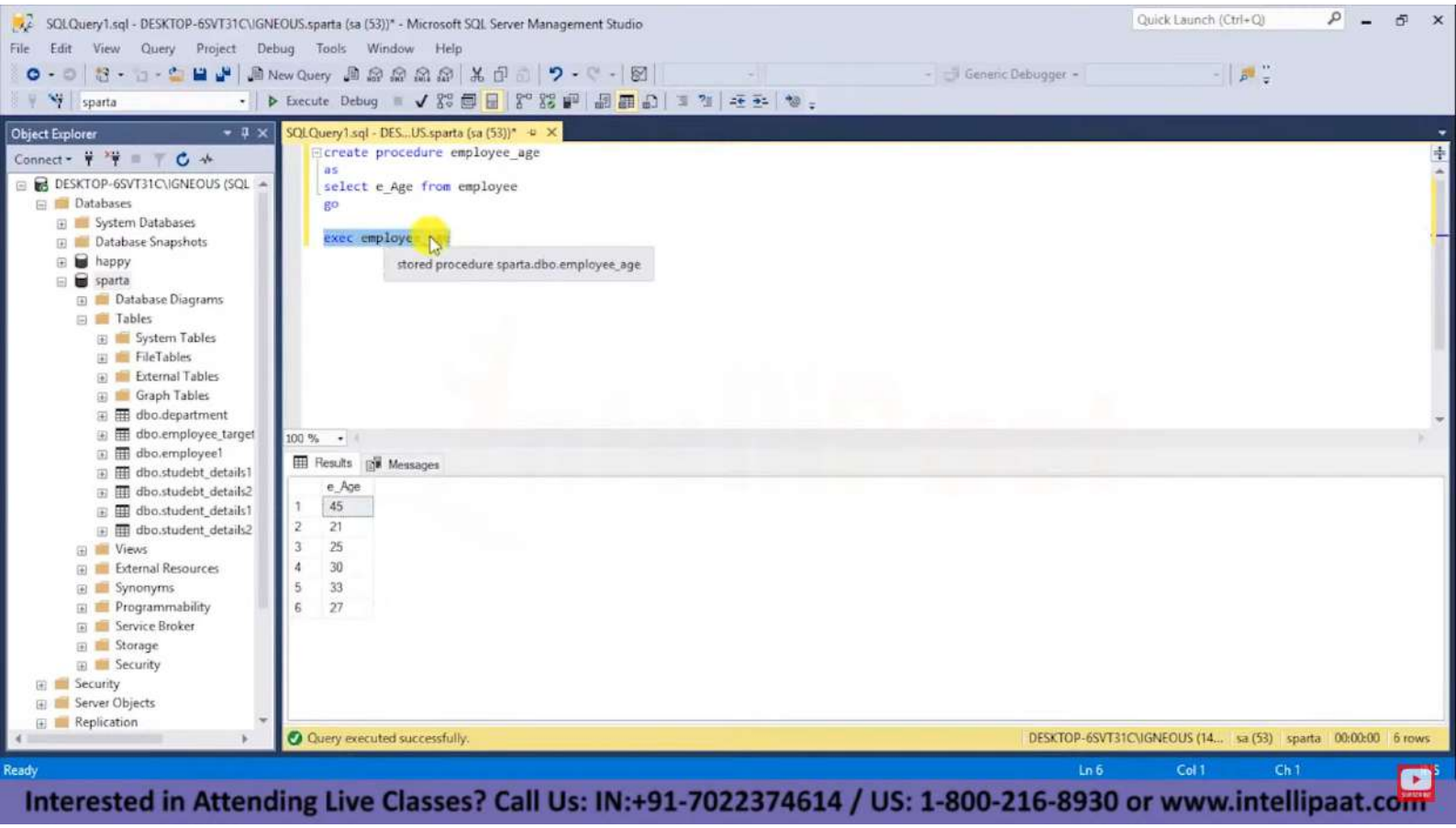
Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Stored Procedure without parameter Syntax

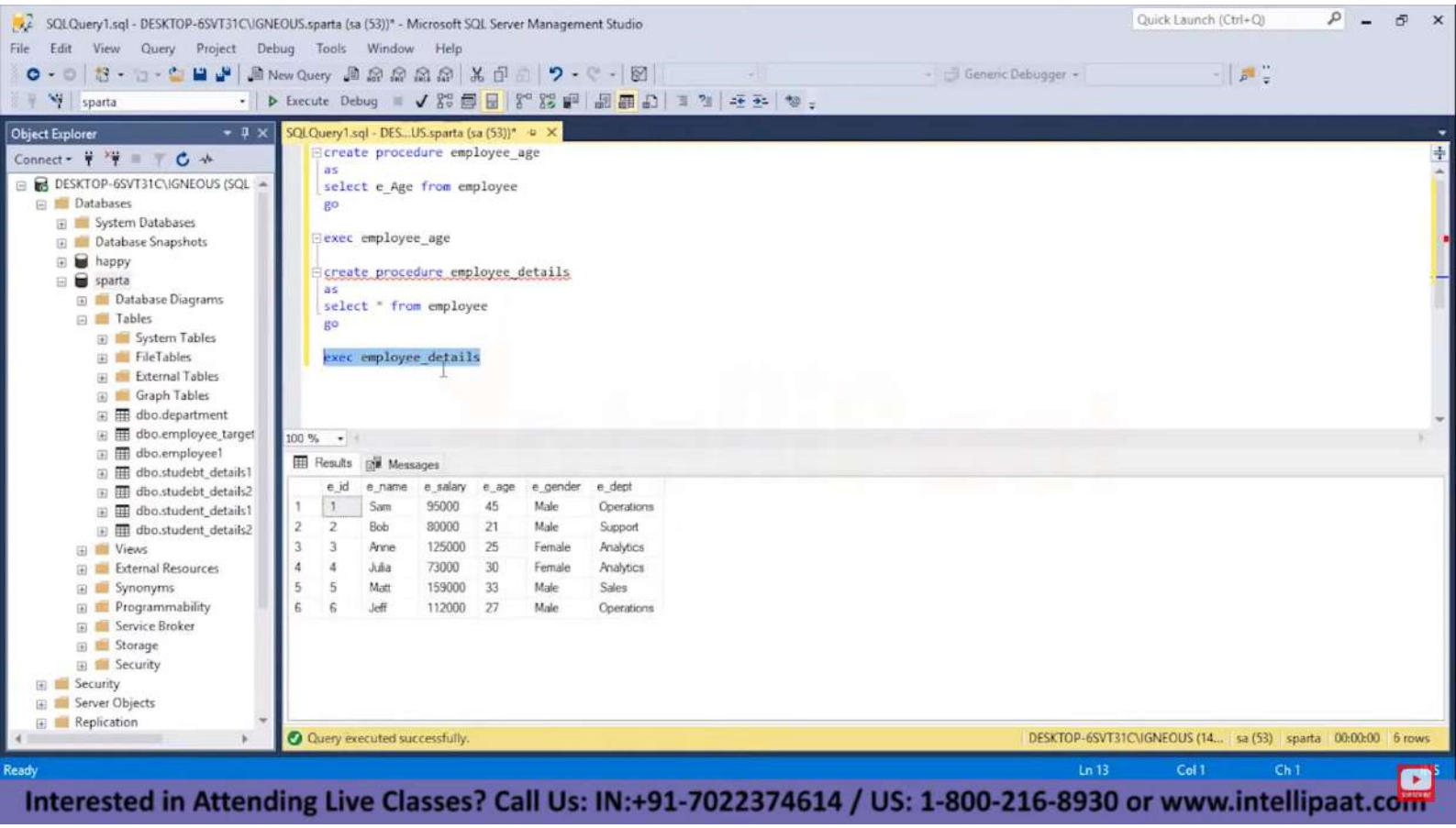

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

```
EXEC procedure_name
```

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com



Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

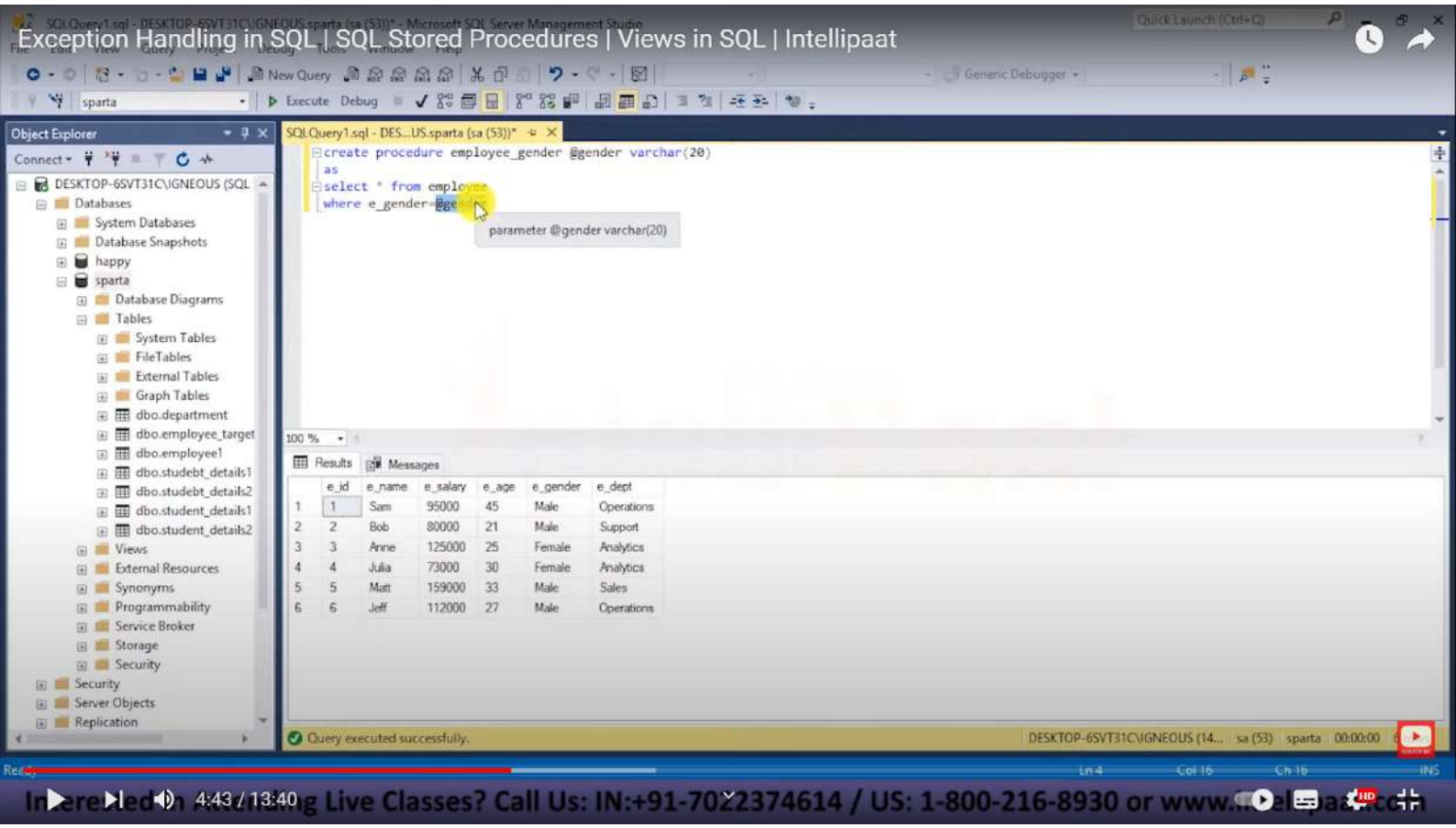


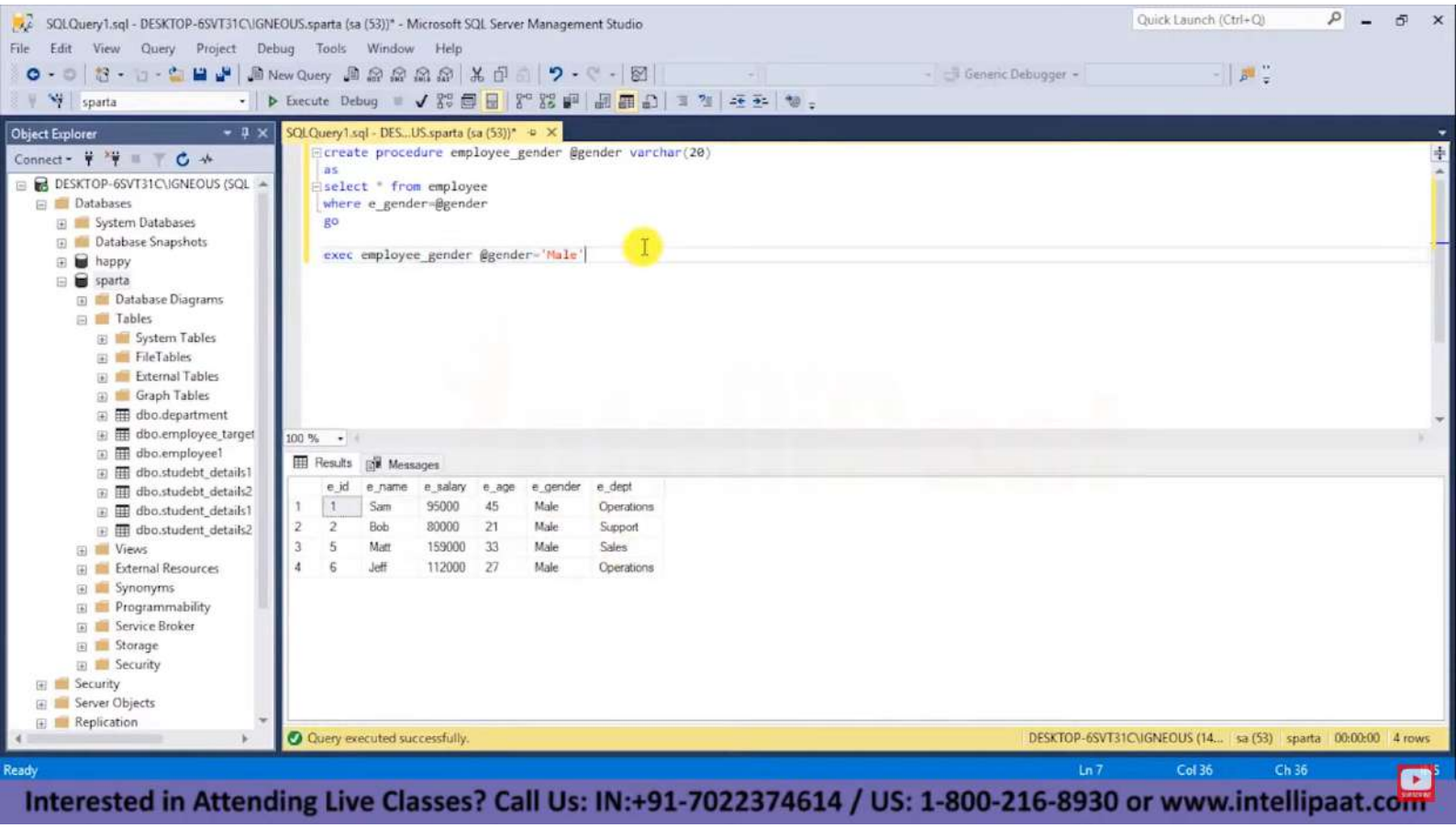
Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Stored Procedure with parameter Syntax


```
CREATE PROCEDURE procedure_name
@param1 data-type, @param2 data-type
AS
sql_statement
GO;
```

Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com


Exception Handling in SQL | SQL Stored Procedures | Views in SQL | Intellipaat
create procedure employee_gender @gender varchar(20)
as
select * from employee
where e_gender=@gender
parameter @gender varchar(20)
Results
Messages
e_id e_name e_salary e_age e_gender e_dept
1 1 Sam 95000 45 Male Operations
2 2 Bob 80000 21 Male Support
3 3 Anne 125000 25 Female Analytics
4 4 Julia 73000 30 Female Analytics
5 5 Matt 159000 33 Male Sales
6 6 Jeff 112000 27 Male Operations
Query executed successfully.
4:43 / 13:40
Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com



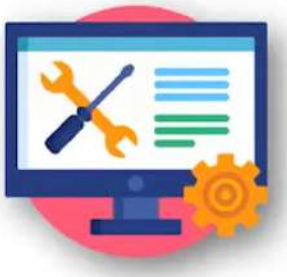
Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

Exception Handling

IntelliPaat

An error condition during a program execution is called an exception

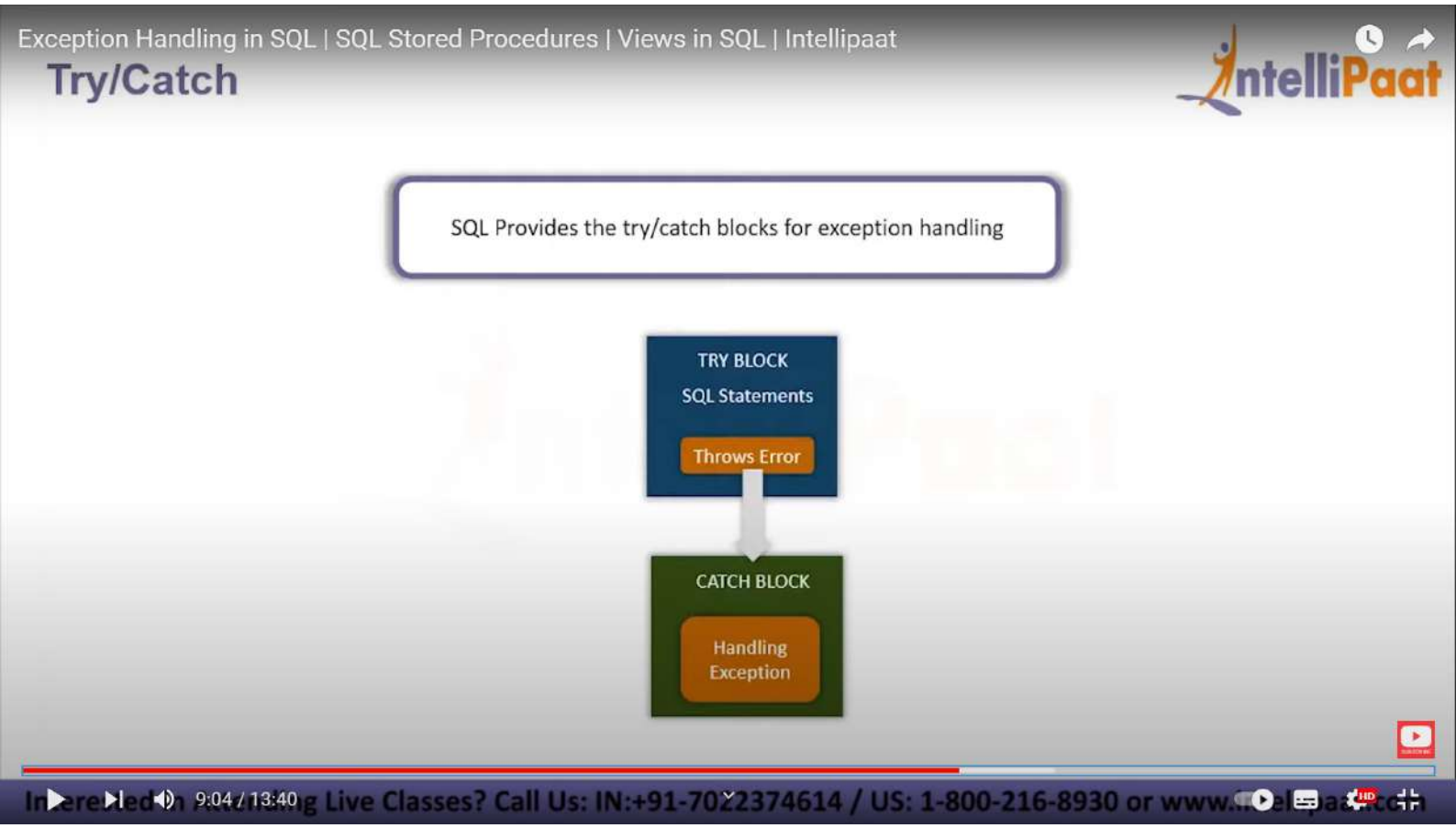
The mechanism for resolving such an exception is exception handling



Exception

Exception Handling

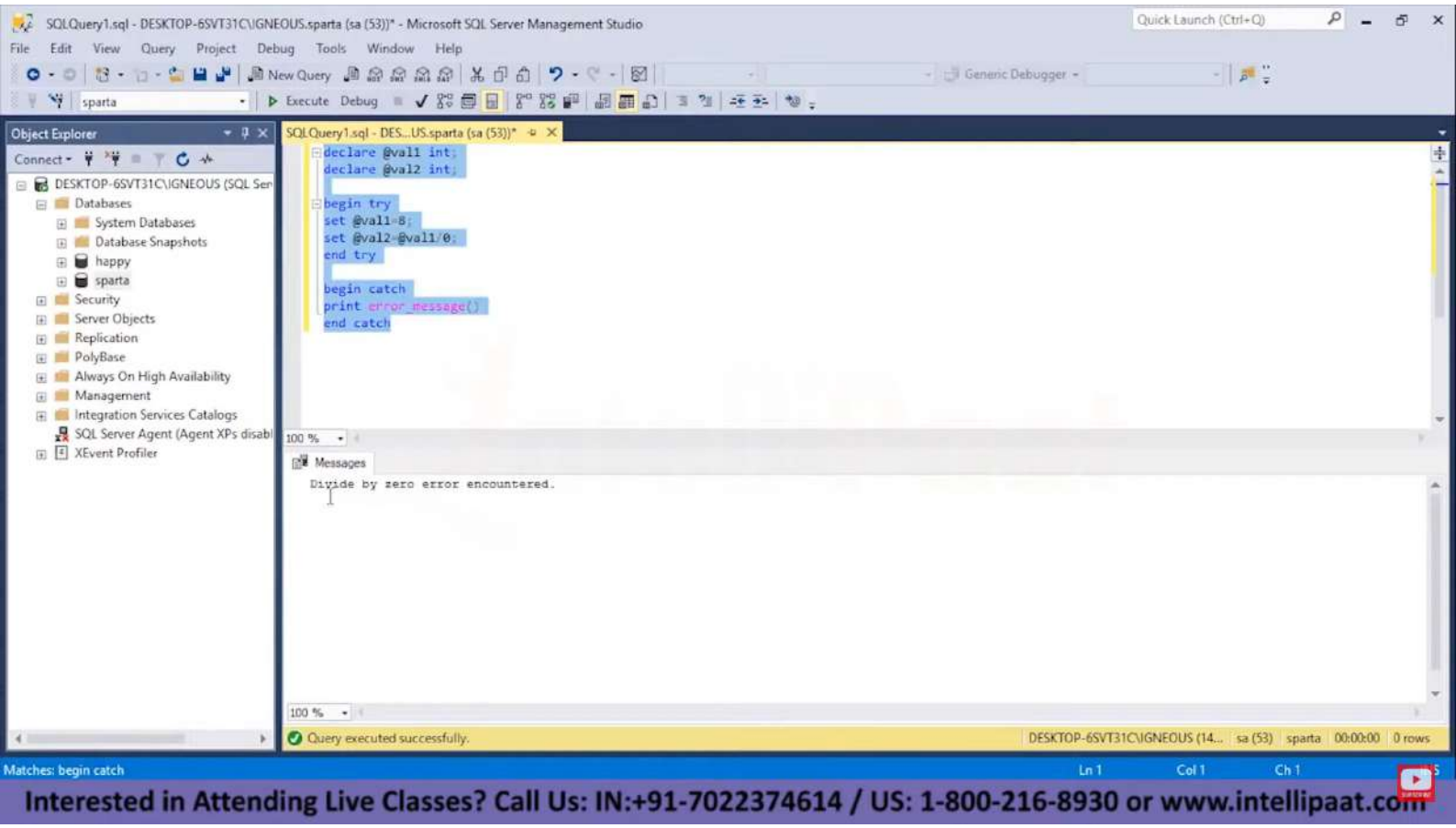
Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com


Exception Handling in SQL | SQL Stored Procedures | Views in SQL | Intellipaat
Try/Catch
IntelliPaat
SQL Provides the try/catch blocks for exception handling
TRY BLOCK
SQL Statements
Throws Error
CATCH BLOCK
Handling Exception
9:04 / 13:40
Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

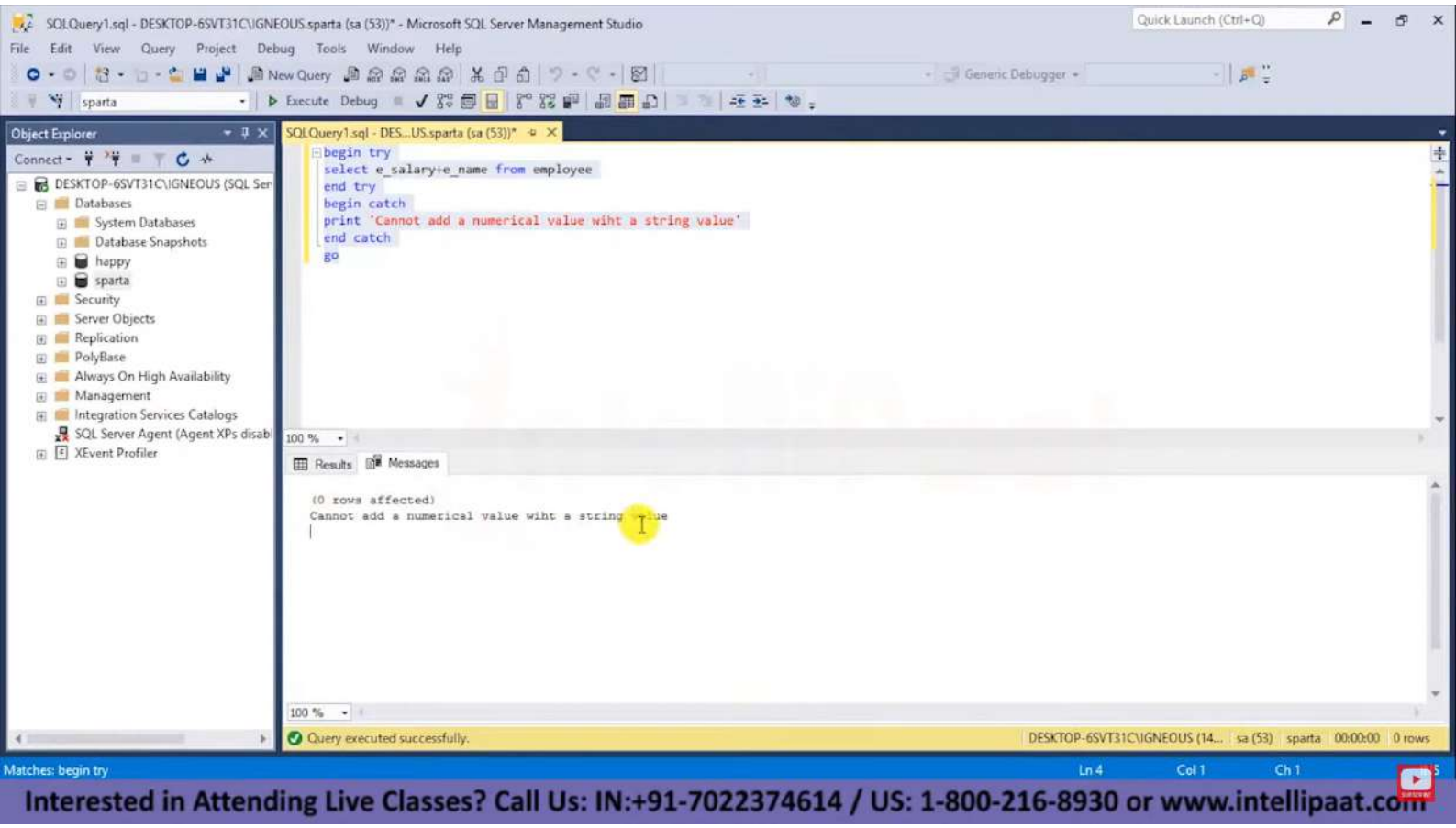
Try/Catch Syntax


```
BEGIN TRY

SQL Statements

END TRY

BEGIN CATCH

- Print Error OR
- Rollback Transaction

END CATCH
```

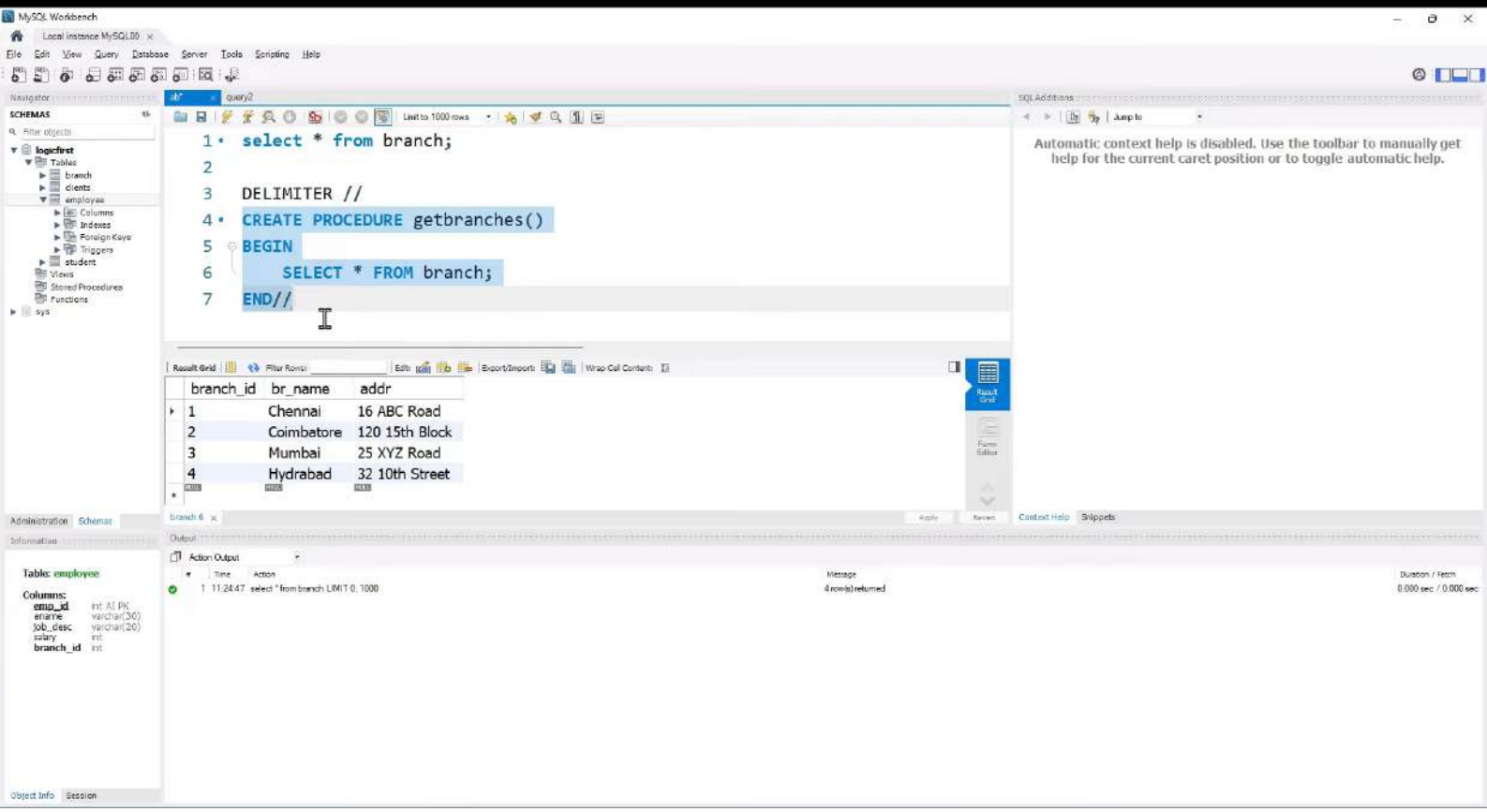
Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

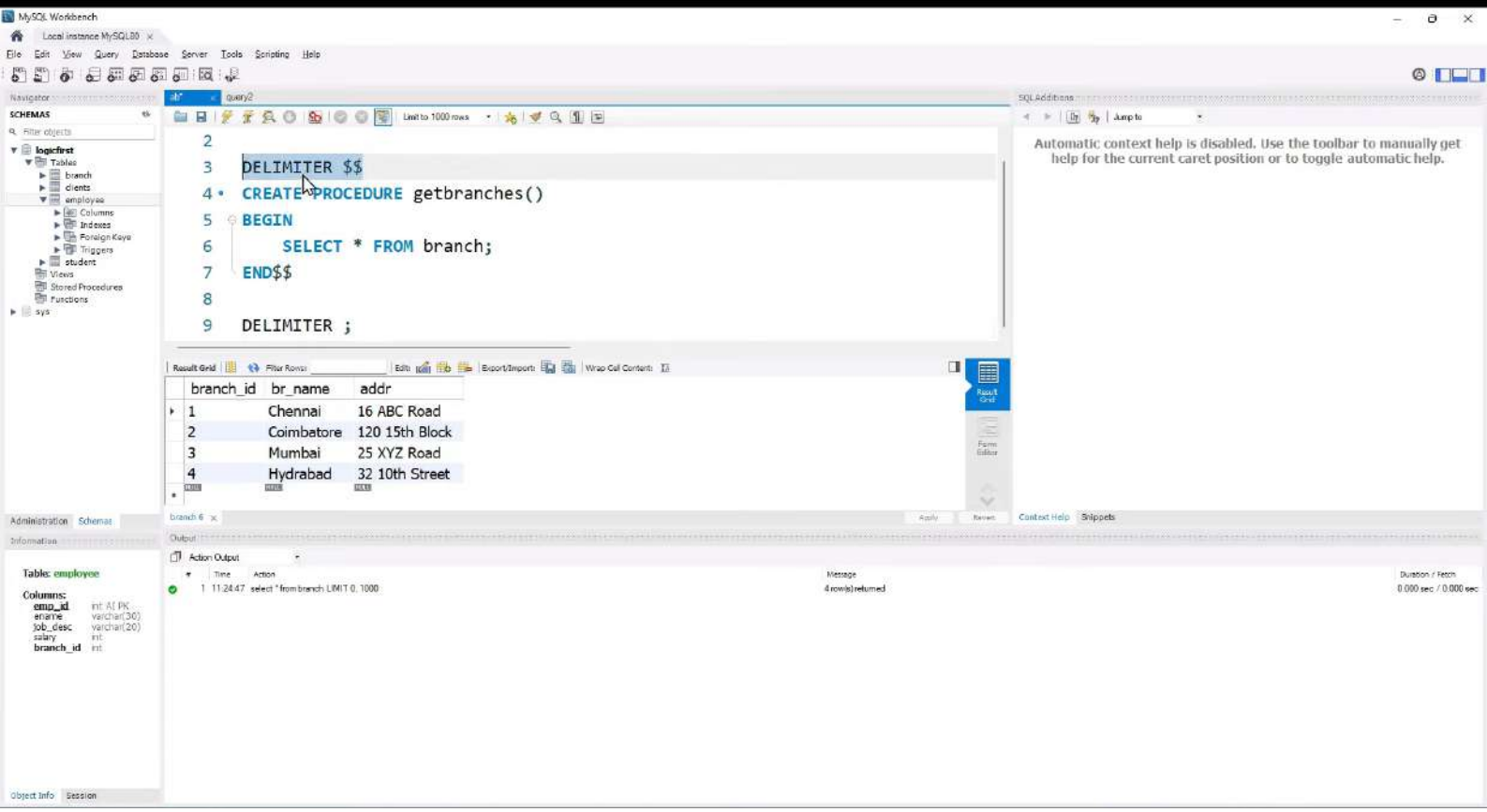


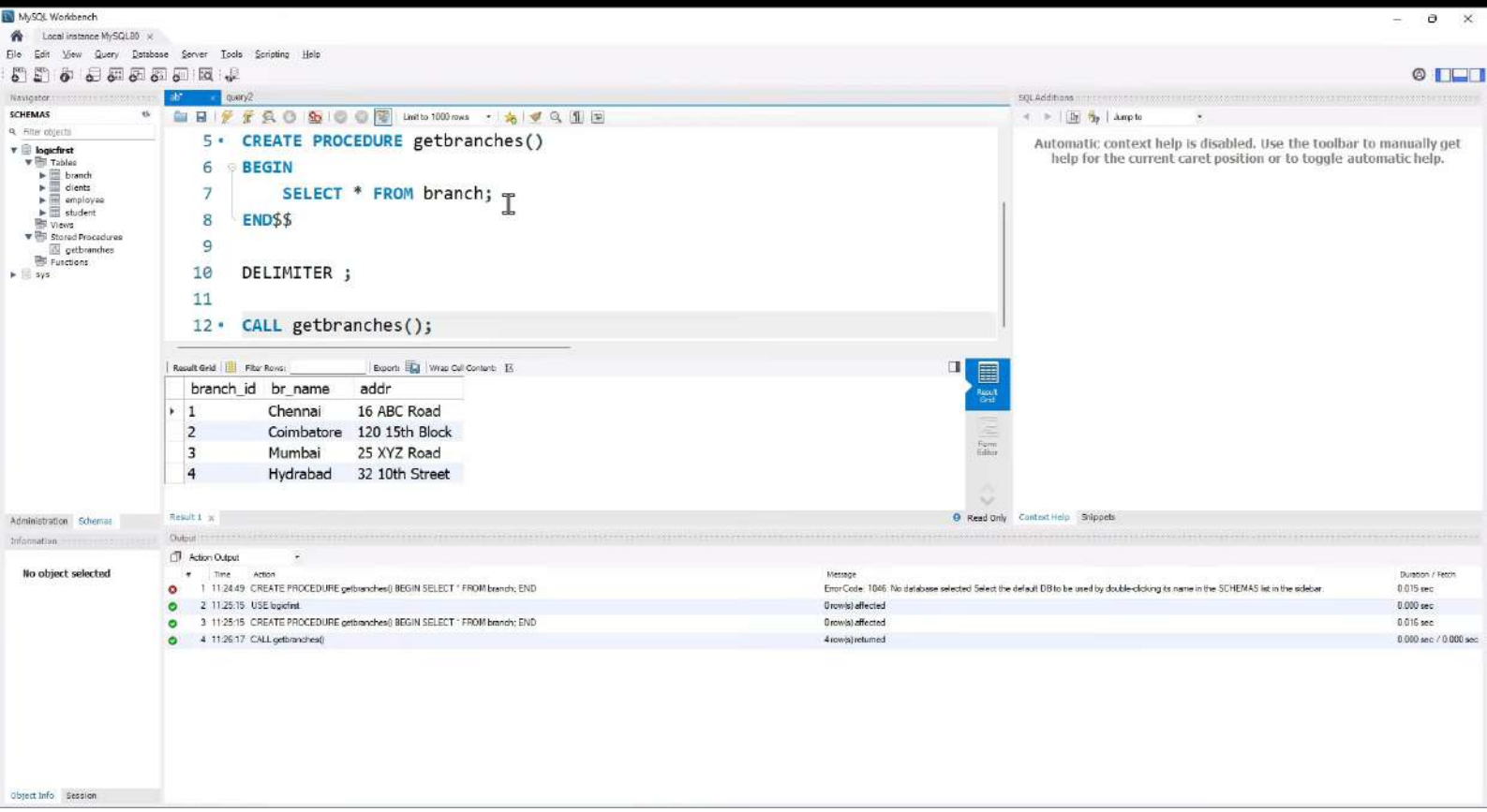
Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com



Interested in Attending Live Classes? Call Us: IN:+91-7022374614 / US: 1-800-216-8930 or www.intellipaat.com

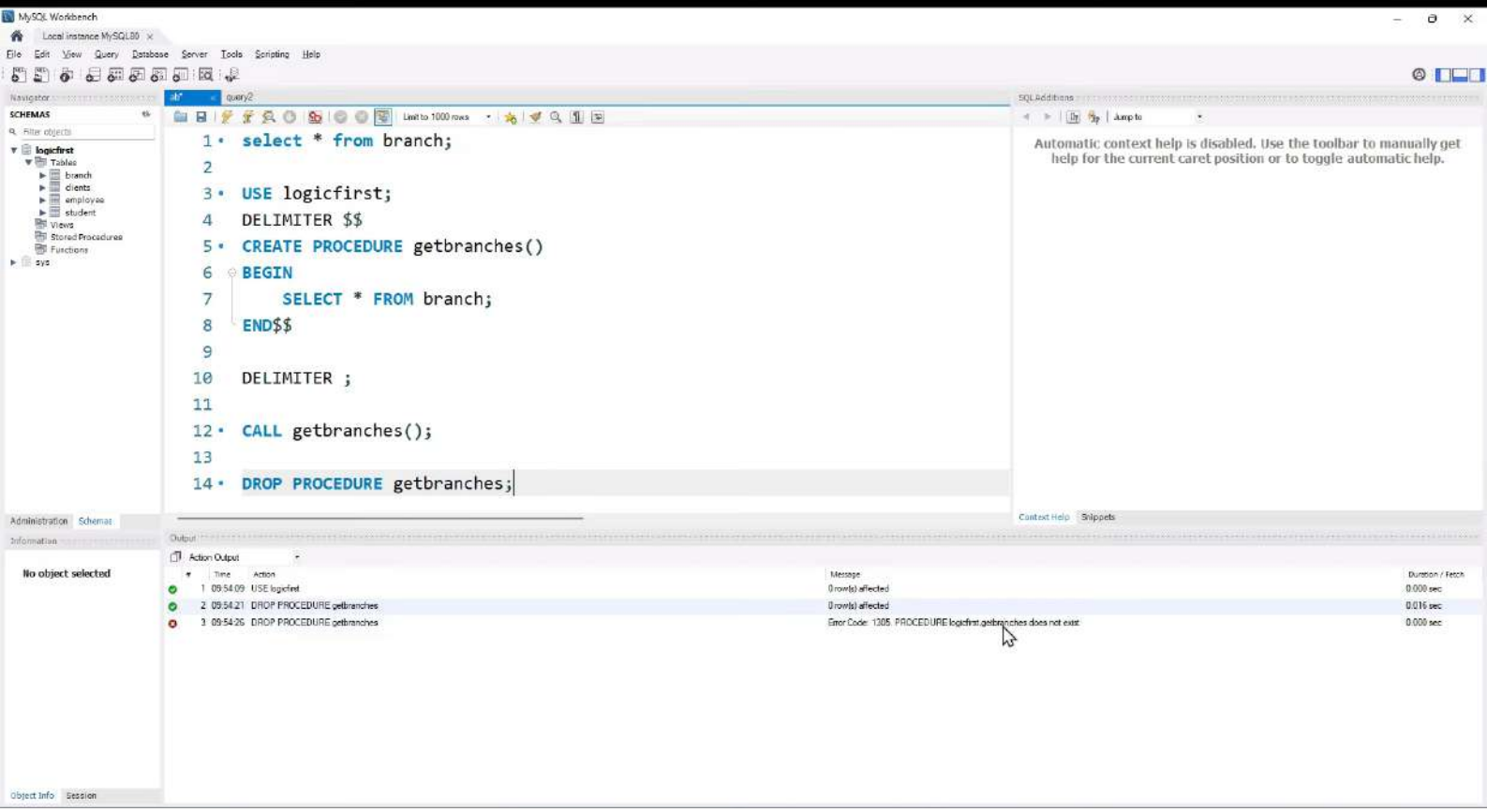

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
Columns
Indexes
Foreign Keys
Triggers
student
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
Table: employee
Columns:
emp_id int AI PK
ename varchar(30)
job_desc varchar(20)
salary int
branch_id int
Object Info Session
query2
Limit to 1000 rows
1 • select * from branch;
2
3 DELIMITER //
4 • CREATE PROCEDURE getbranches()
5 BEGIN
6 SELECT * FROM branch;
7 END//
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
branch_id br_name addr
1 Chennai 16 ABC Road
2 Coimbatore 120 15th Block
3 Mumbai 25 XYZ Road
4 Hydrabad 32 10th Street
branch 6
Apply Revert
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 11:24:47 select * from branch LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec

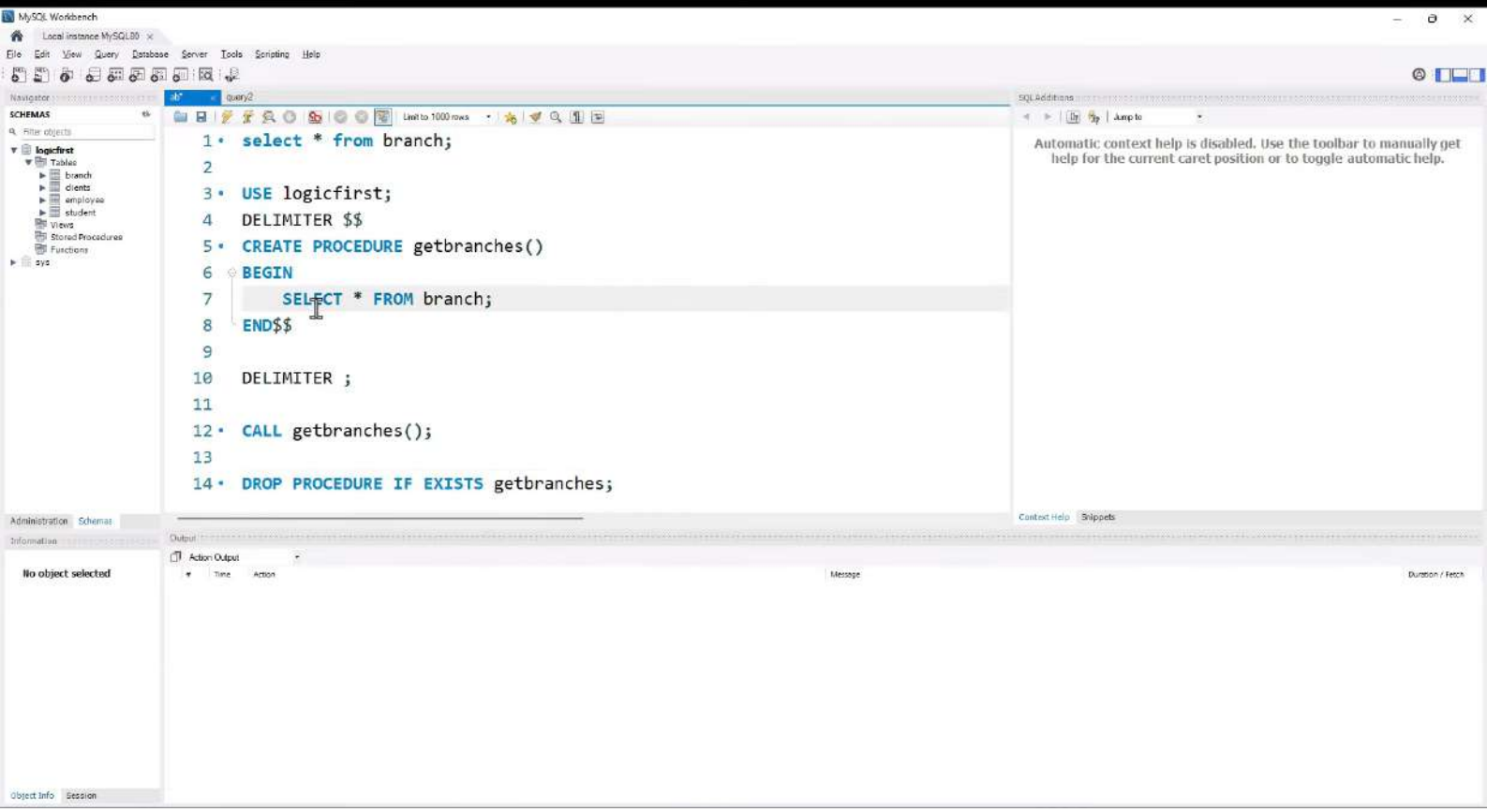

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
Columns
Indexes
Foreign Keys
Triggers
student
Views
Stored Procedures
Functions
sys
query2
Limit to 1000 rows
2
3 DELIMITER $$
4 • CREATE PROCEDURE getbranches()
5 BEGIN
6 SELECT * FROM branch;
7 END$$
8
9 DELIMITER ;
Result Grid
branch_id br_name addr
1 Chennai 16 ABC Road
2 Coimbatore 120 15th Block
3 Mumbai 25 XYZ Road
4 Hydrabad 32 10th Street
SQL Additions
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Administration Schemas
Information
Table: employee
Columns:
emp_id int AI PK
ename varchar(30)
job_desc varchar(20)
salary int
branch_id int
Object Info Session
Output
Action Output
Time Action Message Duration / Fetch
1 11:24:47 select * from branch LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
Context Help Snippets

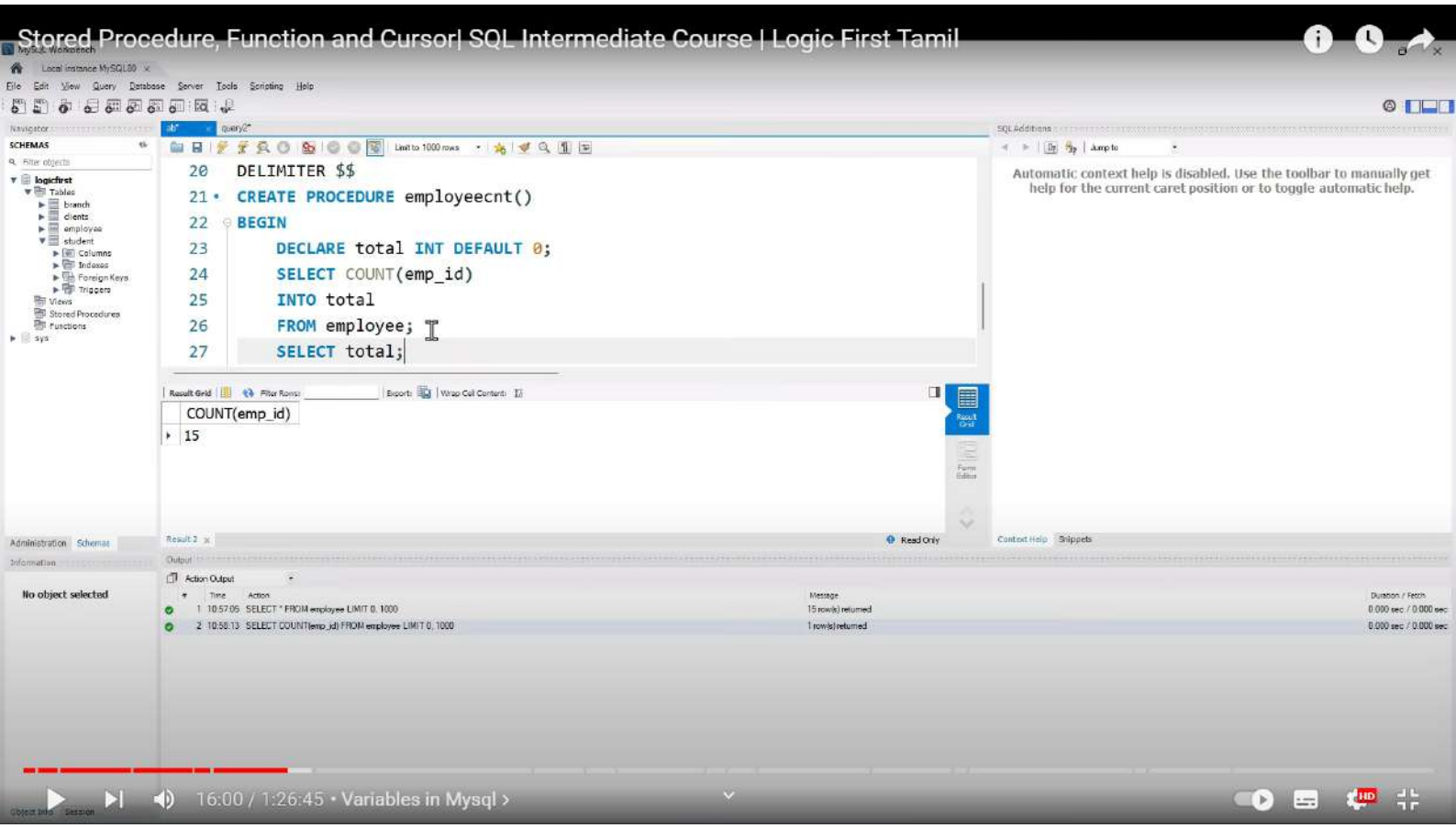

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
student
Views
Stored Procedures
getbranches
Functions
sys
query2
Limit to 1000 rows
5 • CREATE PROCEDURE getbranches()
6 BEGIN
7 SELECT * FROM branch;
8 END$$
9
10 DELIMITER ;
11
12 • CALL getbranches();
Result Grid
branch_id br_name addr
1 Chennai 16 ABC Road
2 Coimbatore 120 15th Block
3 Mumbai 25 XYZ Road
4 Hydrabad 32 10th Street
SQL Additions
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Administration Schemas
Information
No object selected
Object Info Session
Output
Action Output
1 11:24:49 CREATE PROCEDURE getbranches() BEGIN SELECT * FROM branch; END
2 11:25:15 USE logicfirst
3 11:25:15 CREATE PROCEDURE getbranches() BEGIN SELECT * FROM branch; END
4 11:26:17 CALL getbranches()

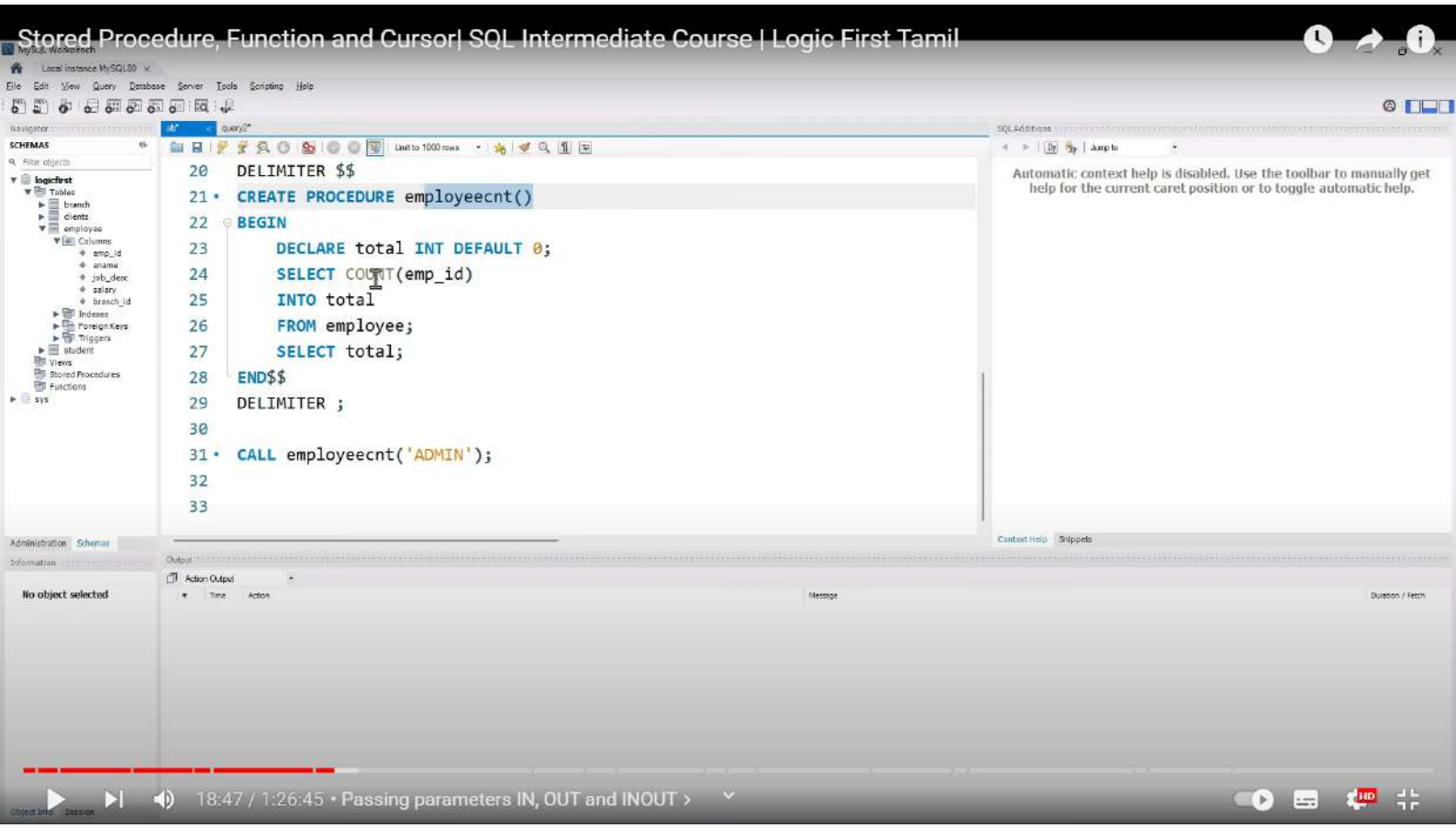
Advantages

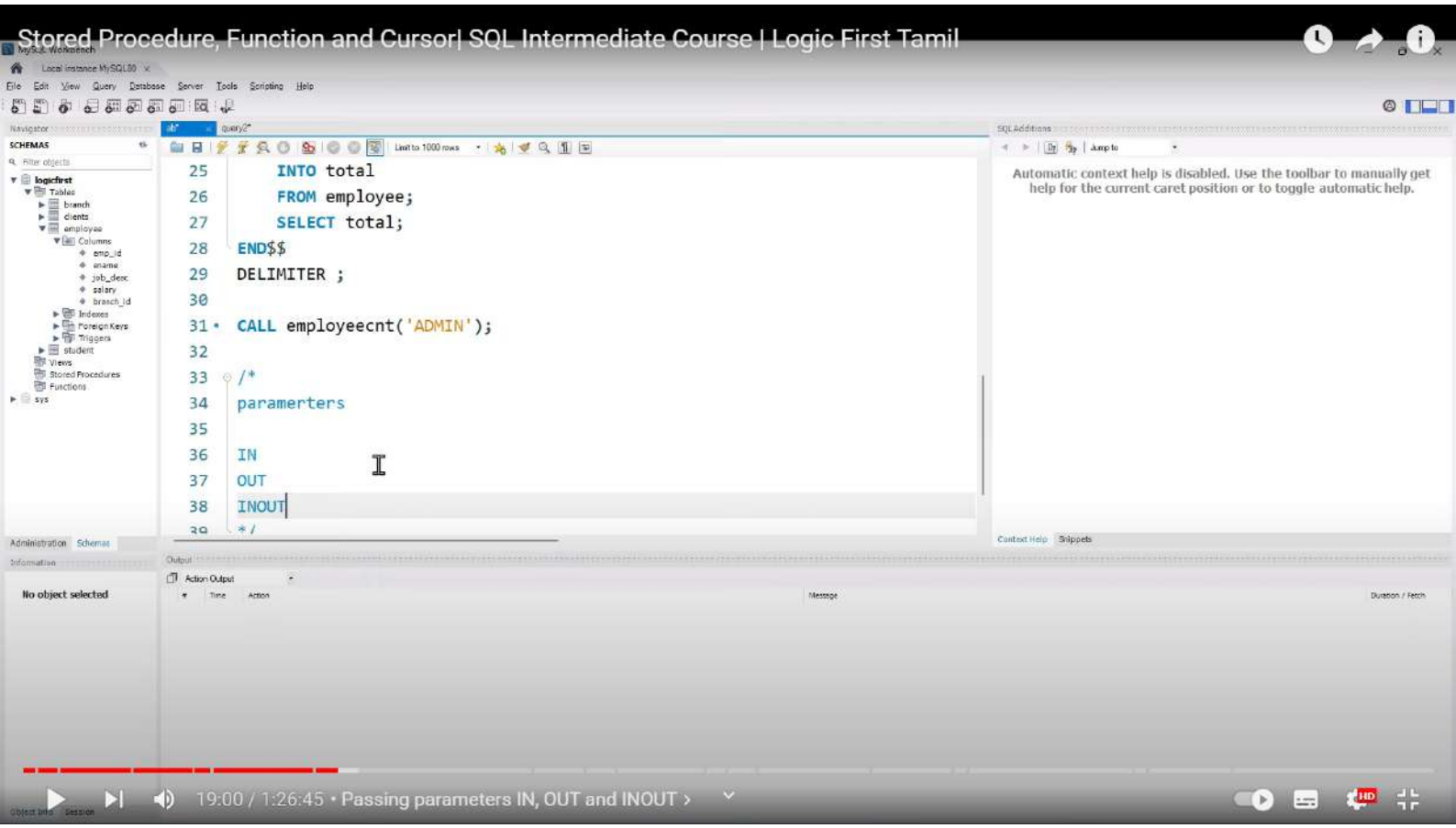
- Reduce Network Traffic
- Centralize Business Logic
- Secure

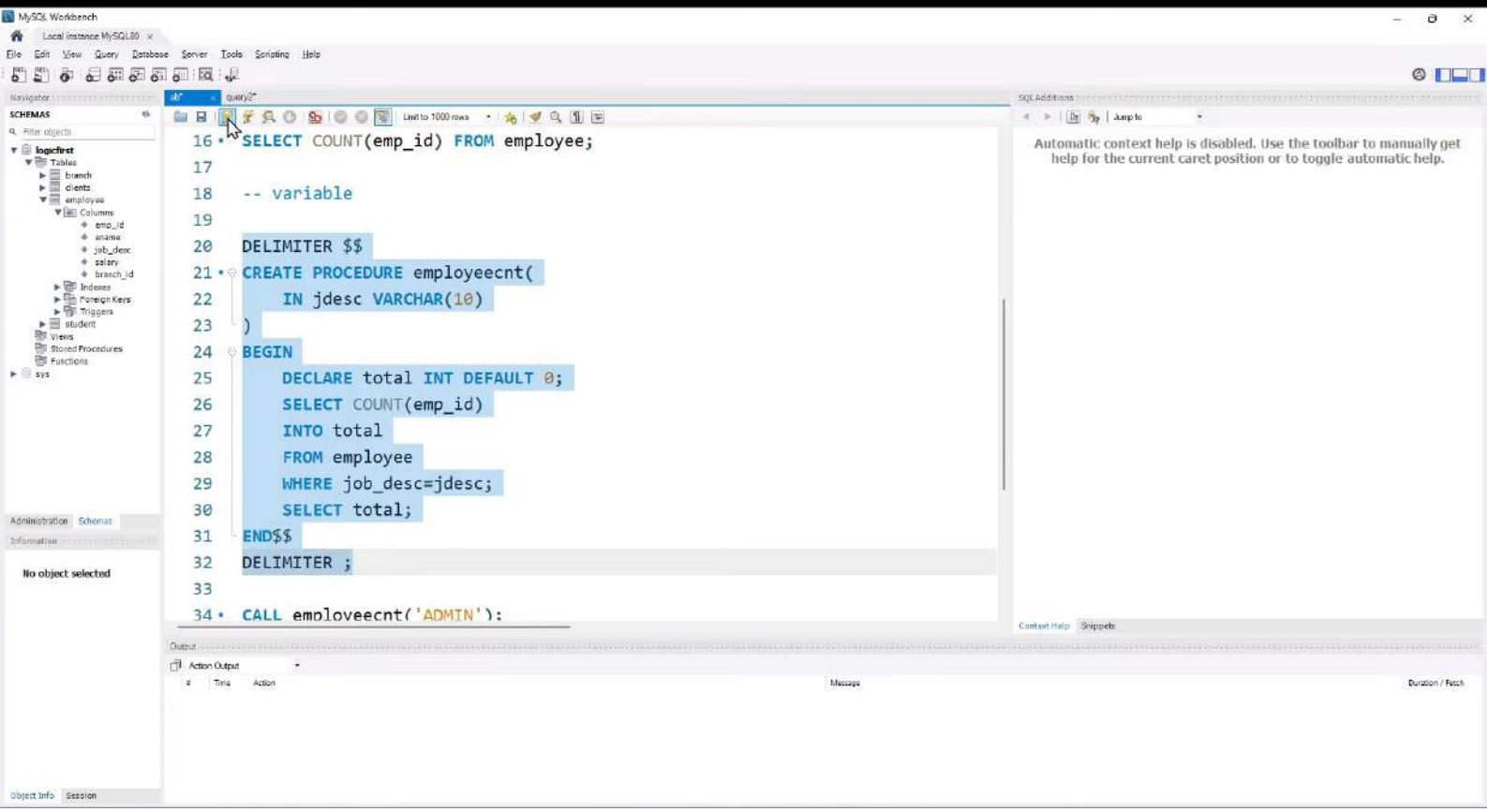

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
student
Views
Stored Procedures
Functions
sys
query2
Limit to 1000 rows
1 • select * from branch;
2
3 • USE logicfirst;
4 DELIMITER $$
5 • CREATE PROCEDURE getbranches()
6 BEGIN
7 SELECT * FROM branch;
8 END$$
9
10 DELIMITER ;
11
12 • CALL getbranches();
13
14 • DROP PROCEDURE getbranches;
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help
Snippets
Administration
Schemas
Information
No object selected
Object Info
Session
Output
Action Output
Time
Action
Message
Duration / Fetch
1 09:54:09 USE logicfirst 0 row(s) affected 0.000 sec
2 09:54:21 DROP PROCEDURE getbranches 0 row(s) affected 0.016 sec
3 09:54:26 DROP PROCEDURE getbranches Error Code: 1305. PROCEDURE logicfirst.getbranches does not exist 0.000 sec

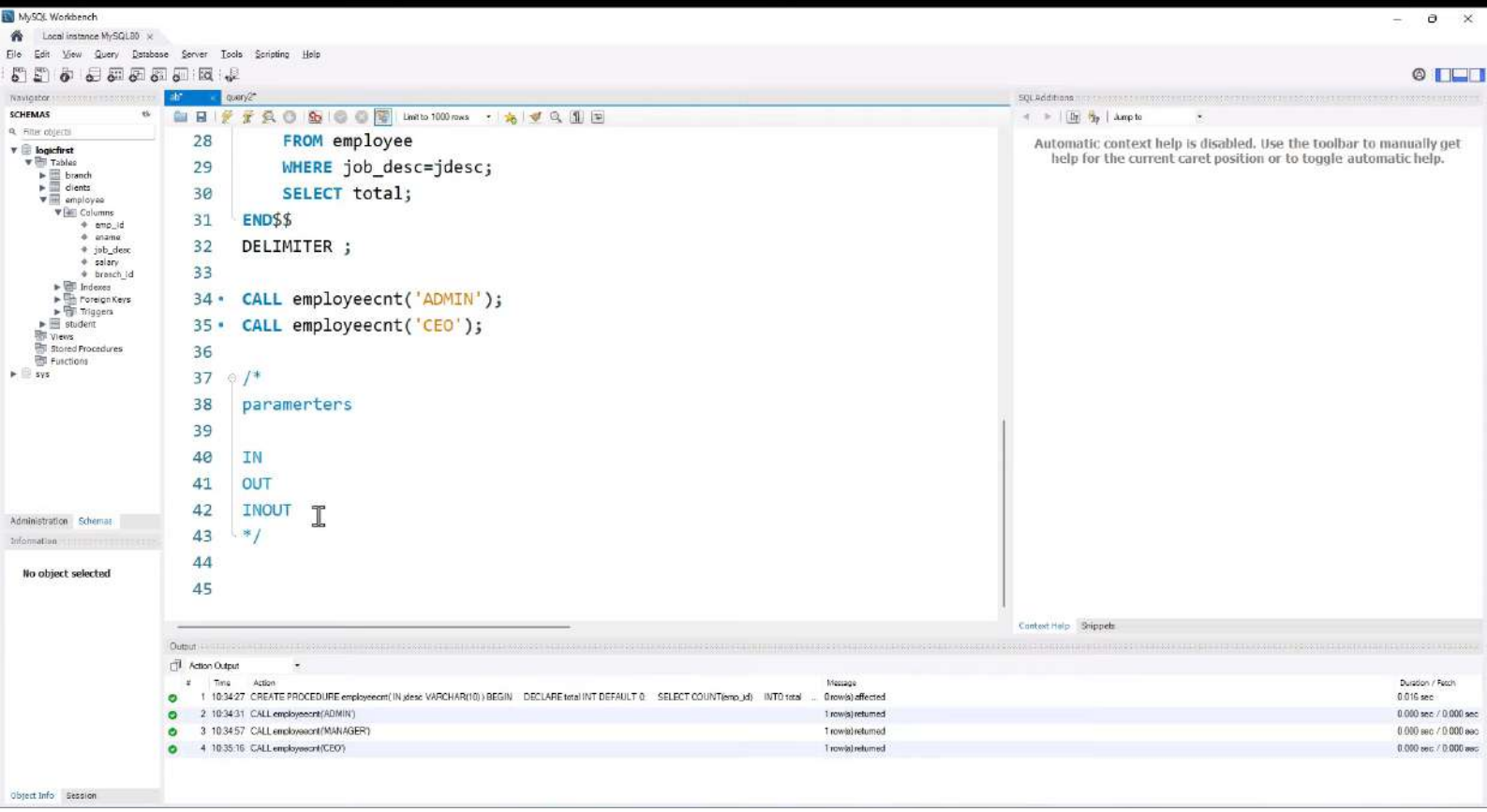

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
student
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
query2
Limit to 1000 rows
1 • select * from branch;
2
3 • USE logicfirst;
4 DELIMITER $$
5 • CREATE PROCEDURE getbranches()
6 BEGIN
7 SELECT * FROM branch;
8 END$$
9
10 DELIMITER ;
11
12 • CALL getbranches();
13
14 • DROP PROCEDURE IF EXISTS getbranches;
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch

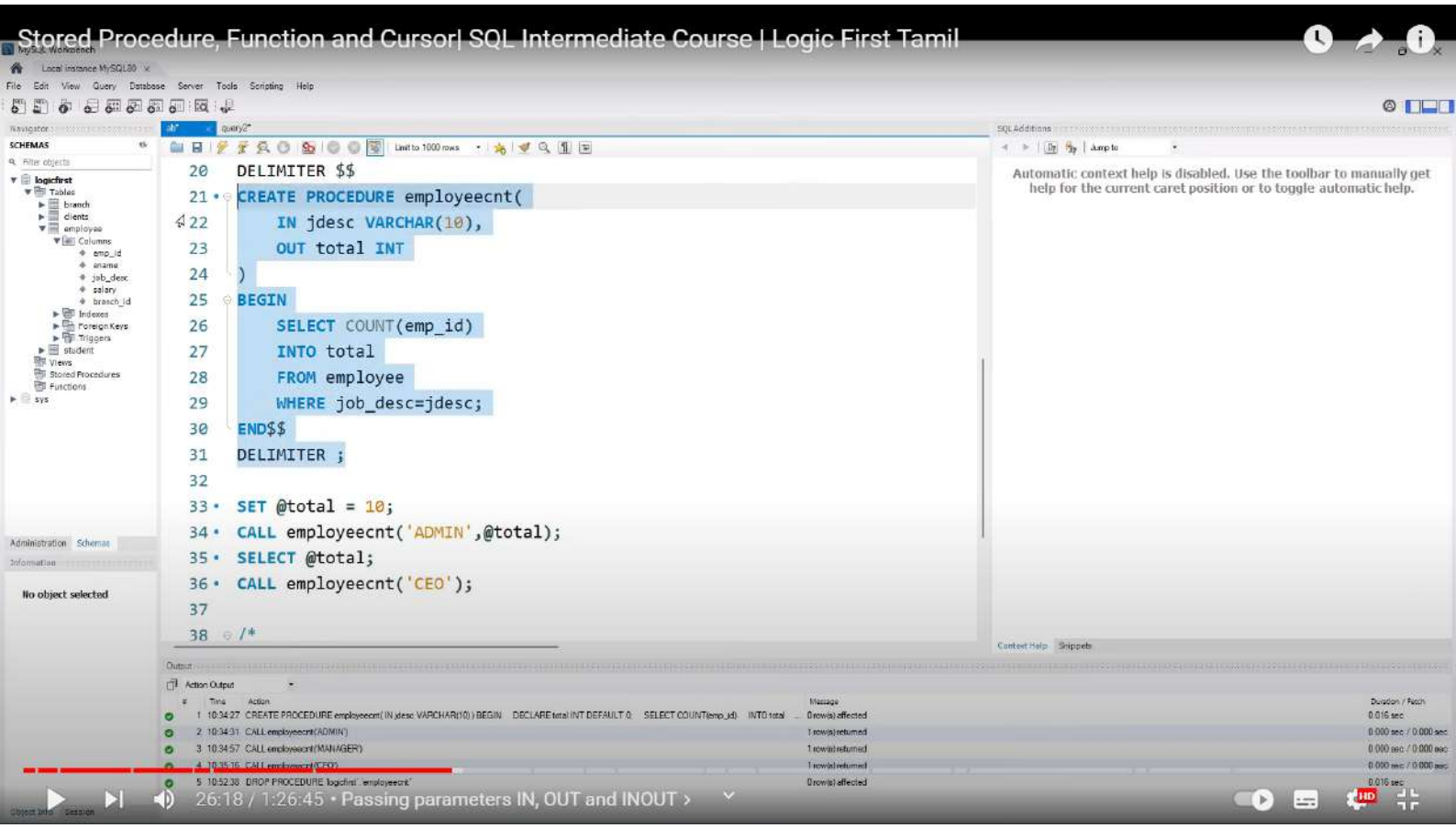

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil
DELIMITER $$
CREATE PROCEDURE employeecnt()
BEGIN
DECLARE total INT DEFAULT 0;
SELECT COUNT(emp_id)
INTO total
FROM employee;
SELECT total;
COUNT(emp_id)
15
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
No object selected
16:00 / 1:26:45 • Variables in Mysql ›

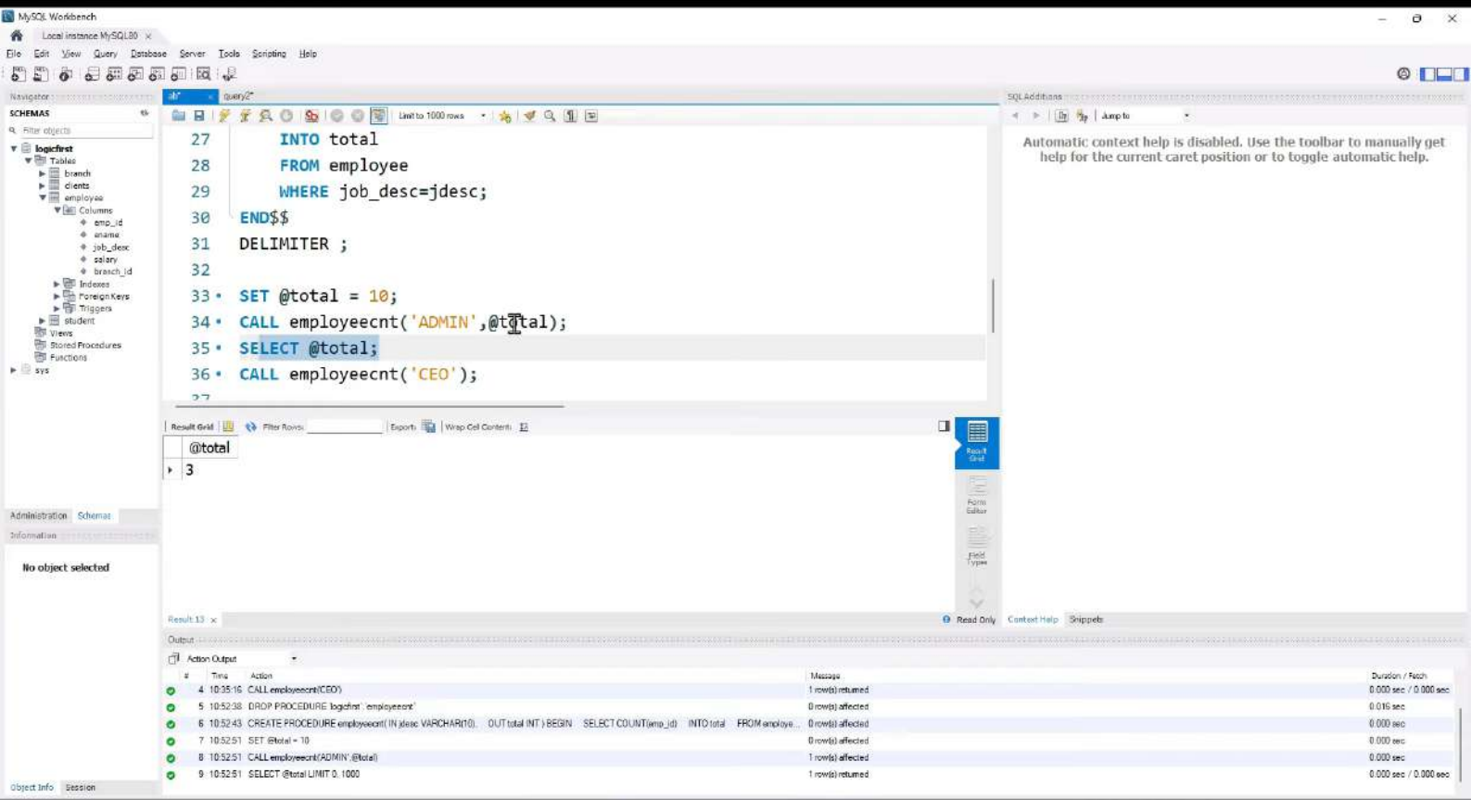

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil
DELIMITER $$
CREATE PROCEDURE employeecnt()
BEGIN
DECLARE total INT DEFAULT 0;
SELECT COUNT(emp_id)
INTO total
FROM employee;
SELECT total;
END$$
DELIMITER ;
CALL employeecnt('ADMIN');
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
No object selected
18:47 / 1:26:45 • Passing parameters IN, OUT and INOUT ›

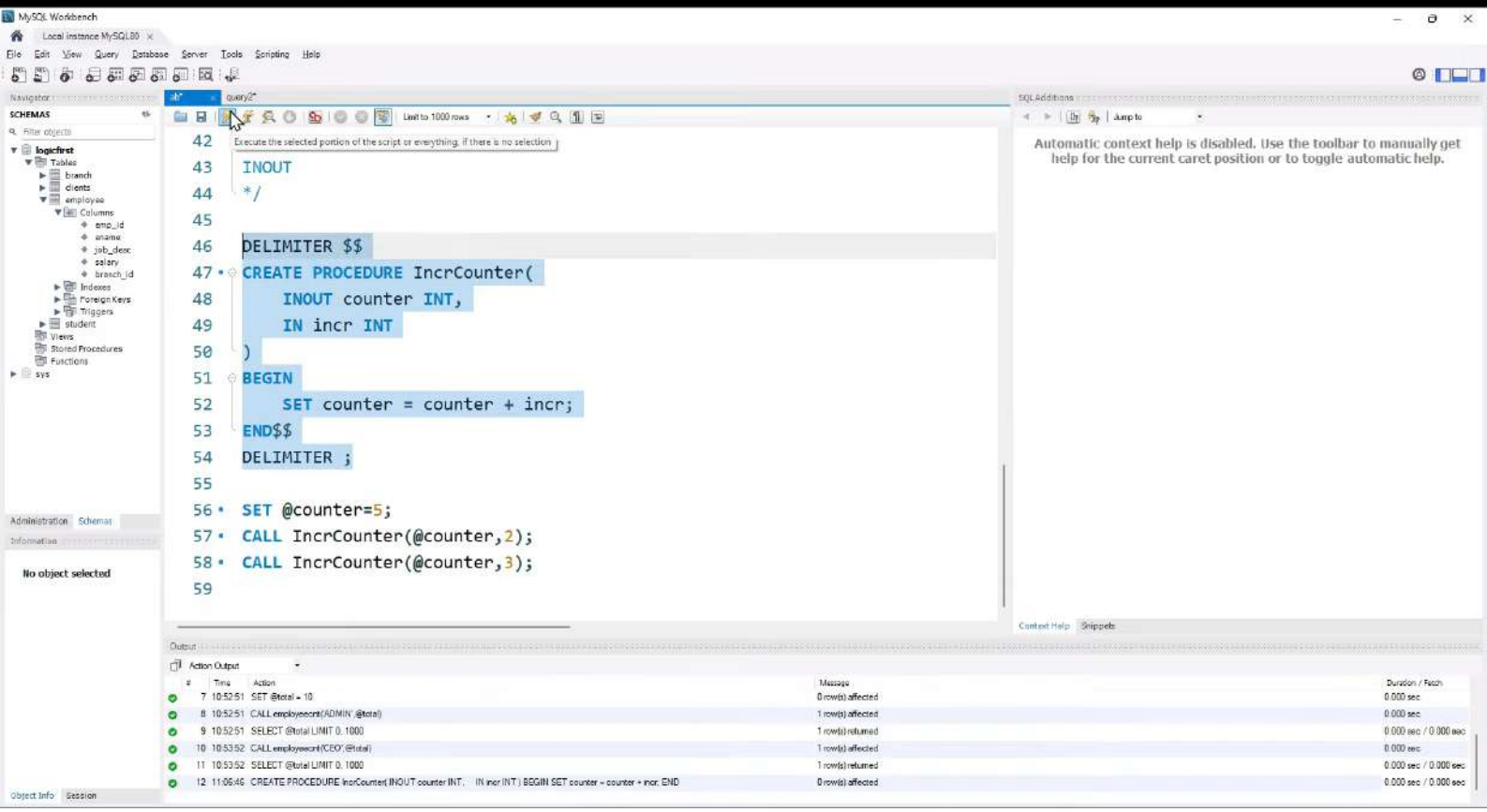

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil
INTO total
FROM employee;
SELECT total;
END$$
DELIMITER ;
CALL employeecnt('ADMIN');
/*
paramerters
IN
OUT
INOUT
*/
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
19:00 / 1:26:45 • Passing parameters IN, OUT and INOUT ›

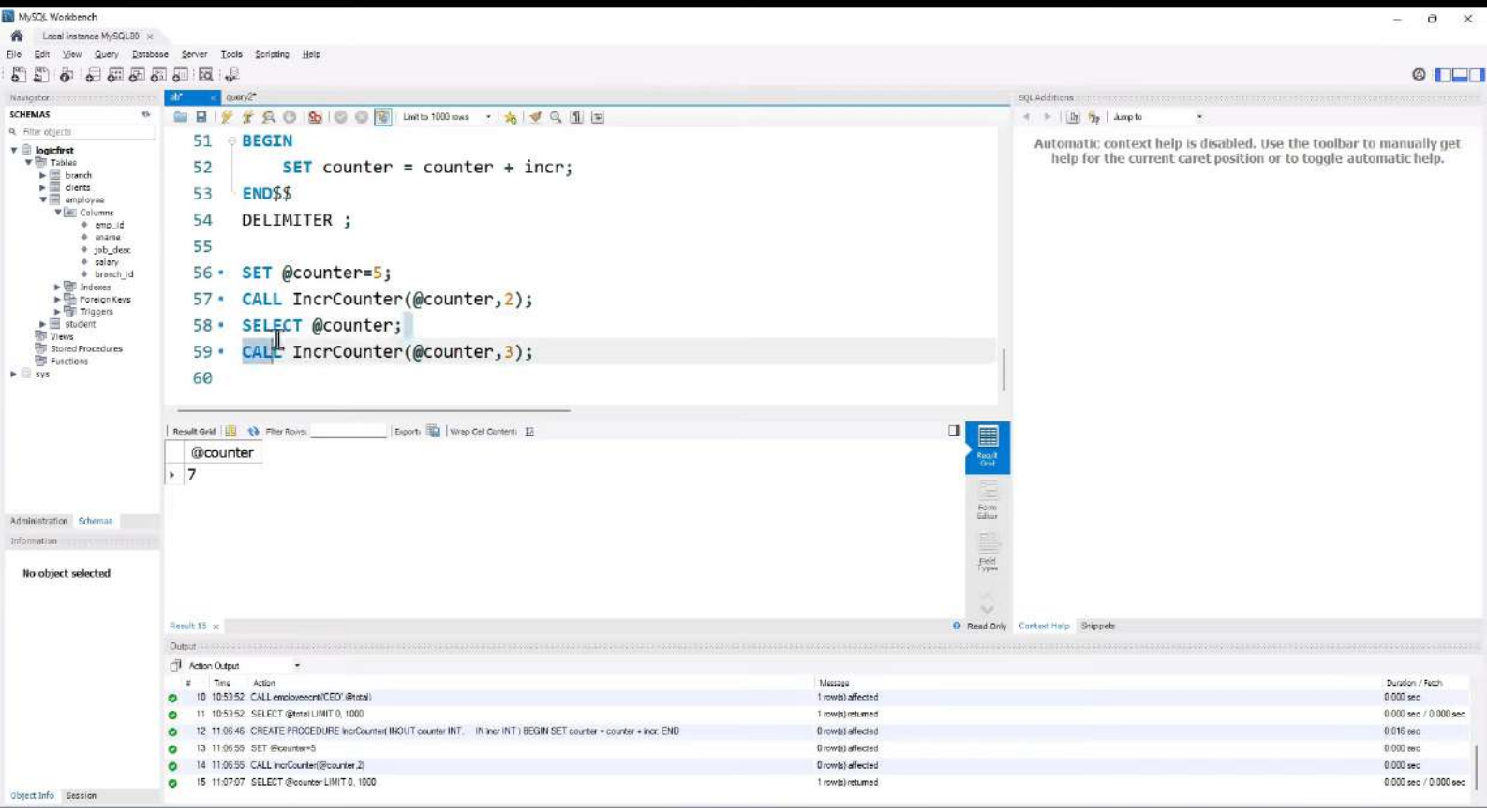

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
Columns
emp_id
ename
job_desc
salary
branch_id
Indexes
Foreign Keys
Triggers
student
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
query2*
Limit to 1000 rows
16 • SELECT COUNT(emp_id) FROM employee;
17
18 -- variable
19
20 DELIMITER $$
21 • CREATE PROCEDURE employeecnt(
22 IN jdesc VARCHAR(10)
23)
24 BEGIN
25 DECLARE total INT DEFAULT 0;
26 SELECT COUNT(emp_id)
27 INTO total
28 FROM employee
29 WHERE job_desc=jdesc;
30 SELECT total;
31 END$$
32 DELIMITER ;
33
34 • CALL employeecnt('ADMIN');
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch

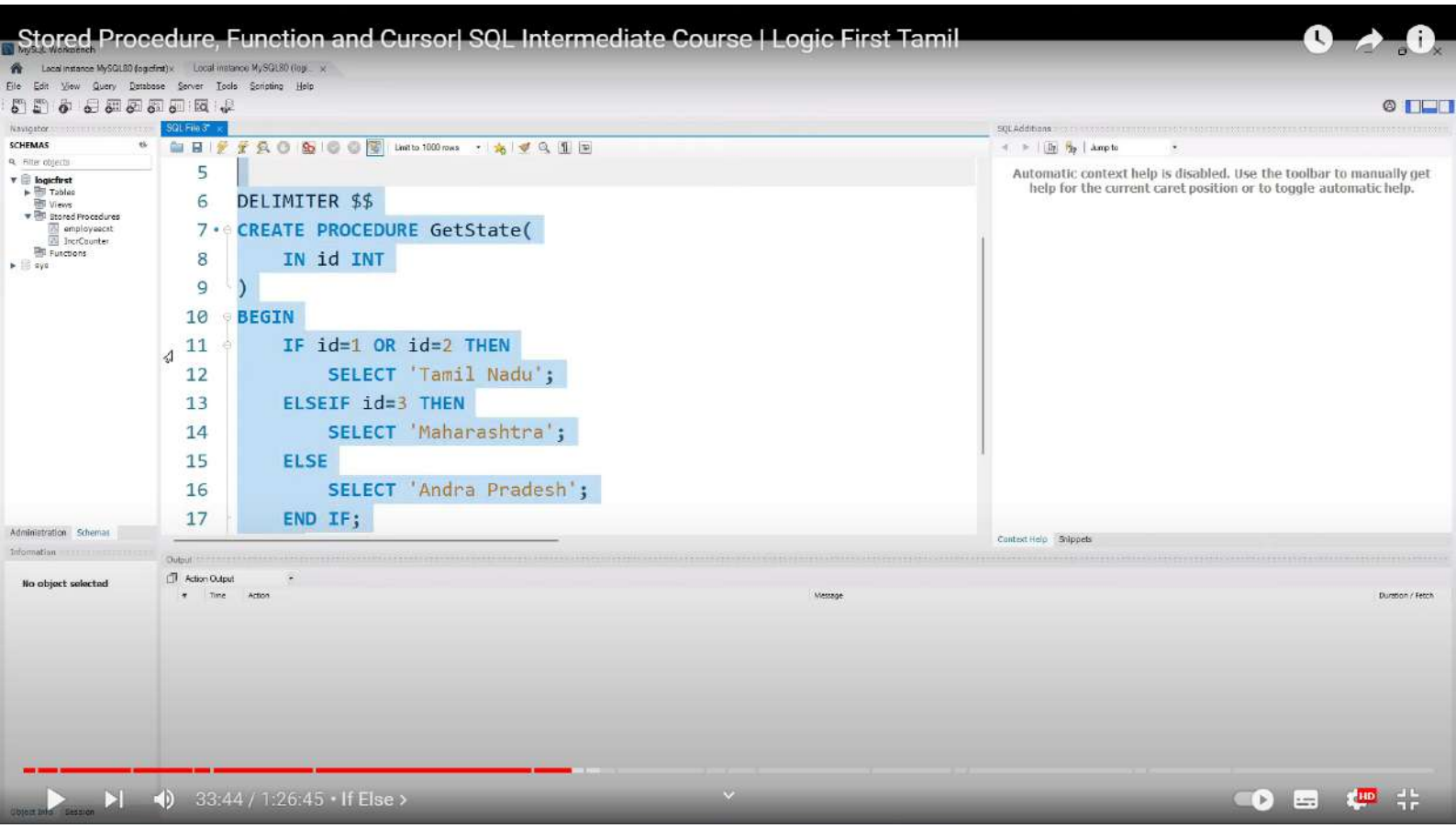

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
Columns
emp_id
ename
job_desc
salary
branch_id
Indexes
Foreign Keys
Triggers
student
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
query2*
Limit to 1000 rows
28 FROM employee
29 WHERE job_desc=jdesc;
30 SELECT total;
31 END$$
32 DELIMITER ;
33
34 • CALL employeecnt('ADMIN');
35 • CALL employeecnt('CEO');
36
37 /*
38 paramerters
39
40 IN
41 OUT
42 INOUT
43 */
44
45
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 10:34:27 CREATE PROCEDURE employeecnt(IN jdesc VARCHAR(10)) BEGIN DECLARE total INT DEFAULT 0 SELECT COUNT(emp_id) INTO total ... 0 row(s) affected 0.016 sec
2 10:34:31 CALL employeecnt('ADMIN') 1 row(s) returned 0.000 sec / 0.000 sec
3 10:34:57 CALL employeecnt('MANAGER') 1 row(s) returned 0.000 sec / 0.000 sec
4 10:35:16 CALL employeecnt('CEO') 1 row(s) returned 0.000 sec / 0.000 sec

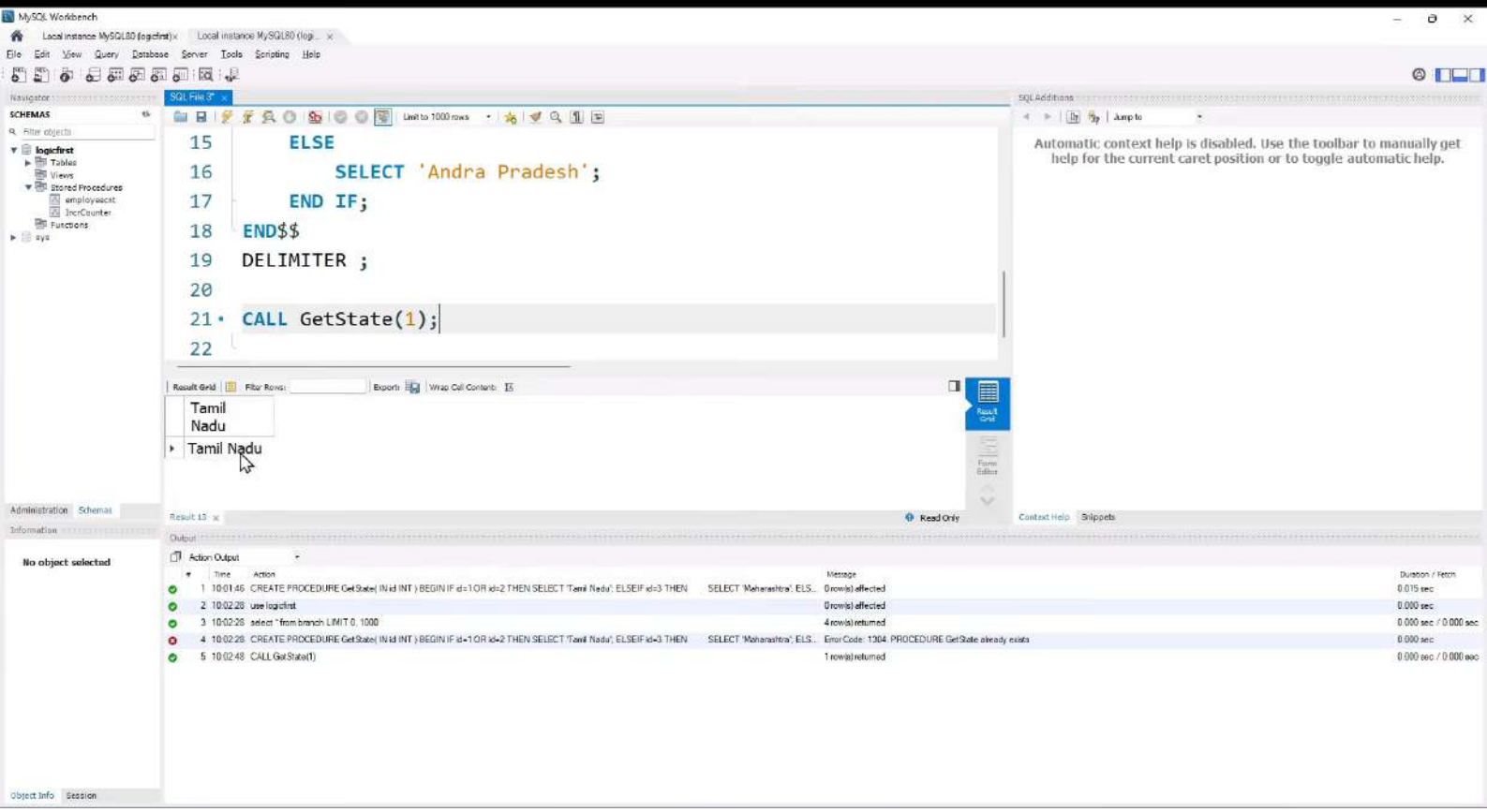

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil
DELIMITER $$
CREATE PROCEDURE employeecnt(
IN jdesc VARCHAR(10),
OUT total INT
)
BEGIN
SELECT COUNT(emp_id)
INTO total
FROM employee
WHERE job_desc=jdesc;
END$$
DELIMITER ;
SET @total = 10;
CALL employeecnt('ADMIN',@total);
SELECT @total;
CALL employeecnt('CEO');
/*
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
26:18 / 1:26:45 • Passing parameters IN, OUT and INOUT ›

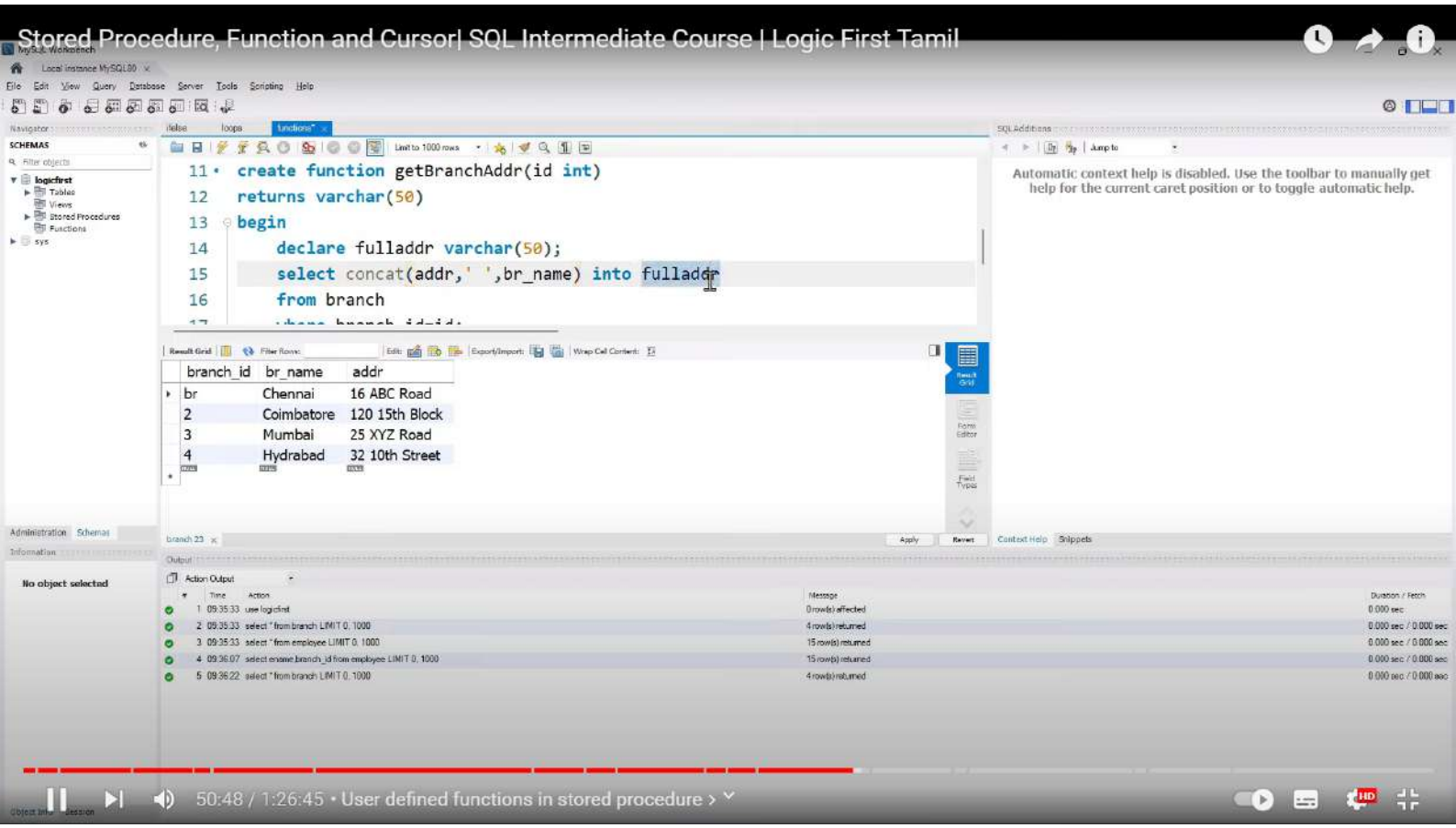

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
Columns
emp_id
ename
job_desc
salary
branch_id
Indexes
Foreign Keys
Triggers
student
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
query2*
Limit to 1000 rows
27 INTO total
28 FROM employee
29 WHERE job_desc=jdesc;
30 END$$
31 DELIMITER ;
32
33 • SET @total = 10;
34 • CALL employeecnt('ADMIN',@total);
35 • SELECT @total;
36 • CALL employeecnt('CEO');
Result Grid
@total
3
Result 13
Read Only
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
4 10:35:16 CALL employeecnt('CEO') 1 row(s) returned 0.000 sec / 0.000 sec
5 10:52:38 DROP PROCEDURE logicfirst.employeecnt 0 row(s) affected 0.016 sec
6 10:52:43 CREATE PROCEDURE employeecnt(IN jdesc VARCHAR(10), OUT total INT) BEGIN SELECT COUNT(emp_id) INTO total FROM employe... 0 row(s) affected 0.000 sec
7 10:52:51 SET @total = 10 0 row(s) affected 0.000 sec
8 10:52:51 CALL employeecnt('ADMIN',@total) 1 row(s) affected 0.000 sec
9 10:52:51 SELECT @total LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

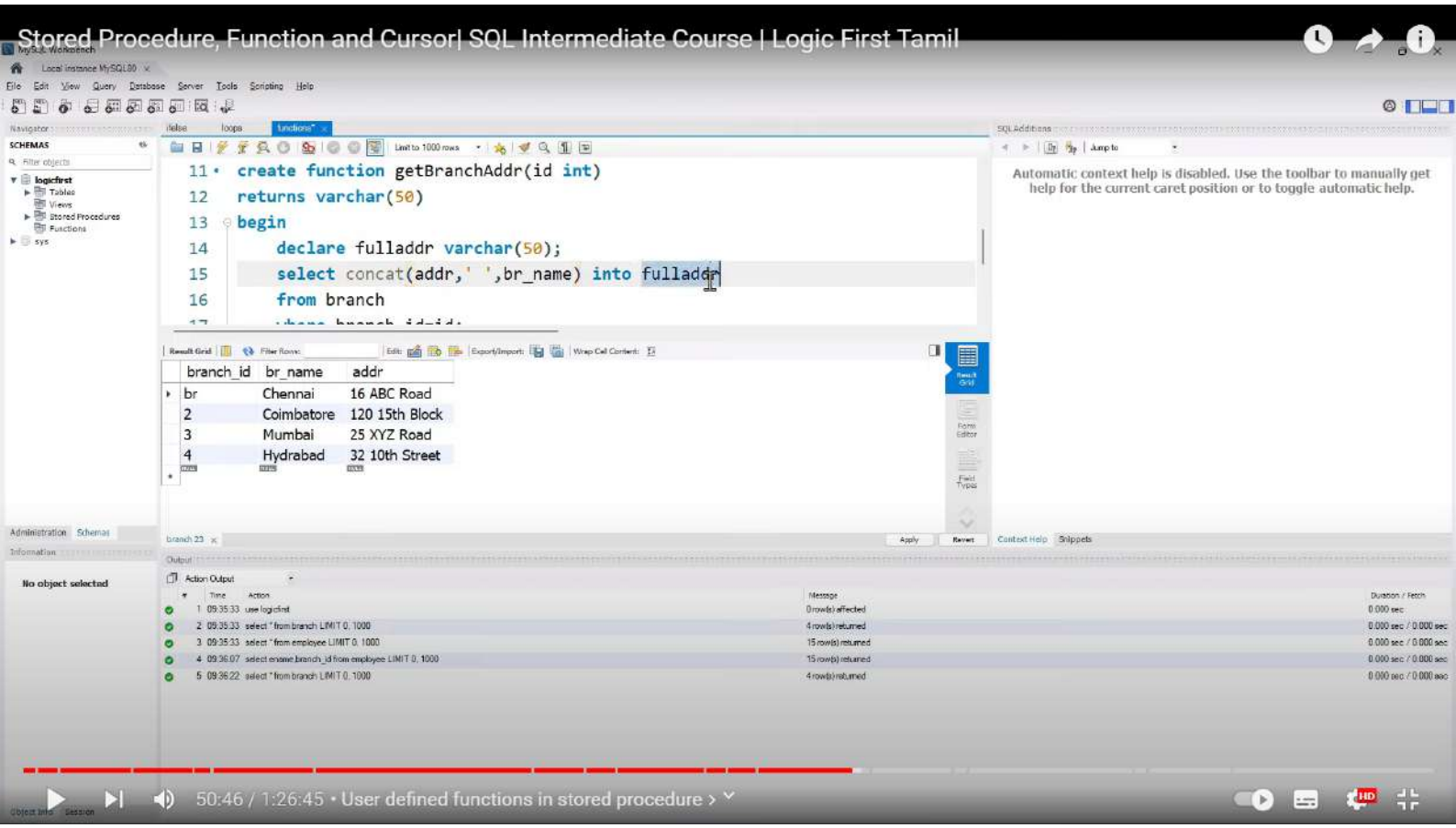

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
Columns
emp_id
ename
job_desc
salary
branch_id
Indexes
Foreign Keys
Triggers
student
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
query2*
Limit to 1000 rows
Execute the selected portion of the script or everything, if there is no selection
42 INOUT
43 */
44
45
46 DELIMITER $$
47 CREATE PROCEDURE IncrCounter(
48 INOUT counter INT,
49 IN incr INT
50)
51 BEGIN
52 SET counter = counter + incr;
53 END$$
54 DELIMITER ;
55
56 SET @counter=5;
57 CALL IncrCounter(@counter,2);
58 CALL IncrCounter(@counter,3);
59
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
7 10:52:51 SET @total = 10 0 row(s) affected 0.000 sec
8 10:52:51 CALL employeecnt('ADMIN',@total) 1 row(s) affected 0.000 sec
9 10:52:51 SELECT @total LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
10 10:53:52 CALL employeecnt('CEO',@total) 1 row(s) affected 0.000 sec
11 10:53:52 SELECT @total LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
12 11:06:46 CREATE PROCEDURE IncrCounter(INOUT counter INT, IN incr INT) BEGIN SET counter = counter + incr; END 0 row(s) affected 0.000 sec / 0.000 sec

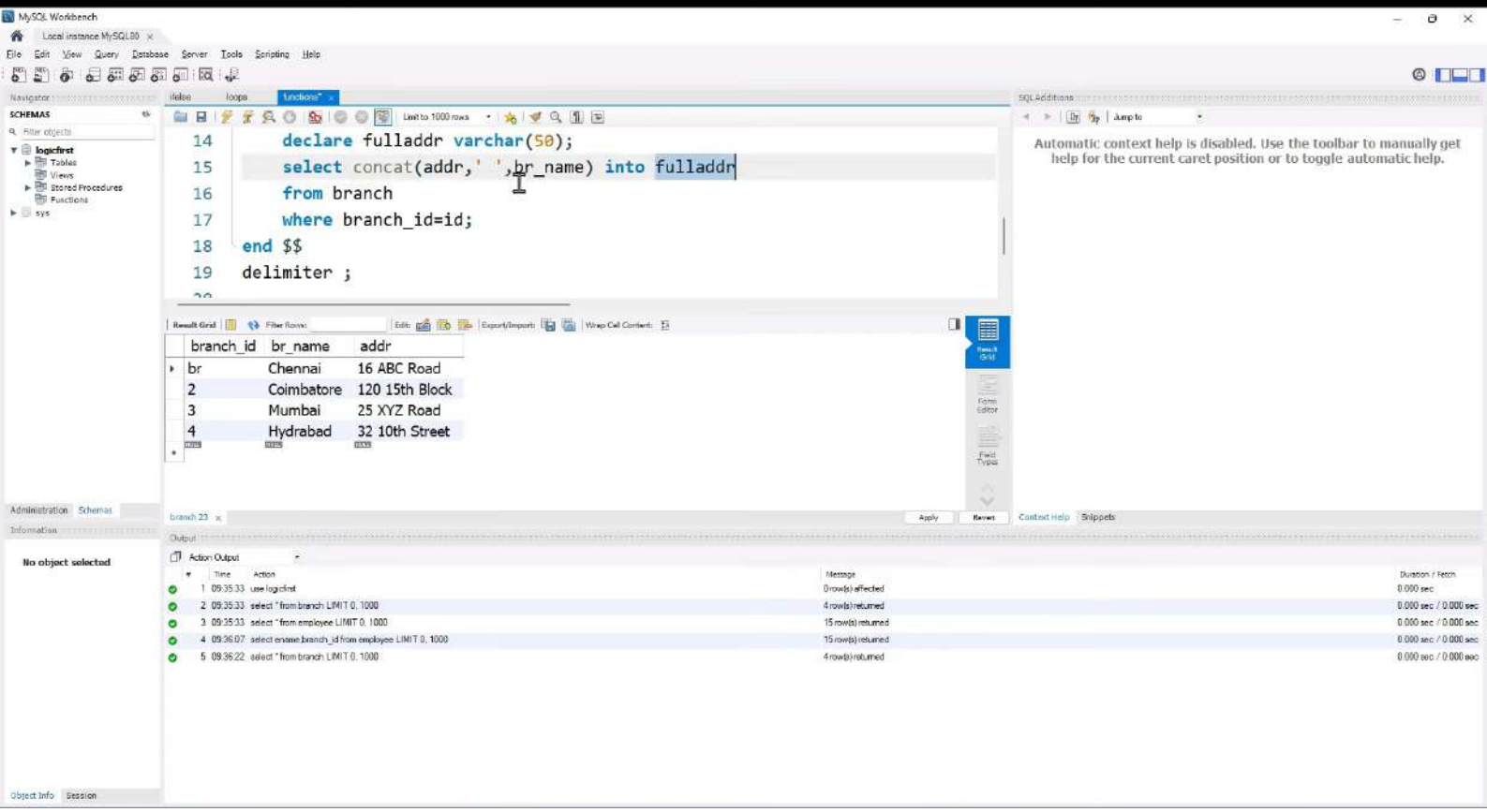

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
branch
clients
employee
Columns
emp_id
aname
job_desc
salary
branch_id
Indexes
Foreign Keys
Triggers
student
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
query2*
Limit to 1000 rows
51 BEGIN
52 SET counter = counter + incr;
53 END$$
54 DELIMITER ;
55
56 • SET @counter=5;
57 • CALL IncrCounter(@counter,2);
58 • SELECT @counter;
59 • CALL IncrCounter(@counter,3);
60
Result Grid Filter Rows: Export: Wrap Cell Content:
@counter
7
Result 15
Read Only
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
10 10:53:52 CALL employeecnt('CEO',@total) 1 row(s) affected 0.000 sec
11 10:53:52 SELECT @total LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
12 11:06:46 CREATE PROCEDURE IncrCounter(INOUT counter INT, IN incr INT) BEGIN SET counter = counter + incr; END 0 row(s) affected 0.016 sec
13 11:06:55 SET @counter=5 0 row(s) affected 0.000 sec
14 11:06:55 CALL IncrCounter(@counter,2) 0 row(s) affected 0.000 sec
15 11:07:07 SELECT @counter LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

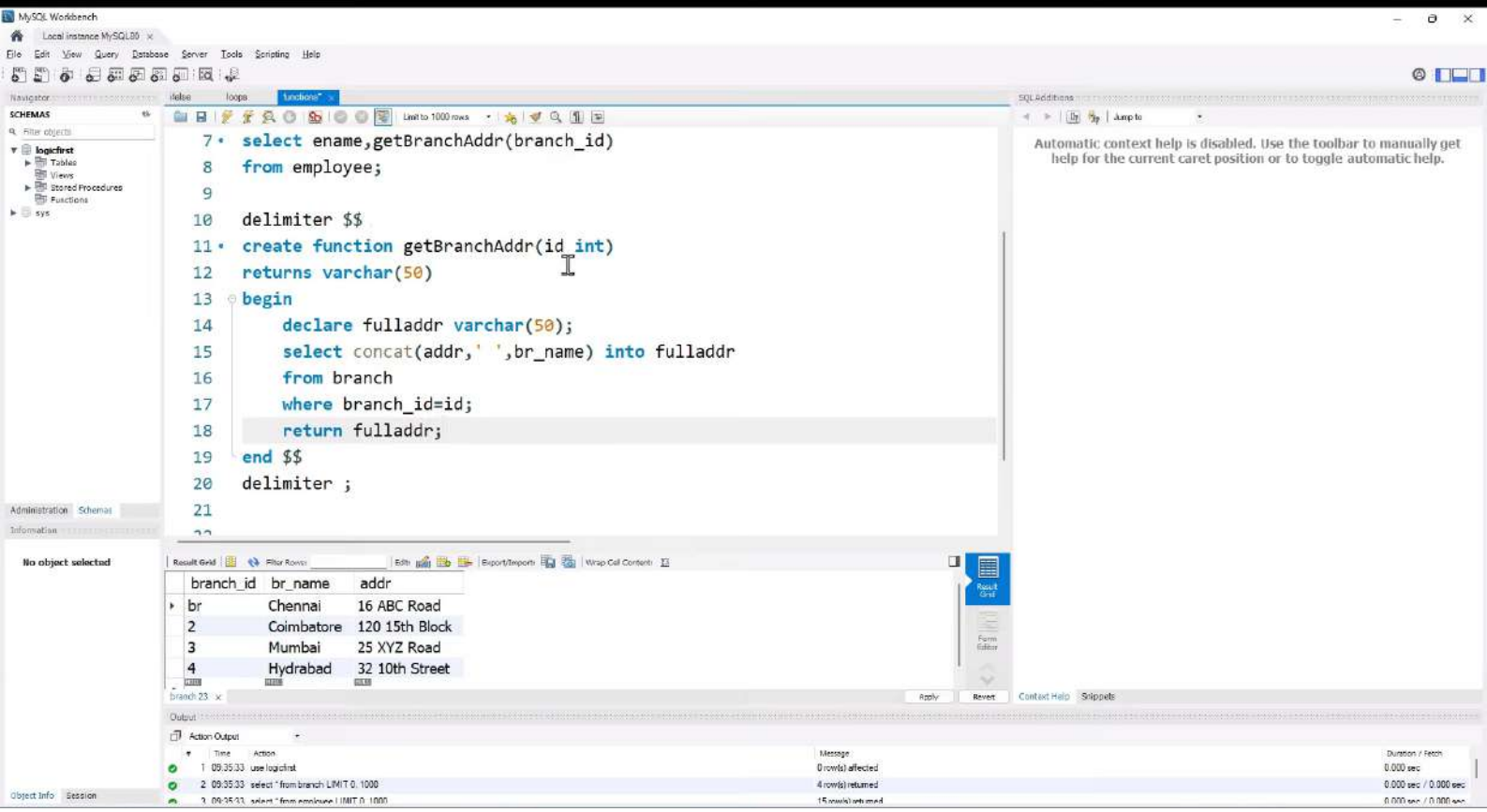

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil
DELIMITER $$
CREATE PROCEDURE GetState(
IN id INT
)
BEGIN
IF id=1 OR id=2 THEN
SELECT 'Tamil Nadu';
ELSEIF id=3 THEN
SELECT 'Maharashtra';
ELSE
SELECT 'Andra Pradesh';
END IF;
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
33:44 / 1:26:45 • If Else ›

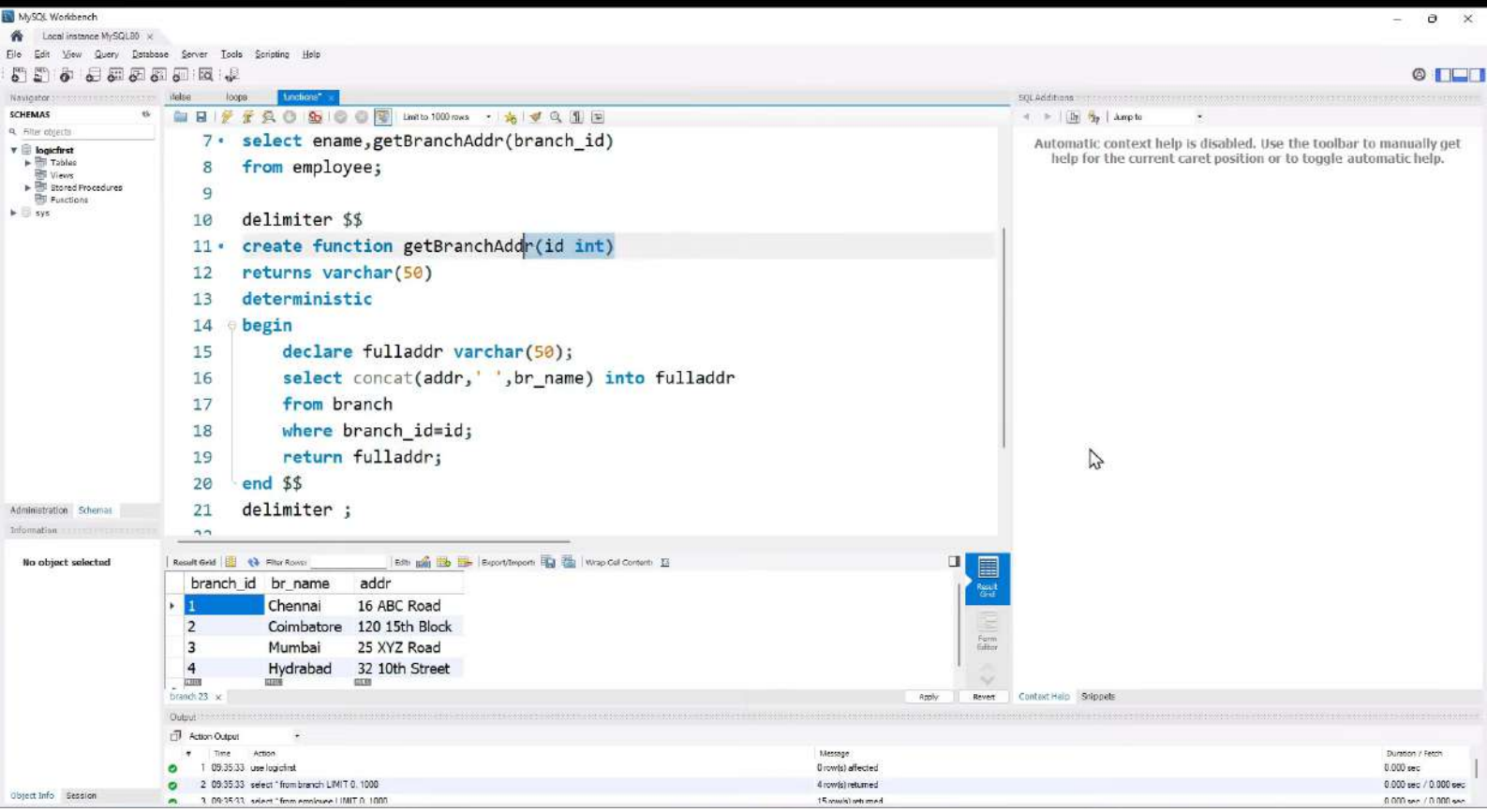

MySQL Workbench
Local instance MySQL80 (logicfirst)
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
employeecct
IncrCounter
Functions
sys
SQL File 3*
Limit to 1000 rows
15 ELSE
16 SELECT 'Andra Pradesh';
17 END IF;
18 END$$
19 DELIMITER ;
20
21 • CALL GetState(1);
22
Result Grid
Tamil Nadu
Tamil Nadu
Read Only
SQL Additions
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 10:01:46 CREATE PROCEDURE GetState(IN id INT) BEGIN IF id=1 OR id=2 THEN SELECT 'Tamil Nadu', ELSEIF id=3 THEN SELECT 'Maharashtra', ELS... 0 row(s) affected 0.015 sec
2 10:02:28 use logicfirst 0 row(s) affected 0.000 sec
3 10:02:28 select * from branch LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
4 10:02:28 CREATE PROCEDURE GetState(IN id INT) BEGIN IF id=1 OR id=2 THEN SELECT 'Tamil Nadu', ELSEIF id=3 THEN SELECT 'Maharashtra', ELS... Error Code: 1304. PROCEDURE GetState already exists 0.000 sec
5 10:02:48 CALL GetState(1) 1 row(s) returned 0.000 sec / 0.000 sec
No object selected

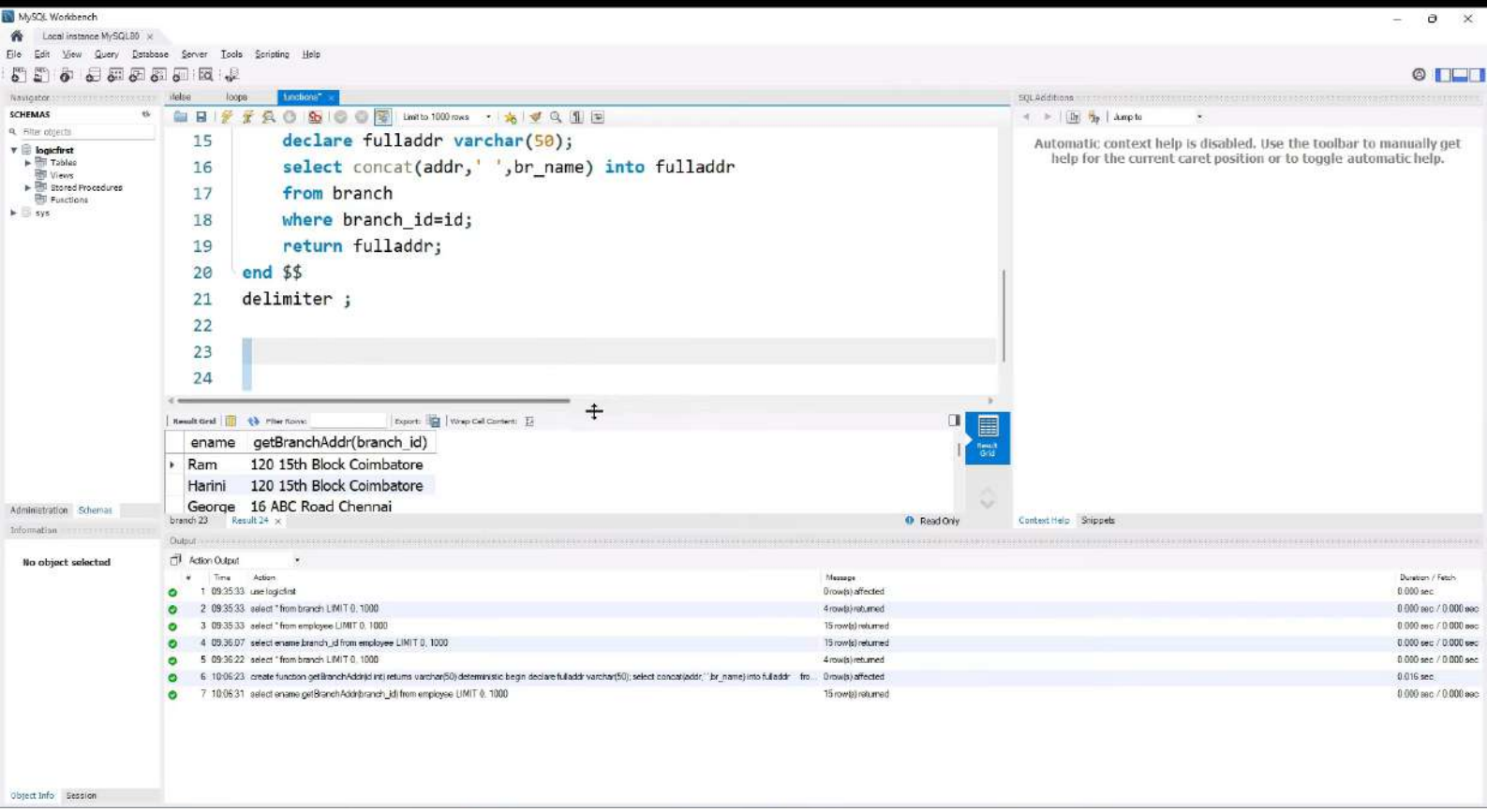

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil
create function getBranchAddr(id int)
returns varchar(50)
begin
declare fulladdr varchar(50);
select concat(addr,' ',br_name) into fulladdr
from branch
branch_id br_name addr
br Chennai 16 ABC Road
2 Coimbatore 120 15th Block
3 Mumbai 25 XYZ Road
4 Hydrabad 32 10th Street
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
50:48 / 1:26:45 • User defined functions in stored procedure >

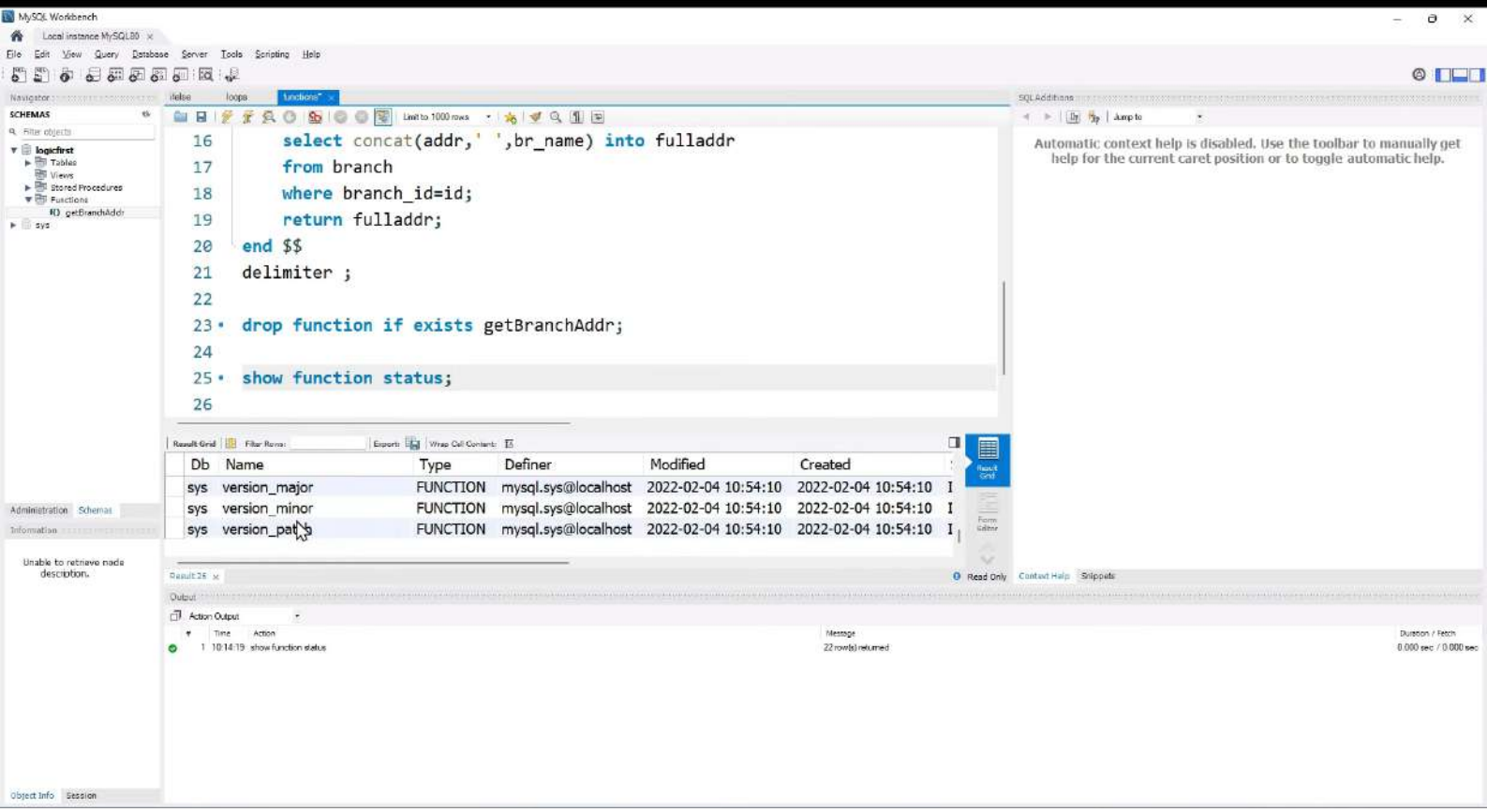

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil
create function getBranchAddr(id int)
returns varchar(50)
begin
declare fulladdr varchar(50);
select concat(addr,' ',br_name) into fulladdr
from branch
branch_id br_name addr
br Chennai 16 ABC Road
2 Coimbatore 120 15th Block
3 Mumbai 25 XYZ Road
4 Hydrabad 32 10th Street
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
50:46 / 1:26:45 • User defined functions in stored procedure >

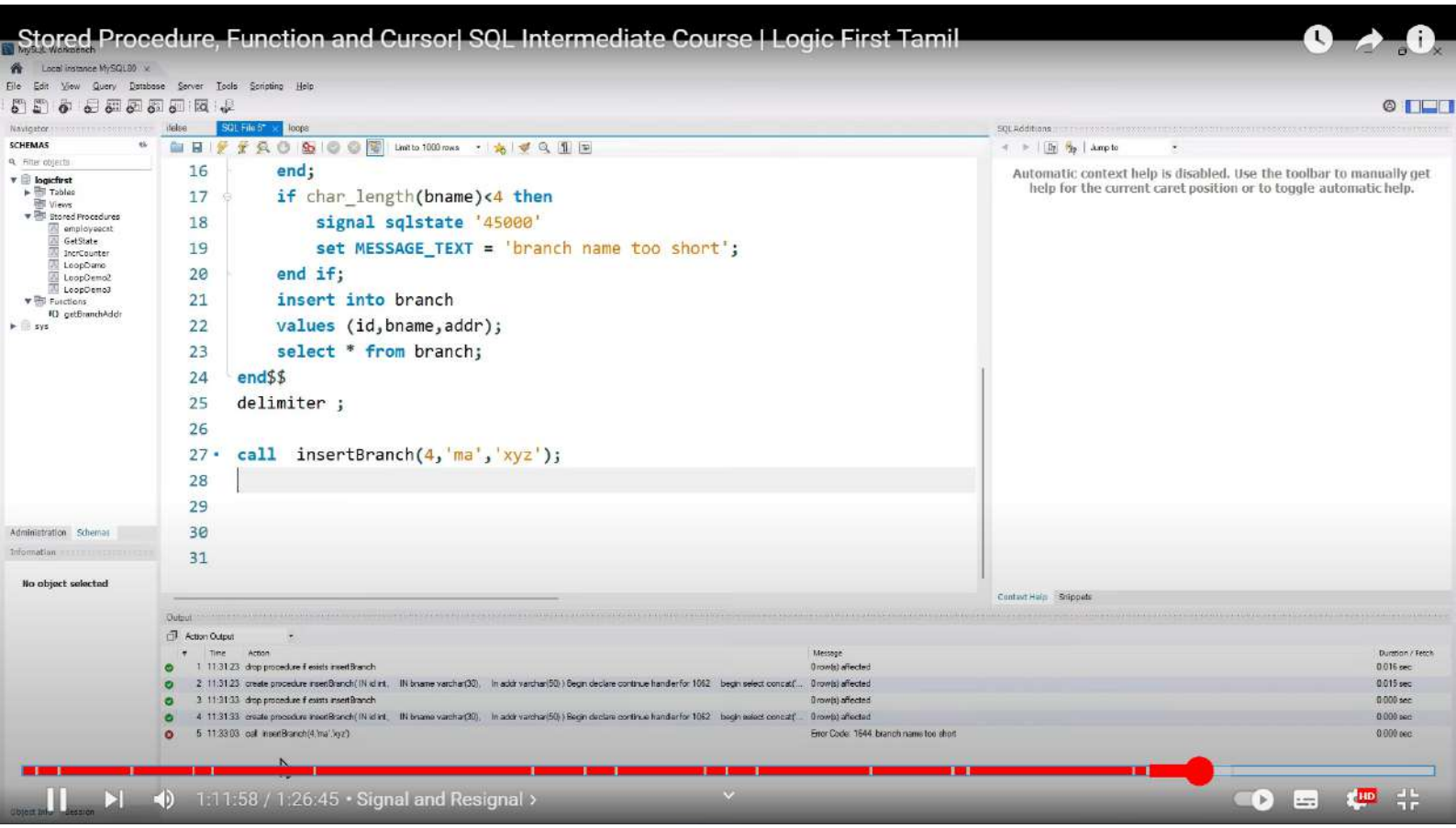

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
delete loops functions*
Limit to 1000 rows
14 declare fulladdr varchar(50);
15 select concat(addr,' ',br_name) into fulladdr
16 from branch
17 where branch_id=id;
18 end $$
19 delimiter ;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
branch_id br_name addr
br Chennai 16 ABC Road
2 Coimbatore 120 15th Block
3 Mumbai 25 XYZ Road
4 Hydrabad 32 10th Street
branch 23
Apply Revert
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 09:35:33 use logicfirst 0 row(s) affected 0.000 sec
2 09:35:33 select * from branch LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
3 09:35:33 select * from employee LIMIT 0, 1000 15 row(s) returned 0.000 sec / 0.000 sec
4 09:36:07 select ename,branch_id from employee LIMIT 0, 1000 15 row(s) returned 0.000 sec / 0.000 sec
5 09:36:22 select * from branch LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec

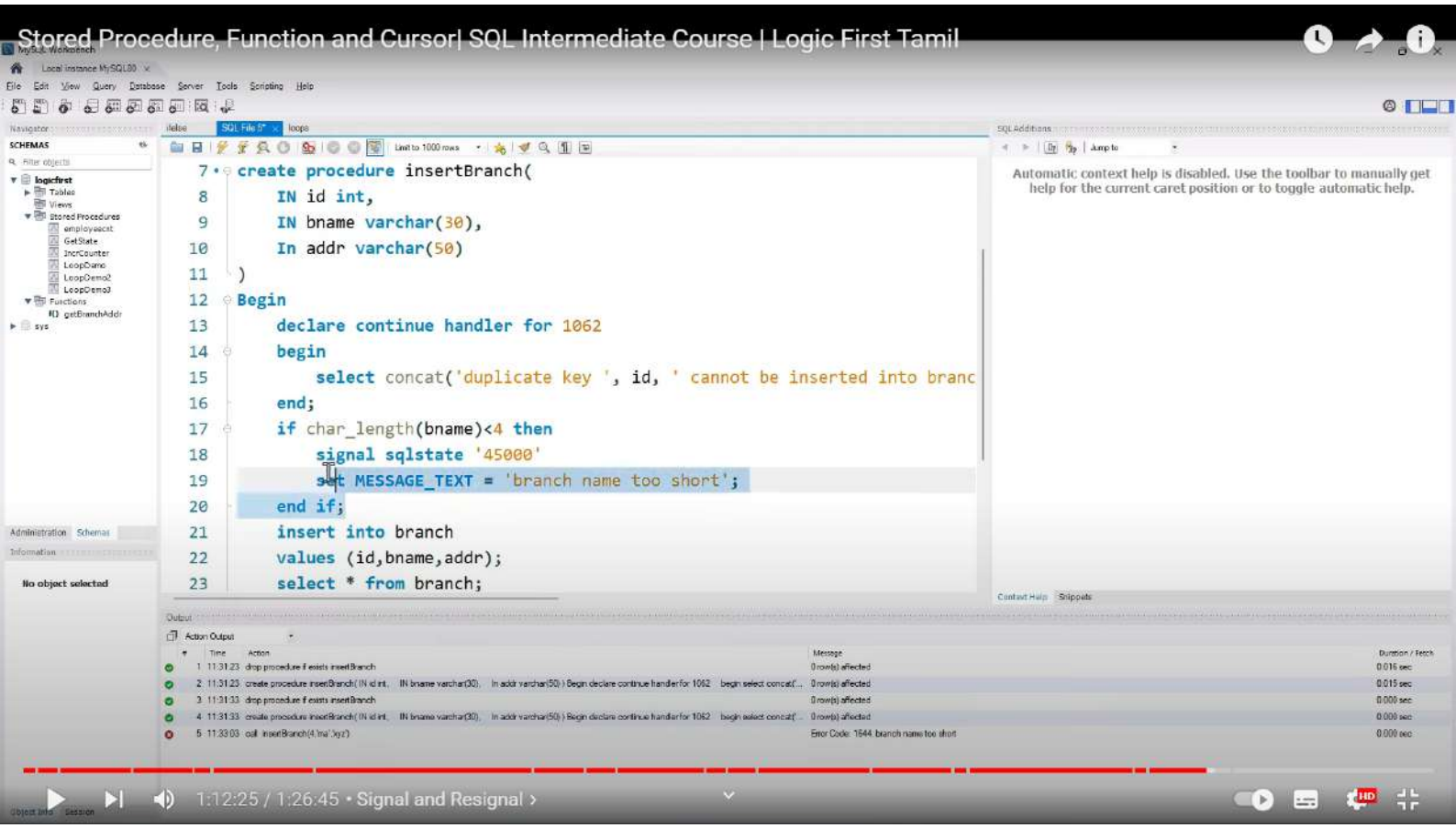

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
delse loops functions*
Limit to 1000 rows
7 • select ename,getBranchAddr(branch_id)
8 from employee;
9
10 delimiter $$
11 • create function getBranchAddr(id int)
12 returns varchar(50)
13 begin
14 declare fulladdr varchar(50);
15 select concat(addr,' ',br_name) into fulladdr
16 from branch
17 where branch_id=id;
18 return fulladdr;
19 end $$
20 delimiter ;
21
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
branch_id br_name addr
br Chennai 16 ABC Road
2 Coimbatore 120 15th Block
3 Mumbai 25 XYZ Road
4 Hydrabad 32 10th Street
branch 23
Apply Revert
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 09:35:33 use logicfirst 0 row(s) affected 0.000 sec
2 09:35:33 select * from branch LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec

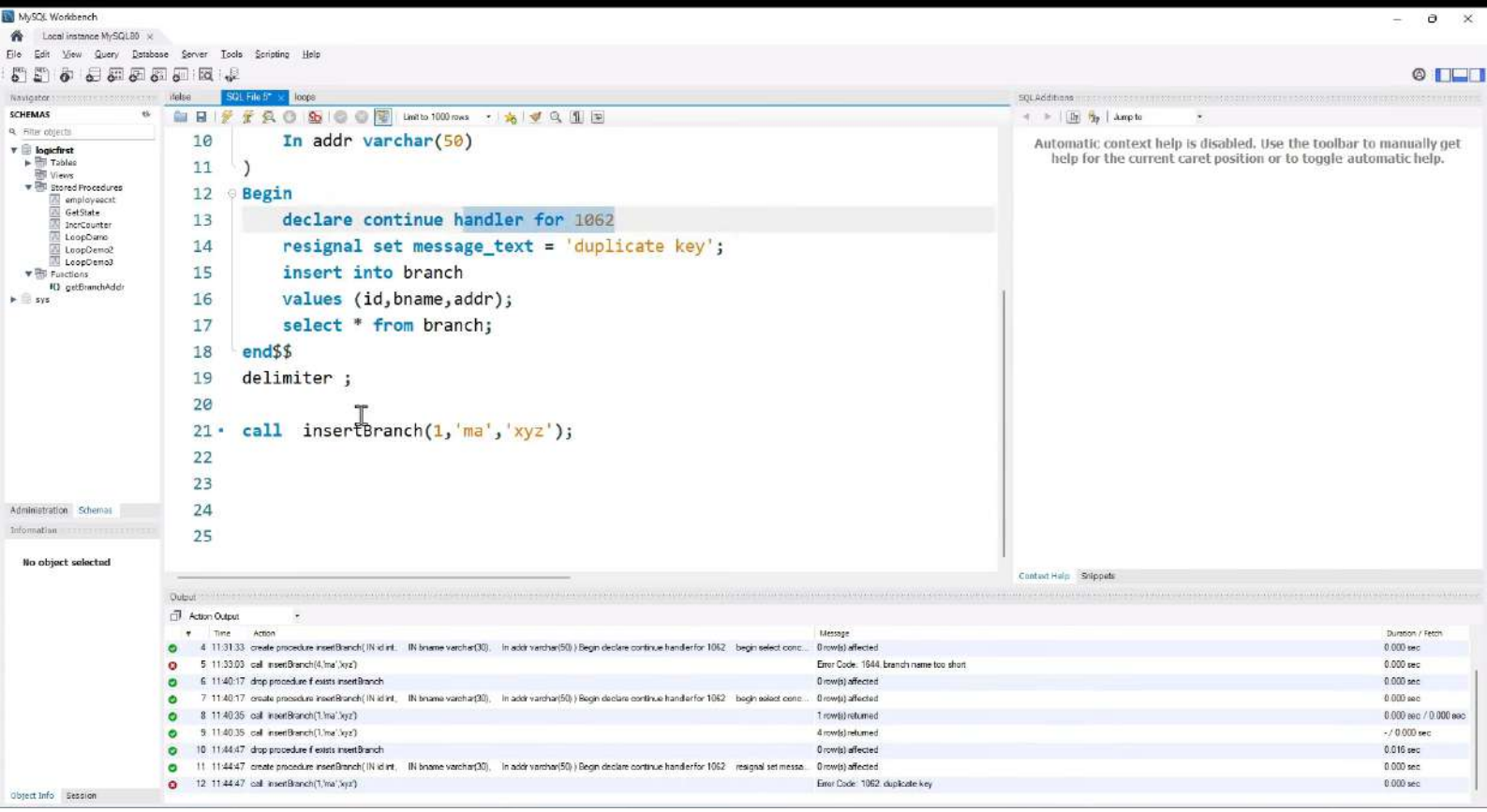

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
delse loops functions*
Limit to 1000 rows
7 • select ename,getBranchAddr(branch_id)
8 from employee;
9
10 delimiter $$
11 • create function getBranchAddr(id int)
12 returns varchar(50)
13 deterministic
14 begin
15 declare fulladdr varchar(50);
16 select concat(addr,' ',br_name) into fulladdr
17 from branch
18 where branch_id=id;
19 return fulladdr;
20 end $$
21 delimiter ;
Result Grid
branch_id br_name addr
1 Chennai 16 ABC Road
2 Coimbatore 120 15th Block
3 Mumbai 25 XYZ Road
4 Hydrabad 32 10th Street
Apply Revert
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
1 09:35:33 use logicfirst 0 row(s) affected 0.000 sec
2 09:35:33 select * from branch LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec

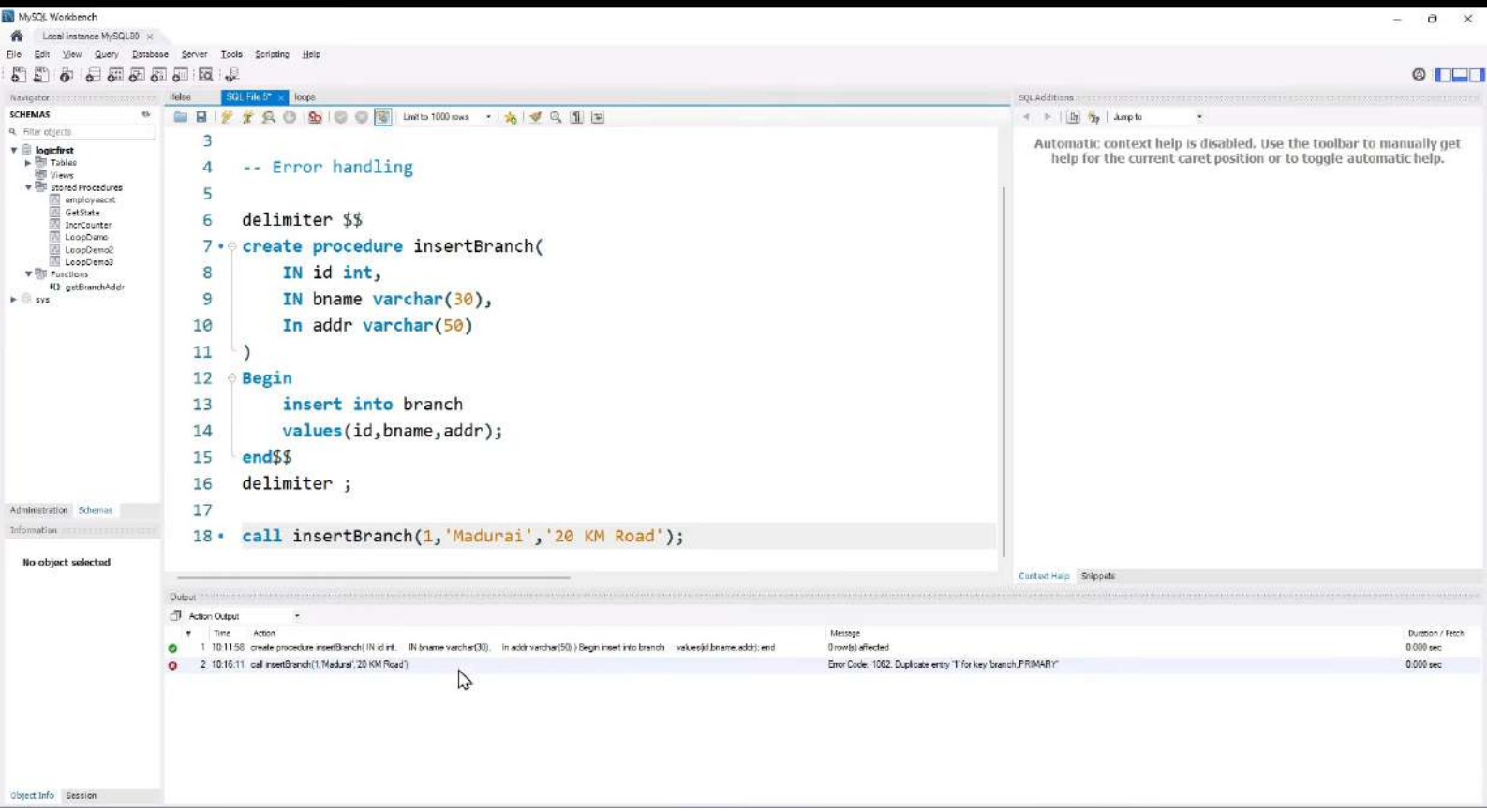

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
delloe loops functions*
Limit to 1000 rows
15 declare fulladdr varchar(50);
16 select concat(addr,' ',br_name) into fulladdr
17 from branch
18 where branch_id=id;
19 return fulladdr;
20 end $$
21 delimiter ;
22
23
24
Result Grid Filter Rows: Export: Wrap Cell Content:
ename getBranchAddr(branch_id)
Ram 120 15th Block Coimbatore
Harini 120 15th Block Coimbatore
George 16 ABC Road Chennai
branch 23 Result 24
Read Only
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 09:35:33 use logicfirst 0 row(s) affected 0.000 sec
2 09:35:33 select * from branch LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
3 09:35:33 select * from employee LIMIT 0, 1000 15 row(s) returned 0.000 sec / 0.000 sec
4 09:36:07 select ename,branch_id from employee LIMIT 0, 1000 15 row(s) returned 0.000 sec / 0.000 sec
5 09:36:22 select * from branch LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
6 10:06:23 create function getBranchAddr(id int) returns varchar(50) deterministic begin declare fulladdr varchar(50); select concat(addr,' ',br_name) into fulladdr fro... 0 row(s) affected 0.016 sec
7 10:06:31 select ename,getBranchAddr(branch_id) from employee LIMIT 0, 1000 15 row(s) returned 0.000 sec / 0.000 sec

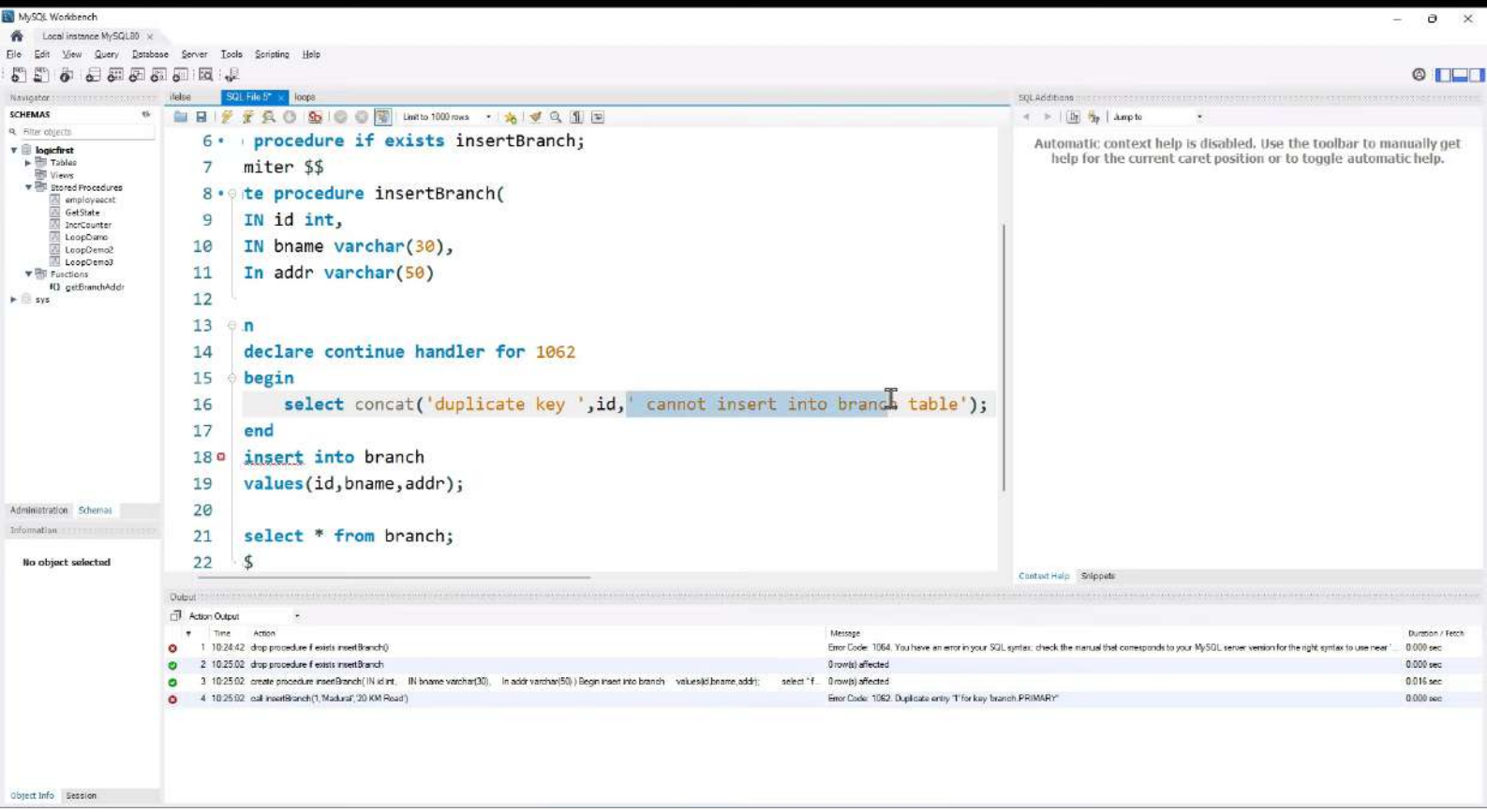

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
Functions
getBranchAddr
sys
Administration Schemas
Information
Unable to retrieve node description.
Object Info Session
delloe loops functions*
Limit to 1000 rows
16 select concat(addr,' ',br_name) into fulladdr
17 from branch
18 where branch_id=id;
19 return fulladdr;
20 end $$
21 delimiter ;
22
23 drop function if exists getBranchAddr;
24
25 show function status;
26
Result Grid Filter Rows: Export: Wrap Cell Content:
Db Name Type Definer Modified Created
sys version_major FUNCTION mysql.sys@localhost 2022-02-04 10:54:10 2022-02-04 10:54:10 I
sys version_minor FUNCTION mysql.sys@localhost 2022-02-04 10:54:10 2022-02-04 10:54:10 I
sys version_patch FUNCTION mysql.sys@localhost 2022-02-04 10:54:10 2022-02-04 10:54:10 I
Result 26
Read Only
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 10:14:19 show function status 22 row(s) returned 0.000 sec / 0.000 sec

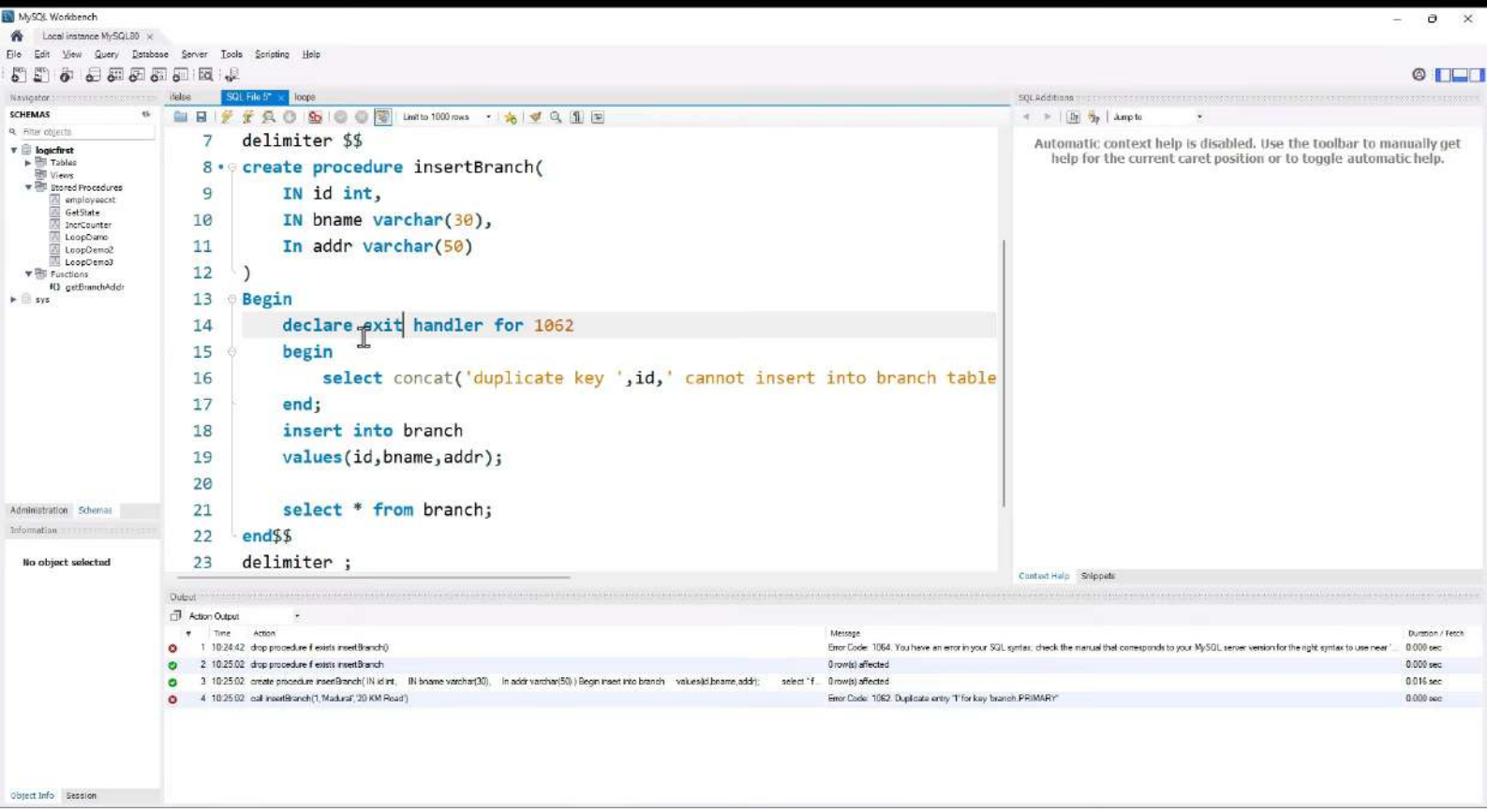

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil
end;
if char_length(bname)<4 then
signal sqlstate '45000'
set MESSAGE_TEXT = 'branch name too short';
end if;
insert into branch
values (id,bname,addr);
select * from branch;
end$$
delimiter ;
call insertBranch(4,'ma','xyz');
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
1:11:58 / 1:26:45 • Signal and Resignal ›


Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil
create procedure insertBranch(
IN id int,
IN bname varchar(30),
In addr varchar(50)
)
Begin
declare continue handler for 1062
begin
select concat('duplicate key ', id, ' cannot be inserted into branc
end;
if char_length(bname)<4 then
signal sqlstate '45000'
set MESSAGE_TEXT = 'branch name too short';
end if;
insert into branch
values (id,bname,addr);
select * from branch;
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
1:12:25 / 1:26:45 • Signal and Resignal ›


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
employeecct
GetState
IncrCounter
LoopDemo
LoopDemo2
LoopDemo3
Functions
getBranchAddr
sys
Administration Schemas
Information
No object selected
Object Info Session
ifelse
SQL File 5*
loops
Limit to 1000 rows
10 In addr varchar(50)
11)
12 Begin
13 declare continue handler for 1062
14 resignal set message_text = 'duplicate key';
15 insert into branch
16 values (id,bname,addr);
17 select * from branch;
18 end$$
19 delimiter ;
20
21 call insertBranch(1,'ma','xyz');
22
23
24
25
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
4 11:31:33 create procedure insertBranch(IN id int, IN bname varchar(30), In addr varchar(50)) Begin declare continue handler for 1062 begin select conc... 0 row(s) affected 0.000 sec
5 11:33:03 call insertBranch(4,'ma','xyz') Error Code: 1644. branch name too short 0.000 sec
6 11:40:17 drop procedure if exists insertBranch 0 row(s) affected 0.000 sec
7 11:40:17 create procedure insertBranch(IN id int, IN bname varchar(30), In addr varchar(50)) Begin declare continue handler for 1062 begin select conc... 0 row(s) affected 0.000 sec
8 11:40:35 call insertBranch(1,'ma','xyz') 1 row(s) returned 0.000 sec / 0.000 sec
9 11:40:35 call insertBranch(1,'ma','xyz') 4 row(s) returned - / 0.000 sec
10 11:44:47 drop procedure if exists insertBranch 0 row(s) affected 0.016 sec
11 11:44:47 create procedure insertBranch(IN id int, IN bname varchar(30), In addr varchar(50)) Begin declare continue handler for 1062 resignal set messa... 0 row(s) affected 0.000 sec
12 11:44:47 call insertBranch(1,'ma','xyz') Error Code: 1062. duplicate key 0.000 sec


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
employeecct
GetState
IncrCounter
LoopDemo
LoopDemo2
LoopDemo3
Functions
getBranchAddr
sys
Administration Schemas
Information
No object selected
Object Info Session
ifelse
SQL File 5*
loops
Limit to 1000 rows
3
4 -- Error handling
5
6 delimiter $$
7 create procedure insertBranch(
8 IN id int,
9 IN bname varchar(30),
10 In addr varchar(50)
11)
12 Begin
13 insert into branch
14 values(id,bname,addr);
15 end$$
16 delimiter ;
17
18 call insertBranch(1,'Madurai','20 KM Road');
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 10:11:58 create procedure insertBranch(IN id int, IN bname varchar(30), In addr varchar(50)) Begin insert into branch values(id,bname,addr); end 0 row(s) affected 0.000 sec
2 10:16:11 call insertBranch(1,'Madurai','20 KM Road') Error Code: 1062. Duplicate entry '1' for key 'branch.PRIMARY' 0.000 sec


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
employeecct
GetState
IncrCounter
LoopDemo
LoopDemo2
LoopDemo3
Functions
getBranchAddr
sys
Administration Schemas
Information
No object selected
Object Info Session
delise
SQL File 5*
loops
Limit to 1000 rows
6 • procedure if exists insertBranch;
7 miter $$
8 • te procedure insertBranch(
9 IN id int,
10 IN bname varchar(30),
11 In addr varchar(50)
12
13 n
14 declare continue handler for 1062
15 begin
16 select concat('duplicate key ',id,' cannot insert into branch table');
17 end
18 insert into branch
19 values(id,bname,addr);
20
21 select * from branch;
22 $
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 10:24:42 drop procedure if exists insertBranch() Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '... 0.000 sec
2 10:25:02 drop procedure if exists insertBranch 0 row(s) affected 0.000 sec
3 10:25:02 create procedure insertBranch(IN id int, IN bname varchar(30), In addr varchar(50)) Begin insert into branch values(id,bname,addr); select "f... 0 row(s) affected 0.016 sec
4 10:25:02 call insertBranch(1,'Madurai','20 KM Road') Error Code: 1062. Duplicate entry '1' for key branch.PRIMARY' 0.000 sec


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
logicfirst
Tables
Views
Stored Procedures
employeecct
GetState
IncrCounter
LoopDemo
LoopDemo2
LoopDemo3
Functions
getBranchAddr
sys
Administration Schemas
Information
No object selected
Object Info Session
fakse
SQL File 5*
loops
Limit to 1000 rows
7 delimiter $$
8 create procedure insertBranch(
9 IN id int,
10 IN bname varchar(30),
11 In addr varchar(50)
12)
13 Begin
14 declare exit handler for 1062
15 begin
16 select concat('duplicate key ',id,' cannot insert into branch table
17 end;
18 insert into branch
19 values(id,bname,addr);
20
21 select * from branch;
22 end$$
23 delimiter ;
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 10:24:42 drop procedure if exists insertBranch() Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '... 0.000 sec
2 10:25:02 drop procedure if exists insertBranch 0 row(s) affected 0.000 sec
3 10:25:02 create procedure insertBranch(IN id int, IN bname varchar(30), In addr varchar(50)) Begin insert into branch values(id,bname,addr); select * f... 0 row(s) affected 0.016 sec
4 10:25:02 call insertBranch(1,'Madurai','20 KM Road') Error Code: 1062. Duplicate entry '1' for key branch.PRIMARY' 0.000 sec