

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 Latest 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 Microsoft 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 instance - we can have 32767 DBs.
- The instance will allocate/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 interact 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. Integration Services (SSIS)

** SSMS will not support MSBI tools (SSAS, SSRS, SSIS)

SERVER TYPES

DATABASE ENGINE :

- its a core component 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 data
- SECURITY -- Provides security for data

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
char 8000 characters	It's a fixed-length character string and we can store maximum Based on defined width
varchar(n) 8000 characters	It's a variable-length character string and we can store maximum 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 4000 characters	It's a fixed-length character string and we can store maximum
nvarchar(n) 4000 characters	It's a variable-length character string and we can store maximum
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 maximum 2GB data	It is used to store XML formatted data and we can store
table later time	It is used to store table result set for processing at a
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 stored 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
- Relational Database, Stores data in form of a table that can be easily retrieved.

=====

=====

What is SQL?

- SQL is a language to communicate with database
- It's a DOMAIN SPECIFIC LANGUAGE (DSL)/Scripting Language
- SEQUEL - Structured English Query Language
- SEQUEL was developed by IBM corporation
- SEQUEL later became [SQL] but still pronounced 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, Data 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 relational 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. Retrive 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 permanent

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

OPERATORS:

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

We have diff. types of OPERATORS:

1. SQL Arithmetic Operators
2. SQL Logical Operators
3. SQL Comparison Operators
4. Assignment Operators

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

1. SQL Arithmetic Operators :

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

SYNTAX :

```
SELECT <Column_Name>, <column_nameX> (arithmetic 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 Operator is useful 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 useful to search for the  
specified value that matches with any value in the set of multiple values or NOT  
// NOT IN - retrieves rest other values
```

-- LIKE // NOT LIKE -- LIKE operator searches for the character string with specified 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 Comparison Operators

```
-- NOT EQUAL           -- != OR <>
-- EQUAL               -- =
-- GREATER THAN EQUAL TO -- >=
-- LESS THAN 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....., aggregate 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>
```

||||| ||||| ||||| ||||| ||||| ||||| ||||| ||||| ||||| ||||| ||||| |||||

aggregate 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/resultSet in the form of ASC (Ascending) order or DESC (Descending) order

SYNTAX :

```
SELECT COL1, COL2, COL3 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 implement 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 - Transaction 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 TRANSACTIONS

--COMMIT

--Chance of ROLLING BACK without COMMIT keyword

BEGIN TRANSACTION

<write statement>

--No chance of ROLLING BACK with COMMIT keyword

BEGIN TRANSACTION

<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 TRANSACTION

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 committed 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 Of --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 permanent, 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

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

DQL -

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

JOIN - Retrive the data from two or more tables

-- Two types :

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

---- INNER JOIN

Retrive 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

Retrieves matching data and unmatching data from
the table

---- LEFT OUTER JOIN

-- Retrives 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

--Retrives 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);
```

```
*/
```

```
--INNER 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
```

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

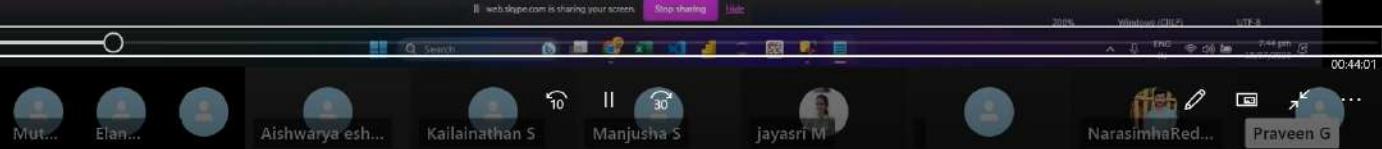
-- we must use ALIAS name for the table

-- 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.



SQLQuery1.sql - LAPTOP-UN6Q23EP\SQLPRAVEEN.nv (LAPTOP-UN6Q23EP)\P6 (67)* - Microsoft SQL Server Management Studio

Object Explorer File Edit View Query Project Tools Window Help

13 -- DQL satatement
14 SELECT * FROM tb
15
16
17 ---SELF JOIN syntax
18 =SELECT * FROM TB as t1, TB as t2
19 WHERE t1.salary = t2.salary
20 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

	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

Query executed successfully.

Snipping Tool

Screenshot copied to clipboard and saved

Select here to mark up and share the image

LAPTOP-UN6Q23EP\SQLPRAVEEN.nv (LAPTOP-UN6Q23EP)\P6 (67)*

Praveen G Ready 00:16:35

Mut... Elan... Aishwarya esh... Kailainathan S Manjusha S jayasri M NarasimhaRed... Praveen G

00:16:39 18/07/2023

```
18 SELECT * FROM TB as t1, TB as t2
19 WHERE t1.salary = t2.salary
20 AND T1.emp_name = 'AA'
```

The screenshot shows a SQL query being run in a Microsoft SQL Server Management Studio (SSMS) window. The query selects all columns from two tables, TB, aliased as t1 and t2, where the salary in t1 equals the salary in t2, and the employee name in t1 is 'AA'. The results are displayed in two tables. The first table, labeled T1, contains six rows of employee data. The second table, labeled T2, contains two rows of data, both corresponding to the row with emp_id 101 and emp_name 'AA' in the first table. Orange hand-drawn annotations highlight the WHERE clause conditions and the matching rows between the two tables.

	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

	emp_id	emp_name	salary	manager_id
1	101	AA	8000.00	101
2	101	AA	8000.00	101



```
--> D12 - 07072023 - JOINS
File Edit View
----- EQUI JOIN

-- When we retrieve the data from multiple tables based on an [EQUILITLY {=} ] 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 compare " " " " " " in the same table
```



SQLQuery4.sql - LAPTOP-UN6223EP\SQLPRAVEEN - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query New Item Object Explorer

equi_join

SQLQuery4.sql - L... UN6223EP\SQLPRAVEEN (55) SQLQuery3.sql - L... UN6223EP\SQLPRAVEEN (57)

Quick Launch (Ctrl+Q)

Execute

Change Type

equi_join

Execute

Results Messages

32
33 SELECT * FROM customer, company
34 WHERE customer.customerId = company.company_id

	id	name	num	customerId	company_id	company_name	salary
1	1	A	99999	101	101	Comp_X	50000.00
2	2	B	88888	102	102	Comp_Y	70000.00
3	3	C	77777	103	103	Comp_Z	30000.00
4					104	Comp_A	30000.00

	id	name	num	customerId	company_id	company_name	salary
1	1	A	99999	101	101	Comp_X	50000.00
2	2	B	88888	102	102	Comp_Y	70000.00
3	3	C	77777	103	103	Comp_Z	30000.00
4					104	Comp_A	30000.00

Query executed successfully.

l web.skype.com is sharing your screen. Stop sharing Hide

LAPTOP-UN6223EP\SQLPRAVEEN ... LAPTOP-UN6223EP\SQLPRAVEEN equi_join 00:00:30 4 rows

Praveen G 00:35:59

Praveen G 00:35:59

Mut... Elan... Aishwarya esh... Kailainathan S Manjusha S jayasri M NarasimhaRed... Praveen G

00:17:15

2. NON-ANSI JOINS

-- with 'WHERE' keyword join condition

---- EQUI JOIN

-- When we retrieve the data from multiple tables based on an [EQUALITY {=}] 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 retrieve 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

T

=====

---- SELF JOIN

Stop sharing Hide

182% Windows (C:\lf) UTF-8
00:40:39 8:16 pm 18/07/2019 00:12:35

Mut... Elan...

Aishwarya esh...

Kailainathan S

Manjusha S

jayasri M

NarasimhaRed...

Praveen G

SQLQuery1.sql - LAPTOP-UN6G23EP\SQLPRAVEEN.non_equi (LAPTOP-UN6G23EP\PG (0)) - Microsoft SQL Server Management Studio

```
18
19  --DQL
20  SELECT * FROM employee
21  SELECT * FROM salaryRange
22
23  ---|
24  SELECT * FROM employee, salaryRange
25  WHERE (salary > lowSal) and (salary < highSal)
26
27
```

Results

	id	name	salary
1	1	A	2500.00
2	2	B	3200.00
3	3	C	4000.00
4	4	D	5000.00
5	5	E	800.00

	no_	lowSal	highSal
1	1	1500	2800
2	2	3500	5200
3	3	6500	8000

Query executed successfully.

LAPTOP-UN6G23EP\SQLPRAVEEN... LAPTOP-UN6G23EP\PG (0) non_equi 00:06:30 | 8 rows

Praveen G Ready 00:47:02

Mut... Elan... Aishwarya esh... Kailainathan S Manjusha S jayasri M NarasimhaRed... Praveen G

00:06:12

```
1 --NOT NULL
2
3 --without NOT NULL
4 CREATE TABLE Notnull_constraint (
5     id int NOT NULL,
6     [name] varchar(100)
7 );
8
9 DROP TABLE no_null_constraint;
10
11 INSERT INTO no_null_constraint (id, [name]) VALUES
12 (1, NULL);
13
14 INSERT INTO no_null_constraint (id, [name]) VALUES
```

Completion time: 2023-07-12T18:19:58.9355895+05:30

SQLQuery1.sql - LAPTOP-UN6Q23EP\SQL_NEWINSTANCE.Constraints (LAPTOP-UN6Q23EP\PG (54)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Constraints

Execute

SQLQuery1.sql - L... - UN6Q23EP\PG (54) Constraints

Execute

Quick Launch (Ctrl+Q)

Object Explorer

Connected to: LAPTOP-UN6Q23EP\SQL_NEWINSTANCE (SQL Server 15)

Databases

System Databases

Database Snapshots

Constraints

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

sys.fn_notnull_constraint

Columns

Keys

Constraints

Triggers

Indexes

Statistics

Dropped Ledger Tables

Views

External Resources

Synonyms

Programmability

Query Store

Service Broker

Storage

Security

Set Operators

Security

Server Objects

Replication

Always On High Availability

Management

Integration Services Catalogs

SQL Server Agent (Agent XPs disabled)

XE Event Profiler

14 (NULL, 'Praveen');

15

16 INSERT INTO Notnull_constraint (id, [name]) VALUES

17 (2, NULL);

18

19 SELECT * FROM Notnull_constraint;

20

21 UPDATE Notnull_constraint

22 SET [name] = 'A'

23 WHERE [name] IS NULL

(2 rows affected)

Completion time: 2023-07-12T18:28:40.3953858+05:30

Query executed successfully.

Stop sharing Help

LAPTOP-UN6Q23EP\SQL_NEWINSTANCE - LAPTOP-UN6Q23EP\PG (54) Constraints 00:06:30 0 rows

Ready 00:16:16

Praveen G Praveen G 00:16:16

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

SQLQuery2.sql - LAPTOP-UN6223EP\SQL_NEWINSTANCE Constraints (LAPTOP-UN6223EP\PG (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect + Constraints

SQLQuery2.sql - L... - UN6223EP\PG (52)* = SQLQuery2.sql - L... - UN6223EP\PG (54)*

1 --UNIQUE

2

3 CREATE TABLE dept (

4 id int UNIQUE,

5 [name] varchar(100)

6)

SQL Server Object Explorer (LAPTOP-UN6223EP\SQL_NEWINSTANCE)

- System Databases
- Database Snapshots
- Constraints
- Database Diagrams
- Tables
- System Tables
- FileTables
- External Tables
- Graph Tables
- sys.schemas_constraint
- Dropped Ledger Tables
- Views
- External Resources
- Synonyms
- Programmability
- Query Store
- Service Broker
- Storge
- Security
- Set Operators
- Server
- Server Objects
- Replication
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

Praveen G 00:29:07 Party summary Nani... Vijaya... muthu lakshmi Aishwarya esh... jayasri M Manjusha S Bala Praveen G

Connected (1/1) Stop sharing Hide In 2 Col 1 Col 2 Ins 00:46:28 12/07/2023

SQLQuery2.sql - LAPTOP-UN6223EP\SQL_NEWINSTANCE.Constraints (LAPTOP-UN6223EP\PC (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer Constraints Execute Quick Launch (Ctrl+Q)

```

14 SELECT * FROM dept
15
16 --
17 ALTER TABLE dept
18 ADD CONSTRAINT name_unique
19 UNIQUE ([name]);

```

Results Messages

	name	varchar	no	100	yes	no	yes	SQL_Latin1_General_CI_AS
1	Identity		Seed		Increment		Not For Replication	
1	No identity column defined.		NULL	NULL	NULL			
1	RowGuidCol							
1	No rowguidcol column defined.							
1	Data_located_on_filegroup							
1	PRIMARY							
1	index_name		index_description		index_keys			
1	name_unique		nonclustered, unique, unique key located on PRIM...	name				
2	UQ_dept_3213E83ED6C6EE08		nonclustered, unique, unique key located on PRIM...	id				

Query executed successfully. 1 row(s) affected. 10 rows returned. 12/07/2023 00:42:08

LAPTOP-UN6223EP\SQL_NEWINST... LAPTOP-UN6223EP\PC (52) Constraints 00:00:01 10 rows

Praveen G 00:33:27 Skype is sharing your screen. Stop sharing Hide 16x16 Col 16 Ch 16 Bits

Jaga... Nari... Vijaya... muthu lakshmi Aishwarya esh... jayasri M Manjusha S Bala Praveen G

SQLQuery2.sql - LAPTOP-UH6Q23EP\SQL_NEWINSTANCE.Constraints (LAPTOP-UH6Q23EP\PG (S2)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer Constraints

Connect + Constraints

SQLQuery2.sql - L...UH6Q23EP\PG (S3) = SQLQuery2.sql - UH6Q23EP\PG (S4)

```

17 ALTER TABLE dept
18 ADD CONSTRAINT name_unique
19 UNIQUE ([name]);
20
21 ALTER TABLE dept
22 DROP CONSTRAINT UQ_dept_3213E83ED6C6EE08;

```

Results Messages

Name	Owner	Type	Created_datetime
dept	dbo	user table	2023-07-12 18:41:42.153

Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
id	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
name	varchar	no	100			yes	no	yes	SQL_Latin1_General_CI_AS

Identity	Seed	Increment	Not For Replication
No identity column defined.	NULL	NULL	NULL

RowGuidCol
No rowguidcol column defined.

Data_located_on_filegroup
PRIMARY

index_name	index_description	index_keys
UQ_dept_3213E83ED6C6EE08	nonclustered, unique, unique key located on PRIM...	id

constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
UNIQUE (non-clustered)	UQ_dept_3213E83ED6C6EE08	(n/a)	(n/a)	(n/a)	(n/a)	id

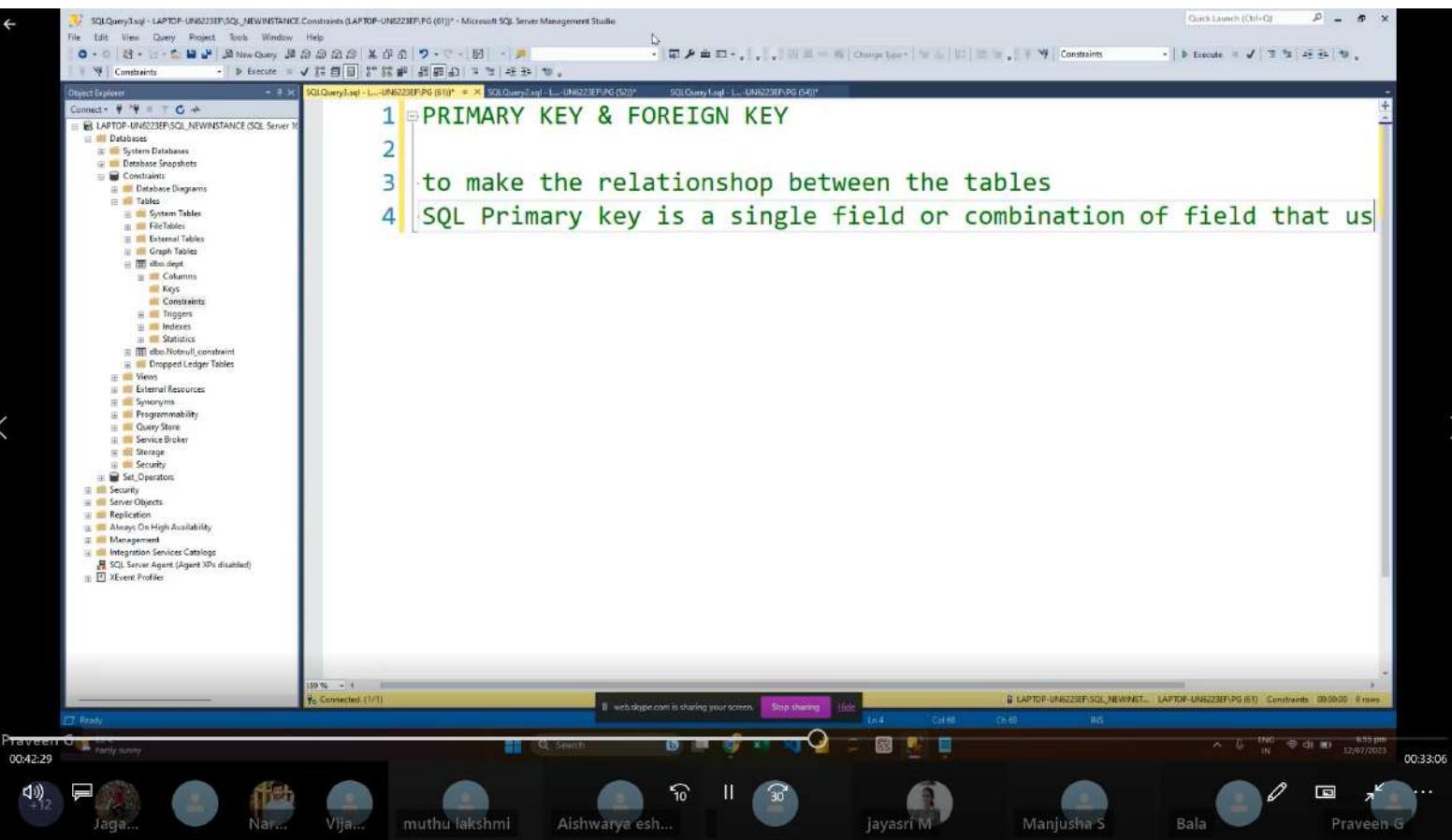
Query executed successfully.

Praveen G Party summary 00:36:12 12/07/2023 00:00:01 8 rows

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

00:39:23

1 PRIMARY KEY & FOREIGN KEY
2
3 to make the relationship between the tables
4 SQL Primary key is a single field or combination of field that us



The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. In the Object Explorer on the left, a connection to 'LAPTOP-UN6223EP\SQL_NEWINSTANCE' is selected. The 'Constraints' node under the 'Tables' section is expanded, showing various constraint types like 'Check', 'Default', 'Unique', and 'Unique Filter'. In the center pane, four lines of T-SQL code are visible:

```
1
2
3.es
4 combination of field that is used UNIQUE(ly) or it should not be NULL
```

The code appears to be a partial definition of a constraint, likely a Unique Constraint, as indicated by the 'es' abbreviation and the note about unique fields.

At the bottom of the screen, a Skype video call interface is visible, showing participants including 'Praveen G', 'Jaga...', 'Nar...', 'Vijaya...', 'muthu lakshmi', 'Aishwarya esh...', 'jayasri M', 'Manjusha S', 'Bala', and 'Praveen G'. The call is currently sharing the screen.

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure of 'LAPTOP-UN6223EP\SQL_NEWINSTANCE'. In the center, three query panes are visible, with the middle one containing the following T-SQL script:

```
1 --PRIMARY KEY & FOREIGN KEY
2
3 --to make the relationship between the tables
4 --SQL Primary key is a single field or combination of field that is used UNIQUE(1)
5 --a table can have only one PK
6
7 --Conditions are :
8     --1. One table should contain PK and other table should contain FK
9     --2. A common Column in tables should there
10    --3. Datatype must be same in both the tables
11
12 --DEPT & EMPLOYEE
13
14
15
```



The screenshot shows a Microsoft SQL Server Management Studio (SSMS) window with two query panes and a taskbar at the bottom.

Object Explorer: Shows the database structure for the 'LAPTOP-UN6223EP\SQL_NEWINSTANCE' instance. It includes nodes for Databases, Views, External Resources, Programmability, Security, and System Objects.

Query Windows:

- SQLQuery1.sql - LAPTOP-UN6223EP\SQL_NEWINSTANCE Constraints (LAPTOP-UN6223EP\PG (6))**: Contains the following T-SQL code:

```
11 --DEPT & EMPLOYEE
12
13
14 CREATE TABLE dept (
15     dep_no int PRIMARY KEY,
16     dep_location varchar(40)
17 )
18 Go
19
20 CREATE TABLE employee (
21     emp_id int,
22     emp_name varchar(100),
23     salary money,
24     dep_no int FOREIGN KEY REFERENCES dept(dep_no)
25 )
26
27
```
- SQLQuery2.sql - LAPTOP-UN6223EP\PG (5)**: Contains the following T-SQL code:

```
10
30
```
- SQLQuery3.sql - LAPTOP-UN6223EP\PG (54)**: Contains the following T-SQL code:

```
10
30
```

Taskbar: Shows the following information:

- Connected (1/1)
- Stop sharing
- LAPTOP-UN6223EP\SQL_NEWINSTANCE - LAPTOP-UN6223EP\PG (6) | Constraints | 00:00:00 | 0 rows
- 00:49:13
- 00:26:22

SQLQuery3.sql - LAPTOP-UN6023EP\SQL_NEWINSTANCE.PK_FK (LAPTOP-UN6023EP\PG (61)) - Microsoft SQL Server Management Studio

Object Explorer

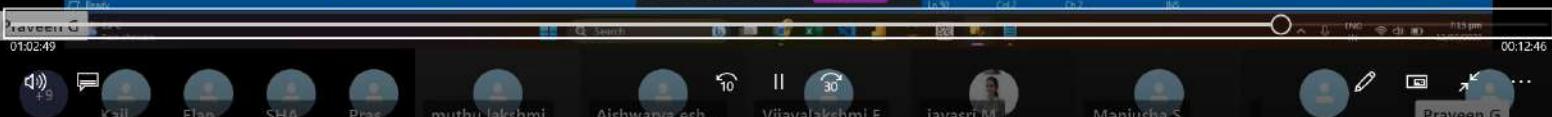
File Edit View Query Project Tools Window Help

PK_FK

```
46 INSERT INTO employee VALUES
47 (104, 'Katie', 16000, 40);
48
49
50 --DROP CONSTRAINT
51 ALTER TABLE dept
52 DROP CONSTRAINT [PK_dept_BB4BB72753B6D054]
53
54 ALTER TABLE employee
55 DROP CONSTRAINT FK_employee_dep_no_38996AB5
```

Commands completed successfully.

Completion time: 2023-07-12T19:15:16.7148724+05:30

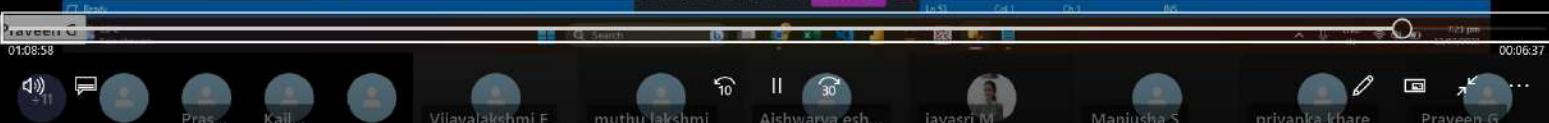


The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows a connection to 'LAPTOP-UN6223EP\SQL_NEWINSTANCE.PK_FK'. The main window contains the following T-SQL script:

```
46 INSERT INTO employee VALUES
47 (104, 'Katie', 16000, 40);
48
49 --ADD CONSTRAINT
50 ALTER TABLE dept
51 ALTER COLUMN dep_no int NOT NULL
52
53 ALTER TABLE dept
54 ADD CONSTRAINT PK_dept_no PRIMARY KEY (dep_no)
55
56
57 --DROP CONSTRAINT
58 ALTER TABLE dept
59 DROP CONSTRAINT [PK_dept_BB4BB72753B6D054]
```

The message pane at the bottom indicates: 'Commands completed successfully.' and 'Completion time: 2023-07-12T19:16:28.9488705+05:30'.

```
SQLQuery3.sql - LAPTOP-UN6Q23EP\SQL_NEWINSTANCE.PK_FK (LAPTOP-UN6Q23EP\PG (61)) Executing... - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
SQLQuery2.Log - L...UN6Q23EP\PG (52) SQLQuery1.Log - L...UN6Q23EP\PG (54)
Object Explorer
Connect + C A
LAPTOP-UN6Q23EP\SQL_NEWINSTANCE (SQL Server 10.0.2210.1)
Databases
System Databases
Database Snapshots
Constraints
PK_FK
Tables
System Tables
File Tables
External Tables
Graph Tables
dbo.dept
Columns
Keys
Triggers
Indexes
Statistics
dbo.employee
Columns
Keys
Constraints
Triggers
Indexes
Statistics
Dropped Ledger Tables
Views
External Resources
Synonyms
Programmability
Query Store
Service Broker
Storage
Security
Set_Operations
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
AEvent Profiler
51 ALTER COLUMN dep_no int NOT NULL
52
53 ALTER TABLE employee
54 ADD CONSTRAINT FK_KEY
55 FOREIGN KEY (dep_no) REFERENCES dept(dep_no)
56
57 SP_HELP dept
58
59 --DROP CONSTRAINT
60 ALTER TABLE dept
61 DROP CONSTRAINT [PK_dept_BB4BB72753B6D054]
62
63 ALTER TABLE employee
64 DROP CONSTRAINT FK_employee_dep_no_38996AB5
Results Messages
Query completed with errors.
Stop sharing Hide
LAPTOP-UN6Q23EP\SQL_NEWINST... | LAPTOP-UN6Q23EP\PG (61) | PK_FK | 00:00:30 | 0 rows
01:08:58 00:06:37
```



```
46 INSERT INTO employee VALUES
47 (104, 'Katie', 16000, 40);
48
49 --ADD CONSTRAINT
50 ALTER TABLE dept
51 ALTER COLUMN dep_no int NOT NULL
52
53 --PRIMARY KEY
54 ALTER TABLE dept
55 ADD CONSTRAINT [PK_dept_no] PRIMARY KEY(dep_no)
56
57 --FOREIGN KEY
58 ALTER TABLE employee
59 ADD CONSTRAINT FK_KEY
60 FOREIGN KEY (dep_no) REFERENCES dept(dep_no)
61
62 SP_HELP dept
63
64 --DROP CONSTRAINT
```

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows a connection to 'LAPTOP-UN6223EF\SQL_NEWINSTANCE'. The 'PK_FK' node under 'Tables' is expanded, revealing 'PK_dept_no' as a primary key constraint. The central pane displays a T-SQL script for managing constraints on the 'employee' and 'dept' tables. The script includes an 'INSERT' statement, alterations to the 'dept' table (adding a NOT NULL constraint to 'dep_no'), a creation of a primary key constraint named 'PK_dept_no' on 'dept', a creation of a foreign key constraint named 'FK_KEY' on 'employee' referencing 'dept', and a help command for 'dept'. The status bar at the bottom indicates 'Query executed successfully'.

```
1 --CHECK & -- DEFAULT
2 USE [Constraints];
3
4 CREATE TABLE dept_check_const (
5     id int NOT NULL,
6     [name] varchar(100),
7     salary int CHECK (salary > 10000)
8     Gender varchar(10) CHECK(gender = 'M' OR gender = 'F')
9 );
10
11 SP_HELP dept_check_const;
12
13 --this will work
14 INSERT dept_check_const VALUES
15 (1, 'Praveen', 15000);
16
17 --this will not work bcs salary must be greater than 10000
18 INSERT dept_check_const VALUES
19 (2, 'Rahul', 5000);
20
```

SQLQuery4.sql - LAPTOP-UH6223EP\SQL_NEWINSTANCE.Constraints (LAPTOP-UH6223EP\PG (3)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connected to: LAPTOP-UH6223EP\SQL_NEWINSTANCE (SQL Server 16.0.105)

Databases

System Databases

FileTables

Constraints

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

Views

Tables

System Tables

FileTables

External Tables

Graph Tables

Views

Triggers

Keys

Constraints

CK_dept_check_const

Columns

Indexes

Statistics

Notnull_constraint

Dropped Edge Tables

Views

External Resources

Synonyms

Programmability

Query Store

Service Broker

Storage

Security

PK_FK

Set_Operators

Security

Server Objects

Replication

Always On High Availability

Management

Integration Services Catalogs

SQL Server Agent (Agent XPs disabled)

XEvent Profiler

SQLQuery4.sql - L-UNH6223EP\PG (3) * - SQL Query (not connected)

SQLQuery2.sql - not connected*

SQLQuery1.sql - not connected*

Execute

Constraints

Execute

Quick Launch (Ctrl+Q)

20 INSERT dept_check_const VALUES
21 (2, 'Ramya', 9000);
22
23
24
--ALTER
25
26
27
28 ALTER TABLE dept_check_const
29 ADD CONSTRAINT CK_check_constraint_for_id
30 CHECK (id > 100);

Completion time: 2023-07-14T18:51:54.1735855+05:30

Msg 547, Level 16, State 0, Line 28
The ALTER TABLE statement conflicted with the CHECK constraint "CK_check_constraint_for_id".

Query completed with errors.

Stop sharing Hide

L-UNH6223EP\SQL_NEWINSTANCE - LAPTOP-UH6223EP\PG (3) | Constraints | 00:00:30 | 0 rows

Praveen G 00:12:05

Manjusha S Aishwarya esh... SHALU

Vijayalakshmi E Praveen G

01:27:19

SQLQuery4.sql - LAPTOP-UN6223EP\SQL_NEWINSTANCE.Constraints (LAPTOP-UN6223EP\PG (3)) - Microsoft SQL Server Management Studio

```

ALTER TABLE dept_check_const
ADD CONSTRAINT CK_check_constraint_for_id
CHECK (id > 100);

ALTER TABLE dept_check_const
DROP CONSTRAINT CK_check_constraint_for_id;

ALTER TABLE dept_check_const
DROP CONSTRAINT CK_check_constraint_for_id;

```

Results

Name	Owner	Type	Created_datetime						
dept_check_const	dbo	user table	2023-07-14 18:45:05.933						
1									
Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1 id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2 name	varchar	no	100			yes	no	yes	SQL_Latin1_General_CI_AS
3 salary	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
	Identity	Seed	Increment			Not For Replication			
1	No identity column defined.	NULL	NULL			NULL			

Query executed successfully.

LAPTOP-UN6223EP\SQL_NEWINSTANCE Constraints 00:00:01 8 rows

Praveen G 00:16:06 01:23:18

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

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows a connection to 'LAPTOP-UN6Q23EP\SQL_NEWINSTANCE'. The central pane displays the following T-SQL code:

```
1 -- DEFAULT
2
3
4 CREATE TABLE emp_table_default (
5     id int not null,
6     age int,
7     [address] varchar(100) DEFAULT 'TN'
8 );
9
10 --DROP TABLE emp_table_default;
11
12 INSERT INTO emp_table_default (id, age) VALUES
13 (1, 18)
14
```

Below the code, the message 'Commands completed successfully.' is displayed. The status bar at the bottom shows 'Completion time: 2023-07-14T19:04:45.9107812+05:30'.

SQL Server Management Studio

Object Explorer

SQLQuery5.sql - LAPTOP-UN6Q23EP\SQL_NEWINSTANCE Constraints (LAPTOP-UN6Q23EP\PG (89)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

SQLQuery5.sql - L-UN6Q23EP\PG (89)* SQLQuery4.sql - L-UN6Q23EP\PG (52)* SQLQuery3.sql - not connected* SQLQuery2.sql - not connected* SQLQuery1.sql - not connected*

Quick Launch (Ctrl+Q)

Constraints

26 -- DML

27 --ALTER & DROP

28

29 ALTER TABLE emp_table_default

30 ADD CONSTRAINT DF_age_18

31 DEFAULT 18 FOR age

32

I 33 ALTER TABLE emp_table_default

34 ALTER COLUMN age

35 DROP CONSTRAINT

Results

	id	age	address
1	1	18	TN
2	2	25	KL
3	3	25	BNG
4	101	18	TN

Query executed successfully.

Stop sharing Hide

LAPTOP-UN6Q23EP\SQL_NEWINSTANCE - LAPTOP-UN6Q23EP\PG (89) Constraints 00:00:30 | 4 rows

CT_Basics

00:30:49 01:08:35

Praveen G

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

SQL Server Management Studio

Object Explorer

SQLQuery1.sql - LAPTOP-UH6Q23EP\SQL_NEWINSTANCE Constraints (LAPTOP-UH6Q23EP\SQL (89)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Constraints

SQLQuery3.sql - L-UH6Q23EP\SQL_NEWINSTANCE (SQL Server 16.0.105) - Constraints

SQLQuery4.sql - L-UH6Q23EP\SQL_NEWINSTANCE (SQL Server 16.0.105) - Constraints

SQLQuery1.sql - not connected

SQLQuery2.sql - not connected

SQLQuery1.sql - not connected

26 -- DML

27 --ALTER & DROP

28

29 ALTER TABLE emp_table_default

30 ADD CONSTRAINT DF_age_18

31 DEFAULT 18 FOR age

32

33 ALTER TABLE emp_table_default

34 DROP CONSTRAINT DF_age_18;

132 %

Messages

Commands completed successfully.

Completion time: 2023-07-14T19:11:25.0311319+05:30

Query executed successfully.

Stop sharing Hide

LAPTOP-UH6Q23EP\SQL_NEWINSTANCE - LAPTOP-UH6Q23EP\SQL (89) - Constraints 00:00:30 0 news

00:31:26

01:07:58

Praveen G

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

```
SQLQuery1 - LAPTOP-UNH623EF\SQL_NEWINSTANCE Constraints (LAPTOP-UNH623EF\PG (89)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Object Explorer
Connect + Constraints
SQLQuery1 - L-UNH623EF\PG (89)* SQLQuery2 - L-UNH623EF\PG (57)* SQLQuery3 - not connected* SQLQuery4 - not connected*
SQLQuery1.sql - not connected*
26 --DML
27 --ALTER & DROP
28
29 ALTER TABLE emp_table_default
30 ADD CONSTRAINT DF_age_18
31 DEFAULT 18 FOR age
32
33 ALTER TABLE emp_table_default
34 DROP CONSTRAINT DF_age_18;
35
36 ALTER TABLE emp_table_default
37 DROP CONSTRAINT [DF_emp_table_address_5535A963]

Commands completed successfully.

Completion time: 2023-07-14T19:11:53.6338676+05:30
```

Video-20230714_145012-Meeting Recording (1)

00:31:49 01:07:35

----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 maintains 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, separator)

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


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

--SUBSTRING
--- SUBSTRING(input_string, start, length)

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() - Absolute - 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 employed 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 employed 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
```

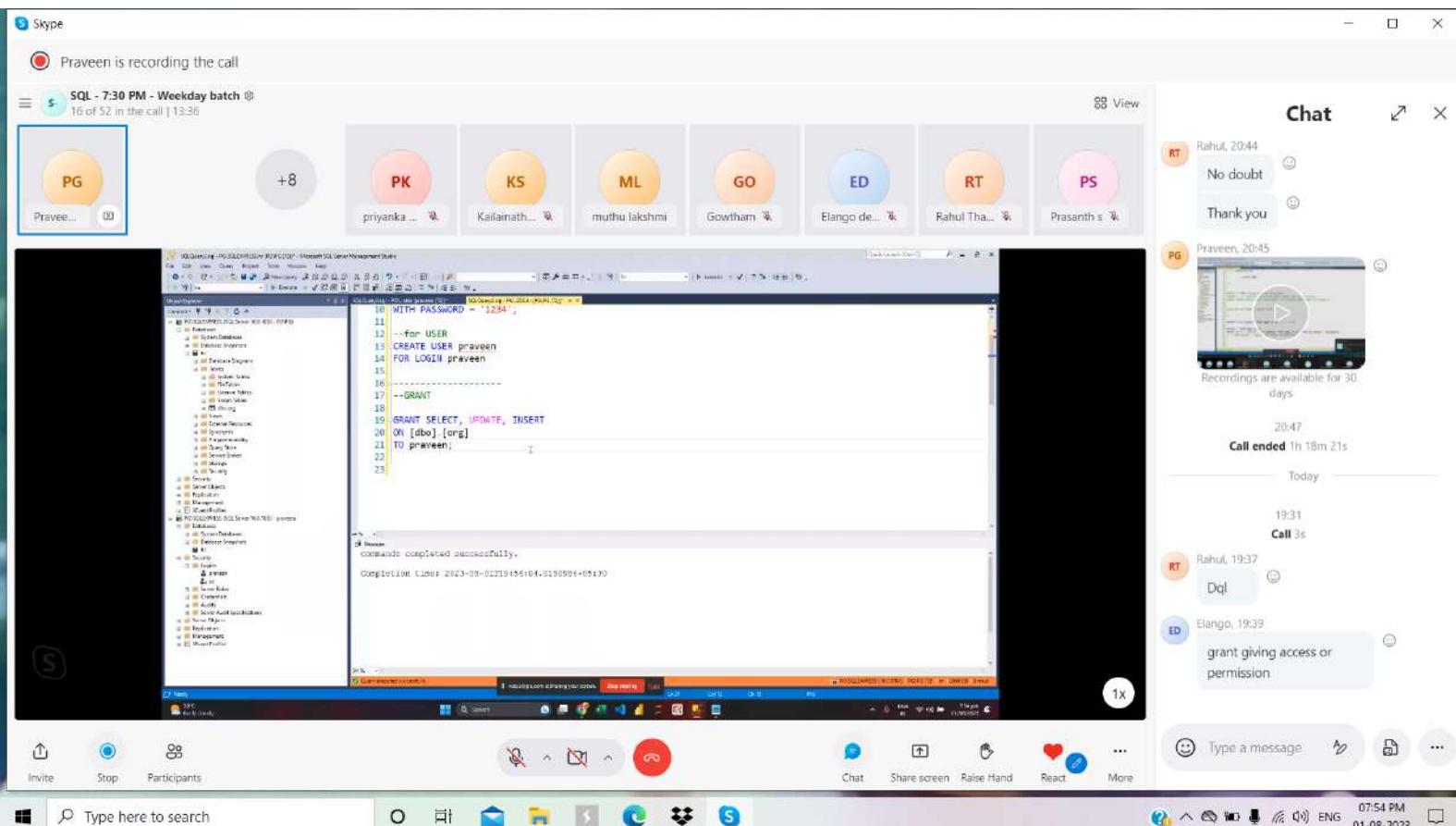
--privilege_names:

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

14 of 52 in the call | 27:34

PG PK KS ML GO ED RT PS

Chat

Rahul, 20:44
No doubt

Thank you

Praveen, 20:45

Recordings are available for 30 days

Call ended 1h 18m 21s

Today

20:47 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 Read More

0 Type here to search 08:02 PM 01-08-2023

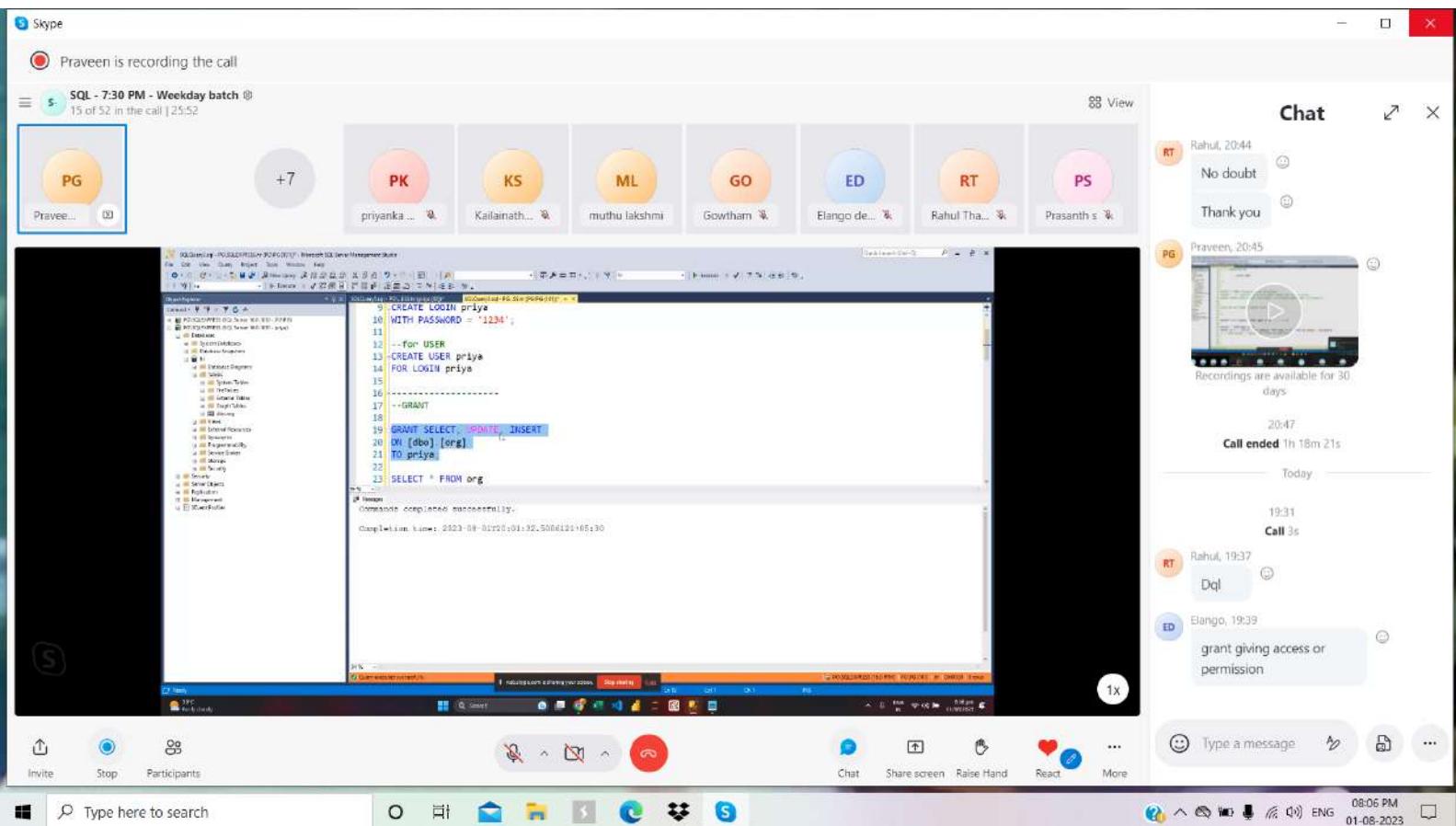
The screenshot shows a Skype video conference with seven participants: PG, PK, KS, ML, GO, ED, and RT. The participant PG is currently recording the call. A Microsoft SQL Server Management Studio (SSMS) window is open in the foreground, showing a query window with the following code:

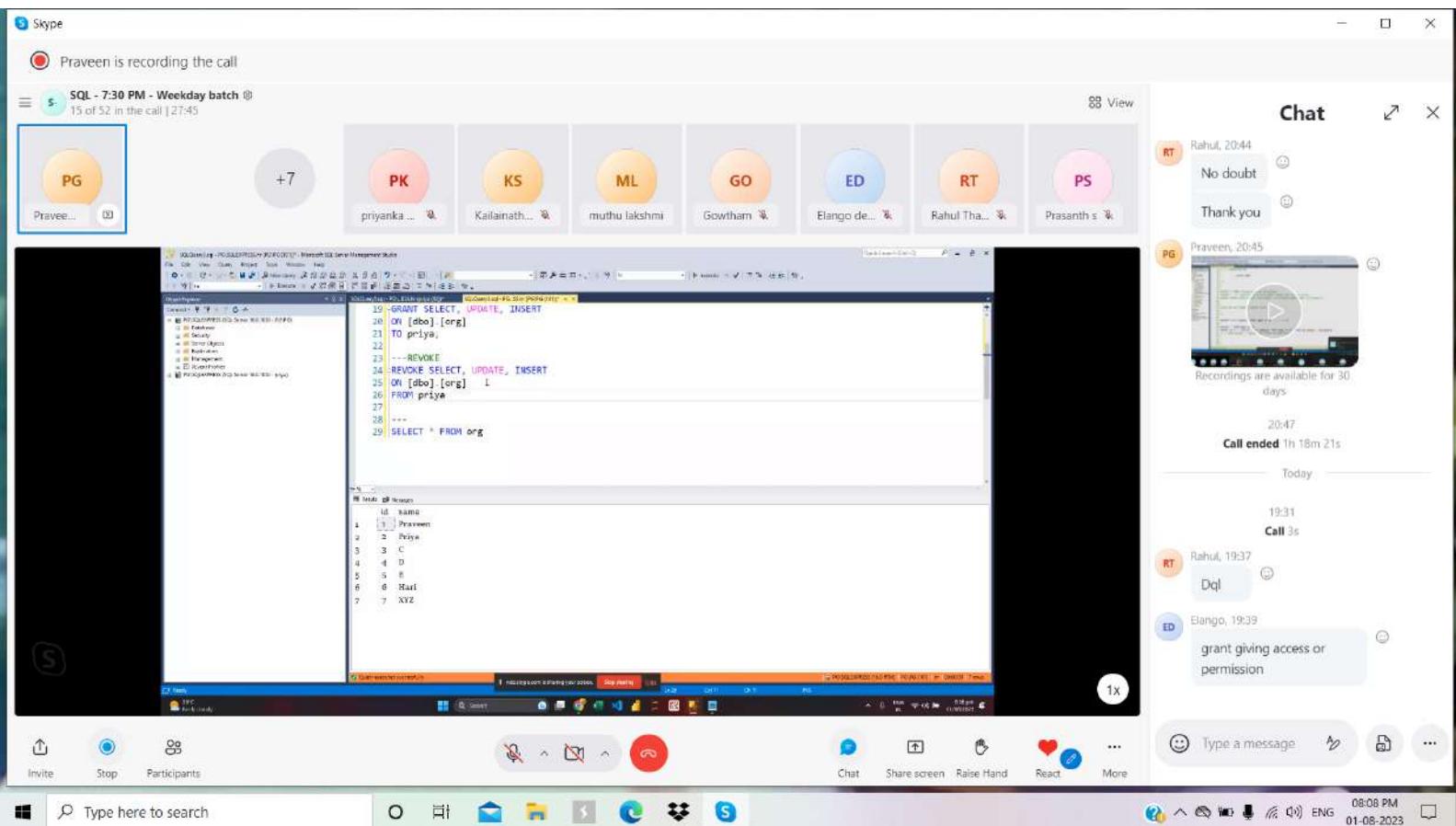
```
1 USE hr;
2 SELECT * FROM org;
```

Below the query window, a results grid displays the following data:

ID	Name
1	Praveen
2	
3	C
4	D
5	E
6	Razi

The system tray at the bottom of the screen shows various icons, and the taskbar includes icons for File Explorer, Mail, Edge, and File Explorer.





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 etc.  
-- dbo 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

```
1 create table random_tab (id int, val decimal);
2
3 insert into random_tab
4 select 1, random() from generate_series(1, 10000000);
5
6 insert into random_tab
7 select 2, random() from generate_series(1, 10000000);
8
9 select count(1) from random_tab;
```

Data Output Explain Messages Notifications

count	bigint
1	2000000



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

```
6  insert into random_tab
7  select 2, random() from generate_series(1, 10000000);
8
9  select id, avg(val), count(*)
10 from random_tab
11 group by id;
12
13 create materialized view mv_random_tab
14 as
15 select id, avg(val), count(*)
16 from random_tab
17 group by id;
18
19
20
21
22
```

Data Output Explain Messages Notifications

SELECT 2

Query returned successfully in 3 secs 165 msec.



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

```
9  select id, avg(val), count(*)
10 from random_tab
11 group by id;
12
13 create materialized view mv_random_tab
14 as
15 select id, avg(val), count(*)
16 from random_tab
17 group by id;
18
19 select * from mv_random_tab;
20
21 delete from random_tab where id = 1;
22
23 refresh materialized view mv_random_tab;
24
25
```

Data Output Explain Messages Notifications

	id integer	avg numeric	count bigint
1	2	0.49986896357199522209930	10000000

Materialized View in SQL | Faster SQL Queries using Materialized Views

```
9  select id, avg(val), count(*)
10 from random_tab
11 group by id;
12
13 create materialized view mv_random_tab
14 as
15 select id, avg(val), count(*)
16 from random_tab
17 group by id;
18
19 select * from mv_random_tab;
20
21 delete from random_tab where id = 1;
22
23 refresh materialized view mv_random_tab;
24
25
```

Data Output Explain Messages Notifications

REFRESH MATERIALIZED VIEW

Query returned successfully in 4 secs 578 msec.



11:33 / 16:39 • Refresh Materialized View >



The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The main window displays a T-SQL script in the Query Editor:

```
1 SELECT * FROM customer_data;
2
3 ---DROP VIEWS
4 DROP VIEW [dbo].[customers_india_more22]
5
6 ---RENAME VIEWS
7 EXEC sp_rename 'old_viewname', 'new_viewName'
8
9
10 ---DML --- INSERT, UPDATE, DELETE
11 ----INSERT
12
13
14
15
16
17 ----
18 ----
```

The Object Explorer on the left shows the database structure for 'PG_SQL_NEWINSTANCE'. The status bar at the bottom indicates the query was executed successfully.

The taskbar at the bottom of the screen shows several pinned user icons, including Praveen G, Nat..., Mut..., muthu lakshmi, SHALU, jayarsi M, NarasimhaRed..., Bala, Guest user, and Praveen G again. The system tray shows the date as 04/12/2023 and the time as 00:11:49.

SQLQuery2.sql - PG_SQL_NEWINSTANCE.m4 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

PG_SQL_NEWINSTANCE (SQL Server 16.0.1050.5)

Databases

Tables

Views

System Views

External Tables

FileTables

Graph Tables

dbo.customers

dbo.employee

Dropped Ledger Tables

Synonyms

Programmability

Query Store

Service Broker

Storage

Security

Server Objects

Replication

Always On High Availability

Management

Integration Services Catalogs

SQL Server Agent (Agent XPs disabled)

XEvent Profiler

SQLQuery2.sql - PG_C.E.Hr (PG (54))

Execute

Quick Launch (Ctrl+Q)

1 SELECT * FROM [dbo].[customers]; -- Basic table
2 SELECT * FROM customer_data; -- virtual table
3
4 ---DROP VIEWS
5 DROP VIEW [dbo].[customers_india_more22]
6
7 ---RENAME VIEWS
8 EXEC sp_rename 'old_viewname', 'new_viewName'
9
10 ---DML --- INSERT, UPDATE, DELETE
11 ---INSERT
12
13

	id	name	age	address	salary
1	1	A	21	India	15000.00
2	2	B	22	USA	25000.00
3	3	C	23	India	30000.00
4	4	D	24	Africa	45000.00
5	5	E	25	Canada	18000.00
6	6	F	26	America	45000.00

Query executed successfully.

PG_SQL_NEWINSTANCE (16.0.1050) PG (54) hr 00:00:30 6 rows

Praveen G Ready 00:28:04

Natty elcny

muthu lakshmi

SHALU

jayasri M

NarasimhaRed...

Bala

Guest user

Praveen G

00:09:48

SQLQuery2.sql - PG_SQL_NEWINSTANCE (PG\PG (S4)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect to... New Query New Object Explorer Group Quick Launch (Ctrl+Q)

SQLQuery2.sql - PG\CE.HR (PG\PG (S4)) * X

```
11: ---DML --- INSERT, UPDATE, DELETE
12: ----INSERT
13: INSERT INTO customer_data VALUES (6, 'F', 26, 'America', 45000);
14:
15: ----UPDATE
16: UPDATE customer_data
17: SET
18:
19: ----
```

Results Messages

	id	name	age	address	salary
1	1	A	21	India	15000.00
2	2	B	22	USA	25000.00
3	3	C	23	India	30000.00
4	4	D	24	Africa	45000.00
5	5	E	25	Canada	18000.00
6	6	F	26	America	45000.00

Query executed successfully.

PG\SQL_NEWINSTANCE (16.0.1050.5) PG\PG (S4) hr 00:00:30 6 rows

Praveen G Ready 00:29:48

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

SQLQuery2.sql - PG_SQL_NEWINSTANCE.m4 - Microsoft SQL Server Management Studio

```
1:SELECT * FROM [dbo].[customers]; -- Basic table
2:SELECT * FROM customer_data; -- virtual table
3:
4:---DROP VIEWS
5:DROP VIEW [dbo].[customers_india_more22]
6:
7:---RENAME VIEWS
8:EXEC sp_rename 'old_viewname', 'new_viewName'
9:
10:---DML --- INSERT, UPDATE, DELETE
11:---INSERT
12:INSERT INTO customer_data VALUES (6, 'F', 26, 'America', 45000);
13:
14:---UPDATE
15:UPDATE customer_data
16:SET age = 40
17:WHERE id = 4;
18:
19:---DELETE
20:DELETE FROM customer_data
21:WHERE id = 4
```

Query executed successfully.

PG_SQL_NEWINSTANCE (160 MB) | PG (54) | hr | 00:06:30 | 3 rows

Praveen G Ready 00:32:59

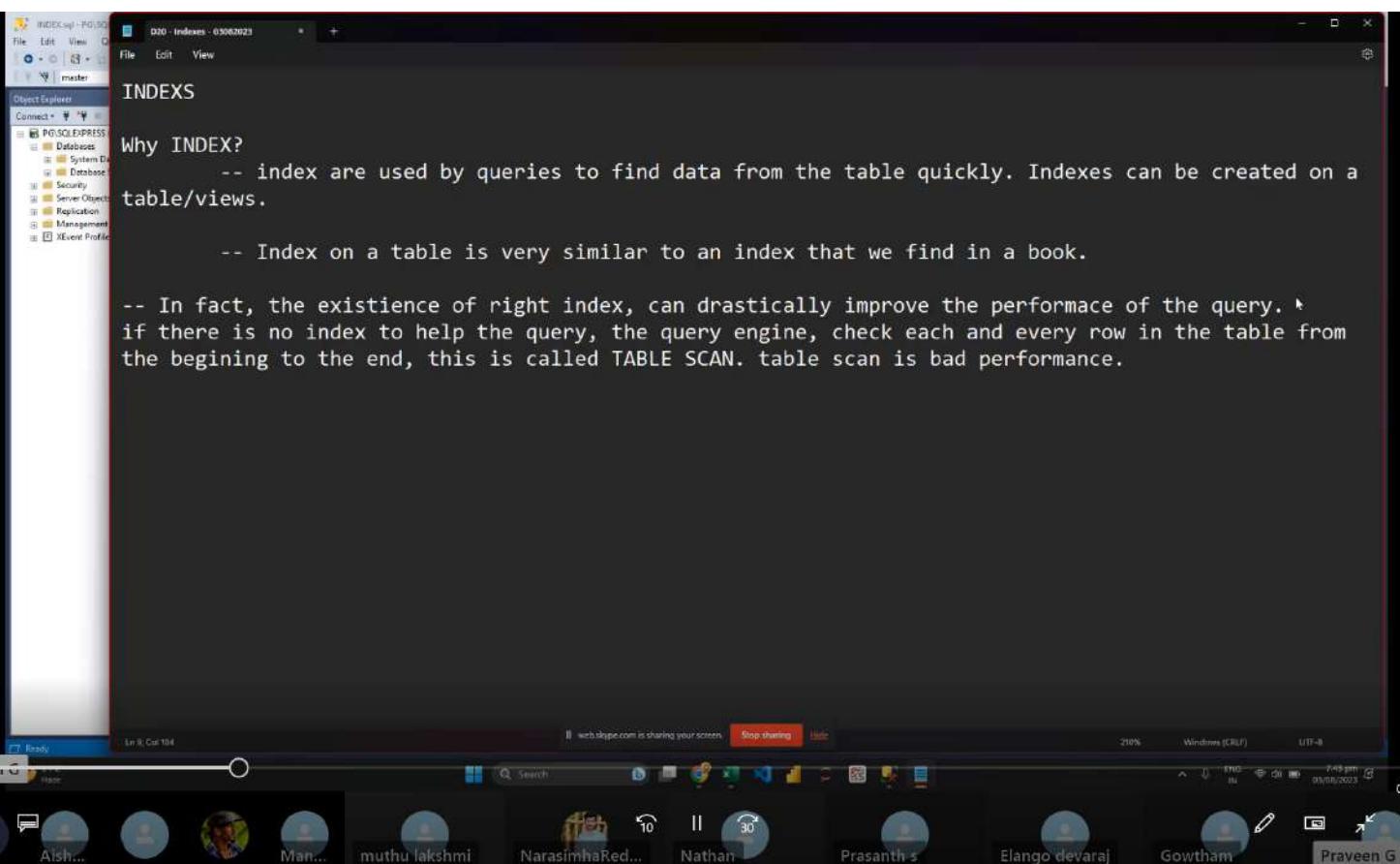
Nat... Mut... muthu lakshmi SHALU jayarsi M NarasimhaRed... Bala Guest user Praveen G

```
INDEXES

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 existence of right index, can drastically improve the performance 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 beginning to the end, this is called TABLE SCAN. table scan is bad performance.
```



INDEXES.sql - P01\SQLEXPRESS\hr (P01\PG (6)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect + P01\SQLEXPRESS (SQL Server 16.0.1050 - P01\PG)

Databases

System Databases

Database Snapshots

hr

Database Diagrams

Tables

Views

External Resources

Synonyms

Programmability

Query Store

Service Broker

Storage

Security

Server Objects

Replication

Management

XEvent Profiler

SQLQuery2019 - P01\hr (P01\PG (6))

```
1 --current/active database
2 CREATE DATABASE hr;
3 USE hr;
4
5 --employee table
6 CREATE TABLE tbmployee
7     id int,
8     Name varchar(50),
9     salary money,
10    gender varchar(10);
11
12
13 --Records for tbmployee table
14 INSERT INTO tbmployee VALUES (1, 'Sam', 2500, 'Male');
15 INSERT INTO tbmployee VALUES (2, 'Pam', 6500, 'Female');
16 INSERT INTO tbmployee VALUES (3, 'John', 4500, 'Male');
17 INSERT INTO tbmployee VALUES (4, 'Linda', 5500, 'Female');
```

85%

Messages

Commands completed successfully.

Completion time: 2023-08-03T19:44:36.7506643+05:30

Query executed successfully.

Stop sharing Hide

CTB Beta

00:11:08 00:39:08

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

INDEX.sql - PG SQL EXPRESS (lr (PG:PG (6))) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect to... Connect to... C +

PQ SQL EXPRESS (SQL Server 16.0.1050 - PG:PG)

Databases System Databases Database Snapshots

Tables

Views

External Resources

Syonyms

Programmability

Query Store

Service Broker

Storage

Security

Server Objects

Replication

Management

XEvent Profiler

INDEX.sql - PG SQL... (PG:PG (6)) = X SQLQueryLog - PG_ESS.lr (PG:PG (9))

```
6 CREATE TABLE tbemployee (
7     id int,
8     Name varchar(50),
9     salary money,
10    gender varchar(10)
11 );
12
13 --Records for tbemployee table
14 INSERT INTO tbemployee VALUES (1, 'Sam', 2500, 'Male');
15 INSERT INTO tbemployee VALUES (2, 'Pam', 6500, 'Female');
16 INSERT INTO tbemployee VALUES (3, 'John', 4500, 'Male');
17 INSERT INTO tbemployee VALUES (4, 'Sara', 5500, 'Female');
18 INSERT INTO tbemployee VALUES (5, 'Todd', 3100, 'Male');
19
20
21 SELECT * FROM tbemployee;
```

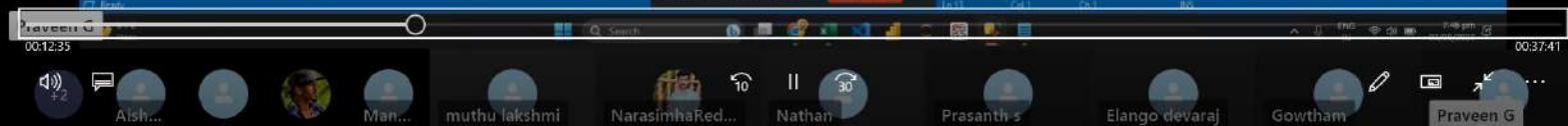
Results Messages

	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.

Stop sharing Hide

00:12:35 00:37:41



INDEX.sql - PG:SQL EXPRESS Jr (PG:PG (6)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect: PG:SQL EXPRESS Jr (SQL Server 16.0.1050 - PG:PG)

Databases
System Databases
Database Snapshots
Logins
Tables
Views
External Resources
Synonyms
Programmability
Query Store
Service Broker
Storage
Security
Server Objects
Replication
Management
XEvent Profiler

Quick Launch (Ctrl+Q)

INDEX.sql - PG:SQL EXPRESS Jr (PG:PG (6)) = X SQLQueryLog - PG:ESS.JR (PG:PG (5))

```
13 --Records for tblemployee table
14 INSERT INTO tblemployee VALUES (1, 'Sam', 2500, 'Male');
15 INSERT INTO tblemployee VALUES (2, 'Pam', 6500, 'Female');
16 INSERT INTO tblemployee VALUES (3, 'John', 4500, 'Male');
17 INSERT INTO tblemployee VALUES (4, 'Sara', 5500, 'Female');
18 INSERT INTO tblemployee VALUES (5, 'Todd', 3100, 'Male');
19
20
21 SELECT * FROM tblemployee;
22
23 --performing TABLE SCAN -- (without/before creating index)
24 --fetch employees whose salary in more than 5000 and less than 7000
25
26 SELECT * FROM tblemployee
27 WHERE salary > 5000 and salary < 7000;
28
```

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.

00:13:10 00:37:06

Praveen G Aish... Muthu Lakshmi Narasimha Red... Nathan Prasanth S Elango Devaraj Gowtham Praveen G

```
INDEXES

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 existence of right index, can drastically improve the performance 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.

SYNTAX :

CREATE INDEX <index_name>
ON <Table_name> (Col_name [ASC/DESC])
```

<



INDEX.sql - PG:SQL EXPRESS (hr) (PG:PG (6)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connected to: PG:SQL EXPRESS (hr) (PG:PG (6))

Databases

System Databases

File

Tables

System Tables

FileTables

External Tables

Graph Tables

Tables

Columns

Keys

Constraints

Triggers

Indexes

Statistics

Views

External Resources

Synonyms

Programmability

Config Store

Service Broker

Storage

Security

Server Objects

Replication

Management

XEvent Profiler

INDEX.sql - PG:SQL EXPRESS (hr) (PG:PG (6)) = < SQLQueryLog - PG:ESS (hr) (PG:PG (5))

```
32 ---CREATE INDEX
33
34
35 CREATE INDEX IX_tblemployee_salary
36 ON tblemployee (salary ASC);
37
38 --CREATE INDEX IX_tblemployee_Name
39 --ON tblemployee (Name ASC);
40
41 ---to Verify index
```

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

	id	Name	salary	gender
1	2	Pam	6500.00	Female
2	4	Sara	5500.00	Female

Query executed successfully.

Stop sharing Hide

Praveen G 00:21:23 00:28:53

INDEX.sql - PG(SQL EXPRESS).lr (PG|PG (6)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect to... C +

PQ(SQL EXPRESS) (SQL Server 15.0.1050 - PG|PG)

- Databases
 - System Databases
 - Database Snapshots
- IAs
 - File Tables
 - External Tables
 - Graph Tables
- Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
- Views
- External Resources
- Synonyms
- Partitioned
- Query Store
- Service Broker
- Storage
- Security
- Server Objects
- Replication
- Management
- XEvent Profiler

INDEX.sql - PG(SQL-EXPRESS).lr [PG|PG (6)]* - X SQL Query3.sql - PG_ESS.lr (PG|PG (9))

```
24 --fetch employees whose salary is more than 5000 and less than 7000
25
26 SELECT * FROM tbemployee
27 WHERE salary > 5000 and salary < 7000;
28
29 --in this case, the SEARCH ENGINE is performing TABLE SCAN for each record in salary column and giving the output.
30 --Because, there is no index created for salary column and salary column is not sorted properly therefore it is very difficult for the sql server to fetch the desired values.
31
32 -----
33 ---CREATE INDEX
34 I
35
36 CREATE INDEX IX_tbemployee_salary
37 ON tbemployee (salary ASC);
38
39 ---CREATE INDEX IX_tbemployee_Name
40 --ON tbemployee (Name ASC);
41
42 ---to Verify index
43 SP_HELPINDEX tbemployee;
44
45
46 ---DROP INDEX
47 DROP INDEX tbemployee.IX_tbemployee_salary;
48 ---DROP INDEX tbemployee.IX_tbemployee_Name;
49
50
51
52
53
```

Query executed successfully

www.deupe.com is sharing your screen. Stop sharing

10 30

00:23:54 00:26:22

Video-20230803_145417-Meeting Recording

Jaga...



gue...



muthu lakshmi

Narasimha Red...

Nathan

Manjusha S

Elango devaraj

Gowtham

Praveen G

D20 - Indexes - 03082023

File Edit View

SYNTAX :

```
CREATE INDEX <index_name>
ON <Table_name> (Col_name [ASC/DESC])
```

```
--to Verify index
SP_HELPINDEX tbmployee;
```

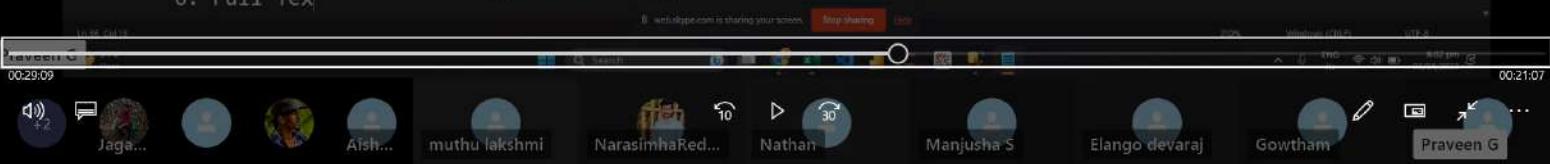
```
--DROP INDEX
--- SYNTAX -- DROP INDEX <TABLE_NAME>.<INDEX_NAME>
```

What is CLUSTERED & NONCLUSTERED INDEX ?

TYPES of INDEX:

1. Clustered
2. Non Clustered
3. Unique
4. Filtered

Video-20230803_145417-Meeting Recording



D2D - Indexes - 03062023

File Edit View

What is CLUSTERED & NONCLUSTERED INDEX ?

TYPES of INDEX:

1. Clustered
2. Non Clustered
3. Unique
4. Filtered
5. XML
6. Full Text
7. Spatial
8. ColumnStore
9. Index with included Columns
10. Index on computed columns

I

Line 6 Col 31

|| web.skype.com is sharing your screen.

Stop sharing Hide

30%

Windows (0.0.0)

LITE 3



Jagade...



Aish...



muthu lakshmi



Narasimha Red...



Nathan



Manjusha S



Elango devaraj



Gowtham



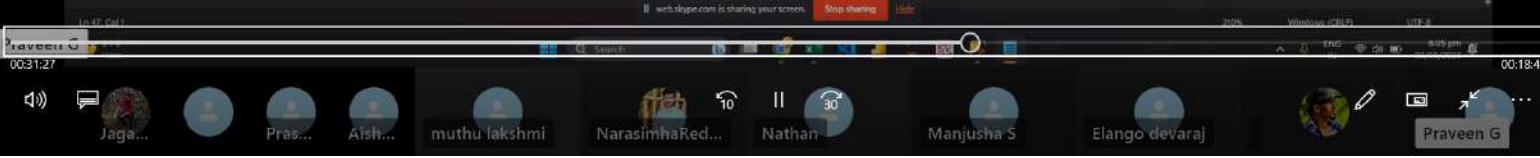
Praveen G

```
D2D - Indexes - 03062023
File Edit View
-----
What is CLUSTERED & NONCLUSTERED INDEX ?

TYPES of INDEX:
1. Clustered
2. Non Clustered
3. Unique
4. Filtered
5. XML
6. Full Text
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,

2. Non Clustered
```



INDEX.sql - PG SQL EXPRESS (PQ|PG (0)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

PG SQL EXPRESS (SQL Server 16.0.1050 - PQ|PG)

Databases
System Databases
Database Snapshots
Tables
Database Diagrams
System Tables
FileTables
External Tables
Graph Tables
tbl_employee
Columns
Keys
Constraints
Triggers
Indexes
Statistics
Views
External Resources
Synonyms
Programmability
Common Store
Service Broker
Sessions
Security
Server Objects
Replication
Management
XEvent Profiler

INDEX.sql - PG SQL EXPRESS (PQ|PG (0)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Quick Launch (Ctrl+C)

49 --DROP INDEX tblemployee.IX_tblemployee_Name;

50

51

52

53

54

55

56 -----

57 --CLUSTERED INDEX vs NON-CLUSTERED INDEX

58

59 --Create a table with primary key

60

61 --employee1 table

62 CREATE TABLE tbl_employee1 (

63 id int primary key, ---- PK constraint is created along with the table

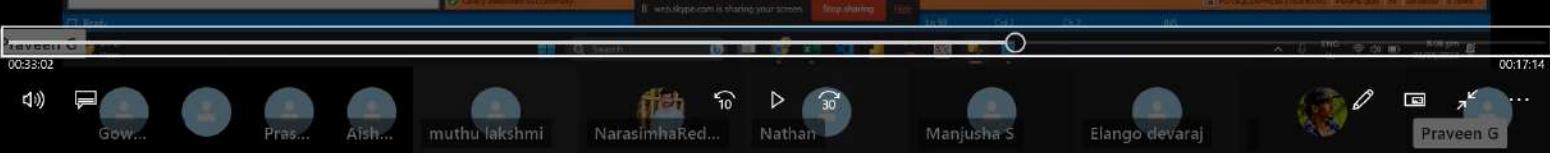
64 Name varchar(50),

65 salary money,

66 gender varchar(10),

Completion time: 2023-08-03T20:01:06.9586194+05:30.

Video-20230803_145417-Meeting Recording



INDEXES.sql - PG:SQL EXPRESS (lr (PQ|PG (6))) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect: PG:SQL EXPRESS (lr (PQ|PG (6)))

Databases

System Databases

File Tables

Tables

System Tables

FileTables

External Tables

Graph Tables

sys.dbo.tbl_employee1

Columns

Keys

Constraints

Triggers

Indexes

Statistics

Views

External Resources

Synonyms

Partitioned Tables

Query Store

Service Broker

Storage

Security

Server Objects

Replication

Management

XEvent Profiler

INdexes.sql - PG:SQL EXPRESS (lr (PQ|PG (6)))

--Create a table with primary key

```
59 --employee1 table
60 CREATE TABLE tbl_employee1 (
61     id int primary key,
62     Name varchar(50),
63     salary money,
64     gender varchar(10),
65 );
66
67
68 --Records for tbl_employee1 table
69
70 INSERT INTO tbl_employee1 VALUES (5, 'Todd', 3100, 'Male');
71 INSERT INTO tbl_employee1 VALUES (2, 'Pam', 6500, 'Female');
72 INSERT INTO tbl_employee1 VALUES (1, 'Sam', 2500, 'Male');
73 INSERT INTO tbl_employee1 VALUES (4, 'Sara', 5500, 'Female');
74 INSERT INTO tbl_employee1 VALUES (3, 'John', 4500, 'Male');
75
76
```

(1 row affected)

Completion time: 2023-08-03T20:07:54.0513345+05:30

85%

Query executed successfully.

Stop sharing Hide

Praveen G 00:34:15

Gow... Pras... Aish... muthu lakshmi NarasimhaRed... Nathan Manjusha S Elango devaraj Praveen G 00:16:01

INDEX.sql - PG:SQL EXPRESS Jr (PG:PG (6)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect to... ? C +

PQ:SQL EXPRESS (SQL Server 16.0.1050 - PG:PG)

Databases System Databases Database Snapshots

Tables

System Tables

FileTables

External Tables

Graph Tables

sys.dbo.tblemployee1

Columns Keys Constraints Triggers Indexes Statistics

Views External Resources Synonyms Partitioned Tables Query Store Service Broker Storage Security

Server Objects Replication Management XEvent Profiler

INdex.sql - PG:SQL-ESS Jr (PG:PG (6))

64 Name varchar(50),
65 salary money,
66 gender varchar(10),
67);
68
69 --Records for tbl_employee1 table
70
71 INSERT INTO tbl_employee1 VALUES (5, 'Todd', 3100, 'Male');
72 INSERT INTO tbl_employee1 VALUES (2, 'Pam', 6500, 'Female');
73 INSERT INTO tbl_employee1 VALUES (1, 'Sam', 2500, 'Male');
74 INSERT INTO tbl_employee1 VALUES (4, 'Sara', 5500, 'Female');
75 INSERT INTO tbl_employee1 VALUES (3, 'John', 4500, 'Male');
76
77 --DQL satatement
78 --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)
79 SELECT * FROM tbl_employee1
80

Results

	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.

Stop sharing Hide

00:34:19 00:15:57

INDEX.sql - PG:SQL EXPRESS (r) (PG (6)) - Microsoft SQL Server Management Studio

File Edit View Project Tools Window Help

Object Explorer

Connect + Connect to database

PG:SQL EXPRESS (SQL Server 15.0.1050 - P0(PG))

Databases

System Databases

Master

tempdb

msdb

model

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

systbl_employee1

Columns

Keys

Constraints

Triggers

Indexes

PK_tbl_employee1 (Clustered)

Statistics

dbo.tb... (1 row(s) affected)

Views

External Resources

Aliases

Programmability

Catalogs

Service Broker

Storage

Security

Security

Server Objects

Replication

Management

XEvent Profiler

SQL Query3.sql - PG_EIS (r) (PG (6))

```
74 INSERT INTO tbl_employee1 VALUES (4, 'Sara', 5500, 'Female');
75 INSERT INTO tbl_employee1 VALUES (3, 'John', 4500, 'Male');
76
77 --DQL satatement
78 --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)
79 SELECT * FROM tbl_employee1
80
81
82 --to verify index name
83 EXECUTE sp_helpindex tbl_employee1           --- output : index keys --> id
84                                         --- because primary key is created
85
86 --lets create a CLUSTERED INDEX
87 CREATE CLUSTERED INDEX IX_tbl_employee1_Gender_Salary
88 ON tbl_employee1 (Gender DESC, Salary ASC);
89
90
91 --output : will throw an error
```

Results

	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.

Stop sharing Hide

CT_Basics

00:35:47

00:14:29

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'POSQL EXPRESS (SQL Server 16.0.1090 - PG/P3)'. The query editor window on the right contains a script named 'INDEX.sql' with the following content:

```
--Records for tbl_employee1 table
--DQL statement
--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);

--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.PK_tbl_empl_3213E83FBE25C9E;

--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
```

What is CLUSTERED & NONCLUSTERED INDEX ?

TYPES of INDEX:

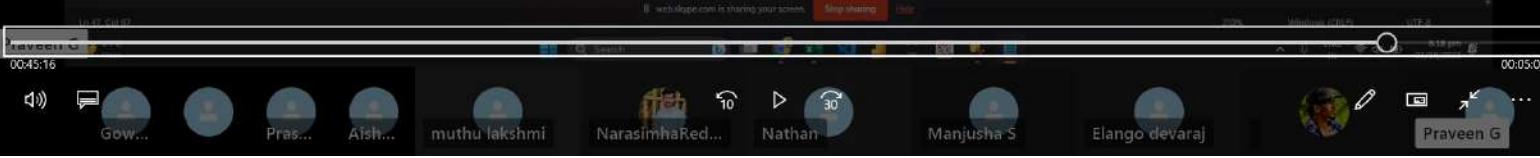
1. Clustered
2. Non Clustered
3. Unique
4. Filtered
5. XML
6. Full Text
7. Spatial
8. ColumnStore
9. Index with included Columns
10. Index on computed columns

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)

Video-20230803_145417-Meeting Recording



INDEX.sql - PO|SQL|EXPRESS.ln (PO|PG (6)) - Microsoft SQL Server Management Studio

File Edit View Project Tools Window Help

Quick Launch (Ctrl+Q)

Object Explorer

Connect + SQL Server 2019 (LocalDB) | New Query | Open Recent | Run | Stop | Refresh | Esc | Execute | Back | Forward | Home | Stop Sharing

INDEX.sql - PO|SQL|EXPRESS.ln (PO|PG (6)) * SQLQuery3.sql - PG_ESS.ln (PO|PG (9))

```
30 --Because, there is no index created for salary column and salary column is not sorted properly therefore it is very difficult for the sql server to fetch the desired values.
```

```
31
32
33 ---CREATE INDEX
34
35
36 CREATE INDEX IX_tbemployee_salary
37 ON tbemployee (salary ASC);
38
39 --CREATE INDEX IX_tbemployee_Name
40 --ON tbemployee (Name ASC);
41
42 --to Verify index
43 SP_HELPINDEX tbemployee;
44
45
46 ---DROP INDEX
47 --- SYNTAX -- DROP INDEX <TABLE_NAME>.<INDEX_NAME>
```

85%  Messages
Commands completed successfully.

Completion time: 2023-08-03T20:19:35.9849400+05:30

Video-20230803_145417-Meeting Recording

INDEXES.sql - PG\SQL EXPRESS (hr [PG\PG (6)]) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connected to: PG\SQL EXPRESS (SQL Server 15.0.1050 - PG\PG)

Databases

- System Databases
- Database Snapshots
- Tables
- System Tables
- FileTables
- External Tables
- Graph Tables
- Tables
- System Tables
- FileTables
- External Tables
- Graph Tables
- Triggers
- Indexes
- IX_tblemployee_salary (Non-Unique, Non-Clustered)
- Statistics

Views

- External Resources
- Synonyms
- Programmability
- Query Store
- Service Broker
- Storage
- Security
- Server Objects
- Replication
- Management
- Server Profiler

Script

30 --Because, there is no index created for salary column and salary column is not sorted properly therefore it is very difficult for the sql server to fetcch the desired values.

31

32

33 ---CREATE INDEX

34

35

36 CREATE INDEX IX_tblemployee_salary

37 ON tblemployee (salary ASC);

38

39 CREATE INDEX IX_tblemployee_Name

40 ON tblemployee (Name ASC);

41

42 ---to Verify index

43 SP_HELPINDEX tblemployee;

44

45

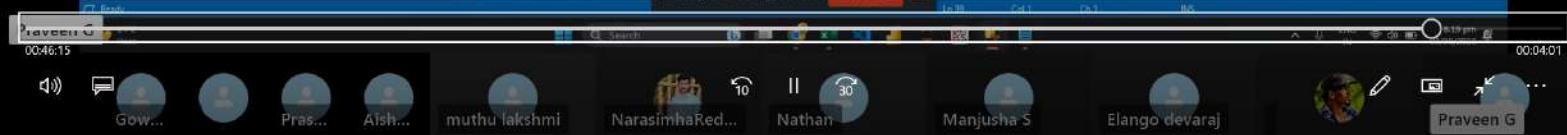
46 ---DROP INDEX

47 --- SYNTAX -- DROP INDEX <TABLE_NAME>.<INDEX_NAME>

Messages

Commands completed successfully.

Completion time: 2023-08-03T20:19:35.9849400+05:30



```
7. Spatial  
8. ColumnStore  
9. Index with included Columns  
10. Index on computed columns
```

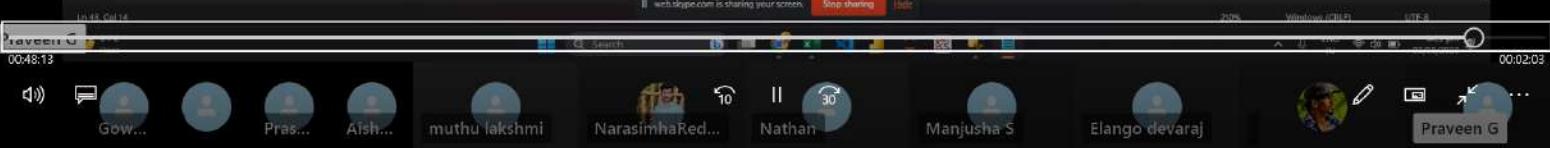
```
1. Clustered
```

```
-- 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 seperately from the actual data, table can have more than one  
clustered index.
```



```
*Untitled - Notepad
File Edit Format View Help
CREATE DATABASE hr;
USE hr;

--employee table
CREATE TABLE tbmployee (
    id int,
    Name varchar(50),
    salary money,
    gender varchar(10)
);
--Records for tbmployee table
INSERT INTO tbmployee VALUES (1, 'Sam', 2500, 'Male');
INSERT INTO tbmployee VALUES (2, 'Pam', 6500, 'Female');
INSERT INTO tbmployee VALUES (3, 'John', 4500, 'Male');
INSERT INTO tbmployee VALUES (4, 'Sara', 5500, 'Female');
INSERT INTO tbmployee VALUES (5, 'Todd', 3100, 'Male');

SELECT * FROM tbmployee;

--performing TABLE SCAN -- (without/before creating index)
--fetch employees whose salary in more than 5000 and less than 7000
SELECT * FROM tbmployee
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 difficult for the sql server to fetch the desired values.

----CREATE INDEX

CREATE INDEX IX_tbmployee_salary
ON tbmployee (salary ASC);

CREATE INDEX IX_tbmployee_Name
ON tbmployee (Name ASC);

---to Verify index
SP_HELPINDEX tbmployee;
<
```

Ln 107, Col 28 | 100% | Windows (CRLF) | UTF-8
11:25 AM | 04-08-2023

*Untitled - Notepad
File Edit Format View Help

```
--DROP INDEX
--- SYNTAX -- DROP INDEX <TABLE_NAME>.<INDEX_NAME>
DROP INDEX tbmployee.IX_tbmployee_salary;
--DROP INDEX tbmployee.IX_tbmployee_Name;

-----CLUSTERED INDEX vs NON-CLUSTERED INDEX
--create a table with primary key
--employee1 table
CREATE TABLE tb1_employee1 (
    id int primary key,          ---- PK constraint is created along with the table
    Name varchar(50),
    salary money,
    gender varchar(10),
);
--Records for tb1_employee1 table
INSERT INTO tb1_employee1 VALUES (5, 'Todd', 3100, 'Male');
INSERT INTO tb1_employee1 VALUES (2, 'Pam', 6500, 'Female');
INSERT INTO tb1_employee1 VALUES (1, 'Sam', 2500, 'Male');
INSERT INTO tb1_employee1 VALUES (4, 'Sara', 5500, 'Female');
INSERT INTO tb1_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 tb1_employee1

--to verify index name
EXECUTE sp_helpindex tb1_employee1      --- output : index keys --> id
                                         --- because primary key is created
```

Ln 107, Col 28 100% Windows (CRLF) UTF-8
Type here to search 11:25 AM 04-08-2023 ENG

```
*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 statement
--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 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
11:25 AM 04-08-2023

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.

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 to fetch the rows from the table, rather than scanning each row in the table.

This is called



Next Video

Create Index

3:3 PRAGIM Technologies | www.pragimtech.com | 9900113931

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

5

Indexes in sql server Part 35

File Edit View Query Project Debug Tools Window Community Help

New Query Execute SQL Server 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

Ready 6:59 / 11:13 • Create Index > Ln 4 Col 15 Ch 16 INS

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
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 Synonyms
SQLQuery2.sql - (l...).Sample (sa (56))
Select * from tblEmployee
Create Index IX_tblEmployee_Salary
ON tblEmployee (Salary ASC)
sp_Helppindex tblEmployee
drop index tblEmployee.IX_tblEmployee_Salary
Messages
Command(s) completed successfully.
Pull up for precise seeking
Create Index
9:41
Query executed successfully.
(llocal) (10.0.1600.0) Sample 00:00:00 0 rows
Ready 9:43 / 11:13 • Create Index >
```

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 creates 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	

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

(local) (SQL Server 10.0.1600 - sa)

Databases

- System Databases
- Database Snapshots
- AdventureWorks2008
- ReportServer
- ReportServerTempDB
- Sample

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

Views

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

	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

Messages

Query executed successfully.

(local) (10.0 RTM) | sa (54) | Sample | 00:00:00 | 5 rows

Ready 5:03 / 16:48 Ln 18 Col 1 Ch 1 INS

NonClustered Index

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

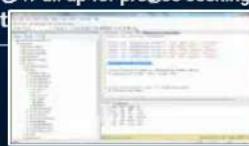
ID	Name	Salary	Gender	City
1	Sam	2500	Male	London
5	Todd	3100	Male	Toronto
3	John	4500	Male	New York
4	Sara	5500	Female	Tokyo
2	Pam	6500	Female	Sydney

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 order according to the index key, which doesn't in any way influence the storage of data in the table.



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.

Unique and Non-Unique Indexes in sql server Part 37

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The top menu bar includes File, Edit, View, Project, Debug, Tools, Window, Community, and Help. Below the menu is a toolbar with various icons for database management tasks.

The Object Explorer on the left displays the database structure for '(local) (SQL Server 10.0.1600 - sa)'. It shows the following hierarchy:

- Databases
 - System Databases
 - AdventureWorks2008
 - ReportServer
 - ReportServerTempDB
 - Sample
- Tables
 - System Tables
 - dbo.PersonDetails
 - dbo.tblDateTime
 - dbo.tblDepartment
 - dbo.tblEmployee
 - Columns
 - Keys (highlighted with a red circle)
 - Constraints
 - Triggers
 - Indexes
 - Statistics
 - dbo.tblEmployeeOld
 - dbo.tblEmployees
 - dbo.tblGender
 - dbo.tblIndiaCustomers
 - dbo.tblPerson
 - dbo.tblRegistrations
 - dbo.tblUKCustomers
 - dbo.tblUSCustomers
- Views

The central pane contains two tabs: 'SQLQuery3.sql - (l...).Sample (sa (55))' and 'SQLQuery1.sql - (l...).Sample (sa (52))'. The code in the tabs is as follows:

```
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
```

The 'Messages' pane at the bottom shows an error message: "Line 1 allowed on index 'tblEmployee.PK_tblEmplo_3214EC0757DD0BE4'. It is being used for PRIMARY KEY constraint enforcement." A red circle highlights the 'Keys' node under the 'tblEmployee' table in the Object Explorer.

The status bar at the bottom indicates 'Query completed with errors.' and shows connection details: '(local) (10.0 RTM) | sa (52) | Sample | 00:00:00 | 0 rows'.

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, whereas 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.

SUBQUERY

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

Syntax :

```
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

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas: logicfirst sys

Query 1 SQL File

```
3 * SELECT * FROM employee;
4 * SELECT * FROM branch;
5
6 * SELECT * FROM employee
7 WHERE branch_id=(SELECT branch_id FROM branch
8 WHERE br_name="Chennai");
```

Result Grid: Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid Form Editor Field Types

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

Information: Output: Apply: Context Help: Snippets

Pull up for precise searching:

```
1:02
1:53 / 16:13
```

Message: Duration / Fetch: 0 rows(0) affected 0.015 sec; 0.016 sec / 0.000 sec; 0.000 sec / 0.000 sec; 0.000 sec / 0.000 sec

Detailed Query Log:

```
01:02:00 SE logined.
01:02:00 SELECT * FROM employee LIMIT 0, 1000.
01:02:00 SELECT * FROM branch LIMIT 0, 1000.
01:02:00 SELECT * FROM employee WHERE branch_id=(SELECT branch_id FROM branch WHERE br_name="Chennai") LIMIT 0, 1000.
```

Subqueries | Exists | Any | All | SQL in Tamil | Logic First Tamil

Local instance MySQL00

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL00

SCHEMAS: logicfirst

Query 1 SQL File

```

21 -- branches containing atleast one admin
22 * SELECT branch_id,br_name
23 FROM branch
24 WHERE EXISTS (
25   SELECT * FROM employee
26   WHERE job_desc = "ADMIN" AND branch.branch_id=employee.branch_id);
27
28

```

Result Grid: Filter Rows: Edit: Export/Import: Wrap Cell Content: Apply: Context Help: Snippets

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

Information: Output: Action: Output: No object selected

Time Action Message Duration / Fetch

- 1 16:34:45 SELECT branch_id,br_name FROM branch LIMIT 0,1000 4 rows returned 0.000 sec / 0.000 sec
- 2 16:43:58 SELECT branch_id,br_name FROM branch WHERE EXISTS (SELECT * FROM employee WHERE job_desc = "ADMIN") LIMIT 0,1000 4 rows returned 0.016 sec / 0.000 sec
- 3 16:45:31 SELECT * FROM employee WHERE job_desc = "ADMIN" LIMIT 0,1000 3 rows returned 0.000 sec / 0.000 sec
- 4 16:46:21 SELECT branch_id,br_name FROM branch WHERE EXISTS (SELECT * FROM employee WHERE job_desc = "ADMIN") LIMIT 0,1000 4 rows returned 0.000 sec / 0.000 sec
- 5 16:46:39 SELECT * FROM employee WHERE job_desc = "ADMIN" LIMIT 0,1000 3 rows returned 0.000 sec / 0.000 sec
- 6 16:48:38 SELECT branch_id,br_name FROM branch LIMIT 0,1000 4 rows returned 0.000 sec / 0.000 sec
- 7 16:51:38 SELECT branch_id,br_name FROM branch LIMIT 0,1000 4 rows returned 0.000 sec / 0.000 sec
- 8 16:51:39 SELECT * FROM employee LIMIT 0,1000 15 rows returned 0.000 sec / 0.000 sec

Object Info Session: 10:01 / 16:13

Subqueries | Exists | Any | All | SQL in Tamil | Logic First Tamil

```
MySQL [Workstation] Local instance MySQL80 < i < 100% < x
File Edit View Query Database Server Tools Scripting Help
Navigator: Local instance MySQL80 < SQL Editor < SQL File < 
SCHEMAS: logicfirst < Filter objects < 
  Tables < Views < Stored Procedures < Functions < sys < 
Query 1 SQL File < 
19 -- EXISTS used with subquery
20 -- branches containing atleast one admin
21 * SELECT branch_id,br_name
22   FROM branch
23 * WHERE EXISTS (
24     SELECT * FROM employee
25     WHERE job_desc = "ADMIN" AND branch.branch_id=employee.branch_id);
26
27 -- ANY
28 * -- branch info in which any employee gets more than 25L
29 * -- SELECT branch_id FROM employee
30
31 * WHERE salary>
32
SQL Editor: 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 Output: Action: Output * Time Action Message Duration / Fetch
No object selected
10:43 / 16:13
```

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

logistics

- Tables
- Views
- Stored Procedures

sys

Query 1 SQL File 2

```
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

Administration Schemas

No object selected

Information

Action Output

Time	Action	Message	Duration / Fetch
1 17:03:59	SELECT branch_id FROM employee WHERE salary>2500000 LIMIT 0, 1000	3rows 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	2rows returned	0.000 sec / 0.000 sec

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

SCHEMAS: logicfirst

Query 1 SQL File Z X

```
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 | Filter Rows | Edit: | Export/Import: | Who Can Content: |

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

Action Output

#	Time	Action	Message	Duration / Freq
1	17:03:59	SELECT branch_id FROM employee WHERE salary>2500000 LIMIT 0, 1000	3rows 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	2rows returned	0.000 sec / 0.000 sec
3	17:13:40	SELECT branch_id FROM branch WHERE br_name IN ("Chennai","Coimbatore") LIMIT 0, 1000	2rows returned	0.000 sec / 0.000 sec
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	3rows returned	0.000 sec / 0.000 sec

Object Info Session



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

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 Products WHERE ProductID = ALL (SELECT ProductID FROM Products WHERE Quantity = 10);`

INSERT INTO SELECT

The `INSERT INTO SELECT` statement inserts one or more rows from one table and inserts it into another table. The `INSERT INTO SELECT` statement preserves data types in source and target table.

The existing records in the target table are not affected.

Example:-
`INSERT INTO table2
SELECT * FROM table1
WHERE condition;`

`INSERT INTO table2(column1, column2, ...)
SELECT column1, column2, column3
FROM table1
WHERE condition;`

INSERT INTO Statement

The `INSERT INTO` statement is used to insert records in a table.

It is possible to write the `INSERT` statement 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/ewS3kh>



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

ble is

- 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 result.
- It returns TRUE if ANY of the subquery satisfies the condition.

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

Example:-

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

ALL Operator

- It returns a boolean value as a result.
 - It returns TRUE if ALL of the subquery satisfies the condition.
 - It is used with SELECT, WHERE and UPDATE.
- ALL means that the condition will be satisfied if the operation is true for all values in the subquery.

Example:-

- SELECT ALL ProductNo
FROM Sales
WHERE TRUE;

ble is
; side-by-side on top
you to
to have
ame table

;

iple

le OR



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

PG +8 Anagha M VR AE ML RT PS

View Chat

Recordings are available for 30 days

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

Participants

Invite Stop

0 Type here to search

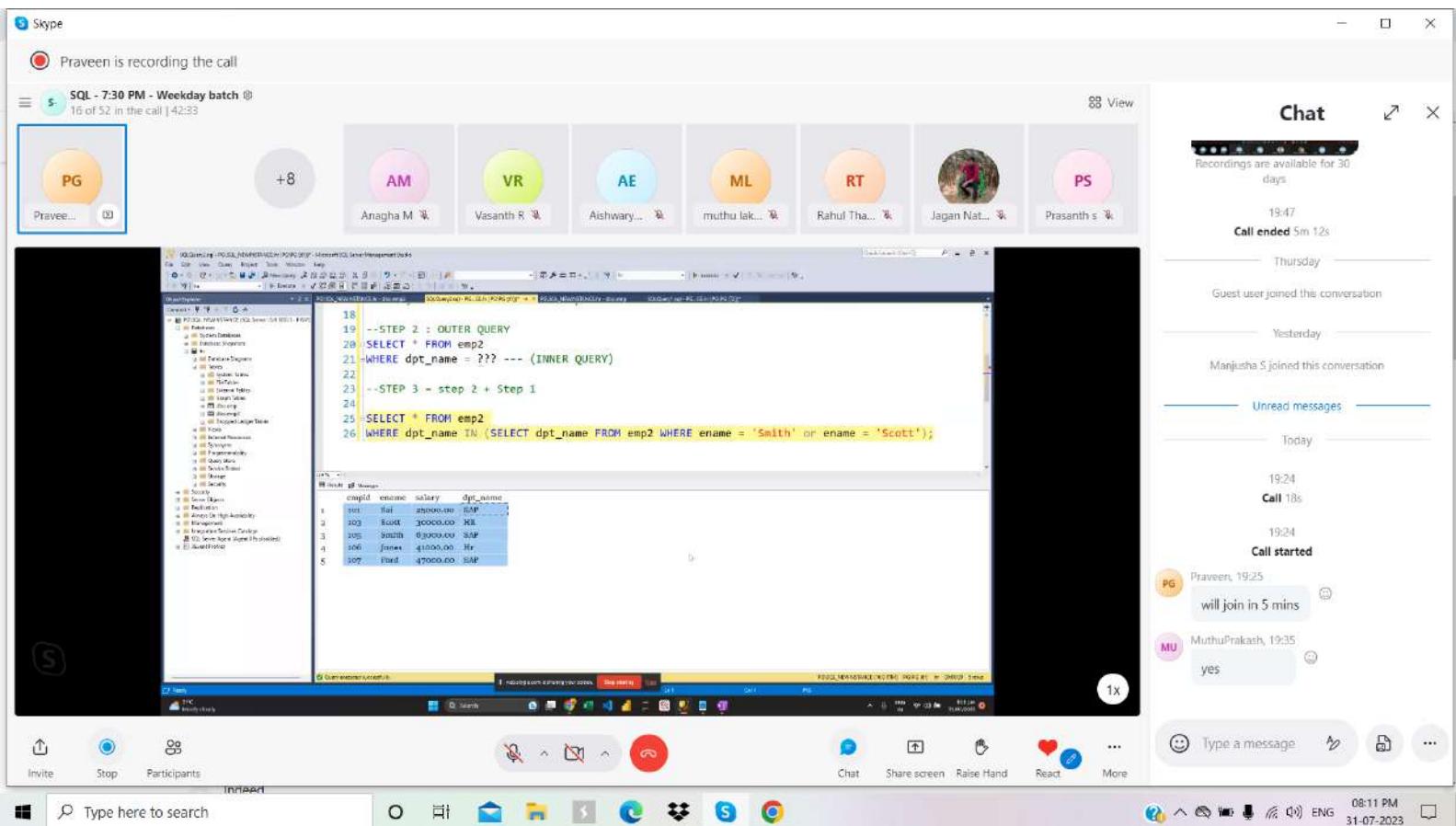
Chat Share screen Raise Hand Read More

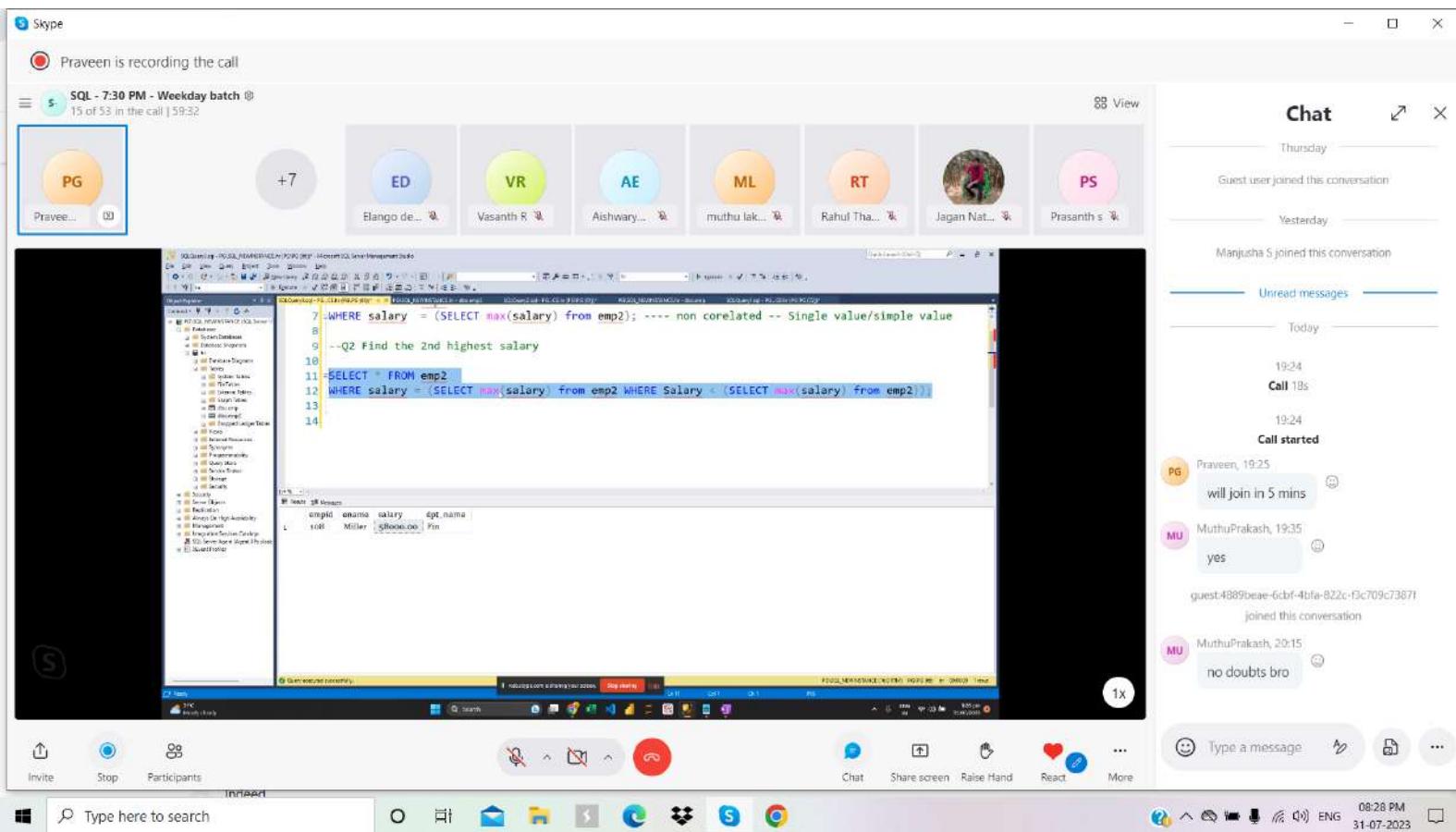
08:08 PM 31-07-2023

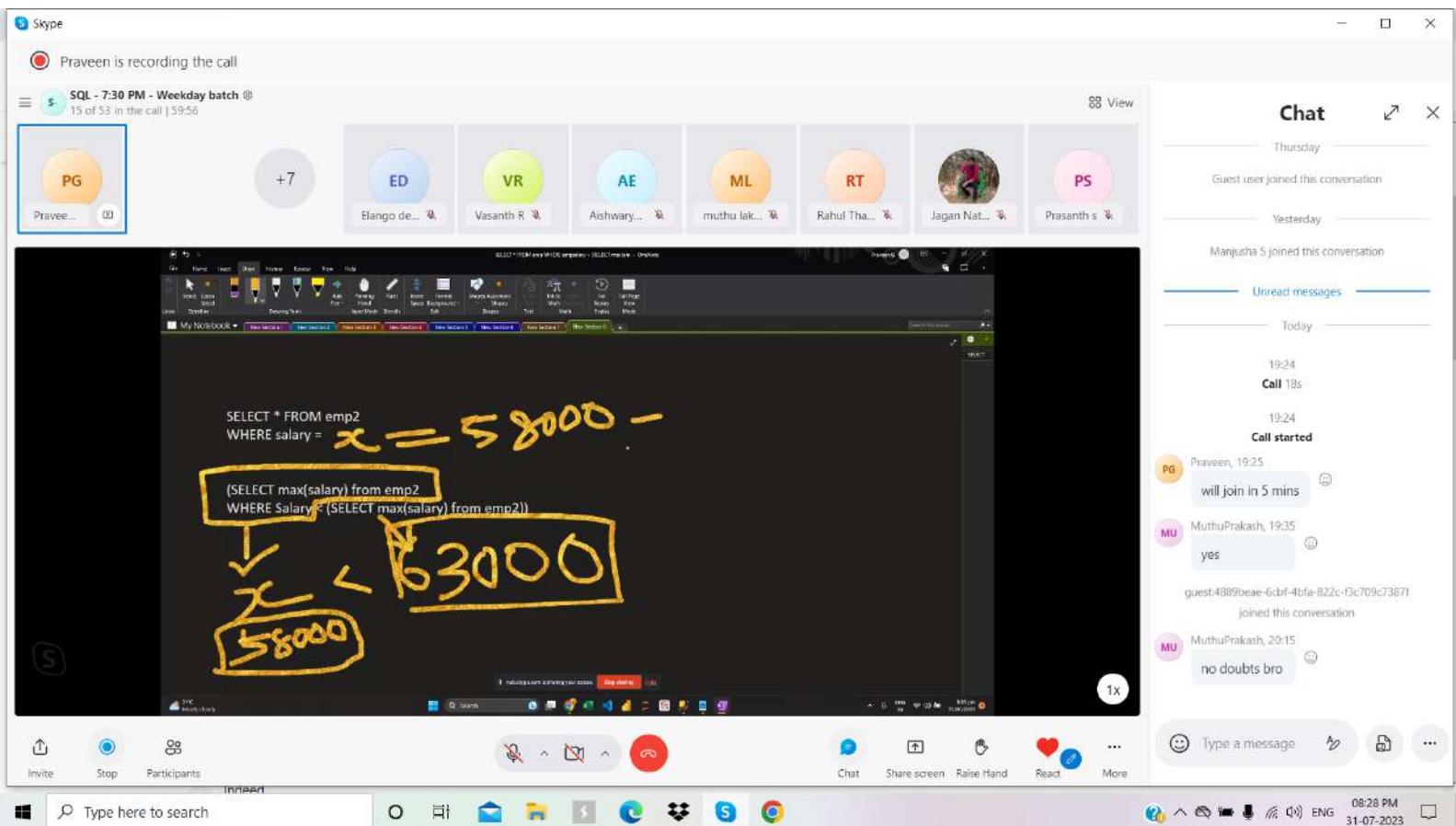
```
13 --STEP 1 : INNER QUERY
14 SELECT dpt_name FROM emp2
15 WHERE ename = 'Smith' or ename = 'SCOTT';
16
17 --- now, we found the value of dept of SCOTT and SMITH
18
19 --STEP 2 : OUTER QUERY
20 SELECT * FROM emp2
21 WHERE dpt_name = ??? --- (INNER QUERY)
22
23 --STEP 3 = step 1 + Step 2
```

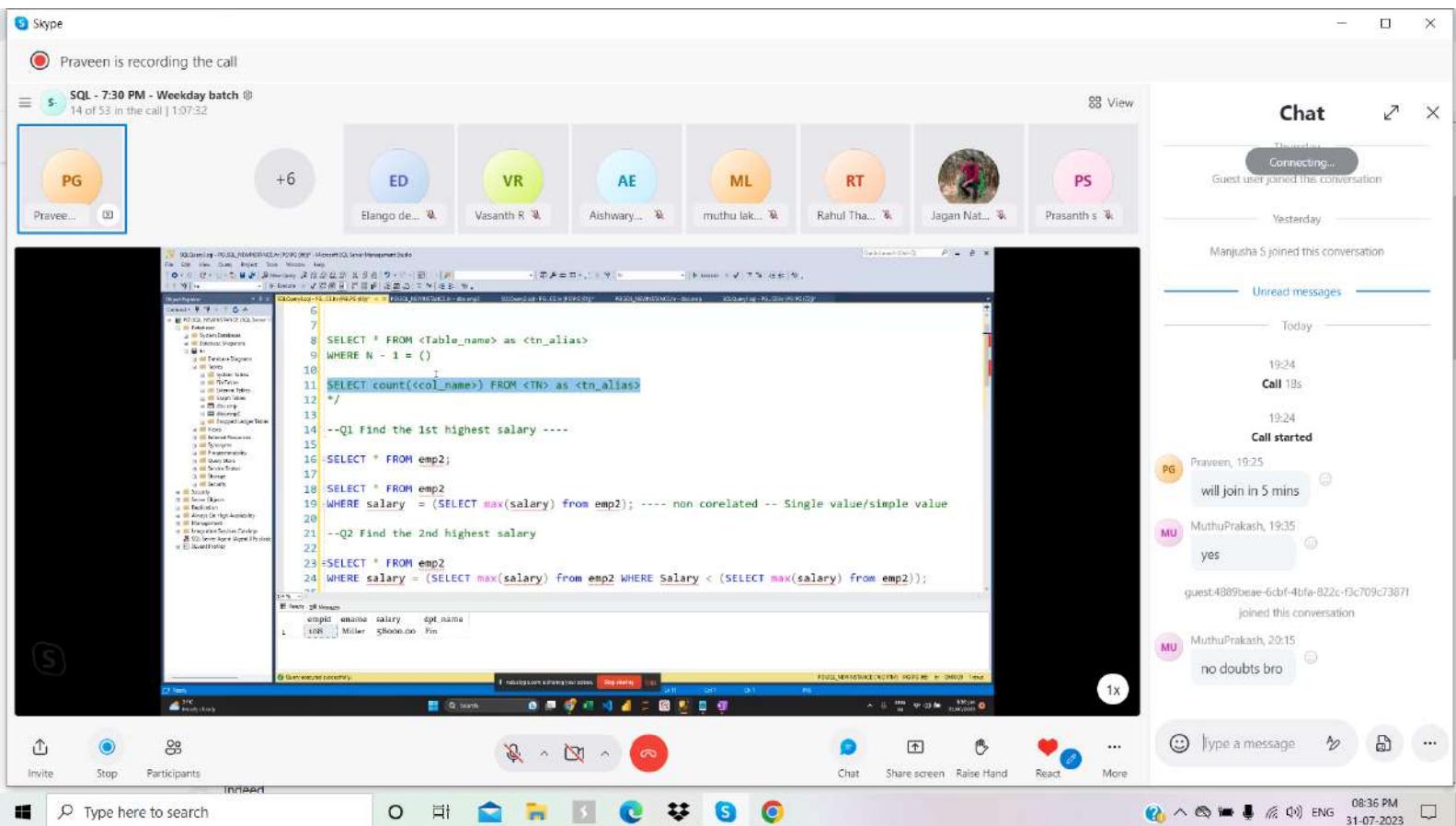
empid	ename	salary	dpt_name
101	Sax	25000.00	SAP
102	Allen	15000.00	IT
103	Scott	30000.00	IT
104	Warner	17000.00	IT
105	Smith	60000.00	SAP
106	Jones	41000.00	IT
107	Ford	47000.00	SAP
108	Millar	50000.00	IT

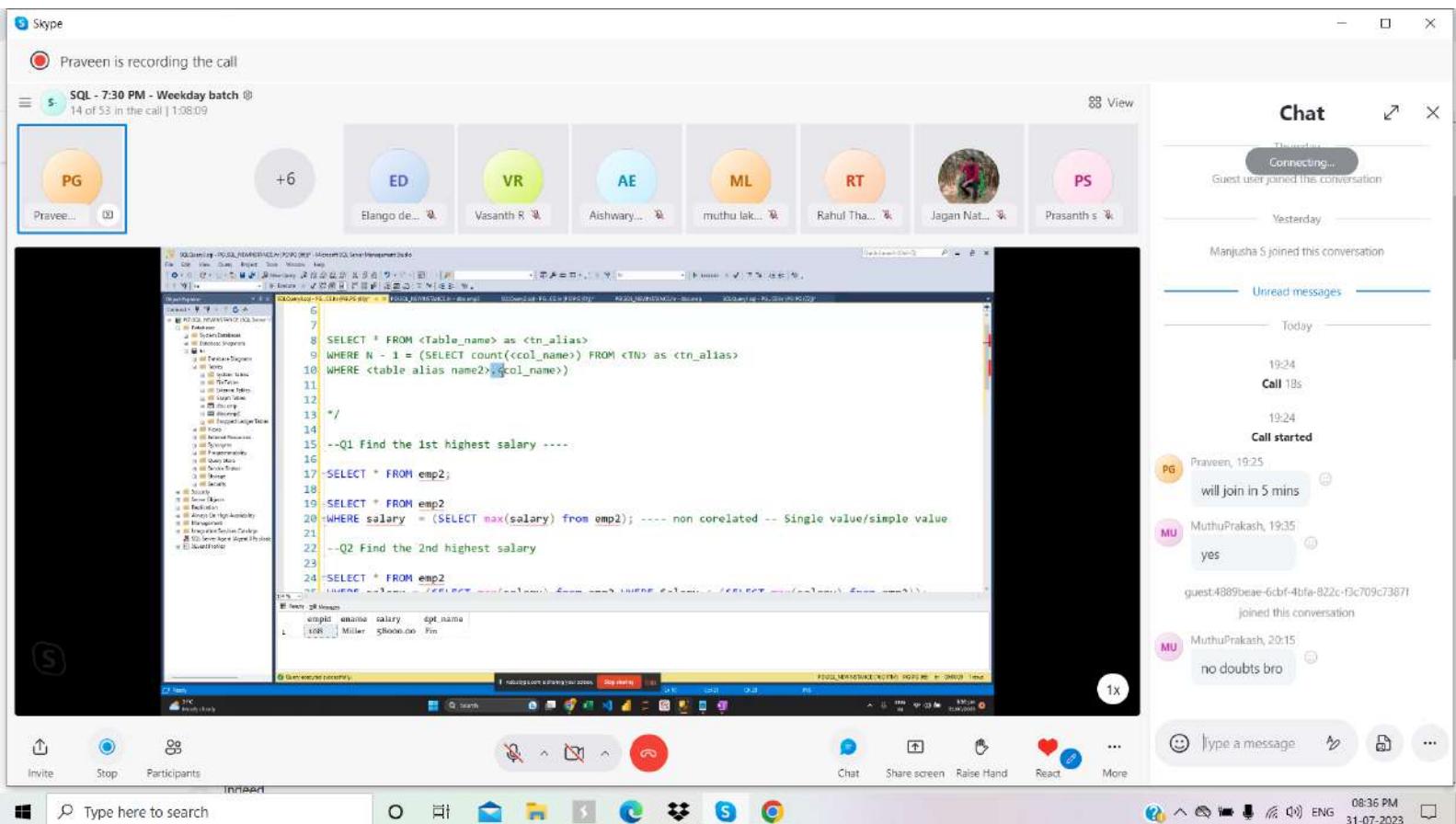
dpt_name
IT
SAP

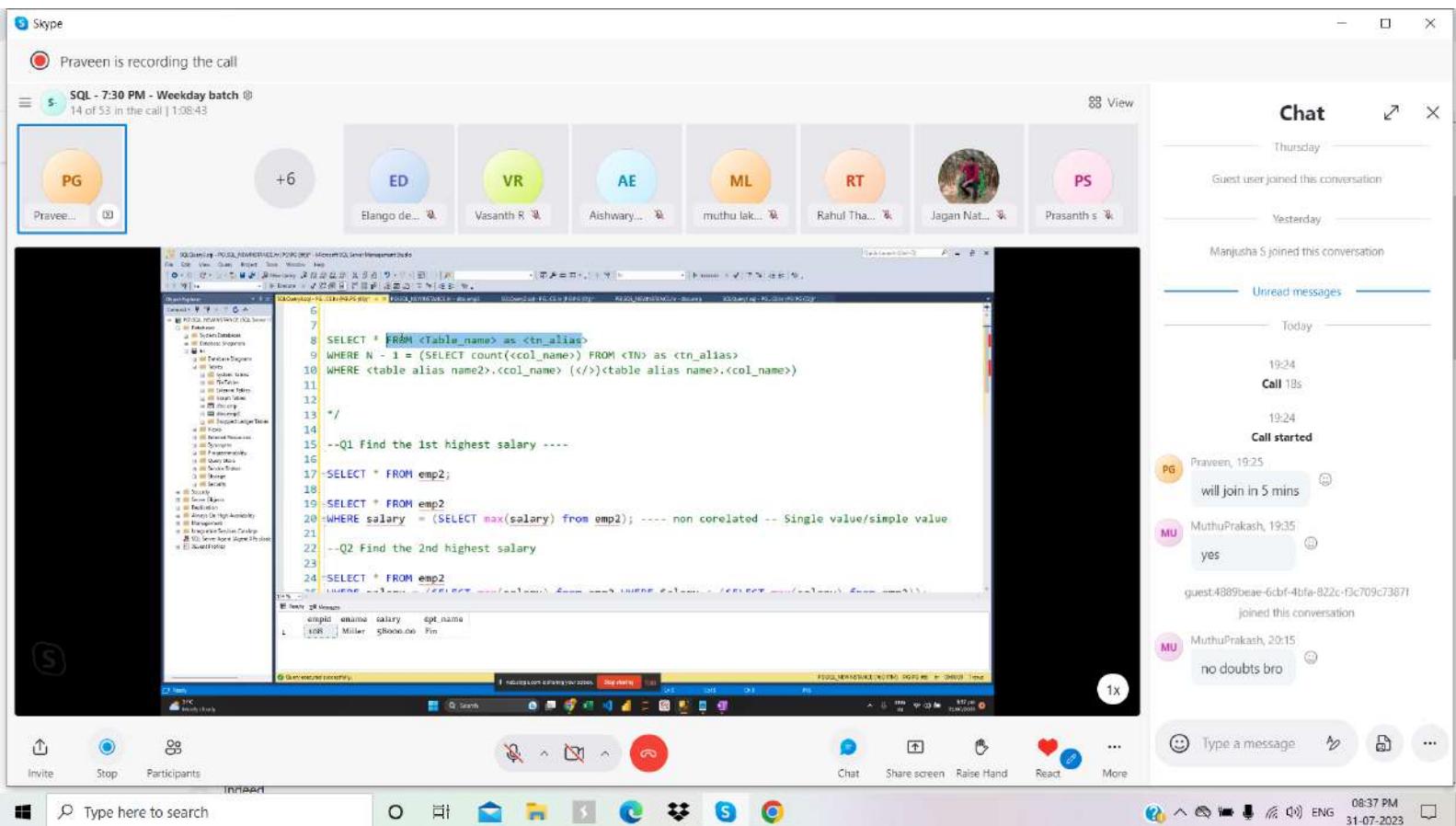


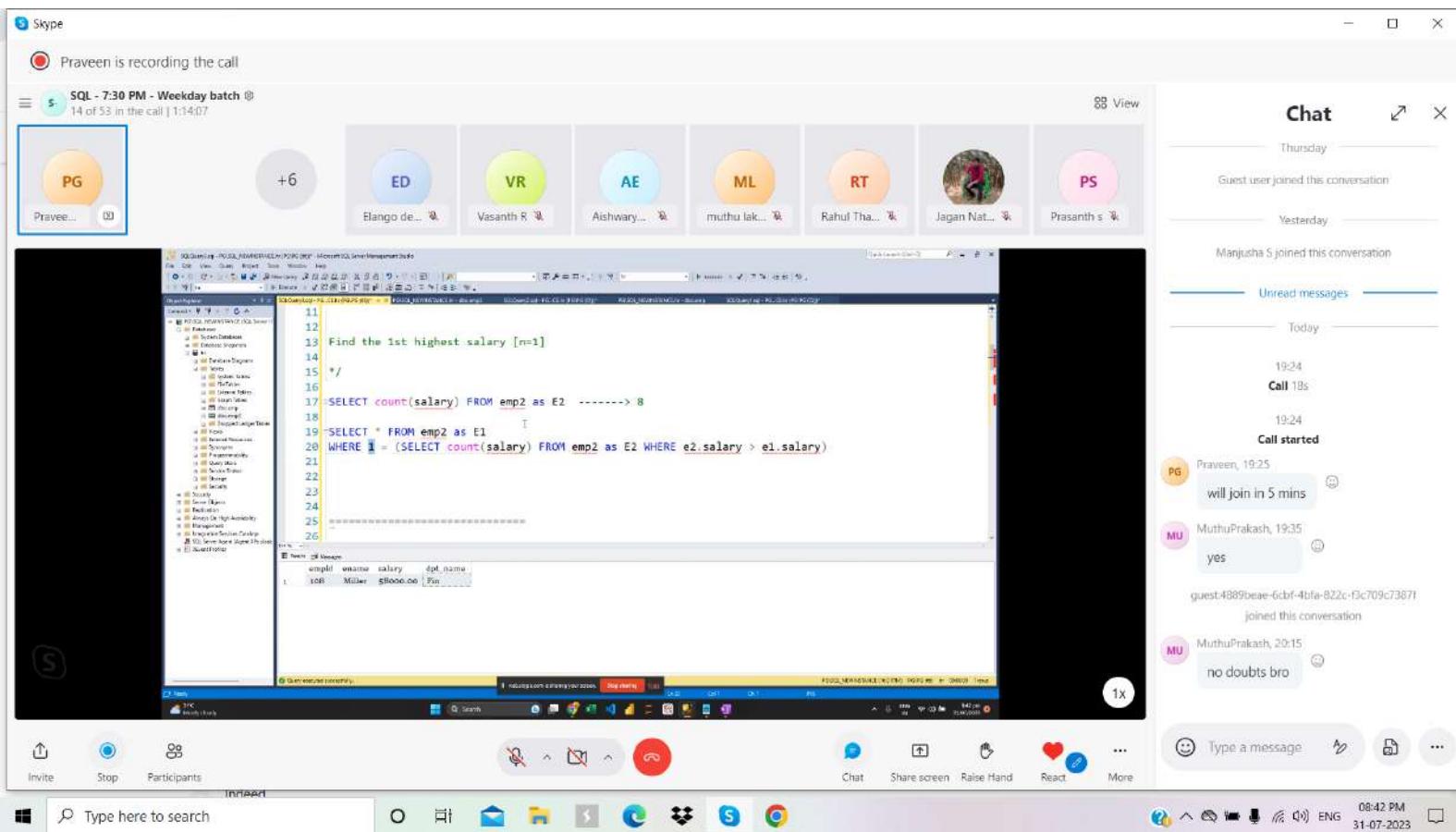












MySQL Cursor with Example https://www.mysqltutorial.org/mysql-cursor/

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

Ten Frame Flashcard.pdf addition and subtrac...pdf adding within 5 ga...pdf Summer Math Pac...pdf Addition_Subtracti...pdf Show all

```
SQLQuery1.sql - LAPTOP-UN6223EF\SQLPRAVEEN (LAPTOP-UN6223EF\PG (84)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query New Item Task List Change Type master Execute
Object Explorer
Connected: LAPTOP-UN6223EF\SQLPRAVEEN (SQL Server 16.0.1000 - LA)
Databases
System Databases
Database Snapshots
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery1.sql - ...UN6223EF\PG (84) = X
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
```

278 % Connected (1/1) web skype.com is sharing your screen Stop sharing Hide

Praveen G 05:58:22 28/09/2023

00:03:09

kart... Man... Aishwarya esh... Gowtham NarasimhaRed... Bala Praveen G

```
1 ---L-UN6233EFG (B4) --- X
2
3 7 ┌─CREATE TABLE dept(
4     8   S_no int IDENTITY,
5     9   Full_Name varchar(100)
6    10 );
7
8 11 Go
9
10 12
11 13 SP_HELP dept;
12 14 Go
13
14 15
15 16 ┌─INSERT INTO dept VALUES
16     17   (1, 'Praveen');           --_this will not work
17 18 Go
18
19 20
20 21
21 22
22
23 23 ┌─--IDENTITY(Seed, Increment)
24     24   --IDENTITY(S, I)
25     25   -- Seed - it indicates the starting value
```

```
19
20  INSERT INTO dept(Full_Name) VALUES
21  ('PRAVEEN')
22  Go
23
24  --DQL/DRL
25  SELECT * FROM dept;
```

S_no	Full_Name
1	PRAVEEN

Query executed successfully.

Praveen G Ready 00:13:33

Man... Aish... NarasimhaRed... Kailainathan S Bala Praveen G

00:47:58

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) window. The left pane displays the Object Explorer with a connection to 'LAPTOP-UN6223EP\SQLPRAVEEN' (SQL Server 16.0.1050 - Local). The right pane contains a query editor window titled 'SQLQuery1.sql - UWA6223EP\PG (34)'. The code in the editor is as follows:

```
25 ('Orange');
26
27 INSERT INTO dept(Full_Name) VALUES
28 ('Apple');
29
30 --DQL/DRL
31 SELECT * FROM dept;
32
33 --SYNTAX
34 SET IDENTITY_INSERT dept OFF
35 -- the user cannot insert the values into identity column by explicitly
36
37 SET IDENTITY_INSERT dept ON
38
39
40
41
42 --IDENTITY(Seed, Increment)
43 --IDENTITY(S, I)
44 -- Seed - it indicates the starting value
45 -- Increment - indicates the values to be add/next value/Step into
```

A yellow vertical bar highlights the code from line 25 to line 45. A tooltip box appears over the line 'SET IDENTITY_INSERT dept OFF' with the text: 'the user cannot insert the values into identity column by explicitly'. The status bar at the bottom shows 'Query completed with errors'.

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The query window displays the following T-SQL code:

```
31  SELECT * FROM dept;
32
33  --SYNTAX
34  SET IDENTITY_INSERT dept OFF;
35  -- the user cannot insert the values into identity column by explicitly
36
37  SET IDENTITY_INSERT dept ON;
38  -- we can insert the values into identity column implicitly/Manually
39
40  INSERT INTO dept(S_no, Full_Name) VALUES
41  (108, 'Hello World')
42
```

The execution results show:

```
(1 row affected)
```

Completion time: 2023-06-28T19:58:43.3310095+05:30

The status bar at the bottom indicates "Query executed successfully." and shows the current user is "Praveen G".

SQLQuery1.sql - LAPTOP-UN6Q23EF\SQLPRAVEEN.hr (LAPTOP-UN6Q23EF)\PG (84) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query New Item Object Explorer

Execute

Quick Launch (Ctrl+Q)

Object Explorer

Connect to... Connect to... Connect to...

LAPTOP-UN6Q23EF\SQLPRAVEEN (SQL Server 15.0.1050 - Local)

Database

System Databases

Database Snapshots

Security

Server Objects

Replication

Management

XEvent Profiler

57
58 SELECT * FROM dept;
59
60 --IDENTITY(Seed, Increment)
61 --IDENTITY(S, I)
62 -- Seed - it indicates the starting value
63 -- Increment - indicates the values to be add/next value/Step into
64
65 --DML
66 ----UPDATE
67
68 UPDATE dept_I
69 SET Full_Name = 'Priyanka'
70 WHERE S_no = 2

Results Messages

S_no	Full_Name
1	PRAVEEN
2	Orange
3	Apple
4	Hello World
5	Kanesh
6	Lemon
7	Lemon
8	Lemon

Query executed successfully.

Stop sharing hide

LAPTOP-UN6Q23EF\SQLPRAVEEN ... LAPTOP-UN6Q23EF\PG (84) hr : 00:00:00 | 8 rows

00:37:36 00:23:55

Pra... 00:37:36 00:23:55

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

SQLQuery1.sql - LAPTOP-UN6Q23EF\SQLPRAVEEN.nv (LAPTOP-UN6Q23EF\PB (84)) - Microsoft SQL Server Management Studio

```
--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');

INSERT INTO dept(S_no, Full_Name) VALUES
(1001, 'Kamesh');

---
```

Results

S_no	Full_Name
1	PRAVEEN
2	Praveen
3	Apple
4	Hello World
5	Kamesh
6	Lemon
7	Lemon
8	Lemon

Query executed successfully.

00:38:28 00:23:03

Video-20230628_150834-Meeting Recording

SQLQuery1.sql - LAPTOP-UH6223EP\SQLPRAVEEN.ln (LAPTOP-UH6223EP\PB (84)) - Microsoft SQL Server Management Studio

```
55  INSERT INTO dept(Full_Name) VALUES
56  ('Lemon');
57
58  SELECT * FROM dept;    I
59
60  --IDENTITY(Seed, Increment)
61  --IDENTITY(S, I)
62  -- Seed - it indicates the starting value
63  -- Increment - indicates the values to be add/next value/Step into
64
65  --DML
66  ----UPDATE
67
68  UPDATE dept
69  SET Full_Name = 'Priyanka'
```

(1 row affected)

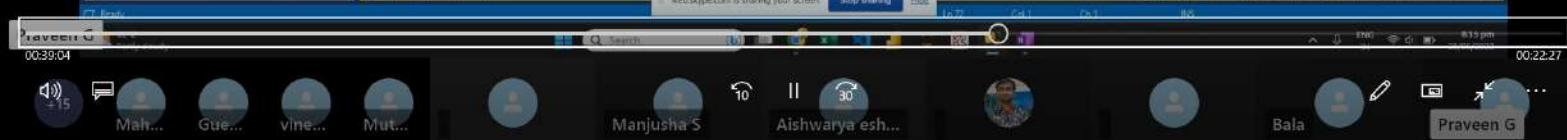
Completion time: 2023-06-28T20:15:42.5030119+05:30

Query executed successfully.

web skype.com is sharing your screen

Stop sharing Hide

LAPTOP-UH6223EP\SQLPRAVEEN ... LAPTOP-UH6223EP\PB (84) hr 00:00:30 0 rows



The screenshot shows a Microsoft SQL Server Management Studio (SSMS) window. The Object Explorer on the left shows a connection to 'LAPTOP-UN6223EP\SQLPRAVEEN'. The main pane displays the following T-SQL code:

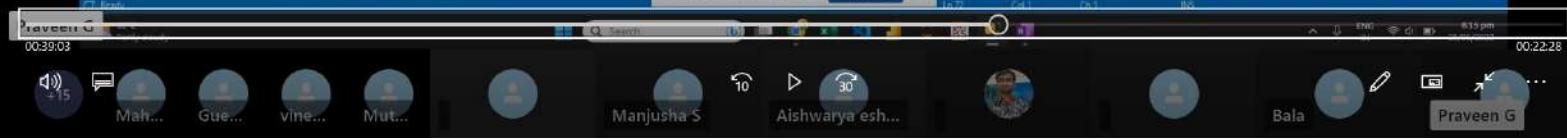
```
67
68 UPDATE dept
69   SET Full_Name = 'Priyanka'
70 WHERE S_no = 2;
71
72 UPDATE dept
73   SET Full_Name = 'Tree'
74 WHERE S_no = 3;
```

After executing the code, the results pane shows:

(1 row affected)

Completion time: 2023-06-28T20:15:42.5030119+05:30

At the bottom of the SSMS window, a status bar indicates: 'Query executed successfully.' and 'LAPTOP-UN6223EP\SQLPRAVEEN ... LAPTOP-UN6223EP\PG (4) hr: 00:00:30 6 rows'.



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.

3. 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.

4. 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 columns.

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) >

Complete guide to Database Normalization in SQL									
A27	B	C	D	E	F	G	H	I	
12	Columns which are not part of the candidate key or primary key								
13									
14	Partial Dependency:								
15	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 columns.								
16	If there is any non key column which depends only on one of the candidate key columns then this results in partial dependency.								
17									
18	ORDER_NUMBER	QUANTITY	ITEM_PRICE	TOTAL_COST	ORDER_DATE	STATUS	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME	DEAL_SIZE
19	10101	30	202	6060	2003-03-24	Shipped	Yu	Kwai	Small
20	10102	34	550	18700	2003-07-05	Cancelled	Henriot	Paul	Medium
21	10103	41	420	17220	2003-01-07	Shipped	Da Cunha	Daniel	Medium
22	10104	45	780	35100	2003-08-25	On Hold	Young	Julie	High
23	10105	49	850	41650	2003-10-10	Shipped	Brown	Julie	High
24	10106	4 780		3120	2005-05-07	Shipped	Perrier	Dominique	Small
25	10106	4 850		3400	2005-05-07	Shipped	Perrier	Dominique	Small
26									
27	Candidate key = Order_number Product_code								
28									
29	PRODUCT_CODE	PRODUCT_NAME	PRODUCT_PRICE						
30	PRD_1001	Motorcycles	200						
31	PRD_1002	Vintage Cars	500						
32	PRD_1003	Classic Cars	400						
33	PRD_1004	Buses	800						
34	PRD_1005	Trucks	900						
35	PRD_1004	Buses	800						
36	PRD_1005	Trucks	900						
37									
38									
39	CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY
40	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA
41	Reims Collectables	2123356666	59 rue de l'Abbaye		Reims		51100	France	EMEA
42	Pause (k)s	2121257800	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA
43	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA

19:14 / 40:50 • Second Normal Form (2NF) > HD

Complete guide to Database Normalization in SQL

	A	B	C	D	E	F	G	H	I	J	K
18	ORDERS										
19	ORDER_ID	ORDER_NUMBER	QUANTITY	ITEM_PRICE	TOTAL_COST	ORDER_DATE	STATUS	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME	DEAL_SIZE	
20	1	10101	30	202	6060	2003-03-24	Shipped	Yu	Kwai	Small	
21	2	10102	34	550	18700	2003-07-05	Cancelled	Henriot	Paul	Medium	
22	3	10103	41	420	17220	2003-01-07	Shipped	Da Cunha	Daniel	Medium	
23	4	10104	45	780	35100	2003-08-25	On Hold	Young	Julie	High	
24	5	10105	49	850	41650	2003-10-10	Shipped	Brown	Julie	High	
25	6	10106	4780		3120	2005-05-07	Shipped	Perrier	Dominique	Small	
26	7	10106	4850		3400	2005-05-07	Shipped	Perrier	Dominique	Small	
27	PK										
28	Candidate key = Order_number Product_code										
29											
30	PRODUCTS										
31	PRODUCT_CODE	PRODUCT_NAME	PRODUCT_PRICE								
32	PRD_1001	Motorcycles	200								
33	PRD_1002	Vintage Cars	500								
34	PRD_1003	Classic Cars	400								
35	PRD_1004	Buses	800								
36	PRD_1005	Trucks	900								
37	PK										
38											
39											
40	CUSTOMERS										
41	CUSTOMER_ID	CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY	
42	C1	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA	
43	C2	Reims Collectables	2123356666	59 rue de l'Abbaye		Reims		51100	France	EMEA	
44	C3	Lyon Souveniers	2121257800	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA	
45	C4	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA	
46	C5	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisco	CA		USA	NA	
47	C6	Auto Canal Petit	6577771000	25, rue Lauriston		Paris		75016	France	EMEA	
48											
49											
50											

23:31 / 40:50 • Second Normal Form (2NF) >

Complete guide to Database Normalization in SQL

	A	B	C	D	E	F	G	H	I	J	K
35	PRD_1004	Buses	800								
36	PRD_1005	Trucks	900								
37	PK										
38											
39											
40	CUSTOMERS										
41	CUSTOMER_ID	CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY	
42	C1	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY		10022 USA	NA	
43	C2	Reims Collectables	2123356666	59 rue de l'Abbaye		Reims			51100 France	EMEA	
44	C3	Lyon Souveniers	2121257800	27 rue du Colonel Pierre Avia		Paris			75508 France	EMEA	
45	C4	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA		90003 USA	NA	
46	C5	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisco	CA		USA	NA	
47	C6	Auto Canal Petit	6577771000	25, rue Lauriston		Paris			75016 France	EMEA	
48											
49	RELATIONSHIPS										
50	ORDER_ID	PRODUCT_CODE	CUSTOMER_ID								
51	1	PRD_1001	C1								
52	2	PRD_1002	C2								
53	3	PRD_1003	C3								
54	4	PRD_1004	C4								
55	5	PRD_1005	C5								
56	6	PRD_1004	C6								
57	7	PRD_1005	C6								
58											
59											
60											
61											
62											
63											
64											
65											
66											
67											

26:14 / 40:50 • Third Normal Form (3NF) >       

	PRODUCT_NAME																
1	3NF (Third Normal Form)																
2	1. Must be in 2NF																
3	2. Avoid Transitive dependencies.																
4																	
5	Transitive Dependency:																
6	Lets say you have a table T which has 3 columns namely A, B and C.																
7	If A is functionally dependent on B and B is functionally dependent on C then we can say that A is functionally dependent on C.																
8																	
9	ORDERS																
10	ORDER_ID	ORDER_NUMBER	QUANTITY	ITEM_PRICE	TOTAL_COST	ORDER_DATE	STATUS	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME	DEAL							
11	1	10101	30	202	6060	2003-03-24	Shipped	Yu	Kwai	Small							
12	2	10102	34	550	18700	2003-07-05	Cancelled	Henriot	Paul	Medium							
13	3	10103	41	420	17220	2003-01-07	Shipped	Da Cunha	Daniel	Medium							
14	4	10104	45	780	35100	2003-08-25	On Hold	Young	Julie	High							
15	5	10105	49	850	41650	2003-10-10	Shipped	Brown	Julie	High							
16	6	10106	4	780	3120	2005-05-07	Shipped	Perrier	Dominique	Small							
17	7	10106	4	850	3400	2005-05-07	Shipped	Perrier	Dominique	Small							
18	PK																
19																	
20	PRODUCTS																
21	PRODUCT_CODE	PRODUCT_NAME	PRODUCT_PRICE														
22	PRD_1001	Motorcycles	200														
23	PRD_1002	Vintage Cars	500														
24	PRD_1003	Classic Cars	400														
25	PRD_1004	Buses	800														
26	PRD_1005	Trucks	900														
27	PK																
28																	
29																	
30	CUSTOMERS																
31	CUSTOMER_ID	CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY							
32	C1	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY		10022 USA								
33	C2	Reims Collectables	2123256666	59 rue de l'Albâtre		Reims			51100 France								

< > Definitions De-Normalized Dataset 1NF 2NF 3NF INSERTION Anomalies DELETION Anomalies UPDATION Anomalies +



Calibri (Body) 12 A A Wrap Text General Conditional Formatting Format as Table Normal Good Neutral Insert Delete Format Auto Sort & Filter Find & Select Analyse Data

A49 EMP_ID

PK										
27	CUSTOMERS									
31	CUSTOMER_ID	CUSTOMER_NAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY
32	C1	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA
33	C2	Reims Collectables	2123356666	59 rue de l'Abbaye		Reims		51100	France	EMEA
34	C3	Lyon Souveniers	2121257800	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA
35	C4	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA
36	C5	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisco	CA		USA	NA
37	C6	Auto Canal Petit	6577771000	25, rue Lauriston		Paris		75016	France	EMEA
38	RELATIONSHIPS									
40	ORDER_ID	PRODUCT_CODE	CUSTOMER_ID							
41	1	PRD_1001	C1							
42	2	PRD_1002	C2							
43	3	PRD_1003	C3							
44	4	PRD_1004	C4							
45	5	PRD_1005	C5							
46	6	PRD_1004	C6							
47	7	PRD_1005	C6							
48	EMP_ID	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME							
50	E1	Yu	Kwai							
51	E2	Henriot	Paul							
52	E3	Da Cunha	Daniel							
53	E4	Young	Julie							
54	E5	Brown	Julie							
55	E6	Perrier	Dominique							
56										
57										
58										
59										
60										

Clipboard (Body) 12 A A Wrap Text General Conditional Formatting as Table Normal Good Bad Neutral Insert Delete Format Auto-Sort Fill Sort & Filter Find & Select Analyse Data

A48 RELATIONSHIPS

	A	B	C	D	E	F	G	H	I	J
33	C2	Reims Collectables	2123356666	59 rue de l'Abbaye	Reims			51100	France	EMEA
34	C3	Lyon Souveniers	2121257800	27 rue du Colonel Pierre Avia	Paris			75508	France	EMEA
35	C4	Toys4GrownUps.com	6265557265	78934 Hillside Dr.	Pasadena	CA		90003	USA	NA
36	C5	Corporate Gift Ideas Co.	6505551386	7734 Strong St.	San Francisco	CA			USA	NA
37	C6	Auto Canal Petit	6577771000	25, rue Lauriston	Paris			75016	France	EMEA
38										
39		EMPLOYEES								
40	EMP_ID	SALES_CONTACT_LASTNAME	SALES_CONTACT_FIRSTNAME							
41	E1	Yu	Kwai							
42	E2	Henriot	Paul							
43	E3	Da Cunha	Daniel							
44	E4	Young	Julie							
45	E5	Brown	Julie							
46	E6	Perrier	Dominique							
47										
48		RELATIONSHIPS								
49	ORDER_ID	PRODUCT_CODE	CUSTOMER_ID	EMP_ID						
50	1	PRD_1001	C1	E1						
51	2	PRD_1002	C2	E2						
52	3	PRD_1003	C3	E3						
53	4	PRD_1004	C4	E4						
54	5	PRD_1005	C5	E5						
55	6	PRD_1004	C6	E6						
56	7	PRD_1005	C6	E6						
57										
58										
59										
60										
61										
62										
63										
64										
65										
66										

Definitions De-Normalized Dataset 1NF 2NF 3NF INSERTION Anomalies DELETION Anomalies UPDATION Anomalies +

CLICK HERE TO SUBSCRIBE

A screenshot of a web browser showing a YouTube video player and a sidebar of recommended videos.

The main video player displays a slide from a presentation about "1NF (First Normal Form)". The slide includes the following text:

1. Every column/attribute need to have a single value.
2. Each row should be unique. Either through a single or multiple columns. Not mandatory to have primary key

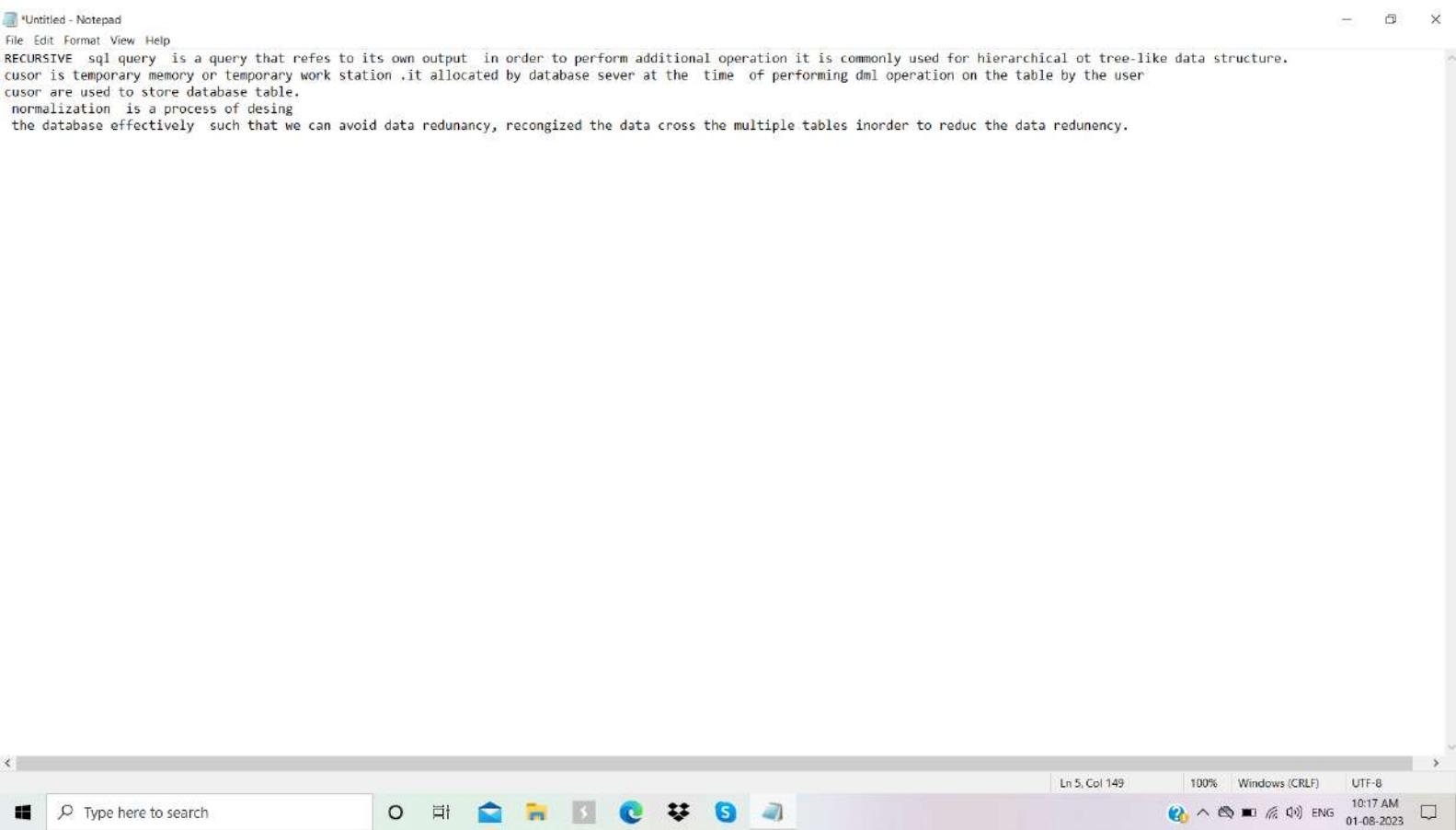
The video progress bar shows it is at 7:54 / 40:50. The video title is "First Normal Form (1NF)".

The sidebar contains the following recommended videos:

- HPE GreenLake** Ad - hpe.com/in/en/greenlake/edge [Learn more](#)
- 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** SQL Views Tutorial

The browser's address bar shows the URL: youtube.com/watch?v=rBPQ5fg_kiY

The taskbar at the bottom shows several open files and applications, including "Screenshot (548).pdf", "D18 - Sub Query -....pdf", "example.pdf", and "D18 - Sub Query -....txt".



en

ance MySQL80 ×

Query Database Server Tools Scripting Help

File Edit Sign Loops cursor ×

Limit to 1000 rows

SQL Additions

Automatic context help is disabled. [help for the current caret position](#)

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

```

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;    I
19 |
20 | end$$
21 | delimiter ;

```

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

employee 3 ->
 Date
 1. Admin Output
 2. 16:59:27 user:root
 3. 16:59:27 user:root:employee 1M:10 0 000

help for the current caret position or click here



Cancel Help / Shutdown

MySQL Workbench

Local instance MySQL 8.0

File Edit View Query Database Server Tools Scripts Help

Navigator Schemas

Filter objects

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

Result Grid File New... Info Wrap Cell Content

emp_id	ename	job_desc	salary	branch_id
1	Ram	ADMIN	1000000	2
2	Hanni	MANAGER	2500000	2
3	George	SALES	2000000	1
4	Ramya	SALES	1300000	2

Automatic context help is disabled. Use the toolbar to manually help for the current caret position or to toggle automatic help.

Run Cancel | ⚙ Disconnect ⚙ Change Connection demo ▾ | ⚙ Explain ⚙ Enable SQLCMD ↗ Export as Notebook

```
1 select * from sales_data;
2
3 select *
4   from (
5     (
6       base query
7     )
8   pivot
9   (
10      aggregate function
11      for column value in ('Jan-21', ' ', ' ')
12    )
```

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

```
1 select customer_id as customer
2 , format(sales_date, 'MMM-yy') as sales_date
3 , replace(amount, '$', '') as amount
4 from sales_data;
5
6 select *
7 from
8 (
9     base query
10 )
11 pivot
12 (
13     aggregate function
14     for column value in ('Jan-21', '', '')
15 )
```

Results Messages

	customer	sales_date	amount
1	Cust-1	Jan-21	50
2	Cust-1	Jan-21	50
3	Cust-1	Jan-21	50
4	Cust-2	Jan-21	100
5	Cust-2	Jan-21	100

▶ Run □ Cancel | ⚙ Disconnect ⚙ Change Connection demo ▾ | ⚙ Explain ⚙ Enable SQLCMD ↗ Export as Notebook

```
1 select *
2 from
3 (
4     select customer_id as customer
5         , format(sales_date, 'MMM-yy') as sales_date
6         , cast[replace(amount, '$', '') as int] as amount
7     from sales_data
8 ) as sales_data
9 pivot
10 (
11     sum(amount)
12     for sales_date in ([Jan-21], [Feb-21], [Mar-21], [Apr-21]
13                         ,[May-21], [Jun-21], [Jul-21], [Aug-21]
14                         ,[Sep-21], [Oct-21], [Nov-21], [Dec-21])
15 ) 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							
2	Cust-2	300	-300	NULL						
3	Cust-3	NULL	NULL	1	1	1	1	-1	-1	-1

Solving SQL Query | Rows to Column in SQL

F12 fx Cust-1

General Conditional Formatting as Table Cell Styles Insert Delete Format Sort & Filter Find & Select Analyse Data

INPUT			OUTPUT													
SalesDate	CustomerId	Amount	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
1-Jan-21	Cust-1	50\$	Cust-1	150\$	0\$	0\$	0\$	0\$	0\$	0\$	0\$	0\$	0\$	0\$	150\$	
2-Jan-21	Cust-1	50\$	Cust-2	300\$	(300)\$	0\$	0\$	0\$	0\$	0\$	0\$	0\$	0\$	0\$	0\$	
3-Jan-21	Cust-1	50\$	Cust-3	0\$	0\$	1\$	1\$	1\$	1\$	(1)\$	(1)\$	(1)\$	(1)\$	(1)\$	(2)\$	
1-Jan-21	Cust-2	100\$	Total	450\$	(300)\$	1\$	1\$	1\$	1\$	(1)\$	(1)\$	(1)\$	(1)\$	(1)\$	(1)\$	
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\$														

* All negative values should be represented with ()

13:47 / 1:13:10 • Solving SQL Query in Microsoft SQL Server using PIVOT >

Sheet1 HD

```
▶ Run □ Cancel | ⌘ Disconnect ⌘ Change Connection demo ▾ | ⚒ Explain ⌘ Enable SQLCMD ↗ Export as Notebook  
14     ) as pivot_table  
15 UNION  
16  
17 select *  
18 from  
19 (        
20     select 'Total' as customer  
21     , format(sales_date, 'MMM-yy') as sales_date  
22     , cast(replace(amount, '$', '') as int) as amount  
23     from sales_data  
24 ) as sales_data  
25 pivot  
26 (        
27     sum(amount)  
28     for sales_date in ([Jan-21], [Feb-21], [Mar-21], [Apr-21]  
29             , [May-21], [Jun-21], [Jul-21], [Aug-21]  
30             , [Sep-21], [Oct-21], [Nov-21], [Dec-21])  
31 ) as pivot_table  
32 ) 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
14      ) as pivot_table
15  UNION
16 select *
17 from (
18   (
19     select 'Total' as customer
20     , format(sales_date, 'MMM-yy') as sales_date
21     , cast(replace(amount, '$', '') as int) as amount
22     from sales_data
23   ) as sales_data
24 ) as pivot
25 (
26   sum(amount)
27   for sales_date in ([Jan-21], [Feb-21], [Mar-21], [Apr-21]
28                      ,[May-21], [Jun-21], [Jul-21], [Aug-21]
29                      ,[Sep-21], [Oct-21], [Nov-21], [Dec-21])
30 )
31 ) 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							
2	Cust-2	300	-300	NULL						
3	Cust-3	NULL	NULL	1	1	1	1	-1	-1	-1
4	Total	Results grid	-300	1	1	1	1	-1	-1	-1

Run Cancel Disconnect Change Connection demo Explain Enable SQLCMD Export as Notebook

Solving SQL Query | Rows to Column in SQL [Sep-21], [Oct-21], [Nov-21], [Dec-21])

```
32      ) as pivot_table ),
33  final_data AS
34    (select customer
35     , coalesce([Jan-21], 0) as Jan_21
36     , coalesce([Feb-21], 0) as Feb_21
37     , coalesce([Mar-21], 0) as Mar_21
38     , coalesce([Apr-21], 0) as Apr_21
39     , coalesce([May-21], 0) as May_21
40     , coalesce([Jun-21], 0) as Jun_21
41     , coalesce([Jul-21], 0) as Jul_21
42     , coalesce([Aug-21], 0) as Aug_21
43     , coalesce([Sep-21], 0) as Sep_21
44     , coalesce([Oct-21], 0) as Oct_21
45     , coalesce([Nov-21], 0) as Nov_21
46     , coalesce([Dec-21], 0) as Dec_21
47   from pivot_data)
48 select *
49 from final_data;
```

Results Messages

1	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

20:03 / 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

```
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

```
(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

1	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 >

Run Cancel | ⚙ Disconnect ⚙ Change Connection demo | ⚙ Explain ⚙ Enable SQLCMD ↗ Export as Notebook

```

46     , coalesce([Dec-21], 0) as Dec_21
47     from pivot_data)
48 select customer
49 , case when Jan_21 < 0 then concat('(', Jan_21 * -1, ')$') else concat(Jan_21, '$') end as "Jan-21"
50 , case when Feb_21 < 0 then concat('(', Feb_21 * -1, ')$') else concat(Feb_21, '$') end as "Feb-21"
51 , case when Mar_21 < 0 then concat('(', Mar_21 * -1, ')$') else concat(Mar_21, '$') end as "Mar-21"
52 , case when Apr_21 < 0 then concat('(', Apr_21 * -1, ')$') else concat(Apr_21, '$') end as "Apr-21"
53 , case when May_21 < 0 then concat('(', May_21 * -1, ')$') else concat(May_21, '$') end as "May-21"
54 , case when Jun_21 < 0 then concat('(', Jun_21 * -1, ')$') else concat(Jun_21, '$') end as "Jun-21"
55 , case when Jul_21 < 0 then concat('(', Jul_21 * -1, ')$') else concat(Jul_21, '$') end as "Jul-21"
56 , case when Aug_21 < 0 then concat('(', Aug_21 * -1, ')$') else concat(Aug_21, '$') end as "Aug-21"
57 , case when Sep_21 < 0 then concat('(', Sep_21 * -1, ')$') else concat(Sep_21, '$') end as "Sep-21"
58 , case when Oct_21 < 0 then concat('(', Oct_21 * -1, ')$') else concat(Oct_21, '$') end as "Oct-21"
59 , case when Nov_21 < 0 then concat('(', Nov_21 * -1, ')$') else concat(Nov_21, '$') end as "Nov-21"
60 , case when Dec_21 < 0 then concat('(', Dec_21 * -1, ')$') else concat(Dec_21, '$') end as "Dec-21"
61 from final_data;

```

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\$	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

```
1 select * from sales_data;  
2  
3 selec|
```

Data Output Explain Messages Notifications

CREATE EXTENSION

Query returned successfully in 83 msec.



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

```
1 select * from sales_data;  
2  
3 create extension htablefunc;
```

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 >

✓ Query in need HD 8 Subscribers



Solving SQL Query | Rows to Column in SQL

```
1 select customer_id as customer
2 , to_char(sales_date, 'Mon-YY') as sales_date
3 , cast(replace(amount, '$', '') as int) as amount
4 from sales_data;
5
6
7 select *
8 from crosstab('select customer_id as customer
9                 , to_char(sales_date, ''Mon-YY'') as sales_date
10                , cast(replace(amount, '''$', '') as int) as amount
11              from sales_data')
12      as (customer varchar, Jan_21 int, Feb_21 int, Mar_21 int, Apr_21 int, May_21 int, Jun_21 int
13           , Jul_21 int, Aug_21 int, Sep_21 int, Oct_21 int, Nov_21 int, Dec_21 int)
```

Data Output Explain Messages Notifications

	customer	sales_date	amount
1	Cust-2	Feb-21	-100
2	Cust-2	Feb-21	-100
3	Cust-3	Mar-21	1
4	Cust-3	Apr-21	1
5	Cust-3	May-21	1
6	Cust-3	Jun-21	1
7	Cust-3	Jul-21	-1
8	Cust-3	Aug-21	-1
9	Cust-3	Sep-21	-1
10	Cust-3	Oct-21	-1
11	Cust-3	Nov-21	-1
12	Cust-3	Dec-21	-1

37:42 / 1:13:10 • Solving SQL Query in PostgreSQL using CROSSTAB >

✓ Successfully run. Total query (1) 6 sec. 1 HD 8 embeds.

Solving SQL Query | Rows to Column in SQL

```
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 bigint
22            , Jul_21 bigint, Aug_21 bigint, Sep_21 bigint, Oct_21 bigint, Nov_21 bigint, Dec_21 bigint)
23
24
25
```

	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
1	Cust-1	150	[null]										
2	Cust-2	300	-300	[null]									
3	Cust-3	-1	-1	1	1	1	-1	-1	[null]	[null]	[null]	[null]	[null]

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

```
1 Select customer_id as customer
2 , date_format(sales_date, '%b-%y') as sales_date
3 , replace(amount, '$', '') as amount
4 from sales_data;
5
```

Result Grid

customer sales_date amount

customer	sales_date	amount
Cust-1	Aug-21	-1
Cust-3	Sep-21	-1
Cust-3	Oct-21	-1
Cust-3	Nov-21	-1
Cust-3	Dec-21	-1

Action Output

57:05 / 1:13:10 Solving SQL Query in MySQL using CASE statement >

Query Completed

```
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 |
```

100% 57 |

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
Cust-2	.Jan-21	100

Result 2.

Action Output

Time	Action	Response	Duration / Fetch Ti...
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

```

1 • with base_query as
2     (select customer_id as customer
3      , date_format(sales_date, '%b-21') as sales_date
4      , replace(amount, '$', '') as amount
5     from sales_data)
6 select customer
7   , case when sales_date = 'Jan-21' then amount else 0 end as Jan_21
8   , case when sales_date = 'Feb-21' then amount else 0 end as Feb_21
9   , case when sales_date = 'Mar-21' then amount else 0 end as Mar_21
10  , case when sales_date = 'Apr-21' then amount else 0 end as Apr_21
11  , case when sales_date = 'May-21' then amount else 0 end as May_21
12  , case when sales_date = 'Jun-21' then amount else 0 end as Jun_21
13  , case when sales_date = 'Jul-21' then amount else 0 end as Jul_21
14  , case when sales_date = 'Aug-21' then amount else 0 end as Aug_21
15  , case when sales_date = 'Sep-21' then amount else 0 end as Sep_21
16  , case when sales_date = 'Oct-21' then amount else 0 end as Oct_21
17  , case when sales_date = 'Nov-21' then amount else 0 end as Nov_21
18  , 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
Cust-2	100	0	0	0	0	0	0	0	0	0	0	0

Result 5

Action Output

Time	Action	Response	Duration / Fetch Ti...
6 13:42:52	with base_query as (select customer_id...	19 row(s) returned	0.00058 sec / 0.00

Query Completed

Read Only

Solving SQL Query | Rows to Column in SQL

```
6  select customer
7 , sum(case when sales_date = 'Jan-21' then amount else 0 end) as Jan_21
8 , sum(case when sales_date = 'Feb-21' then amount else 0 end) as Feb_21
9 , sum(case when sales_date = 'Mar-21' then amount else 0 end) as Mar_21
10 , sum(case when sales_date = 'Apr-21' then amount else 0 end) as Apr_21
11 , sum(case when sales_date = 'May-21' then amount else 0 end) as May_21
12 , sum(case when sales_date = 'Jun-21' then amount else 0 end) as Jun_21
13 , sum(case when sales_date = 'Jul-21' then amount else 0 end) as Jul_21
14 , sum(case when sales_date = 'Aug-21' then amount else 0 end) as Aug_21
15 , sum(case when sales_date = 'Sep-21' then amount else 0 end) as Sep_21
16 , sum(case when sales_date = 'Oct-21' then amount else 0 end) as Oct_21
17 , sum(case when sales_date = 'Nov-21' then amount else 0 end) as Nov_21
18 , sum(case when sales_date = 'Dec-21' then amount else 0 end) as Dec_21
19 from base_query
20 group by customer;
21
```

100% 50:8

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	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

Action Output

Time Action Response Duration / Fetch Time

13:47 1:05:56 / 1:13:10 • Solving SQL Query in MySQL using CASE statement > 3081 HD 00:00

Query Completed

Solving SQL Query | Rows to Column in SQL

```

20 group by customer
21 UNION
22 select 'Total' customer
23 , sum(case when sales_date = 'Jan-21' then amount else 0 end) as Jan_21
24 , sum(case when sales_date = 'Feb-21' then amount else 0 end) as Feb_21
25 , sum(case when sales_date = 'Mar-21' then amount else 0 end) as Mar_21
26 , sum(case when sales_date = 'Apr-21' then amount else 0 end) as Apr_21
27 , sum(case when sales_date = 'May-21' then amount else 0 end) as May_21
28 , sum(case when sales_date = 'Jun-21' then amount else 0 end) as Jun_21
29 , sum(case when sales_date = 'Jul-21' then amount else 0 end) as Jul_21
30 , sum(case when sales_date = 'Aug-21' then amount else 0 end) as Aug_21
31 , sum(case when sales_date = 'Sep-21' then amount else 0 end) as Sep_21
32 , sum(case when sales_date = 'Oct-21' then amount else 0 end) as Oct_21
33 , sum(case when sales_date = 'Nov-21' then amount else 0 end) as Nov_21
34 , sum(case when sales_date = 'Dec-21' then amount else 0 end) as Dec_21
35 from base_query;
36

```

100% 16:35

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	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

Action Output

Time Action Response Duration / Fetch Time

13:47 1:06:50 / 1:13:10 • Solving SQL Query in MySQL using CASE statement >

Query Completed

Limit to 1000 rows

```

1 • with base_query as
2     (select customer_id as customer
3      , date_format(sales_date, '%b-21') as sales_date
4      , replace(amount, '$', '') as amount
5     from sales_data)
6 select customer
7   , sum(case when sales_date = 'Jan-21' then amount else 0 end) as Jan_21
8   , sum(case when sales_date = 'Feb-21' then amount else 0 end) as Feb_21
9   , sum(case when sales_date = 'Mar-21' then amount else 0 end) as Mar_21
10  , sum(case when sales_date = 'Apr-21' then amount else 0 end) as Apr_21
11  , sum(case when sales_date = 'May-21' then amount else 0 end) as May_21
12  , sum(case when sales_date = 'Jun-21' then amount else 0 end) as Jun_21
13  , sum(case when sales_date = 'Jul-21' then amount else 0 end) as Jul_21
14  , sum(case when sales_date = 'Aug-21' then amount else 0 end) as Aug_21
15  , sum(case when sales_date = 'Sep-21' then amount else 0 end) as Sep_21
16  , sum(case when sales_date = 'Oct-21' then amount else 0 end) as Oct_21
17  , sum(case when sales_date = 'Nov-21' then amount else 0 end) as Nov_21
18  , sum(case when sales_date = 'Dec-21' then amount else 0 end) as Dec_21

```

100% 1:36

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	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

Action Output

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

Query Completed

Read Only

Result Grid Form Editor

5 from sales_data)

6 select customer

7 , sum(case when sales_date = 'Jan-21' then amount else 0 end) as Jan_21

8 , sum(case when sales_date = 'Feb-21' then amount else 0 end) as Feb_21

9 , sum(case when sales_date = 'Mar-21' then amount else 0 end) as Mar_21

10 , sum(case when sales_date = 'Apr-21' then amount else 0 end) as Apr_21

11 , sum(case when sales_date = 'May-21' then amount else 0 end) as May_21

12 , sum(case when sales_date = 'Jun-21' then amount else 0 end) as Jun_21

13 , sum(case when sales_date = 'Jul-21' then amount else 0 end) as Jul_21

14 , sum(case when sales_date = 'Aug-21' then amount else 0 end) as Aug_21

15 , sum(case when sales_date = 'Sep-21' then amount else 0 end) as Sep_21

16 , sum(case when sales_date = 'Oct-21' then amount else 0 end) as Oct_21

17 , sum(case when sales_date = 'Nov-21' then amount else 0 end) as Nov_21

18 , sum(case when sales_date = 'Dec-21' then amount else 0 end) as Dec_21

19 , sum(amount) as Total

20 from base_query

21 group by customer

22 UNION

23

100% 15:19 |

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	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

Action Output

Time	Action	Response	Duration / Fetch Time
8 13:49:05	with base_query as (select customer,...	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**.



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.



Stored Procedure vs Trigger

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



Stored Procedure vs Trigger

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



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



Types of Trigger

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



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
```



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



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
```



Stored Procedure in sql



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;
```



Stored Procedure without parameter Syntax



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

```
EXEC procedure_name
```



SQLQuery1.sql - DESKTOP-6SVT31C\IGNEOUS.sparta (sa (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Execute Debugger

sparta

Object Explorer

SQLQuery1.sql - DES...US.sparta (sa (53))

```
create procedure employee_age
as
select e_Age from employee
go
exec employee
```

stored procedure sparta.dbo.employee_age

Results Messages

	e_Age
1	45
2	21
3	25
4	30
5	33
6	27

Query executed successfully.

DESKTOP-6SVT31C\IGNEOUS (14...) sa (53) sparta 00:00:00 6 rows

Ready

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

SQLQuery1.sql - DESKTOP-6SVT31C\IGNEOUS.sparta (sa (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Generic Debugger

sparta Execute Debug

Object Explorer

Connect Connect to Database

DESKTOP-6SVT31C\IGNEOUS (SQL Server) Databases happy sparta Tables Views External Resources Synonyms Programmability Service Broker Storage Security Server Objects Replication

SQLQuery1.sql - DES...US.sparta (sa (53))

```
create procedure employee_age
as
select e_Age from employee
go

exec employee_age

create procedure employee_details
as
select * from employee
go

exec employee_details
```

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.

Ln 13 Col 1 Ch 1

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

SQLQuery1.sql - DESKTOP-6SVT31C\IGNEOUS (sa (53)) - Microsoft SQL Server Management Studio

Exception Handling in SQL | SQL Stored Procedures | Views in SQL | Intellipaat

New Query Execute Debug Generic Debugger

sparta

Object Explorer

SQLQuery1.sql - DESKTOP-6SVT31C\IGNEOUS (sa (53))

```
create procedure employee_gender @gender varchar(20)
as
begin
    select * from employee
    where e_gender=@gender
end
```

parameter @gender varchar(20)

Results

	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

Messages

Query executed successfully.

DESKTOP-6SVT31C\IGNEOUS (sa (53)) sparta 00:00:00

Ln 4 Col 16 Ch 16 IN5

Inquire about our live classes? Call Us: IN:+91-702374614 / US: 1-800-216-8930 or www.intellipaat.com

SQLQuery1.sql - DESKTOP-6SVT31C\IGNEOUS.sparta (sa (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Generic Debugger

sparta Execute Debug ✓

Object Explorer

Connect ▾

DESKTOP-6SVT31C\IGNEOUS (SQL Server) ▾

- Databases
 - System Databases
 - Database Snapshots
 - happy
 - sparta
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.department
 - dbo.employee_target
 - dbo.employee1
 - dbo.student_details1
 - dbo.student_details2
 - dbo.student_details1
 - dbo.student_details2
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Service Broker
 - Storage
 - Security
- Security
- Server Objects
- Replication

SQLQuery1.sql - DESKTOP-6SVT31C\IGNEOUS (sa (53))

```
create procedure employee_gender @gender varchar(20)
as
begin
select * from employee
where e_gender=@gender
go
exec employee_gender @gender='Male'
```

Results Messages

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Matt	159000	33	Male	Sales
4	Jeff	112000	27	Male	Operations

Query executed successfully.

DESKTOP-6SVT31C\IGNEOUS (14...) | sa (53) | sparta | 00:00:00 | 4 rows

Ready Ln 7 Col 36 Ch 36

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

Exception Handling



An error condition during a program execution is called an exception



Exception

The mechanism for resolving such an exception is exception handling



Exception Handling

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



Try/Catch

SQL Provides the try/catch blocks for exception handling



Try/Catch Syntax



```
BEGIN TRY  
SQL Statements  
END TRY  
  
BEGIN CATCH  
  
- Print Error OR  
- Rollback Transaction  
  
END CATCH
```



SQLQuery1.sql - DESKTOP-6SVT31C\IGNEOUS.sparta (sa (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Execute Debugger

sparta

Object Explorer

Connect

DESKTOP-6SVT31C\IGNEOUS (SQL Server)

- Databases
 - System Databases
 - Database Snapshots
 - happy
 - sparta
- Security
- Server Objects
- Replication
- PolyBase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

SQLQuery1.sql - DES...US.sparta (sa (53))*

```
declare @val1 int;
declare @val2 int;

begin try
set @val1=8;
set @val2=@val1/0;
end try

begin catch
print error.message();
end catch
```

100 %

Messages

Divide by zero error encountered.

Query executed successfully.

DESKTOP-6SVT31C\IGNEOUS (14...) sa (53) sparta 00:00:00 0 rows

Ln 1 Col 1 Ch 1

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

SQLQuery1.sql - DESKTOP-6SVT31C\IGNEOUS.sparta (sa (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Generic Debugger

sparta Execute Debug

Object Explorer

Connect DESKTOP-6SVT31C\IGNEOUS (SQL Server)

- Databases
 - System Databases
 - Database Snapshots
 - happy
 - sparta
- Security
- Server Objects
- Replication
- PolyBase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

SQLQuery1.sql - DES..US.sparta (sa (53))

```
begin try
    select e_salary+e_name from employee
end try
begin catch
    print 'Cannot add a numerical value with a string value'
end catch
go
```

100 %

Results Messages

(0 rows affected)
Cannot add a numerical value with a string value

Query executed successfully.

DESKTOP-6SVT31C\IGNEOUS (14...) sa (53) sparta 00:00:00 0 rows

Matches: begin try

Ln 4 Col 1 Ch 1

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

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- Filter objects
- logistics
 - Tables
 - branch
 - clients
 - employee
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - student
 - Views
 - Stored Procedures
 - Functions
- sys

query2

```
1 * select * from branch;
2
3 DELIMITER //
4 * CREATE PROCEDURE getbranches()
5 BEGIN
6   SELECT * FROM branch;
7 END//
```

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

Administration Schemas

Information

Table: employee

Columns:

- emp_id int AI PK
- ename varchar(30)
- job_desc varchar(20)
- salary int
- branch_id int

Action Output

Time	Action	Message	Duration / Fetch
1 11:24:47	selected * from branch LIMIT 0, 1000	4 rows returned	0.000 sec / 0.000 sec

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- logistics
- Tables
- branch
- clients
- employee
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
- student
- Views
- Stored Procedures
- Functions

sys

query2

DELIMITER \$\$

CREATE PROCEDURE getbranches()

BEGIN

SELECT * FROM branch;

END\$\$

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

Administration Schemas

Information

Table: employee

Columns:

- emp_id int AI PK
- ename varchar(30)
- job_desc varchar(20)
- salary int
- branch_id int

Action Output

Time	Action	Message	Duration / Fetch
11:24:47	selected * from branch LIMIT 0,1000	4 rows returned	0.000 sec / 0.000 sec

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- loginst
- Tables
- branch
- clients
- employee
- student
- Views
- Stored Procedures
- getbranches
- Functions

sys

query2

```
5 * CREATE PROCEDURE getbranches()
6 BEGIN
7     SELECT * FROM branch;
8 END$$
9
10 DELIMITER ;
11
12 * CALL getbranches();
```

Result Grid | Filter Rows | Export | 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

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result 1 x

Output

Action / Output

Time	Action	Message	Duration / Fetch
1 11:24:49	CREATE PROCEDURE getbranches() BEGIN SELECT * FROM branch; END	Error Code: 1046 No database selected. Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.	0.015 sec
2 11:25:15	USE loginst;	0 rows affected	0.000 sec
3 11:25:15	CREATE PROCEDURE getbranches() BEGIN SELECT * FROM branch; END	0 rows affected	0.016 sec
4 11:26:17	CALL getbranches()	4 rows returned	0.000 sec / 0.000 sec

Object Info Session

Advantages

- Reduce Network Traffic
- Centralize Business Logic
- Secure

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator query2

Limit to 1000 rows

SQL Editor

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

SCHEMAS

Filter objects

logifirst

- Tables
 - branch
 - clients
 - employee
 - student
- Views
- Stored Procedures
- Functions

sys

1 * select * from branch;

2

3 * USE logifirst;

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;|

Administration Schemas

Information

No object selected

Action Output

Time	Action	Message	Duration / Fetch
1 09:54:09	USE logifirst	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 logifirst.getbranches does not exist.	0.000 sec

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

logicfirst

- Tables branch, clients, employee, student
- Views
- Stored Procedures
- Functions

sys

query2

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

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 Output

No object selected

Action Output Time Action Message Duration / Fetch

Object Info Session

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil

The screenshot shows the MySQL Workbench interface with a query editor window. The code in the editor is:

```
20  DELIMITER $$  
21 * CREATE PROCEDURE employeecnt()  
22  BEGIN  
23      DECLARE total INT DEFAULT 0;  
24      SELECT COUNT(emp_id)  
25      INTO total  
26      FROM employee; ||  
27      SELECT total;
```

The result grid shows the output of the last SELECT statement:

COUNT(emp_id)
15

The information panel at the bottom shows the execution log:

Action	Time	Message	Duration / Fetch
1	10:57:05	SELECT * FROM employee LIMIT 0, 1000	0.000 sec / 0.000 sec
2	10:58:13	SELECT COUNT(emp_id) FROM employee LIMIT 0, 1000	0.000 sec / 0.000 sec

The status bar at the bottom indicates the session is active.

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Local instance MySQL80

query

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

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');

Information

No object selected

Action Output

Time Action

Message

Duration / Fetch

18:47 / 1:26:45 • Passing parameters IN, OUT and INOUT > ▾

HD

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil

The screenshot shows the MySQL Workbench interface with a query editor window. The code in the editor is:

```
25    INTO total
26    FROM employee;
27    SELECT total;
28  END$$
29  DELIMITER ;
30
31 • CALL employeecnt('ADMIN');
32
33 • /*
34  parameters
35
36  IN      I
37  OUT
38  INOUT|  */
39 */
```

The code includes several syntax errors, notably the use of `•` instead of `DELIMITER` and `/*` instead of `*/` for comments. The `INOUT` parameter type is highlighted with a cursor.

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

Localhost

- branch
- clients
- employee
 - Columns
 - emp_id
 - ename
 - job_desc
 - salary
 - branch_id
- Indexes
- Foreign Keys
- Triggers
- student
- Views
- Stored Procedures
- Functions

Information

No object selected

Output

Action Output

Time Action

Message

Duration / Rech

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
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');
```

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator query?

Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

SQL Editor

```
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  parameters
39
40  IN
41  OUT
42  INOUT  */
43
44
45
```

Administration Schemas Information

No object selected

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	0.016 sec
2	10:34:31	CALL employeecnt(ADMIN)	1 row(s) affected	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

Object Info Session

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Local instance MySQL80, Filter objects, Schemas (logfirst), Tables, Views, Stored Procedures, Functions, sys.
- Query Editor:** A large text area containing the SQL code for creating a stored procedure. The code is as follows:

```
20 DELIMITER $$  
21 CREATE PROCEDURE employeecnt(  
22     IN jdesc VARCHAR(10),  
23     OUT total INT  
24 )  
25 BEGIN  
26     SELECT COUNT(emp_id)  
27     INTO total  
28     FROM employee  
29     WHERE job_desc=jdesc;  
30 END$$  
31 DELIMITER ;  
32  
33 SET @total = 10;  
34 CALL employeecnt('ADMIN',@total);  
35 SELECT @total;  
36 CALL employeecnt('CEO');  
37  
38 /*
```

- SQL Additions:** A note stating "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Output:** A table showing the execution log with columns: #, Time, Action, Message, Duration / Epoch. The log entries are:

#	Time	Action	Message	Duration / Epoch
1	10:34:27	CREATE PROCEDURE employeecnt(IN jdesc VARCHAR(10)) BEGIN	DECLARE total INT DEFAULT 0; SELECT COUNT(emp_id) INTO total	0.016 sec
2	10:34:31	CALL employeecnt('ADMIN')	1 rows affected	0.000 sec / 0.000 sec
3	10:34:57	CALL employeecnt('MANAGER')	1 rows returned	0.000 sec / 0.000 sec
4	10:35:16	CALL employeecnt('CEO')	1 rows returned	0.000 sec / 0.000 sec
5	10:52:38	DROP PROCEDURE logfirst.employeecnt;	0 rows affected	0.016 sec

- Bottom Status:** 26:18 / 1:26:45 • Passing parameters IN, OUT and INOUT > ▶ HD

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

SCHEMAS: logicfirst

Tables: branch, clients, employee, logins, student, sys

Columns: emp_id, fname, lname, job_desc, salary, branch_id

Indexes: primary, foreignkeys

Triggers: Views: Stored Procedures: Functions: sys

query? Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

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 | Filter Rows | Export | Wrap Cell Content | Result Grid | Form Editor | Field Types

@total
3

Administration Schemas Information

No object selected

Result 13 x

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 employee WHERE job_desc=jdesc; END;	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

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- logistics
 - Tables
 - branch
 - clients
 - employee
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
- sys

Administration Schemas

No object selected

query2

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

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

SQL Additions

Context Help Snippets

Output

Action	Time	Action	Message	Duration / Fetch
7	10:52:51	SET @total = 10;	0 rows(s) affected	0.000 sec
8	10:52:51	CALL employee('ADMIN', @total)	1 row(s) affected	0.000 sec / 0.000 sec
9	10:52:51	SELECT @total LIMIT 0,1000	1 row(s) returned	
10	10:53:52	CALL employee('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 rows(s) affected	0.000 sec / 0.000 sec

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator: query

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

SCHEMAS

- logistics
 - Tables
 - branch
 - clients
 - employee
 - Columns
 - emp_id
 - ename
 - job_desc
 - salary
 - branch_id
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - sys

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 | Read Only | Context Help | Snippets

@counter
7

Administration Schemas Information

No object selected

Result 15 x

Action Output

#	Time	Action	Message	Duration / Fetch
10	10:53:52	CALL employee(CEO, @total)	1 row(s) affected	0.000 sec / 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

Object Info Session

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil

The screenshot shows the MySQL Workbench interface with a SQL editor window. The code in the editor is:

```
5
6  DELIMITER $$;
7  CREATE PROCEDURE GetState(
8      IN id INT
9  )
10 BEGIN
11 IF id=1 OR id=2 THEN
12     SELECT 'Tamil Nadu';
13 ELSEIF id=3 THEN
14     SELECT 'Maharashtra';
15 ELSE
16     SELECT 'Andra Pradesh';
17 END IF;
```

The code defines a stored procedure named `GetState` that takes an integer parameter `id`. It uses an IF-THEN-ELSEIF-ELSE structure to return different state names based on the value of `id`. The code is syntax-highlighted, with keywords in blue and identifiers in black.

MySQL Workbench

Local instance MySQL80 (loginst)
Local instance MySQL80 (loginst)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

logicfirst

- Tables
- Views
- Stored Procedures
 - employeeact
 - IncrCounter
- Functions

aya

SQL File

```
15 ELSE
16     SELECT 'Andra Pradesh';
17 END IF;
18 END$$
19 DELIMITER ;
20
21 • CALL GetState(1);
22
```

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 | Export | Wrap Cell Content: 15

Tamil Nadu
▶ Tamil Nadu

Result Grid | Form Editor

Administration Schemas Information

No object selected

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'; ELSE...;	Rows(s) affected	0.015 sec
2 10:02:28	use logicfirst	0 rows affected	0.000 sec
3 10:02:28	select * from branch LIMIT 0, 1000	4 rows 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'; ELSE...;	Error Code: 1304 PROCEDURE GetState already exists	0.000 sec
5 10:02:48	CALL GetState()	1 rows(s) returned	0.000 sec / 0.000 sec

Object Info Session

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil

MySQL Workbench

Local instance MySQL80 · X

File Edit View Query Database Server Tools Scripting Help

Navigator · Schemas · Tables · Views · Stored Procedures · Functions · sys

Filter objects

SCHEMAS

logistics

Tables

Views

Stored Procedures

Functions

sys

File loops Functions

SQL Editor · Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
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;
```

Result Grid · Filter Rows · Edit: Export/Import: Wrap Cell Content: Result Form Editor Field Types

branch_id	br_name	addr
1	Chennai	16 ABC Road
2	Coimbatore	120 15th Block
3	Mumbai	25 XYZ Road
4	Hyderabad	32 10th Street

Administration · Schemas · Information

No object selected

Action Output

Time	Action	Message	Duration / Fetch
1 09:35:33	use logistics	0 rows/affected	0.000 sec
2 09:35:33	select * from branch LIMIT 0, 1000	4 rows/returned	0.000 sec / 0.000 sec
3 09:35:33	select * from employee LIMIT 0, 1000	15 rows/returned	0.000 sec / 0.000 sec
4 09:36:07	select e.empno, branch_id from employee LIMIT 0, 1000	15 rows/returned	0.000 sec / 0.000 sec
5 09:36:22	select * from branch LIMIT 0, 1000	4 rows/returned	0.000 sec / 0.000 sec

Object Info · Session 50:48 / 1:26:45 · User defined functions in stored procedure > ▶ HD

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil

MySQL Workbench

Local instance MySQL80 · X

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80 · X

SCHEMAS: logicfirst

Filter objects

Tables

Views

Stored Procedures

Functions

sys

File loops Functions

SQL Editor: Local instance MySQL80 · X

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

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: Result Form Editor Field Types

branch_id	br_name	addr
1	Chennai	16 ABC Road
2	Coimbatore	120 15th Block
3	Mumbai	25 XYZ Road
4	Hyderabad	32 10th Street

Administration Schemas Information

No object selected

branch 23 · X

Action Output

Time	Action	Message	Duration / Fetch
1 09:35:33	use logicfirst	0 rows/affected	0.000 sec
2 09:35:33	select * from branch LIMIT 0, 1000	4 rows/returned	0.000 sec / 0.000 sec
3 09:35:33	select * from employee LIMIT 0, 1000	15 rows/returned	0.000 sec / 0.000 sec
4 09:36:07	select empno,branch_id from employee LIMIT 0, 1000	15 rows/returned	0.000 sec / 0.000 sec
5 09:36:22	select * from branch LIMIT 0, 1000	4 rows/returned	0.000 sec / 0.000 sec

50:46 / 1:26:45 • User defined functions in stored procedure > ▶

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

logistics

Tables Views Stored Procedures Functions sys

SQL Editor

Limit to 1000 rows

declare fulladdr varchar(50);
select concat(addr, ' ', br_name) into fulladdr
from branch
where branch_id=id;
end \$\$
delimiter ;

Result Grid

branch_id	br_name	addr
1	Chennai	16 ABC Road
2	Coimbatore	120 15th Block
3	Mumbai	25 XYZ Road
4	Hyderabad	32 10th Street

Administration Schemas Information

No object selected

branch 23 X

Action Output

Time	Action	Message	Duration / Fetch
1 09:35:33	use logistics	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 e.name,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

Object Info Session

MySQL Workbench

Local instance MySQL00 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

loginstat

Tables

Views

Stored Procedures

Functions

sys

SQL Editor

Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
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
22
```

Administration Schemas Information

No object selected

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	Hyderabad	32 10th Street

branch 23 x

Action Output

Time	Action	Message	Duration / Fetch
1	09:35:33 use loginstat	0 rows affected	0.000 sec
2	09:35:33 select * from branch LIMIT 0,1000	4 rows returned	0.000 sec / 0.000 sec
3	09:35:33 select * from employee LIMIT 0,1000	16 rows returned	0.000 sec / 0.000 sec

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

logistics

- Tables
- Views
- Stored Procedures
- Functions

sys

SQL Editor

Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
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 ;
```

No object selected

Result Grid

branch_id	br_name	addr
1	Chennai	16 ABC Road
2	Coimbatore	120 15th Block
3	Mumbai	25 XYZ Road
4	Hyderabad	32 10th Street

Object Info Session

Action Output

Time	Action	Message	Duration / Fetch
1 09:35:33	use logistics	0 rows affected	0.000 sec
2 09:35:33	select * from branch LIMIT 0, 1000	4 rows returned	0.000 sec / 0.000 sec
3 09:35:33	select * from employee LIMIT 0, 1000	16 rows returned	0.000 sec / 0.000 sec

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

SCHEMAS

Filter objects

logistics

- Tables
- Views
- Stored Procedures
- Functions

sys

File loops Landings

Limit to 1000 rows

SQL Editor

```
declare fulladdr varchar(50);
select concat(addr, ' ', br_name) into fulladdr
from branch
where branch_id=id;
return fulladdr;
end $$
```

delimiter ;

Result Grid

ename	getBranchAddr(branch_id)
Ram	120 15th Block Coimbatore
Harini	120 15th Block Coimbatore
George	16 ABC Road Chennai

branch:23 Result:24 x

Administration Schemas Information

No object selected

Action Output

Time	Action	Message	Duration / Fetch
1 09:35:33	use loginstat	0 rows affected	0.000 sec / 0.000 sec
2 09:35:33	select * from branch LIMIT 0, 1000	4 rows returned	0.000 sec / 0.000 sec
3 09:35:33	select * from employee LIMIT 0, 1000	15 rows returned	0.000 sec / 0.000 sec
4 09:36:07	select ename,branch_id from employee LIMIT 0, 1000	15 rows returned	0.000 sec / 0.000 sec
5 09:36:22	select * from branch LIMIT 0, 1000	4 rows 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 from branch where branch_id=id; return fulladdr; end	0 rows affected	0.016 sec / 0.000 sec
7 10:06:31	select ename, getBranchAddr(branch_id) from employee LIMIT 0, 1000	15 rows returned	0.000 sec / 0.000 sec

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator: tables loops functions

SCHEMAS: logstash sys

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
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 Contents: | Read Only | Context Help | Snippets:

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
sys	version_minor	FUNCTION	mysql.sys@localhost	2022-02-04 10:54:10	2022-02-04 10:54:10
sys	version_patch	FUNCTION	mysql.sys@localhost	2022-02-04 10:54:10	2022-02-04 10:54:10

Output: Action Output | Read Only | Context Help | Snippets:

Action Output | Time Action | Message | Duration / Fetch

1 10:14:19 show function status 22 rows returned 0.000 sec / 0.000 sec

Object Info Session

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil

```

16      end;
17      if char_length(bname)<4 then
18          signal sqlstate '45000'
19          set MESSAGE_TEXT = 'branch name too short';
20      end if;
21      insert into branch
22      values (id,bname,addr);
23      select * from branch;
24  end$$
25 delimiter ;
26
27 • call insertBranch(4,'ma','xyz');
28
29
30
31

```

Output

Action	Time	Message	Duration / Fetch
drop procedure if exists insertBranch	11:31:23	0 rows(a) affected	0.016 sec
create procedure insertBranch(IN id int, IN bname varchar(30), IN addr varchar(50))	11:31:23	Begin declare continue handler for 1062 begin select concat('...',	0.015 sec
drop procedure if exists insertBranch	11:31:33	0 row(s) affected	0.000 sec
create procedure insertBranch(IN id int, IN bname varchar(30), IN addr varchar(50))	11:31:33	Begin declare continue handler for 1062 begin select concat('...',	0.000 sec
call insertBranch(4,'ma','xyz')	11:33:03	Error Code: 1644: branch name too short	0.000 sec

1:11:58 / 1:26:45 • Signal and Resignal >

Stored Procedure, Function and Cursor| SQL Intermediate Course | Logic First Tamil

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80 · Schemas: logicfirst · Tables: · Views: · Stored Procedures: employeeact, GetState, IncrCounter, LoopDemo, LoopDemo2, LoopDemo3 · Functions: getbranchAddr · sys

SQL Editor: SQL File 5 · loops

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
7 • create procedure insertBranch(
8     IN id int,
9     IN bname varchar(30),
10    In addr varchar(50)
11 )
12 Begin
13     declare continue handler for 1062
14     begin
15         select concat('duplicate key ', id, ' cannot be inserted into branch');
16     end;
17     if char_length(bname)<4 then
18         signal sqlstate '45000';
19         set MESSAGE_TEXT = 'branch name too short';
20     end if;
21     insert into branch
22     values (id,bname,addr);
23     select * from branch;
```

Output:

Action	Time	Action	Message	Duration / Fetch
drop procedure if exists insertBranch	1 11:31:23		0 row(s) affected	0.016 sec
create procedure insertBranch(IN id int, IN bname varchar(30), IN addr varchar(50))	2 11:31:23	Begin declare continue handler for 1062 begin select concat(...)	0 row(s) affected	0.015 sec
drop procedure if exists insertBranch	3 11:31:33		0 row(s) affected	0.000 sec
create procedure insertBranch(IN id int, IN bname varchar(30), IN addr varchar(50))	4 11:31:33	Begin declare continue handler for 1062 begin select concat(...)	0 row(s) affected	0.000 sec
call insertBranch(4,In0yz)	5 11:33:03		Error Code: 1644: branch name too short	0.000 sec

1:12:25 / 1:26:45 • Signal and Resignal >

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator SQL Editor loops

Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

SCHEMAS

- logicfirst
 - Tables
 - Views
 - Stored Procedures
 - employeeacct
 - GetState
 - IncrCounter
 - LoopDemo
 - LoopDemo2
 - LoopDemo3
 - Functions
 - RD_getBranchAddr
- sys

Administration Schemas Information

No object selected

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 rows(a) 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 rows(b) 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 rows(a) 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 rows(s) returned	- / 0.000 sec
10 11:44:47	drop procedure if exists insertBranch	0 rows(a) 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 begin select conc...	0 rows(a) affected	0.000 sec
12 11:44:47	call insertBranch(1,ma'xyz')	Error Code: 1062, duplicate key	0.000 sec

Object Info Session

```

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

```

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator SQL Editor loops

SCHEMAS Filter objects

logistics Tables Views

Stored Procedures

- employeeacct
- GetState
- IncrCounter
- LoopDemo
- LoopDemo2
- LoopDemo3

Functions

- getbranchAddr

sys

Administration Schemas Information

No object selected

SQL Editor

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');

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

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 rows 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

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator: SQL Editor: loops

SCHEMAS: logicfirst

Stored Procedures: insertBranch

Functions: getbranchAddr

sys

SQL Editor:

```
6 • 1 procedure if exists insertBranch;
7   miter $$;
8 • 2 create 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  • 3 insert into branch
19    values(id,bname,addr);
20
21  select * from branch;
22  $;
```

SQL Additions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Output:

Action	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 '' at line 1	0.000 sec
2	10:25:02	drop procedure if exists insertBranch()	0 rows(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 * from branch;	0 rows(s) affected	0.016 sec
4	10:25:02	call insertBranch(1,'Madras','20 KM Road')	Error Code: 1062. Duplicate entry '1' for key 'branch.PRIMARY'	0.000 sec

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

SCHEMAS: logicfirst

- Tables
- Views
- Stored Procedures
 - employeeacct
 - GetState
 - IncrCounter
 - LoopDemo
 - LoopDemo2
 - LoopDemo3
- Functions
 - getbranchAddr

SQL Editor: SQL File 57 loops

```
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 Editor Notes: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

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 * from branch;	0 row(s) affected	0.016 sec
4 10:25:02	call insertBranch(1,'Madras','20 KM Road')	Error Code: 1062. Duplicate entry '1' for key 'branch.PRIMARY'	0.000 sec

Object Info Session