1. What is the difference between Drop, Delete and Truncate statements in SQL Server?

.

DELETE:

1. The DELETE command is used to remove some or all rows from a table.
2. A WHERE clause can be used with a DELETE command to remove some specific rows from a table.
3. If the WHERE condition is not specified, then all rows are removed.
4. The DELETE operation will cause all DELETE triggers on the table to fire.
5. It does not reset the identity of the column value.
6. It removes rows on a row-by-row basis and hence for each deleted row it records an entry in the transaction logs, thus this is slower than truncate.
7. This is a DML command so it is just used to manipulate or modify the table data and it does not change any property of a table.

TRUNCATE:

1. TRUNCATE removes **all rows** from a table, but the table structure and its columns, constraints, indexes, and so on remain.
2. It does not require a **WHERE** clause, so we cannot filter rows while Truncating.
3. **IDENTITY** columns are re-seeded on this operation if no seed was defined then the default value 1 is used.
4. **No Triggers** are fired on this operation because it does not operate on individual rows.
5. TRUNCATE removes the data by deallocating the data pages used to store the table's data instead of rows and records, and only the page deallocations are recorded in the transaction log thus it is faster than delete.
6. We cannot use TRUNCATE TABLE on a table referenced by a FOREIGN KEY constraint.
7. TRUNCATE is DDL Command

DROP:

1. The DROP command removes a table from the database.
2. All the related Data, Indexes, Triggers, Constraints, and Permission specifications for the Table are dropped by this operation.
3. Some objects like Views, Stored Procedures that reference the dropped table are not dropped and must be explicitly dropped.
4. Cannot drop a table that is referenced by any Foreign Key constraint.
5. **No Triggers** are fired on this operation because it does not operate on individual rows.

Note: If TRUNCATE is written in Query Editor surrounded by TRANSACTION and if the session is closed, it cannot be rolled back but DELETE can be rolled back.

2. What is the difference between where clause and having clause in SQL Server?

This is one of the most frequently asked SQL Server Interview Questions and in almost all interviews this question being asked.

1. WHERE clause cannot be used with aggregate functions whereas the HAVING clause can be used with aggregate functions. This means the WHERE clause is

used for filtering individual rows on a table whereas the HAVING clause is used to filter groups.

2. WHERE comes before GROUP BY. This means the WHERE clause filters rows before aggregate calculations are performed. HAVING comes after GROUP BY. This means the HAVING clause filters rows after aggregate calculations are performed. So, from a performance standpoint, HAVING is slower than WHERE and should be avoided when possible.
3. WHERE and HAVING clause can be used together in a SELECT query. In this case WHERE clause is applied first to filter individual rows. The rows are then grouped and aggregate calculations are performed, and then the HAVING clause filters the groups.
4. WHERE clause can be used with – Select, Insert, and Update statements whereas the HAVING clause can only be used with the Select statement.

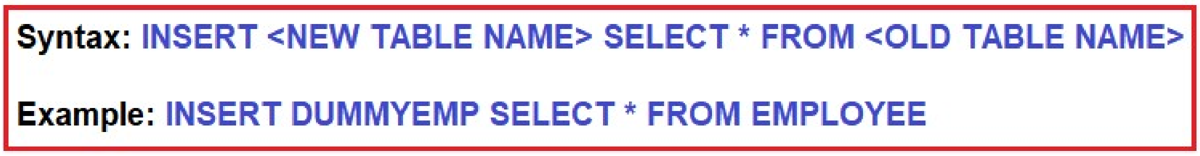
3. What are the differences between primary key and unique key in SQL Server?

This is of the most asked SQL Server Interview Questions in Interviews. Let discuss this question in detail.

1. A table can have only one primary key. On the other hand, a table can have more than one unique key.
2. The primary key column does not accept any null values whereas a unique key column accepts one null value.
3. Both Primary key and unique key enforce the uniqueness of the column on which they are defined. But By default, the primary key creates a unique clustered index on the column whereas a unique key creates a unique non clustered index.

4.How can we copy the data from one table to another?

This is one of the most frequently asked SQL Server Interview Questions and the interviewer basically asked to write the query. When we copy the data from one table to another table then the two tables should contain the same structure.



Syntax: INSERT <NEW TABLE NAME> SELECT * FROM <OLD TABLE NAME>

Example: INSERT DUMMYEMP SELECT * FROM EMPLOYEE

When we copy the data from one table to another table we use insert and select query. Tables always independent objects that mean a table does not depend on other tables

5. How to create a new table from an existing table or in how many ways we can create a new table from an existing table?

If required we can create a new table from an existing table as below.
Syntax1: (with all column from an existing table)
SELECT * INTO <NEW TABLE NAME> FROM <OLD TABLE NAME>
Example: SELECT * INTO NEWEMPLOYEE FROM EMPLOYEE
When we execute the above query it will create a new table with all records from an existing table.
Syntax2: (with specific columns from an existing table)
SELECT <REQUIREDCOLUMN> INTO <NEW TABLE NAME> FROM <OLD TABLE N AME>

Example: **SELECT EID, SALARY INTO SPECEMP FROM EMPLOYEE**
When we execute the above query it will create a new table with the specific column data from an existing table.
Syntax3: (creating a new table without data)
SELECT * INTO <NEW TABLE NAME> FROM <OLD TABLE NAME> WHERE 1 = 0
Example: SELECT * INTO DUMMYEMP FROM EMPLOYEE WHERE 1 = 0
OR
SELECT <REQUIRED COLUMNS> INTO <NEW TABLE NAME> FROM <OLD TABLE NAME>
SELECT EID, SALARY INTO TAB1 FROM EMPLOYEE WHERE 1 = 0
When we execute the above query it will create a new table without records from an existing table.

6.What is normalization?

Database normalization is a data design and organization process applied to data structures based on rules that help build relational databases. In relational database design the process of organizing data to minimize redundancy. Normalization usually involves dividing a database into two or more tables and defining relationships between the tables. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.

7.What are the different normalization forms?

This is one of the frequently asked SQL Server Interview Questions.
1NF: Eliminate Repeating Groups: Make a separate table for each set of related attributes, and give each table a primary key. Each field contains at most one value from its attribute domain.
2NF: Eliminate Redundant Data: If an attribute depends on only part of a multi-valued key, remove it to a separate table.
3NF: Eliminate Columns Not Dependent On Key: If attributes do not contribute to a description of the key, remove them to a separate table. All attributes must be directly dependent on the primary key
BCNF: Boyce-Codd Normal Form: If there are non-trivial dependencies between candidate key attributes, separate them out into distinct tables.
4NF: Isolate Independent Multiple Relationships: No table may contain two or more 1:n or n:m relationships that are not directly related.
5NF: Isolate Semantically Related Multiple Relationships: There may be practical constrains on information that justifies separating logically related many-to-many relationships.
ONF: Optimal Normal Form: A model limited to only simple (elemental) facts, as expressed in Object Role Model notation.
DKNF: Domain-Key Normal Form: A model free from all modification anomalies.
Remember, these normalization guidelines are cumulative. For a database to be in 3NF, it must first fulfill all the criteria of a 2NF and 1NF database. Please **click here** to learn **Database Normalization** in detail step by step with some real-time examples.

8.What is the Cursor?

A cursor is a database object used by applications to manipulate data in a set on a row-by-row basis, instead of the typical SQL commands that operate on all the rows in the set at one time.
In order to work with a cursor we need to perform some steps in the following order:

1. Declare cursor
2. Open cursor
3. Fetch row from the cursor Process fetched row Close cursor
4. Deallocate cursor

9.What is sub-query? Explain the properties of sub-query.

This is one of the frequently asked SQL Server Interview Questions. Sub-queries are often referred to as sub-selects, as they allow a SELECT statement to be executed arbitrarily within the body of another SQL statement. A sub-query is executed by enclosing it in a set of parentheses. Sub-queries are generally used to return a single row as an atomic value, though they may be used to compare values against multiple rows with the IN keyword.
A subquery is a SELECT statement that is nested within another T-SQL statement. A subquery SELECT statement if executed independently of the T-SQL statement, in which it is nested, will return a result set. Meaning a subquery SELECT statement can stand alone and is not dependent on the statement in which it is nested. A subquery SELECT statement can return any number of values and can be found in, the column list of a SELECT statement, a FROM, GROUP BY, HAVING, and/or ORDER BY clauses of a T-SQL statement. A Subquery can also be used as a parameter to a function call. Basically, a subquery can be used anywhere an expression can be used.

10.What is the SQL Profiler?

This is one of the most frequently asked SQL Server Interview Questions. SQL Profiler is a graphical tool that allows system administrators to monitor events in an instance of Microsoft SQL Server. You can capture and save data about each event to a file or SQL Server table to analyze later. For example, you can monitor a production environment to see which stored procedures are hampering performance by executing too slowly.
Use SQL Profiler to monitor only the events in which you are interested. If traces are becoming too large, you can filter them based on the information you want, so that only a subset of the event data is collected. Monitoring too many events adds overhead to the server and the monitoring process and can cause the trace file or trace table to grow very large, especially when the monitoring process takes place over a long period of time.

11.What are the different types of Constraints available in SQL Server?

SQL Server supports five constraints for maintaining data integrity which are

1. **UNIQUE KEY constraint**
2. **NOT NULL constraint**
3. **CHECK KEY constraint**
4. **PRIMARY KEY constraint**
5. **FOREIGN KEY constraint.**

12.What is the Primary Key Constraint in SQL Server?

1. PRIMARY KEY is the combination of UNIQUE and NOT NULL constraint which does not allow either NULL or Duplicate values into a column or columns.
2. Using the primary key we can enforce entity integrity.

3. PRIMARY KEY is also used to make a relationship with the FOREIGN KEY constraint on the table.
4. A table should contain 1 PRIMARY KEY constraint only which can be either on a single or multiple columns i.e. composite primary key also allowed.
5. Every TABLE should have a primary key constraint to uniquely identify each row and only one primary key constraint can be created for each table.
6. PRIMARY KEY can apply to any data type like integer, character, decimal, money, etc.

13.What is Foreign Key Constraint in SQL Server?

1. One of the most important concepts in a database is creating a relationship between database tables.
2. These relationships provide a mechanism for linking data stores in multiple tables and retrieving them in an efficient manner.
3. In order to create a link between two tables, we must specify a FOREIGN KEY in one table that references a column in another table.
4. FOREIGN KEY constraints are used for relating or binding two tables with each other and then verify the existence of one table data in other tables.
5. A foreign key in one TABLE points to a primary key in another table. Foreign keys prevent actions that would leave rows with foreign key values when there are no primary keys with that value. Foreign key constraints are used to enforce referential integrity.

14.What is the difference between primary key and foreign key?

This is one of the frequently asked SQL Server Interview Questions. Let us understand the difference between a primary key and a foreign key.
Primary key:

1. A primary key uniquely identifies a record in the table.
2. Primary Key can't accept null values and duplicate values by default,
3. The primary key is a clustered index and data in the database table is physically organized in the sequence of the clustered index.
4. We can have only one Primary key in a table.

Foreign key:

1. A foreign key is a field in the table that is the primary key (or unique key) in another table.
2. The foreign key can accept null values and duplicate values.
3. A foreign key does not automatically create an index, clustered, or non-clustered. We can manually create an index on the foreign key.
4. We can have more than one foreign key in a table.

15.What are Temporary tables?

SQL Server provides the concept of the **temporary table** which helps us in a great way. These tables can be created at **runtime** and can do all kinds of operations that one normal table can do. But, based on the table types, the scope is limited.
Permanent tables get created in the database we specify and remain in the database permanently until we delete (drop) them. On the other hand, temporary tables get created in the TempDB database and are automatically deleted, when they are no longer used.

16.What is an index?

1. An **index** is a database object which is used by the SQL server to speed up the performance of queries by providing query access to rows in the data table.
2. By using indexes, we can save time and we can improve the performance of database queries and applications.
3. When we create an index on any column SQL Server internally maintains a separate table called index table so that when any user trying to retrieve the data from the existing table depends on index table SQL Server directly go to the table and retrieve required data very quickly.
4. In a table, we can use a maximum of 250 indexes. The index type refers to the way the index is stored internally by SQL Server.

17.What are the types of Indexes available in SQL Server?

1. **Clustered Index**
2. **Non-Clustered Index**

What is a Clustered Index in SQL Server?

1. In the case of a **clustered index**, the arrangement of the data in the index table will be the same as the arrangement of the data of the actual table.
2. Example: The index we find the start of a book.
3. When a table has a clustered index then the table is called a clustered table.
4. If a table has no clustered index its data rows are stored in an unordered structure.'
5. A table can have only one clustered index in it which will be created when the primary key constraint used in a table.
6. A clustered index determines the physical order of data in a table. For this reason, a table can have only one clustered index.

What is a Non-Clustered Index in SQL Server?

1. In the case of a **non-clustered index**, the arrangement of data in the index table will be different from the arrangement of the actual table.
2. A non-clustered index is analogous to an index in a textbook. The data is stored in one place, the index is another place. The index will have pointers to the storage location of the data.
3. Since, the non-clustered 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.
4. In the index itself, the data is stored in an ascending or descending order of the index key which does not in any way influence the storage of data in the table.
5. In a table, we can create a maximum of 249 non clustered indexes.

18.What is the difference between clustered and non-clustered index in SQL Server?

This is one of the frequently asked SQL Server Indexes Interview Questions. Let us understand the differences.

1. Only one clustered index per table whereas we can have more than one non-clustered index.
2. **Clustered Index** is slightly faster than the Non-Clustered Index. This is because when a Non Clustered Index is used there is an extra look up from the Non-Clustered Index to the table, to fetch the actual rows.

3. The clustered index determines the storage order of rows in the table and hence does not require additional disk space whereas a non-clustered index is stored separately from the table, additional storage space is required.
4. **A clustered index** is a special type of index that reorders the way records in the table are physically stored. Therefore a table can have only one clustered index.
5. **A non-clustered index** is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk.

19.What is a Trigger in SQL Server?

A **Trigger** is a database object which can also be treated as a special kind of stored procedure that automatically executes when language events (INSERT, DELETE or UPDATE) occurs. Triggers are stored in and managed by the DBMS. Triggers are similar to stored procedures in that both consist of procedural logic that is stored at the database level.

However, the difference between a trigger and a stored procedure is that the trigger is attached to a table and is only fired when an INSERT, UPDATE or DELETE operation occurred whereas stored procedures are not attached to a specific table and can be executed explicitly by making a call to the stored procedure. In addition to that, triggers can also execute stored procedures.

Nested Trigger: A trigger can also contain INSERT, UPDATE and DELETE logic within itself; so, when the trigger is fired because of data modification, it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself which causes another trigger to be fired is called the nested trigger.

20.What are the types of Triggers available in SQL Server?

The following are the two types of triggers in SQL Server.

1. **After Triggers (For Triggers):** Fired after Insert, Update and Delete operations on a table.
2. **Instead of Triggers:** Fired instead of Insert, Update and Delete operations on a table.

In SQL Server, there are 4 types of triggers

1. **DML Triggers – Data Manipulation Language**
2. **DDL Triggers – Data Definition Language**
3. **CLR triggers – Common Language Runtime**
4. **Logon triggers**

21.What is a View in SQL Server?

A view is nothing more than a saved SQL query. A view can also be considered as a virtual table. So, we can think of a view either as a compiled SQL query or a virtual table. As a view represents a virtual table it does not physically store any data by default. When we query a view we actually, retrieve the data from the underlying database tables.

22.Why we need the stored procedure?

This is one of the most frequently asked SQL Server Stored Procedure Interview Questions in the SQL Server Interview. So let’s discuss this question in details
Whenever we want to execute a SQL query from an application the SQL query (statement) what we send from an application will be first compiled(parsed) for execution, where the process of compiling(parsing) is time-consuming because compilation occurs each time when we execute the query or statements.
To overcome the above problem, we write SQL statements or query under the stored procedure and execute because a stored procedure is a pre-compiled block of code, without compiling(parsing) the statements get executed whenever the procedures are called which can increase the performance of database server which ultimately increases the performance of the application.
If we have a situation where we write the same query again and again, we can save that specific query as a stored procedure and call it’s just by its name whenever required that query to execute.

23.SWhat are the different procedure attributes in SQL Server?

This is one of the frequently asked SQL Server Stored Procedure Interview Questions. There are two types of attributes

1. The With Encryption
2. With Recompile

With Encryption Attribute:

If this attribute is used on the procedure the text of this procedure is encrypted and will not be shown in the text column of the syscomments table so no one will be having an option to view the content of it.
Note: When an application is developed for a client at the time of installing this application on the client system we will be using the encryption option on all the views, procedures, functions, triggers, etc. and install on the client machine. So that they will not have the chance of viewing the source code or altering the source code.

With Recompiled Attribute:

1. Whenever a procedure is compiled for the first time it prepares the best query plan according to the current state of the database and executes the query plan when the procedure is called.
2. The compilation of the procedure and preparing a query plan is prepared not only at the time of procedure creation but each and every time the server is restarted (Implicitly occurs).
3. If the procedure is created by using with Recompile procedure attribute, it is forced to be compiled each time it is executed and whenever it compiles it prepares the query plan.
4. Forcing a procedure for recompilation and prepared a query plan is required when the database undergoes significant changes to its data or structure.
5. Another reason to force a procedure to recompile is if at all the tables is added with new indexes from which the procedure might be benefited forcing for recompilation is very important because we cannot wait until the server is restarted for preparing a new query plan

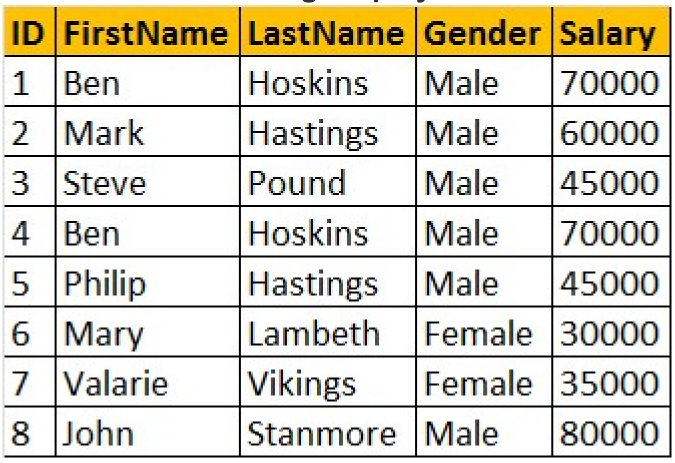
Part 1 - How to find nth highest salary in sql

Suggested Video Tutorials:
SQL Server Tutorial
ASP.NET Tutorial
WCF Tutorial

This is a very common SQL Server Interview Question. There are several ways of finding the nth highest salary.

By the end of this video, we will be able to answer all the following questions as well.
1. How to find nth highest salary in SQL Server using a Sub-Query
2. How to find nth highest salary in SQL Server using a CTE
3. How to find the 2nd, 3rd or 15th highest salary

Let's use the following Employees table for this demo.



ID	FirstName	LastName	Gender	Salary
1	Ben	Hoskins	Male	70000
2	Mark	Hastings	Male	60000
3	Steve	Pound	Male	45000
4	Ben	Hoskins	Male	70000
5	Philip	Hastings	Male	45000
6	Mary	Lambeth	Female	30000
7	Valarie	Vikings	Female	35000
8	John	Stanmore	Male	80000

Use the following script to create Employees table

```
Create table Employees
(
   ID int primary key identity,
   FirstName nvarchar(50),
   LastName nvarchar(50),
   Gender nvarchar(50),
   Salary int
)
GO
```

```
Insert into Employees values ('Ben', 'Hoskins', 'Male', 70000)
Insert into Employees values ('Mark', 'Hastings', 'Male', 60000)
Insert into Employees values ('Steve', 'Pound', 'Male', 45000)
Insert into Employees values ('Ben', 'Hoskins', 'Male', 70000)
Insert into Employees values ('Philip', 'Hastings', 'Male', 45000)
Insert into Employees values ('Mary', 'Lambeth', 'Female', 30000)
Insert into Employees values ('Valarie', 'Vikings', 'Female', 35000)
Insert into Employees values ('John', 'Stanmore', 'Male', 80000)
GO
```

To find the highest salary it is straight forward. We can simply use the **Max**() function as shown below.

```
Select Max(Salary) from Employees
```

To get the **second highest salary** use a **sub query** along with **Max()** function as shown below.

```
Select Max(Salary) from Employees where Salary < (Select Max(Salary) from Employees)
```

To find nth highest salary using Sub-Query

```
SELECT TOP 1 SALARY
FROM (
      SELECT DISTINCT TOP N SALARY
      FROM EMPLOYEES
      ORDER BY SALARY DESC
      ) RESULT
ORDER BY SALARY
```

To find nth highest salary using CTE

```
WITH RESULT AS
(
    SELECT SALARY,
           DENSE_RANK() OVER (ORDER BY SALARY DESC) AS DENSERANK
    FROM EMPLOYEES
)
SELECT TOP 1 SALARY
```

```
FROM RESULT
WHERE DENSERANK = N
```

To find 2nd highest salary we can use any of the above queries. Simple replace N with 2.

Similarly, to find 3rd highest salary, simple replace N with 3.

Please Note: On many of the websites, you may have seen that, the following query can be used to get the nth highest salary. The below query will only work if there are no duplicates.

```
WITH RESULT AS
(
   SELECT SALARY,
          ROW_NUMBER() OVER (ORDER BY SALARY DESC) AS ROWNUMBER
   FROM EMPLOYEES
)
SELECT SALARY
FROM RESULT
WHERE ROWNUMBER = 3
```

Part 4 - Delete duplicate rows in sql

Suggested Videos:
Part 1 - How to find nth highest salary in sql
Part 2 - SQL query to get organization hierarchy
Part 3 - How does a recursive CTE work

In this video, we will discuss **deleting all duplicate rows except one from a sql server table**.

Let me explain what we want to achieve. We will be using **Employees** table for this demo.

ID	FirstName	LastName	Gender	Salary
1	Mark	Hastings	Male	60000
1	Mark	Hastings	Male	60000
1	Mark	Hastings	Male	60000
2	Mary	Lambeth	Female	30000
2	Mary	Lambeth	Female	30000
3	Ben	Hoskins	Male	70000
3	Ben	Hoskins	Male	70000
3	Ben	Hoskins	Male	70000

SQL Script to create **Employees** table

```
Create table Employees
(
    ID int,
    FirstName nvarchar(50),
    LastName nvarchar(50),
    Gender nvarchar(50),
    Salary int
)
GO

Insert into Employees values (1, 'Mark', 'Hastings', 'Male', 60000)
Insert into Employees values (1, 'Mark', 'Hastings', 'Male', 60000)
Insert into Employees values (1, 'Mark', 'Hastings', 'Male', 60000)
Insert into Employees values (2, 'Mary', 'Lambeth', 'Female', 30000)
Insert into Employees values (2, 'Mary', 'Lambeth', 'Female', 30000)
Insert into Employees values (3, 'Ben', 'Hoskins', 'Male', 70000)
Insert into Employees values (3, 'Ben', 'Hoskins', 'Male', 70000)
Insert into Employees values (3, 'Ben', 'Hoskins', 'Male', 70000)
```

The delete query should delete all duplicate rows except one. The output should be as shown below, after the delete query is executed.

ID	FirstName	LastName	Gender	Salary
1	Mark	Hastings	Male	60000
2	Mary	Lambeth	Female	30000
3	Ben	Hoskins	Male	70000

Here is the SQL query that does the job. **PARTITION BY** divides the query result set into partitions.

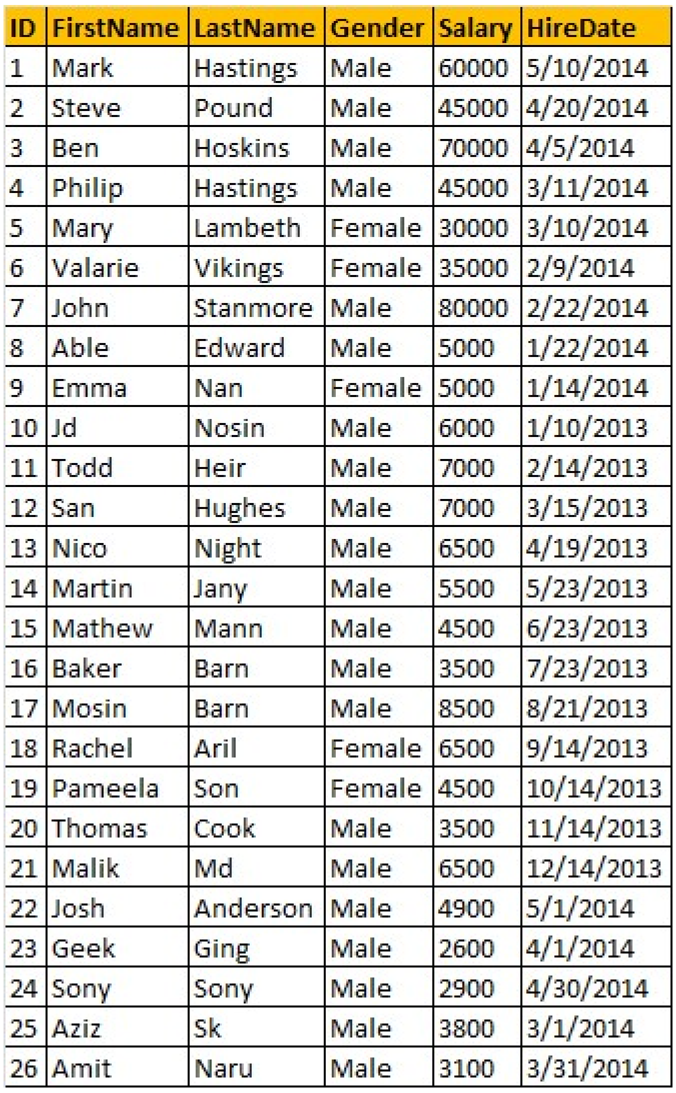
```
WITH EmployeesCTE AS
(
  SELECT *, ROW_NUMBER()OVER(PARTITION BY ID ORDER BY ID) AS RowNumber
  FROM Employees
)
DELETE FROM EmployeesCTE WHERE RowNumber > 1
```

Part 5 - SQL query to find employees hired in last n months

Suggested Videos:
Part 2 - SQL query to get organization hierarchy
Part 3 - How does a recursive CTE work
Part 4 - Delete duplicate rows in sql

This question is asked is many sql server interviews. If you have used **DATEDIFF**() sql server function then you already know the answer.

We will be using the following **Employees** table for this demo.



ID	FirstName	LastName	Gender	Salary	HireDate
1	Mark	Hastings	Male	60000	5/10/2014
2	Steve	Pound	Male	45000	4/20/2014
3	Ben	Hoskins	Male	70000	4/5/2014
4	Philip	Hastings	Male	45000	3/11/2014
5	Mary	Lambeth	Female	30000	3/10/2014
6	Valarie	Vikings	Female	35000	2/9/2014
7	John	Stanmore	Male	80000	2/22/2014
8	Able	Edward	Male	5000	1/22/2014
9	Emma	Nan	Female	5000	1/14/2014
10	Jd	Nosin	Male	6000	1/10/2013
11	Todd	Heir	Male	7000	2/14/2013
12	San	Hughes	Male	7000	3/15/2013
13	Nico	Night	Male	6500	4/19/2013
14	Martin	Jany	Male	5500	5/23/2013
15	Mathew	Mann	Male	4500	6/23/2013
16	Baker	Barn	Male	3500	7/23/2013
17	Mosin	Barn	Male	8500	8/21/2013
18	Rachel	Aril	Female	6500	9/14/2013
19	Pameela	Son	Female	4500	10/14/2013
20	Thomas	Cook	Male	3500	11/14/2013
21	Malik	Md	Male	6500	12/14/2013
22	Josh	Anderson	Male	4900	5/1/2014
23	Geek	Ging	Male	2600	4/1/2014
24	Sony	Sony	Male	2900	4/30/2014
25	Aziz	Sk	Male	3800	3/1/2014
26	Amit	Naru	Male	3100	3/31/2014

SQL Script to create the table and populate with test data

```
Create table Employees

(

    ID int primary key identity,

    FirstName nvarchar(50),

    LastName nvarchar(50),

    Gender nvarchar(50),

    Salary int,

    HireDate DateTime

)

GO


Insert into Employees values('Mark','Hastings','Male',60000,'5/10/2014')
```

```sql
Insert into Employees values('Steve','Pound','Male',45000,'4/20/2014')

Insert into Employees values('Ben','Hoskins','Male',70000,'4/5/2014')

Insert into Employees values('Philip','Hastings','Male',45000,'3/11/2014')

Insert into Employees values('Mary','Lambeth','Female',30000,'3/10/2014')

Insert into Employees values('Valarie','Vikings','Female',35000,'2/9/2014')

Insert into Employees values('John','Stanmore','Male',80000,'2/22/2014')

Insert into Employees values('Able','Edward','Male',5000,'1/22/2014')

Insert into Employees values('Emma','Nan','Female',5000,'1/14/2014')

Insert into Employees values('Jd','Nosin','Male',6000,'1/10/2013')

Insert into Employees values('Todd','Heir','Male',7000,'2/14/2013')

Insert into Employees values('San','Hughes','Male',7000,'3/15/2013')

Insert into Employees values('Nico','Night','Male',6500,'4/19/2013')

Insert into Employees values('Martin','Jany','Male',5500,'5/23/2013')

Insert into Employees values('Mathew','Mann','Male',4500,'6/23/2013')

Insert into Employees values('Baker','Barn','Male',3500,'7/23/2013')

Insert into Employees values('Mosin','Barn','Male',8500,'8/21/2013')

Insert into Employees values('Rachel','Aril','Female',6500,'9/14/2013')

Insert into Employees values('Pameela','Son','Female',4500,'10/14/2013')

Insert into Employees values('Thomas','Cook','Male',3500,'11/14/2013')

Insert into Employees values('Malik','Md','Male',6500,'12/14/2013')

Insert into Employees values('Josh','Anderson','Male',4900,'5/1/2014')

Insert into Employees values('Geek','Ging','Male',2600,'4/1/2014')

Insert into Employees values('Sony','Sony','Male',2900,'4/30/2014')

Insert into Employees values('Aziz','Sk','Male',3800,'3/1/2014')

Insert into Employees values('Amit','Naru','Male',3100,'3/31/2014')
```

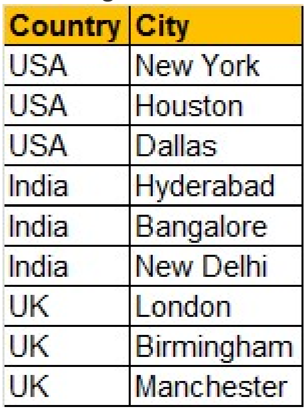
Here is the SQL Query that does the job

```sql
-- Replace N with number of months

Select *

FROM Employees

Where DATEDIFF(MONTH, HireDate, GETDATE()) Between 1 and N
```

Part 6 - Transform rows into columns in sql server

Suggested Videos:
Part 3 - How does a recursive CTE work
Part 4 - Delete duplicate rows in sql
Part 5 - SQL query to find employees hired in last n months

This is another common sql server interview question. We will be using the following **Countries** table in this example.

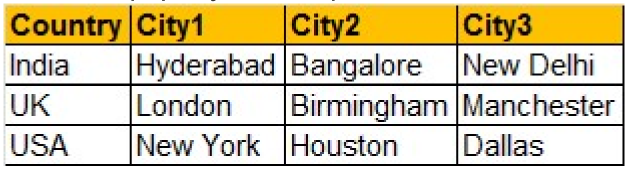


Country	City
USA	New York
USA	Houston
USA	Dallas
India	Hyderabad
India	Bangalore
India	New Delhi
UK	London
UK	Birmingham
UK	Manchester

SQL to create the table

```
Create Table Countries

(

    Country nvarchar(50),

    City nvarchar(50)

)

GO


Insert into Countries values ('USA','New York')

Insert into Countries values ('USA','Houston')

Insert into Countries values ('USA','Dallas')


Insert into Countries values ('India','Hyderabad')

Insert into Countries values ('India','Bangalore')

Insert into Countries values ('India','New Delhi')
Insert into Countries values ('UK','London')

Insert into Countries values ('UK','Birmingham')

Insert into Countries values ('UK','Manchester')
```

Here is the interview question:
Write a sql query to transpose rows to columns. The output should be as shown below.



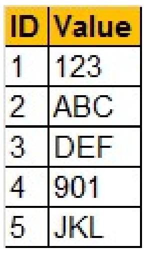
Country	City1	City2	City3
India	Hyderabad	Bangalore	New Delhi
UK	London	Birmingham	Manchester
USA	New York	Houston	Dallas

Using PIVOT operator we can very easily transform rows to columns

```
Select Country, City1, City2, City3

From

(

  Select Country, City,

    'City'+

      cast(row_number() over(partition by Country order by Country)

             as varchar(10)) ColumnSequence

  from Countries

) Temp

pivot

(

  max(City)

  for ColumnSequence in (City1, City2, City3)

) Piv
```

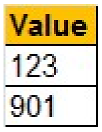
Part 7 - SQL query to find rows that contain only numerical data
Suggested Videos:
Part 4 - Delete duplicate rows in sql
Part 5 - SQL query to find employees hired in last n months
Part 6 - Transform rows into columns in sql server

Let me explain the **scenario mentioned in one of the sql server interview**. We have the following table.



ID	Value
1	123
2	ABC
3	DEF
4	901
5	JKL

Write a SQL query to retrieve rows that contain only numerical data. The output of the query should be as shown below.



Value
123
901

SQL Script to create the TestTable

```
Create Table TestTable

(

    ID int identity primary key,

    Value nvarchar(50)

)


Insert into TestTable values ('123')

Insert into TestTable values ('ABC')

Insert into TestTable values ('DEF')

Insert into TestTable values ('901')

Insert into TestTable values ('JKL')
```

This is very easy to achieve. If you have used **ISNUMERIC**() function in SQL Server, then you already know the answer. Here is the query

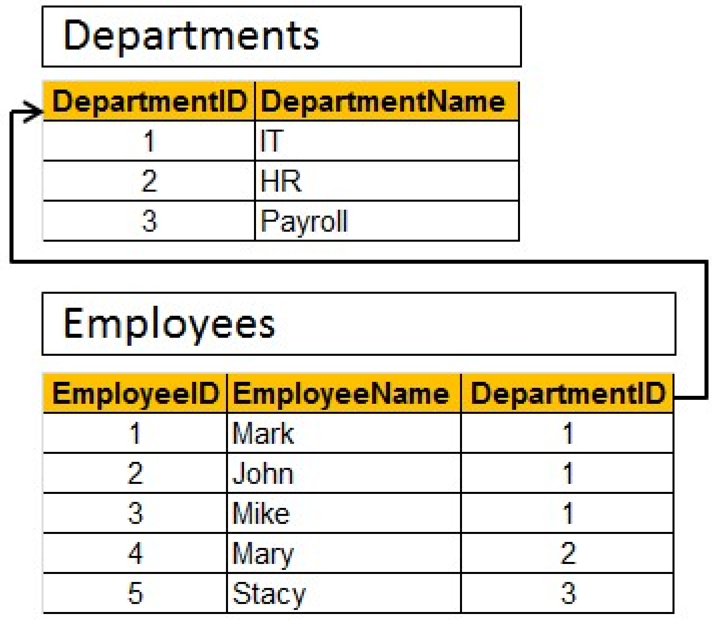
```
Select Value from TestTable Where ISNUMERIC(Value) = 1
```

ISNUMERIC function returns 1 when the input expression evaluates to a valid numeric data type, otherwise it returns 0. For the list of all valid numeric data types in SQL Server please visit the following MSDN link.
http://technet.microsoft.com/en-us/library/ms186272(v=sql.110).aspx

Part 8 - SQL Query to find department with highest number of employees

Suggested Videos:
Part 5 - SQL query to find employees hired in last n months
Part 6 - Transform rows into columns in sql server
Part 7 - SQL query to find rows that contain only numerical data

We will be using the following tables (**Employees** & **Departments**) to answer this question. In **Employees** table **DepartmentID** is the **foreign key** referencing **DepartmentID** column in **Departments** table.



Departments

DepartmentID	DepartmentName
1	IT
2	HR
3	Payroll

Employees

EmployeeID	EmployeeName	DepartmentID
1	Mark	1
2	John	1
3	Mike	1
4	Mary	2
5	Stacy	3

SQL Script to create the required tables

```
Create Table Departments

(

    DepartmentID int primary key,

    DepartmentName nvarchar(50)

)

GO


Create Table Employees

(

    EmployeeID int primary key,
    EmployeeName nvarchar(50),
    DepartmentID int foreign key references Departments(DepartmentID)
)
GO

Insert into Departments values (1, 'IT')
Insert into Departments values (2, 'HR')
Insert into Departments values (3, 'Payroll')
GO

Insert into Employees values (1, 'Mark', 1)
Insert into Employees values (2, 'John', 1)
Insert into Employees values (3, 'Mike', 1)
Insert into Employees values (4, 'Mary', 2)
Insert into Employees values (5, 'Stacy', 3)
GO
```

Scenario asked in the SQL Server Interview
Based on the above two tables write a SQL Query to get the **name of the Department that has got the maximum number of Employees**. To answer this question it will be helpful if you the knowledge of JOINS & GROUP BY in SQL Server. We discusses these in Parts 11 & 12 of SQL Server Tutorial video series.

SQL query that retrieves the department name with maximum number of employees

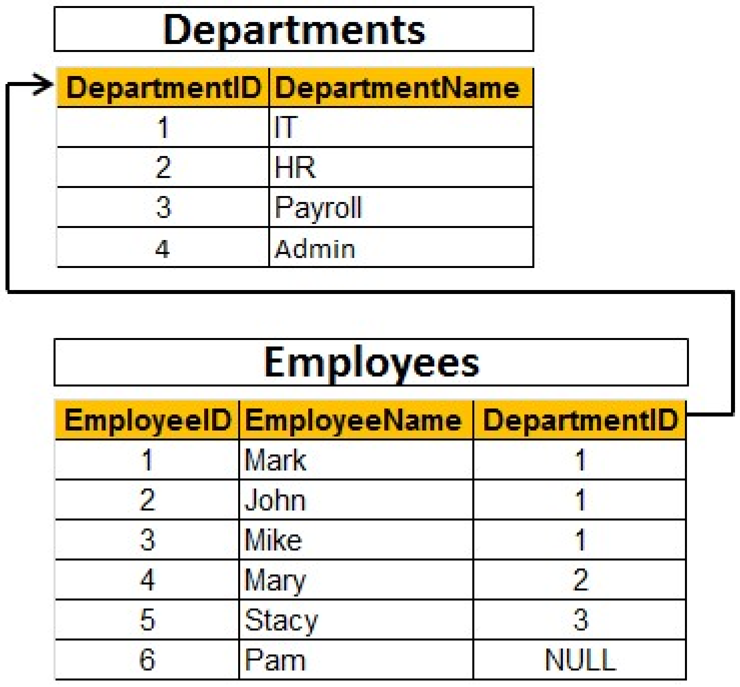
```
SELECT TOP 1 DepartmentName
FROM Employees
JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID
GROUP BY DepartmentName
ORDER BY COUNT(*) DESC
```

Part 9 - Difference between inner join and left join

Suggested Videos:
Part 6 - Transform rows into columns in sql server
Part 7 - SQL query to find rows that contain only numerical data
Part 8 - SQL Query to find department with highest number of employees

This is one of the very common sql server interview question. Different JOINS in SQL Server are discussed in detail in Part 12 of SQL Server tutorial for beginners video series.

Let's understand the difference with an example. We will be using the following tables in this demo



Departments

DepartmentID	DepartmentName
1	IT
2	HR
3	Payroll
4	Admin

Employees

EmployeeID	EmployeeName	DepartmentID
1	Mark	1
2	John	1
3	Mike	1
4	Mary	2
5	Stacy	3
6	Pam	NULL

SQL Script to create and populate the required tables with test data

```
Create Table Departments

(

     DepartmentID int primary key,

     DepartmentName nvarchar(50)

)

GO


Create Table Employees

(
    EmployeeID int primary key,
    EmployeeName nvarchar(50),
    DepartmentID int foreign key references Departments(DepartmentID)
)
GO

Insert into Departments values (1, 'IT')
Insert into Departments values (2, 'HR')
Insert into Departments values (3, 'Payroll')
Insert into Departments values (4, 'Admin')
GO

Insert into Employees values (1, 'Mark', 1)
Insert into Employees values (2, 'John', 1)
Insert into Employees values (3, 'Mike', 1)
Insert into Employees values (4, 'Mary', 2)
Insert into Employees values (5, 'Stacy', 3)
Insert into Employees values (6, 'Pam', NULL)
GO
```

INNER JOIN returns only the matching rows between the tables involved in the JOIN. Notice that, Pam employee record which does not have a matching DepartmentId in departments table is eliminated from the result-set.

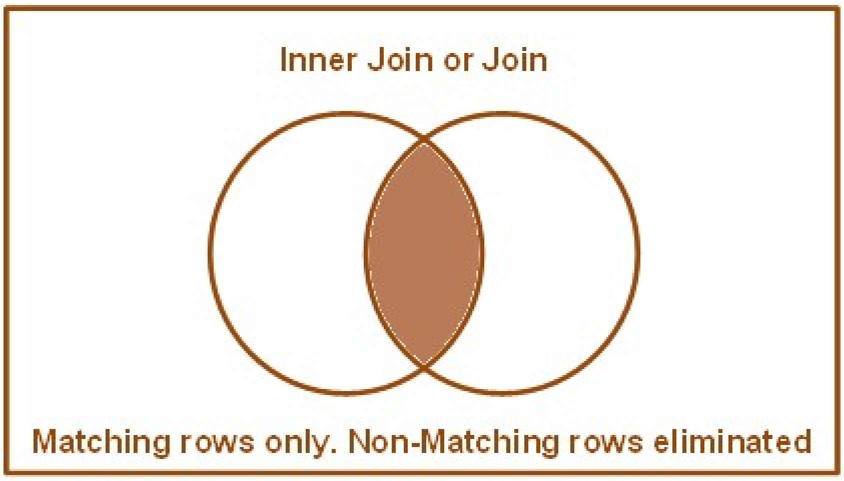
```
SELECT EmployeeName, DepartmentName
FROM Employees
INNER JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID
```

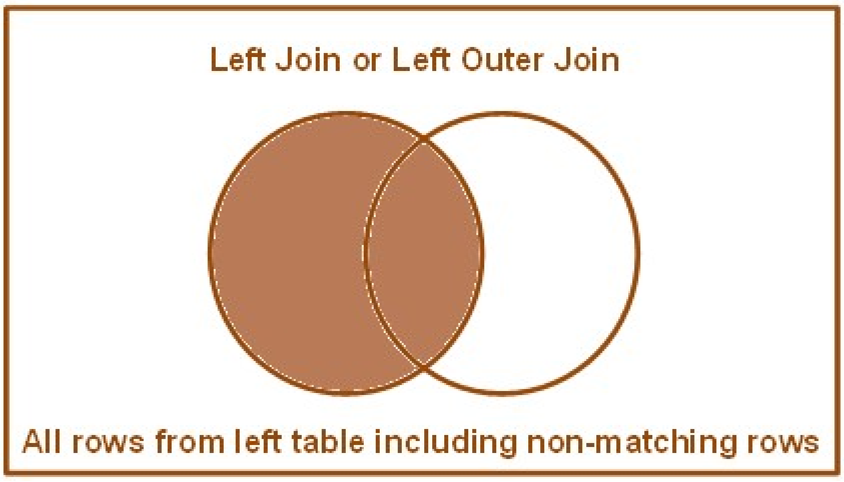
LEFT JOIN returns all rows from left table including non-matching rows. Notice that, Pam employee record which does not have a matching DepartmentId in departments table is also included in the result-set.

```
SELECT EmployeeName, DepartmentName
FROM Employees
LEFT JOIN Departments
```

```
ON Employees.DepartmentID = Departments.DepartmentID
```

A picture speaks thousand words. So, here is the difference between inner join and left join.




Left Join or Left Outer Join
All rows from left table including non-matching rows

In general there could be several questions on JOINS in a sql server interview. If we understand the basics of JOINS properly, then answering any JOINS related questions should be a cakewalk.

What is the difference between INNER JOIN and RIGHT JOIN

INNER JOIN returns only the matching rows between the tables involved in the JOIN, where as RIGHT JOIN returns all the rows from the right table including the NON-MATCHING rows.

What is the difference between INNER JOIN and FULL JOIN

FULL JOIN returns all the rows from both the left and right tables including the NON-MATCHING rows.

What is the Difference between INNER JOIN and JOIN

There is no difference they are exactly the same. Similarly there is also no difference between

LEFT JOIN and LEFT OUTER JOIN

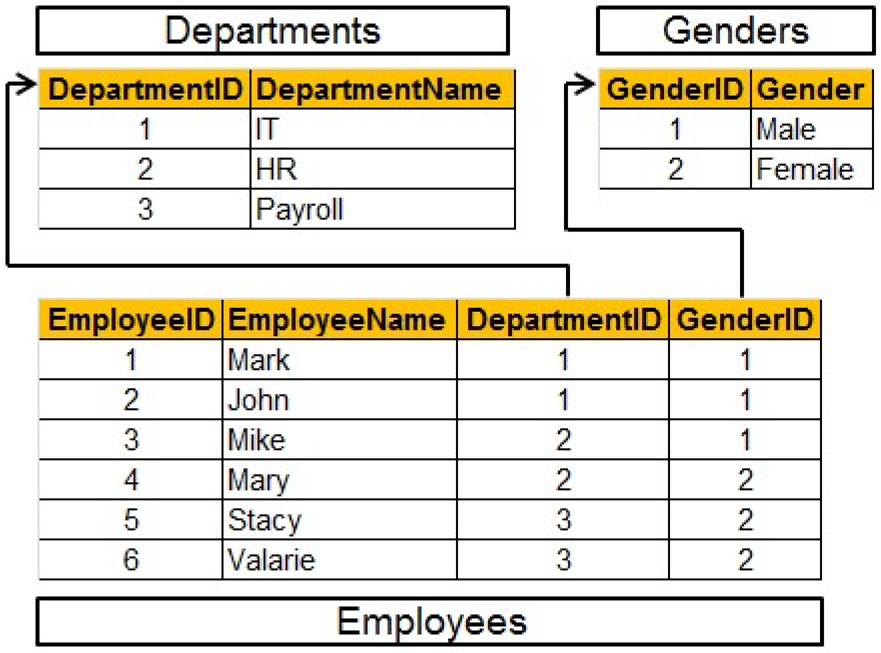
RIGHT JOIN and RIGHT OUTER JOIN

FULL JOIN and FULL OUTER JOIN

Part 10 - Join 3 tables in sql server
Suggested Videos:
Part 7 - SQL query to find rows that contain only numerical data
Part 8 - SQL Query to find department with highest number of employees
Part 9 - Difference between inner join and left join

In this video we will discuss **joining 3 tables in SQL Server**. Joining 3 tables (or even more) is very similar to joining 2 tables.

We will be using the following **3 tables** in this demo.



Departments

DepartmentID	DepartmentName
1	IT
2	HR
3	Payroll

Genders

GenderID	Gender
1	Male
2	Female

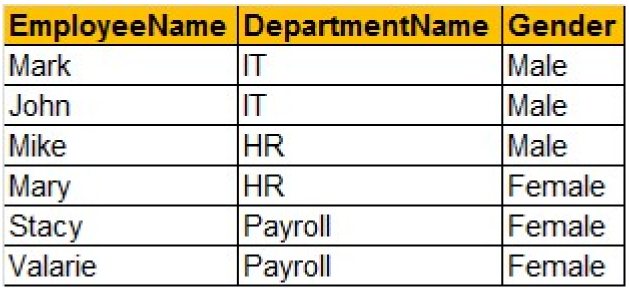
EmployeeID	EmployeeName	DepartmentID	GenderID
1	Mark	1	1
2	John	1	1
3	Mike	2	1
4	Mary	2	2
5	Stacy	3	2
6	Valarie	3	2

Employees

SQL Script to create the required tables

```
Create Table Departments
(
    DepartmentID int primary key,
    DepartmentName nvarchar(50)
)
GO

Create Table Genders
(
    GenderID int primary key,
    Gender nvarchar(50)
)
GO

Create Table Employees
(
    EmployeeID int primary key,
    EmployeeName nvarchar(50),
    DepartmentID int foreign key references Departments(DepartmentID),
    GenderID int foreign key references Genders(GenderID)
)
GO

Insert into Departments values (1, 'IT')
Insert into Departments values (2, 'HR')
Insert into Departments values (3, 'Payroll')
GO

Insert into Genders values (1, 'Male')
Insert into Genders values (2, 'Female')
GO

Insert into Employees values (1, 'Mark', 1, 1)
Insert into Employees values (2, 'John', 1, 1)
Insert into Employees values (3, 'Mike', 2, 1)
Insert into Employees values (4, 'Mary', 2, 2)
Insert into Employees values (5, 'Stacy', 3, 2)
Insert into Employees values (6, 'Valarie', 3, 2)
GO
```

Write a query to **join 3 the tables and retrieve EmployeeName, DepartmentName and Gender**. The output should be as shown below.

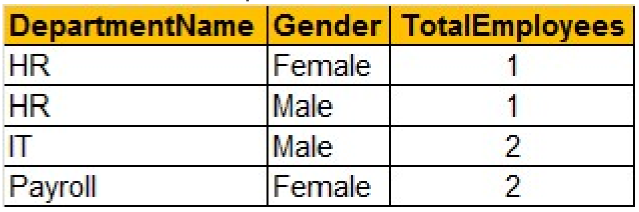


EmployeeName	DepartmentName	Gender
Mark	IT	Male
John	IT	Male
Mike	HR	Male
Mary	HR	Female
Stacy	Payroll	Female
Valarie	Payroll	Female

Query:

```
SELECT EmployeeName, DepartmentName, Gender

FROM Employees

JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID

JOIN Genders ON Employees.GenderID = Genders.GenderID
```

Write a query to show the **total number of employees by DEPARTMENT and by GENDER**. The output should be as shown below.

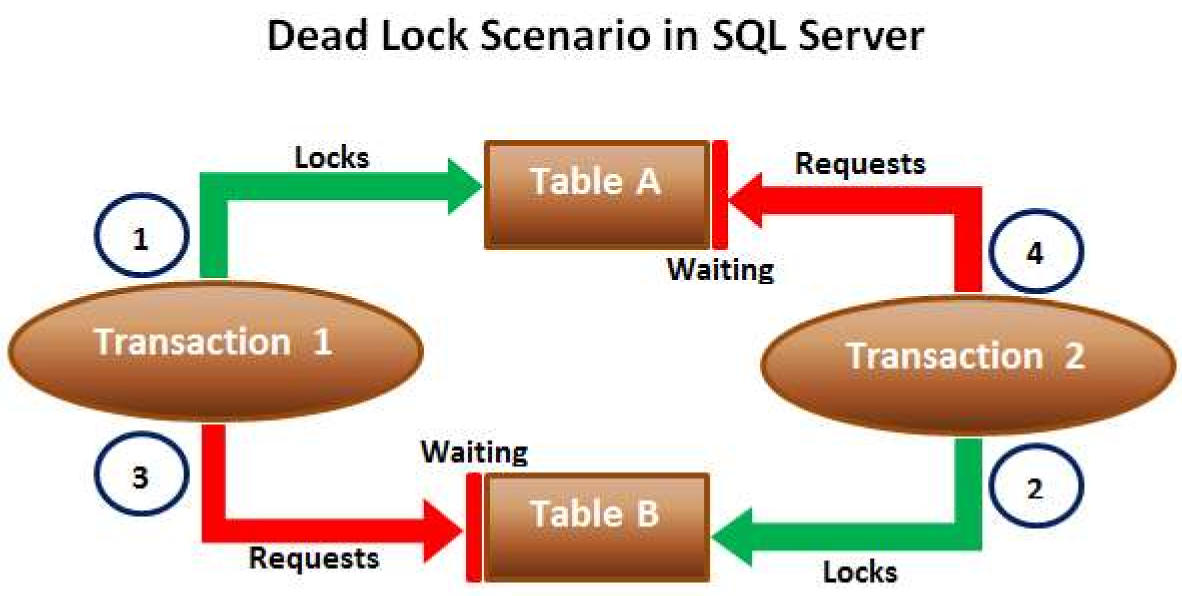


DepartmentName	Gender	TotalEmployees
HR	Female	1
HR	Male	1
IT	Male	2
Payroll	Female	2

Query:

```
SELECT DepartmentName, Gender, COUNT(*) as TotalEmployees

FROM Employees

JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID

JOIN Genders ON Employees.GenderID = Genders.GenderID

GROUP BY DepartmentName, Gender

ORDER BY DepartmentName, Gender
```

Deadlock : Occurs when two or more transactions have a resource locked, and each transaction requests a lock on the resource that another transaction has already locked. Neither of the transactions here can move forward, as each one is waiting for the other to release the lock. So in this case, SQL Server intervenes and ends the deadlock by cancelling one of the transactions, so the other transaction can move forward. The following diagram explains this.



Example : Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code. Notice that there is a deadlock between Transaction 1 and Transaction 2.

```
-- Transaction 1
Begin Tran
Update TableA Set Name = 'Mark Transaction 1' where Id = 1
 -- From Transaction 2 window execute the first update statement
 pdate TableB Set Name = 'Mary Transaction 1' where Id = 1
 -- From Transaction 2 window execute the second update statement
Commit Transaction

-- Transaction 2
Begin Tran
Update TableB Set Name = 'Mark Transaction 2' where Id = 1
-- From Transaction 1 window execute the second update statement
Update TableA Set Name = 'Mary Transaction 2' where Id = 1
-- After a few seconds notice that one of the transactions complete
-- successfully while the other transaction is made the deadlock victim
Commit Transaction
```

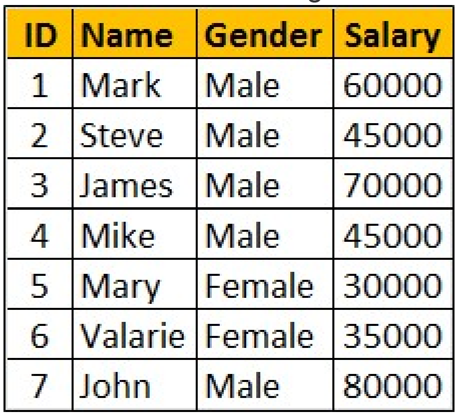
Sql query to select all names that start with a given letter without like operator

Suggested Videos:

Part 11 - Real time example for right join
Part 12 - Can we join two tables without primary foreign key relation
Part 13 - Difference between blocking and deadlocking

In this video we will discuss writing a SQL query to retrieve all student names that start with letter 'M' without using the LIKE operator.

We will use the following Students table for this example



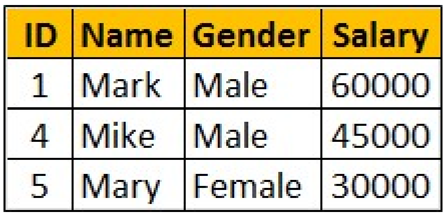
ID	Name	Gender	Salary
1	Mark	Male	60000
2	Steve	Male	45000
3	James	Male	70000
4	Mike	Male	45000
5	Mary	Female	30000
6	Valarie	Female	35000
7	John	Male	80000

SQL Script to create the Students table

```
Create table Students
(
     ID int primary key identity,

     Name nvarchar(50),

     Gender nvarchar(50),

     Salary int

)

Go


Insert into Students values ('Mark', 'Male', 60000)

Insert into Students values ('Steve', 'Male', 45000)

Insert into Students values ('James', 'Male', 70000)

Insert into Students values ('Mike', 'Male', 45000)

Insert into Students values ('Mary', 'Female', 30000)

Insert into Students values ('Valarie', 'Female', 35000)

Insert into Students values ('John', 'Male', 80000)

Go
```

Interview question : Write a query to select all student rows whose Name starts with letter 'M' without using the LIKE operator

The output should be as shown below



ID	Name	Gender	Salary
1	Mark	Male	60000
4	Mike	Male	45000
5	Mary	Female	30000

If the interviewer has not mentioned not to use LIKE operator, we would have written the query using the LIKE operator as shown below.

```
SELECT * FROM Students WHERE Name LIKE 'M%'
```

We can use any one of the following 3 SQL Server functions, to achieve exactly the same thing
CHARINDEX
LEFT
SUBSTRING

The following 3 queries retrieve all student rows whose Name starts with letter 'M'. Notice none of the queries are using the LIKE operator.

```
SELECT * FROM Students WHERE CHARINDEX('M',Name) = 1
SELECT * FROM Students WHERE LEFT(Name, 1) = 'M'

SELECT * FROM Students WHERE SUBSTRING(Name, 1, 1) = 'M'
```