

SQL Server Interview Questions & Answers



By Shailendra Chauhan

Microsoft MVP, Founder & CEO - Dot Net Tricks



SQL Server Interview Questions & Answers

All rights reserved. No part of this book can be reproduced or stored in any retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, uploading on server and scanning without the prior written permission of the Dot Net Tricks Innovation Pvt. Ltd.

The author of this book has tried their best to ensure the accuracy of the information described in this book. However, the author cannot guarantee the accuracy of the information contained in this book. The author or Dot Net Tricks Innovation Pvt. Ltd. will not be liable for any damages, incidental or consequential caused directly or indirectly by this book.

Further, readers should be aware that the websites or reference links listed in this book may have changed or disappeared between when this book was written and when it is read.

All other trademarks referred to in this book are the property of their respective owners.

Release History

- Initial Release 1.0 - 23rd Nov 2018
- Second Release 1.1 - 21st Dec 2018
- Third Release 1.2 - 4th Jan 2019
- Fourth Release 1.3 - 31st Jan 2019



About Dot Net Tricks

Dot Net Tricks is founded by Shailendra Chauhan (Microsoft MVP), in Jan 2010. Dot Net Tricks came into existence in form of a blog post over various technologies including .NET, C#, SQL Server, ASP.NET, ASP.NET MVC, JavaScript, Angular, Node.js and Visual Studio etc.

The company which is currently registered by a name of Dot Net Tricks Innovation Pvt. Ltd. came into the shape in 2015. Dot Net Tricks website has an average footfall on the tune of 300k+ per month. The site has become a cornerstone when it comes to getting skilled-up on .NET technologies and we want to gain the same level of trust in other technologies. This is what we are striving for.

We have a very large number of trainees who have received training from our platforms and immediately got placement in some of the reputed firms testifying our claims of providing quality training. The website offers you a variety of free study material in form of articles.

Dot Net Tricks Training Solutions

Dot Net Tricks provide you training in traditional as well as new age technologies, via various formats.

Master Courses (Instructor-led)

For a beginner who needs regular guidance, we have a fully packed Master Courses. They are almost equal to semester courses taught in engineering colleges when it comes to length, breadth of content delivery, the only difference instead of 5-6 months, they take approx. 16-weekend classes (2 months).

The detail about Master courses can be found here: <https://www.dotnettricks.com/instructor-led-courses>

Hands-On Learning (Learn to code)

Hands-On Learning courses give you the confidence to code and equally helpful to work in real-life scenarios. This course is composed of hands-on exercise using IDE or cloud labs so that you can practice each and everything by yourself. You can learn to code at your own pace, time and place.

The detail about Hands-On Learning courses can be found here: <https://www.scholarhat.com>

Skill Bootcamps (Instructor-led)

Professionals who don't have two months' time and want to get skilled up in least possible time due to some new project that their company has to take in very near future, we have designed Skill Bootcamps Concept, where you will get trained on consecutive days in a fast-paced manner, where our full focus is going to be on hands-on delivery of technological exercises.

The detail about Skill Bootcamps can be found here: <https://www.dotnettricks.com/skill-bootcamp>

Self-paced Courses

Self-paced courses give you the liberty to study at your own pace, time and place. We understand everyone has their own comfort zone, some of you can afford to dedicate 2 hours a day, some of you not. Keeping this thing in

mind, we created these self-paced courses. While creating these courses we have ensured that quality of courses doesn't get compromise at any parameter, and they also will be able to produce the same results as our other course formats, given the fact you will be able to put your own honest effort.

The detail about Self-paced courses can be found here: <https://www.dotnettricks.com/self-paced-courses>

Corporate Training (Online and Classroom)

Dot Net Tricks having a pool of mentors who help the corporate to enhance their employment skills as per changing technology landscape. Dot Net Tricks offers customized training programs for new hires and experienced employees through online and classroom mode. As a trusted and resourceful training partner, Dot Net Tricks helps the corporate to achieve success with its industry-leading instructional design and customer training initiatives.

The detail about Corporate Training can be found here: <https://www.dotnettricks.com/corporate-training>

Learning Platform

We have a very robust technology platform to answer the needs of all our trainees, no matter which program they enrolled in. We have a very self-intuitive Learning Management System (LMS), which help you in remain focused and keeping an eye over your progress.

We offer two Learning Platforms as given below:

1. Dot Net Tricks: <https://www.dotnettricks.com>
2. Scholar Hat: <https://www.scholarhat.com>

Apart from these, we also provide on-demand Skill bootcamps and personalized project consultation.



Dedication

My mother Mrs Vriksha Devi and my wife Reshu Chauhan deserve to have their name on the cover as much as I do for all their support made this possible. I would like to say thanks to all my family members Virendra Singh(father), Jaishree and Jyoti(sisters), Saksham and Pranay(sons), friends, to you and to readers or followers of my articles at <https://www.dotnettricks.com/mentors/shailendra-chauhan> to encourage me to write this book.

-Shailendra Chauhan



Introduction

If you are looking for a book on SQL Server interview, you've come across the right book. This is the book which will master you SQL Server fundamentals. This book will teach you the SQL Server from scratch so that you have the confidence to give the answer of all the questions being asked in your interview.

This book is designed specifically to teach you the DBMS and RDBMS concepts along with SQL server features. This book covers database concepts, SQL queries, SQL statements, functions, procedures, cursor, triggers and exception handling.

So, what where my qualification to write this book? My qualification and inspiration come from my enthusiasm for and the experience with the technology and from my analytic and initiative nature. Being a consultant, corporate trainer, and blogger, I have thorough knowledge and understandings of .NET technologies. My inspiration and knowledge have also come from many years of my working experience and research over it.

So, the next question is who this book is for? This book is best suited for beginners and professionals. It is intended for anyone who so far has not engaged seriously in SQL and SQL Server and would like to start a career in Database. This book covers the mainly following topics.

- DBMS and RDBMS Core concepts
- Various T-SQL Queries asked in interviews
- Database Constraints like a Primary key, foreign key, unique key etc.
- Various Statements including DDL, DML, TCL etc.
- Query multiple tables by using joins, subqueries, table expressions, and set operators
- Use transactions in a concurrent environment
- Get started with programmable objects—from variables and batches to user-defined functions, stored procedures, cursor, triggers etc.

This book is not only the SQL Server interview book but it is more than that. This book helps you to get an in-depth knowledge of SQL Server with a simple and elegant way.

To get the latest information on SQL Server, I encourage you to follow the official Microsoft document website at <https://www.microsoft.com/en-us/sql-server/sql-server-2017>.

All the best for your interview and happy programming!

About the Author

Shailendra Chauhan - An Entrepreneur, Author, Architect, Corporate Trainer, and Microsoft MVP



He is the **Founder and CEO** of DotNetTricks which is a brand when it comes to e-Learning. DotNetTricks provides training and consultation over an array of technologies like **Cloud, .NET, Angular, React, Node and Mobile Apps development**. He has been awarded as **Microsoft MVP** three times in a row (2016-2018).

He has changed many lives from his writings and unique training programs. He has a number of most sought-after books to his name which have helped job aspirants in **cracking tough interviews** with ease.

Moreover, and to his credit, he has delivered **1000+ training sessions** to professionals worldwide in Microsoft .NET technologies and other technologies including JavaScript, AngularJS, Node.js, React and NoSQL Databases. In addition, he provides **Instructor-led online training, hands-on workshop** and **corporate training** programs.

Shailendra has a strong combination of **technical skills and solution development for complex application architecture with proven leadership and motivational skills** have elevated him to world-renowned status, placing him at the top of the list of most sought-after trainers.

"I always keep up with new technologies and learning new skills to deliver the best to my students," says Shailendra Chauhan, he goes on to acknowledge that the betterment of his followers and enabling his students to realize their goals are his prime objective and a great source of motivation and satisfaction.

Shailendra Chauhan - **"Follow me and you too will have the key that opens the door to success"**

How to Contact Us

Although the author of this book has tried to make this book as accurate as it possible but if there is something strikes you as odd, or you find an error in the book please drop a line via e-mail.

The e-mail addresses are listed as follows:

- mentor@dotnettricks.com
- info@dotnettricks.com

We are always happy to hear from our readers. Please provide your valuable feedback and comments!

You can follow us on [YouTube](#), [Facebook](#), [Twitter](#), [LinkedIn](#) and [Google Plus](#) or subscribe to [RSS feed](#).



Table of Contents

SQL Server Interview Questions & Answers	1
Release History	1
About Dot Net Tricks	2
Dot Net Tricks Training Solutions	2
Dedication	4
Introduction	5
About the Author	6
How to Contact Us	7
Introducing SQL Server	13
Q1. What is SQL Server?	13
Q2. What are the different versions of SQL Server?	13
Q3. What is the default running port for a default SQL Server instance?	13
Q4. What is the various system-defined databases in SQL Server?	13
Q5. What is a master database in SQL Server?	14
Q6. What is the TempDB database in SQL Server?	14
Q7. What is Model database in SQL Server?	14
Q8. What is the MSDB database?	14
Q9. What is the Resource database?	15
Q10. Can you run SQL Server on Linux?	15
Q11. What new features are coming to SQL Server 2019?	15
Q12. Can you access or query remote SQL Server database from a Mac, Linux or Ubuntu machine?	15
Q13. What is Azure Data Studio?	15
Database Basics	16
Q1. What is the difference between DBMS and RDMS?	16
Q2. What is normalization?	16
Q3. What are the different normal forms?	16
Q4. Do explain each normal form?	17
Q5. What are the differences between char and nchar?	22
Q6. What are the differences between varchar and nvarchar?	23
Q7. What are the differences between varchar(MAX) and text?	23

Q8.	What is SQL key?	23
Q9.	What are different types of SQL Keys?.....	23
Q10.	Can you define all the keys in a database table?	24
Q11.	What are the differences between Primary Key and Unique Key?.....	24
Q12.	How to define Primary Key and Unique Key?	25
Q13.	What is the difference between Primary Key and Foreign Key?.....	25
Q14.	How to define Primary Key and Foreign Key?	25
Q15.	What are SQL Commands?	26
Q16.	What are different types of SQL Commands?.....	26
Q17.	What are DDL Commands?	26
Q18.	What are DML Commands?.....	27
Q19.	What are DQL Commands?	27
Q20.	What are TCL Commands?	27
Q21.	What are DCL Commands?.....	27
Q22.	What is Grant and Revoke Commands?	27
Q23.	What are SQL Constraints or SQL Integrity Constraints?	27
Q24.	What are different types of SQL Constraints or SQL Integrity Constraints?	27
Q25.	What is Database Table?	29
Q26.	What is View?.....	29
Q27.	What is the need for a SQL view?.....	30
Q28.	What are different types of SQL Server Views?	30
Q29.	What is Simple View and Complex View?	31
Q30.	What is a transaction?	31
Q31.	Why use transaction in SQL Server?.....	31
Q32.	What are different types of transaction in SQL Server?.....	31
Q33.	What are implicit and explicit transactions in SQL Server?.....	31
Q34.	How to use an explicit transaction in SQL Server?	32
SQL Queries		33
Q1.	Write SQL Query to calculate running total, the total of a column and row?	33
Q2.	How to insert values to an identity column in SQL Server?	34
Q3.	How to reseed the identity column in SQL Server?.....	34

Q4.	Write a query to swap the values of two columns in SQL Server?.....	35
Q5.	Write SQL queries to drop all tables, procedure, views and triggers in SQL Server?.....	35
Q6.	Write SQL queries to get the nth highest and lowest salary of an employee?	36
Q7.	Write SQL query to get field name, data type and size of a database table?	36
Q8.	Write a SQL query to remove duplicate records from a table in SQL Server?	37
Q9.	Write SQL query to create a comma-separated list from the column?	38
SQL Statements		39
Q1.	What are SQL Joins?	39
Q2.	What are different types of SQL Joins?	39
Q3.	Describe each type of SQL Join?	39
Q4.	What is Self-Join?	41
Q5.	What is the difference between inner join and equi-join?	42
Q6.	What is Natural Join?.....	43
Q7.	How Group by and Having Clause are related to each other?	43
Q8.	What is Rollup operator in SQL Server?	44
Q9.	What are SQL injection attacks?.....	44
Q10.	What are the recommended solutions to stop SQL Injection Attacks?	45
Functions, Procedures and Exceptions.....		46
Q1.	What is SQL function?	46
Q2.	What are the different types of SQL Server functions?	46
Q3.	What are the System Defined functions?.....	46
Q4.	What is a User Defined function?.....	47
Q5.	What are the limitations of SQL function?	48
Q6.	What are the stored procedures?	48
Q7.	What are different types of SQL Server stored procedure?.....	48
Q8.	What are the limitations of SQL Server stored procedure?	49
Q9.	What is the difference between Stored Procedure and Function in SQL Server?	50
Q10.	What is stored procedure plan recompilation and performance tuning?	50
Q11.	What are various ways to recompile Stored Procedure?.....	50
Q12.	What is Exception?	52
Q13.	What are different types of exception in SQL Server?	52

Q14.	What is Statement-Level exception in SQL Server?	53
Q15.	What is a batch-Level exception in SQL Server?	53
Q16.	What is Parsing and Scope-Resolution exception in SQL Server?	53
Q17.	How to try...catch works in SQL Server?.....	54
Q18.	What are the limitations of try...catch in SQL Server?	55
Q19.	What are different error functions used in SQL Server for error handling?	55
Q20.	What is case expression in SQL Server?	55
Q21.	What are the different ways to write case expression in SQL Server?	55
Q22.	When to use case expression in SQL Server?	56
Triggers and Tables		57
Q1.	What are triggers?	57
Q2.	Write syntax to create a trigger in SQL Server?	57
Q3.	What are different types of triggers in SQL Server?.....	58
Q4.	What are DDL triggers in SQL Server?	58
Q5.	What are DML triggers in SQL Server?	58
Q6.	What are different types of DML triggers in SQL Server?	58
Q7.	What are CLR triggers in SQL Server?.....	58
Q8.	What are Logon triggers in SQL Server?	58
Q9.	What are the limitations of triggers in SQL Server?	59
Q10.	What are different ways for setting triggers firing order in SQL Server?.....	59
Q11.	What are logical tables in SQL Server?	59
Q12.	What are inserted and deleted logical tables in SQL Server?	59
Q13.	When Inserted and Deleted Logical table are used in SQL Server?	61
Q14.	What is CTE (Common Table Expressions) in SQL Server?	61
Q15.	What are temporary tables?	62
Q16.	What are different types of temporary tables?	62
Q17.	What is the table variable?.....	63
Q18.	What is a local and global variable in SQL Server?	63
Cursors and Indexes		64
Q1.	What is cursor?.....	64
Q2.	Explain the life cycle of the cursor in SQL Server?	64

Q3.	What are different types of Cursor in SQL Server?	65
Q4.	What is Static Cursor in SQL Server?	65
Q5.	How to write a Static Cursor in SQL Server?	65
Q6.	What are the limitations of a Static Cursor in SQL Server?	66
Q7.	What is Dynamic Cursor in SQL Server?	66
Q8.	How to write a Dynamic Cursor in SQL Server?	66
Q9.	What is Forward Only Cursor in SQL Server?	67
Q10.	How to write a Forward only Cursor in SQL Server?	67
Q11.	What is Keyset-Driven Cursor in SQL Server?.....	68
Q12.	How to write a Keyset-driven Cursor in SQL Server?	69
Q13.	What are SQL Server cursor alternatives?.....	69
Q14.	How XML data type works in SQL Server?.....	70
Q15.	What are the limitations of the XML data type in SQL Server?.....	70
Q16.	What are various ways for querying XML data?	70
Q17.	What is an index?	72
Q18.	What are the advantages of the index?	72
Q19.	What are different types of indexes?.....	72
Q20.	What is a Clustered index?	72
Q21.	What is a Non-Clustered index?	72
Q22.	What is full-text search in SQL Server?	73
References		74

Introducing SQL Server

Q1. What is SQL Server?

Ans. Microsoft SQL Server is a Relational Database Management System (RDBMS) developed by Microsoft. It is designed to run on a central server so that multiple users can access the same data simultaneously. Generally, users access the database through an application.

Q2. What are the different versions of SQL Server?

Ans. There are the following versions of SQL Server have been released at the time of writing this book:

Version	Year	Release Name
8.0	2000	SQL Server 2000
8.0	2003	SQL Server 2000 (64-bit)
9.0	2005	SQL Server 2005
10.0	2008	SQL Server 2008
10.5	2010	SQL Server 2008 R2
11.0	2012	SQL Server 2012
12.0	2014	SQL Server 2014
13.0	2016	SQL Server 2016
14.0	2017	SQL Server 2017

Q3. What is the default running port for a default SQL Server instance?

Ans. The default running port number for the default instance of SQL Server is 1433. The named instances can be configured on dynamic ports.

Q4. What is the various system-defined databases in SQL Server?

Ans. SQL Server 2008 and 2005 have five system defined databases: master, model, tempdb, msdb, and resource. These databases are used by SQL Server for its own operation and management. There are following system-defined databases in SQL Server:

- Master
- TempDB
- Model
- MSDB
- Resource DB

Q5. What is a master database in SQL Server?

Ans. The Master database contains the system level information like system configuration details, database information, login account information. login user details for SQL Server. The following database tables are generally used for checking all these details.

```
Select * from SYSCONFIGURES -- for System configuration
Select * from SYSDATABASES -- for database information
Select * from SYSLOGINS -- for login account details
Select * from SYSUSERS -- for user details
```

Primary data of Master database is stored in master.mdf file which default size is 11 MB and Master database log are stored in masterlog.ldf file which default size is 1.25 MB.

Q6. What is the TempDB database in SQL Server?

Ans. TempDB database includes information about all the temporary objects such as temporary tables. The temporary objects are created by preceding hash “#” symbol with its name.

For Example - #Student table gets created as a temporary table in the school database.

```
Use School
CREATE TABLE #Student
(
  StudentID int
  Name Varchar(50)
)
```

TempDB database is recreated every time when the instance of SQL Server has been restarted. Hence, every time temporary objects will be removed from the tempDB database.

Primary data of the tempDB database is stored in tempDB.mdf file which default size is 8 MB and tempDB database log is stored in templog.ldf file which default size is 0.5MB.

Q7. What is Model database in SQL Server?

Ans. The model database acts as a template for all the databases created in a SQL Server. It is useful to copy some commonly used database objects like as tables, functions, stored procedure into the newly created database. These commonly used database objects we create in the model database.

Primary data of the model database is stored in the model.mdf file which default size is 0.75 MB and model database log is stored in modellog.ldf which default size is 0.75MB.

Q8. What is the MSDB database?

Ans. MSDB database is used by the SQL Server Agent to schedule alerts, jobs and to record operators.

For Example- When you create a database maintenance plan for taking backup of a specific database on a daily basis, then this scheduled job will be stored in system-defined table “SYSJOBS” in MSDB database.

Primary data of Msdb database is stored in msdbdata.mdf file which default size is 12 MB and Msdb database log are stored in msdblog.ldf which default size is 2.25MB.

Q9. What is the Resource database?

Ans. A Resource database is hidden from user view like as another system defined database. This is read-only database containing all the system-defined objects for a SQL Server. A Resource database file can be seen in your system by navigating to the location C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn.

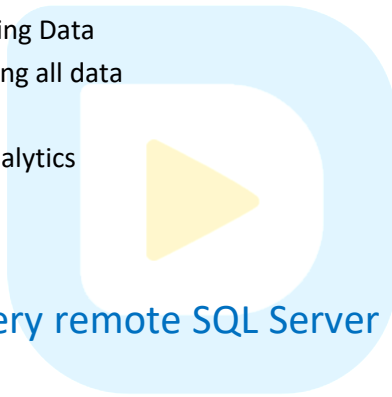
Q10. Can you run SQL Server on Linux?

Ans. From SQL Server 2017, you can run SQL Server on Linux OS. But, SQL Server version from 2000 to 2016, you can't run on Linux. They only supported by Windows OS.

Q11. What new features are coming to SQL Server 2019?

Ans. At the time of writing this book, SQL Server 2019 RTM was not released. The following features are coming in SQL Server 2019 release.

- The single virtual data layer
- Data virtualization and Integrating Data
- No data replication and Managing all data
- Spark Built-In
- Unified platform for big data analytics
- Spark jobs
- Train machine learning models



Q12. Can you access or query remote SQL Server database from a Mac, Linux or Ubuntu machine?

Ans. Yes, you can connect or query your remote SQL Server database from your Mac, Linux or Ubuntu machines using Azure Data Studio tool.

Q13. What is Azure Data Studio?

Ans. Azure Data Studio is an alternative way to SQL Server Management Studio (SSMS) which you can run only on windows machines to query, editing and data development tasks.

Azure Data Studio offers a modern editor experience to connect with a remote SQL Server database. It helps us to query and manage data across multiple sources with intellisense.

2

Database Basics

Q1. What is the difference between DBMS and RDBMS?

Ans. DBMS and RDBMS, both are used to store, manage and query the data. But both have some important differences as listed below-

DBMS (Database Management System)	RDBMS (Relational Database Management System)
DBMS stands for Database Management System and treats data as files internally.	RDBMS stands for Relational Database Management System and treats data as relations means tables.
It defines the relationship between the files programmatically.	It defines the relationship between the relations called tables at the time of table creation.
It does not impose any constraints or security with regard to data manipulation.	It imposes constraints or security with regard to data manipulation.
It does not support distributed architecture.	It supports distributed architecture
It does not support Client-Server architecture	It supports Client-Server Architecture
Only one user can access data at a time	Multiple users can access the data at the same time
It satisfies maximum 6 to 7 rules of E.F. Codd (Edgar Frank "Ted" Codd) out of 12 rules.	It satisfies more than 6 to 7 rules of E.F. Codd out of 12 rules.
Example- File System, XML, FoxPro, IMS	Example – SQL Server, Oracle, DB2, MySQL

Q2. What is normalization?

Ans. Normalization or data normalization is a process to organize the data into a tabular format i.e. database tables. It mainly used for implementing the following two things.

- **Reducing data redundancy.**
- **Ensuring data dependency.**

A good database design implements normalization practices to make a database system fast, efficient and to produce the expected result.

Q3. What are the different normal forms?

Ans. We organize the data into database tables by using normal forms rules or conditions. Normal forms help us to make a good database design. There are following normal forms:

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. BCNF
5. Fourth Normal Form (4NF)
6. Fifth Normal Form (5NF)

Generally, we organize the data up to third normal form. We rarely use the fourth and fifth normal form.

Q4. Do explain each normal form?

Ans. To understand normal forms, consider the following un-normalized database table. Now we will normalize the data of below table using normal forms.

Project Code	Project Name	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	Reservation System	Mr. Ajay	120500	S100	Mohan	D03	Database	21.00
PC010	Reservation System	Mr. Ajay	120500	S101	Vipul	D02	Testing	16.50
PC010	Reservation System	Mr. Ajay	120500	S102	Riyaz	D01	IT	22.00
PC011	HR System	Mrs. Charu	500500	S103	Pavan	D03	Database	18.50
PC011	HR System	Mrs. Charu	500500	S104	Jitendra	D02	Testing	17.00
PC011	HR System	Mrs. Charu	500500	S315	Pooja	D01	IT	23.50
PC012	Attendance System	Mr. Rajesh	710700	S137	Rahul	D03	Database	21.50
PC012	Attendance System	Mr. Rajesh	710700	S218	Avneesh	D02	Testing	15.50
PC012	Attendance System	Mr. Rajesh	710700	S109	Vikas	D01	IT	20.50

UNF

First Normal Form (1NF) - A database table is said to be in 1NF if it contains no repeating fields/columns. The process of converting the UNF table into 1NF is as follows:

- Separate the repeating fields into the new database tables along with the key from a un-normalized database table.
- The primary key of new database tables may be a composite key

1NF of above UNF table is as follows:

Primary Key

Project Code	Project Name	Project Manager	Project Budget
PC010	Reservation System	Mr. Ajay	120500
PC011	HR System	Mrs. Charu	500500
PC012	Attendance System	Mr. Rajesh	710700

Composite Key (Unique Key)

Project Code	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	S100	Mohan	D03	Database	21.00
PC010	S101	Vipul	D02	Testing	16.50
PC010	S102	Riyaz	D01	IT	22.00
PC011	S103	Pavan	D03	Database	18.50
PC011	S104	Jitendra	D02	Testing	17.00
PC011	S315	Pooja	D01	IT	23.50
PC012	S137	Rahul	D03	Database	21.50
PC012	S218	Avneesh	D02	Testing	15.50
PC012	S109	Vikas	D01	IT	20.50

1NF

Second Normal Form (2NF) - A database table is said to be in 2NF if it is in 1NF and contains only those fields/columns that are functionally dependent (means the value of the field is determined by the value of another field(s)) on the primary key. In 2NF we remove the partial dependencies of any non-key field.

The process of converting the database table into 2NF is as follows:

- Remove the partial dependencies (A type of functional dependency where a field is only functionally dependent on the part of primary key) of any non-key field.
- If field B depends on field A and vice versa. Also, for a given value of B, we have only one possible value of A and vice versa, Then we put the field B into new database table where B will be the primary key and also marked as a foreign key in the parent table.

2NF of above 1NF tables is as follows:

Primary Key

Project Code	Project Name	Project Manager	Project Budget
PC010	Reservation System	Mr. Ajay	120500
PC011	HR System	Mrs. Charu	500500
PC012	Attendance System	Mr. Rajesh	710700

Composite Key

Project Code	Employee No.	Hourly Rate
PC010	S100	21.00
PC010	S101	16.50
PC010	S102	22.00
PC011	S103	18.50
PC011	S104	17.00
PC011	S315	23.50
PC012	S137	21.50
PC012	S218	15.50
PC012	S109	20.50

Primary Key

Employee No.	Employee Name	Department No.	Department Name
S100	Mohan	D03	Database
S101	Vipul	D02	Testing
S102	Riyaz	D01	IT
S103	Pavan	D03	Database
S104	Jitendra	D02	Testing
S315	Pooja	D01	IT
S137	Rahul	D03	Database
S218	Avneesh	D02	Testing
S109	Vikas	D01	IT

2NF

Third Normal Form (3NF) - A database table is said to be in 3NF if it is in 2NF and all non-keys fields should be dependent on primary key or We can also say a table to be in 3NF if it is in 2NF and no fields of the table is transitively dependent on the primary key. The process of converting the table into 3NF is as follows:

- Remove the transitive dependencies (A type of functional dependency where a field is functionally dependent on the Field that is not the primary key. Hence its value is determined, indirectly by the primary key).
- Make a separate table for transitive dependent Field.

3NF of above 2NF tables is as follows:

Primary Key

Project Code	Project Name	Project Manager	Project Budget
PC010	Reservation System	Mr. Ajay	120500
PC011	HR System	Mrs. Charu	500500
PC012	Attendance System	Mr. Rajesh	710700

Composite Key

Project Code	Employee No.	Hourly Rate
PC010	S100	21.00
PC010	S101	16.50
PC010	S102	22.00
PC011	S103	18.50
PC011	S104	17.00
PC011	S315	23.50
PC012	S137	21.50
PC012	S218	15.50
PC012	S109	20.50

Primary Key

Employee No.	Employee Name	Department No.
S100	Mohan	D03
S101	Vipul	D02
S102	Riyaz	D01
S103	Pavan	D03
S104	Jitendra	D02
S315	Pooja	D01
S137	Rahul	D03
S218	Avneesh	D02
S109	Vikas	D01

Primary Key **FK_Relationship**

Department No.	Department Name
D01	IT
D02	Testing
D03	Database

3NF

Boyce Code Normal Form (BCNF) - A database table is said to be in BCNF if it is in 3NF and contains each and every determinant as a candidate key. The process of converting the table into BCNF is as follows:

- Remove the non-trivial functional dependency.
- Make a separate table for the determinants.

BCNF of below table is as follows:

Supplier ID	Supplier Name	Product ID	Quantity
S001	Mr. X	P001	120
S002	Mr. Y	P002	102
S003	Mr. Z	P001	100

Determinants

Supplier ID	Supplier Name
S001	Mr. X
S002	Mr. Y
S003	Mr. Z

Supplier ID	Product ID	Quantity
S001	P001	120
S002	P002	102
S003	P001	100

BCNF

Fourth Normal Form (4NF) - A database table is said to be in 4NF if it is in BCNF and primary key has a one-to-one relationship to all non-keys fields or We can also say a table to be in 4NF if it is in BCNF and contains no multi-valued dependencies. The process of converting the table into 4NF is as follows:

- Remove the multivalued dependency.
- Make a separate table for multivalued Fields.

4NF of below table is as follows:

Employee Name	Skills	Language
Mohan	C Sharp	Hindi
Mohan	Asp.Net	Hindi
Mohan	SQL Server	Hindi
Mohan	C Sharp	English
Mohan	Asp.Net	English
Mohan	SQL Server	English

Employee Name	Skills
Mohan	C Sharp
Mohan	Asp.Net
Mohan	SQL Server

4NF

Employee Name	Language
Mohan	Hindi
Mohan	English

Fifth Normal Form (5NF) - A database table is said to be in 5NF if it is in 4NF and contains no redundant values or We can also say a table to be in 5NF if it is in 4NF and contains no join dependencies. The process of converting the table into 5NF is as follows:

- Remove the join dependency.
- Break the database table into smaller and smaller tables to remove all data redundancy.

5NF of below table is as follows:

Company	Product	Supplier
Godrej	Soap	Mr. Amit
Godrej	Shampoo	Mr. Pavan
Godrej	Shampoo	Mr. Amit
H.Lever	Soap	Mr. Amit
H.Lever	Shampoo	Mr. Pavan
H.Lever	Soap	Mr. Sachin

Company	Product
Godrej	Soap
Godrej	Shampoo
H.Lever	Soap
H.Lever	Shampoo

Company	Supplier
Godrej	Mr. Amit
Godrej	Mr. Pavan
H.Lever	Mr. Amit
H.Lever	Mr. Pavan
H.Lever	Mr. Sachin

Product	Supplier
Soap	Mr. Amit
Shampoo	Mr. Pavan
Shampoo	Mr. Amit
Soap	Mr. Sachin

5NF

Q5. What are the differences between char and nchar?

Ans. These data type is used to stores characters but these are different in many cases as given below:

char - This is a fixed length characters data type. It takes one byte per character and used to store non-Unicode characters.

Suppose, you declare a field with *char(20)* then it will allocate memory for 20 characters whether you are using only 10 characters. Hence memory for 10 characters which is empty will be wasted.

nchar - This is like as char data type but it takes two bytes per character and used to store Unicode characters means multiple languages (like Hindi, Chinese etc.) characters in the database.

Q6. What are the differences between varchar and nvarchar?

Ans. There are following differences between varchar and nvarchar:

varchar - This is a variable length characters data type. It takes one byte per character and can store non-Unicode characters (like English). This data type allocates the memory based on a number of characters inserted. Hence, no wastage of memory.

nvarchar – This is like as char data type but it takes 2 bytes per character and used to store Unicode characters means multiple languages (like Hindi, Chinese etc.) characters in the database.

Q7. What are the differences between varchar(MAX) and text?

Ans. VARCHAR(MAX) and TEXT both are used to store larger texts in the database. The varchar (MAX) was introduced with SQL Server 2005 as a replacement for TEXT.

The TEXT type always stores the data in a blob while the VARCHAR(MAX) type store the data directly into the table row. When stored data size exceeds the 8000 characters in VARCHAR(MAX), it stores the data in a blob just like TEXT type.

The TEXT is used when you do not require search on the value of a column and you don't have any plan to use it in the join. But VARCHAR (MAX) column can be used in join and you can do a search on the column value.

Q8. What is SQL key?

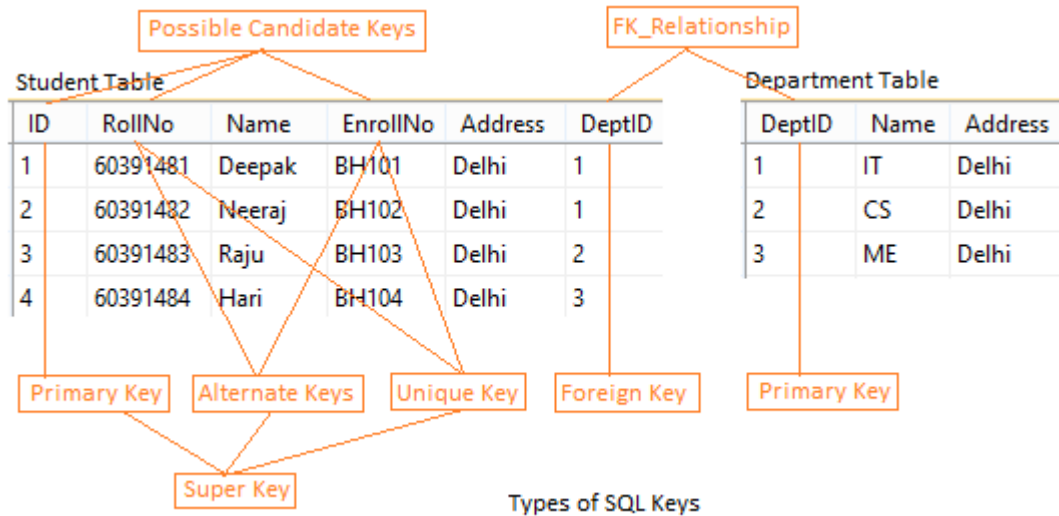
Ans. A key can be a single field/column or a combination of multiple fields/columns in a table. It is used to retrieve records or data-rows from data table based on the condition. Keys are also used to create a relationship among different database tables or views.

Q9. What are different types of SQL Keys?

Ans. There are following types of SQL Keys –

1. **Super Key** - A Super key is a set of one or more than one keys that can be used to identify a record uniquely in a database table.
For Example: Primary key, Unique key, Alternate key are a subset of Super Keys.
2. **Candidate Key** - A Candidate key is a set of one or more fields/columns that can identify a record uniquely in a database table. There can be multiple Candidate keys in one database table and each of them can work as a Primary Key.
For Example: In below diagram ID, RollNo and EnrollNo are Candidate Keys since all these three fields can be work as Primary Key.
3. **Primary Key** - A Primary key is a set of one or more fields/columns in a database table that can identify a record uniquely. It cannot accept null and duplicate values. A Candidate Key can become a Primary Key.
4. **Alternate key** - An Alternate key is a key that can work as a primary key if needed. Basically, it is a candidate key that currently not a primary key.

For Example: In the below diagram, *RollNo* and *EnrollNo* become Alternate Keys when you define ID as Primary Key.



- Composite/Compound Key** - A Composite key is a combination of more than one fields/columns of a database table. It can be a Candidate key, Primary key.
- Unique Key** - A Unique key is a set of one or more fields/columns of a database table that can identify a record uniquely. It is like a Primary key but it can accept only one null value. It cannot have duplicate values.
- Foreign Key** - A Foreign Key is a field/column in a database table that is Primary key in another database table. Unlike, primary key it can accept multiple null values and duplicate values.

For Example: You can have a *DeptID* column in the *Employee* table which is pointing to the *DeptID* column in a department table where it is a primary key.

Q10. Can you define all the keys in a database table?

Ans. Practically in a database table, you can have only three types of keys: Primary Key, Unique Key and Foreign Key. Other types of keys are only concepts of RDBMS that you need to know.

Q11. What are the differences between Primary Key and Unique Key?

Ans. In SQL Server, we have two keys which distinctively or uniquely identify a record in the database. Both the keys seem identical, but actually, both are different in features and behaviours.

Primary Key	Unique Key
Primary Key can't accept null values.	The unique key can accept only one null value.
By default, Primary key is clustered index and data in the database table is physically organized in the sequence of a clustered index.	By default, Unique key is a unique non-clustered index.

You can have only one Primary key (it may be a composite primary key means key on multiple fields) in a table.	You can have more than one unique key in a table.
The primary key can be made foreign key into another table.	In SQL Server, Unique key can be made foreign key into another table.

Q12. How to define Primary Key and Unique Key?

Ans. The code for defining the Primary Key and Unique Key is given below:

Example for defining a Primary key and Unique key

```
CREATE TABLE Employee
(
    EmpID int PRIMARY KEY, --define primary key
    Name varchar (50) NOT NULL,
    MobileNo int UNIQUE, --define unique key
    Salary int NULL
)
```

Q13. What is the difference between Primary Key and Foreign Key?

Ans. The difference between Primary Key and Foreign Key is given below-

Primary Key	Foreign Key
A primary key uniquely identifies a record in the table	A foreign key is a field in the table that is the primary key in another table.
Primary Key can't accept null values.	A foreign key can accept multiple null values.
By default, Primary key is clustered index and data in the database table is physically organized in the sequence of the clustered index.	A foreign key does not automatically create an index, clustered or non-clustered. You can manually create an index on the foreign key.
You can have only one Primary key in a table.	You can have more than one foreign key in a table.

Q14. How to define Primary Key and Foreign Key?

Ans. The code for defining the Primary Key and Foreign Key is given below:

Example for defining Primary and Foreign Key

```
--Create Parent Table
CREATE TABLE Department
(
    DeptID int PRIMARY KEY, --define a primary key
    Name varchar (50) NOT NULL
)
GO
--Create Child Table
CREATE TABLE Employee
(
    EmpID int PRIMARY KEY, --define a primary key
    Name varchar (50) NOT NULL,
```

```
Salary int NULL,  
DeptID int FOREIGN KEY REFERENCES Department(DeptID) --define a foreign key  
)
```

Q15. What are SQL Commands?

Ans. SQL commands are a set of instructions that are used to interact with the database like SQL Server, MySQL and Oracle etc. SQL commands are responsible to create and to do all the manipulation on the database. These are also responsible to give or take out access rights on a particular database.

Q16. What are different types of SQL Commands?

Ans. We have different SQL commands for the different-different purpose. We can group SQL Commands into five major categories depending on their functionality.

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Query Language (DQL)
4. Transaction Control Language (TCL)
5. Data Control Language (DCL)

Q17. What are DDL Commands?

Ans. DDL commands are used to create, modify, and drop the structure of database objects like table, view, procedure, indexes etc. DDL commands are: CREATE, ALTER, DROP and TRUNCATE.

For Example:

```
--Here "TABLE" is a keyword that is used to create table "TABLE_NAME"  
CREATE TABLE TABLE_NAME  
(  
  COL1 VARCHAR(10),  
  COL2 VARCHAR(20),  
);  
  
--Here "VIEW" is a keyword that is used to create VIEW "VIEW_NAME"  
CREATE VIEW VIEW_NAME  
AS  
BEGIN  
  SELECT * FROM TABLE_NAME  
END;
```

Note

- Only with DDL commands you need to write keyword (like table, procedure, view, index, function) with the syntax of the command.
- These commands are used only to create/modify the structure of the database object.

Q18. What are DML Commands?

Ans. DML commands are used to store, modify, and delete data from database tables. The DML commands are INSERT, UPDATE, and DELETE commands.

Q19. What are DQL Commands?

Ans. DQL commands are used to fetch/retrieve data from database tables. There are only one DQL command that is a SELECT command.

Q20. What are TCL Commands?

Ans. TCL commands are used to handle changes which affect the data in the database. Basically, these commands are used with the transaction or used to make a stable point during changes in the database at which You can roll back the database state if needed. The TCL commands are SAVEPOINT, ROLLBACK and COMMIT.

Q21. What are DCL Commands?

Ans. DCL commands are used to provide access or to provide authorization access on the database objects like table, view, stored procedure etc. The DCL commands are: GRANT and REVOKE.

Q22. What is Grant and Revoke Commands?

Ans. The Grant and Revoke commands are explained as:

- **Grant Command:** This command is used to give permission to specific users on specific database objects like table, view etc.
- **Revoke Command:** This command is used to take out permission from specific users on specific database objects like table, view etc.

Q23. What are SQL Constraints or SQL Integrity Constraints?

Ans. Constraints are some rules that enforce on the data to be entered into the database table. Basically, constraints are used to restrict the type of data that can insert into a database table. Constraints can be defined at two levels as given below-

1. **Column Level** - These constraints are defined with the column definition inside a CREATE TABLE statement.
2. **Table Level** - These constraints are defined after the table created using the ALTER TABLE statement.

Q24. What are different types of SQL Constraints or SQL Integrity Constraints?

Ans. In SQL Server, there are six types of SQL Constraints as given below:

1. **Primary Key Constraints** - A Primary key is defined as one or more fields/columns of a database table that identify each record in a database table uniquely. It cannot accept null and duplicate values.
 - Primary key constraint at column level

```
CREATE TABLE table_name
```

```
(
col1 <datatype> [CONSTRAINT constraint_name] PRIMARY KEY,
col2 <datatype>
);
```

- Primary key constraint at table level

```
ALTER TABLE table_name
ADD[CONSTRAINT constraint_name] PRIMARY KEY (col1,col2)
```

2. **Unique Key Constraints** - A Unique key is a set of one or more fields/columns of a database table that can identify a record uniquely. It is like a Primary key but it can accept only one null value. It cannot have duplicate values.

- Unique key constraint at the column level

```
CREATE TABLE table_name
(
col1 <datatype> [CONSTRAINT constraint_name] UNIQUE,
col2 <datatype>
);
```

- Unique key constraint at table level

```
ALTER TABLE table_name
ADD[CONSTRAINT constraint_name] UNIQUE (col1,col2)
```

3. **Foreign Key Constraints** - A Foreign Key is a field/column in a database table that is Primary key in another database table. Unlike, primary key it can accept multiple null values and duplicate values.

- Foreign key constraint at column level

```
CREATE TABLE table_name
(
col1 <datatype> [CONSTRAINT constraint_name] REFERENCES
referenced_table_name(referenced_table_column_name),
col2 <datatype>
)
```

- Foreign key constraint at table level

```
ALTER TABLE table_name
ADD[CONSTRAINT constraint_name] REFERENCES
referenced_table_name(referenced_table_col)
```

4. **Not Null Constraints** - This constraint ensures a field/column cannot accept a null value.

- Not Null constraint at column level

```
CREATE TABLE table_name
(
col1 <datatype> [CONSTRAINT constraint_name] NOT NULL,
col2 <datatype>
);
```

- Not Null constraint at table level

```
ALTER TABLE table_name
ALTER COLUMN col1 <datatype> NOT NULL
```

5. **Check Constraints** - This constraint is used to define a business rule at a field/column in the database table that each record in the database table must follow.

- Check constraint at column level

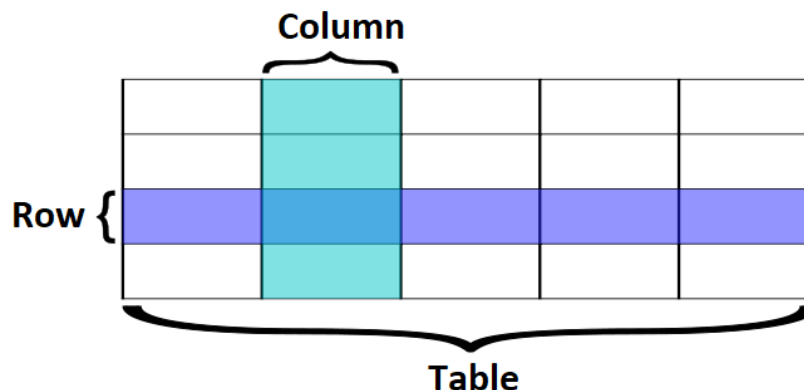
```
CREATE TABLE table_name
(
col1 <datatype> [CONSTRAINT constraint_name] CHECK (<condition>),
col2 <datatype>
);
```

- Check constraint at table level

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name
CHECK(<condition>)
```

Q25. What is Database Table?

Ans. An RDBMS store the data using one than a database table. A database table manages the data in row and columns format. Each row in a table has its own primary key which uniquely identifies that row or record. The data associated with tables are physically stored in the database memory.



Q26. What is View?

Ans. Views are virtual tables that are compiled at runtime. The data associated with views are not physically stored in the view, but it is stored in the base tables of the view. A view can be made over one or more database

tables. Generally, we put those columns in view that we need to retrieve/query again and again. Once you have created the view, you can query view like as table. We can make an index, trigger on the view.

Q27. What is the need for a SQL view?

Ans. In SQL Server we make views for security purpose since it restricts the user to view some columns/fields of the table(s). Views show only those columns that are present in the query which is used to make a view. One more advantage of Views is, data abstraction since the end user is not aware of all the data present in the table.

Syntax:

```
CREATE VIEW view_name
AS
select_statement[]
```

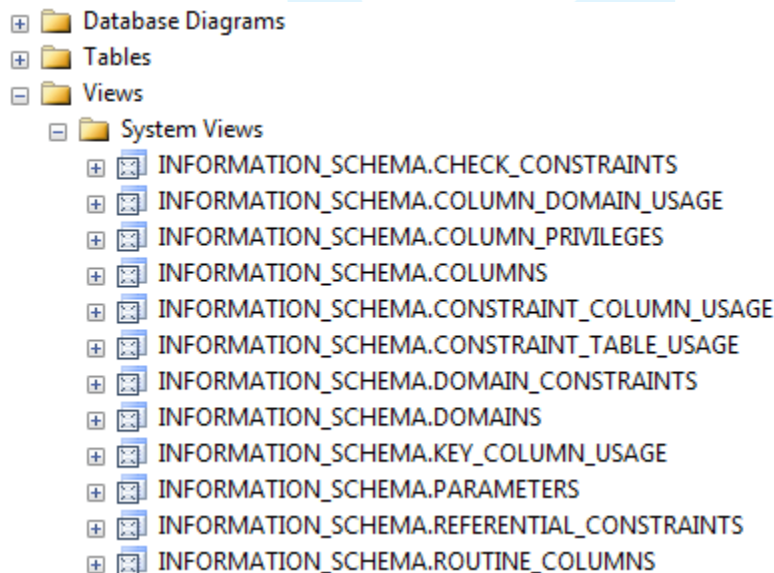
Key points about Views:

- We make views for security purpose since it restricts the user to view some columns/fields of the table(s).
- One more advantage of Views is, data abstraction since the end user is not aware of all the data present in database table

Q28. What are different types of SQL Server Views?

Ans. There are two types of View as given below –

1. **System Defined Views-** System defined Views are predefined Views that already exist in the Master database of SQL Server. These are also used as template Views for all newly created databases. These system Views will be automatically attached to any user-defined database.



2. **User Defined Views** - These types of view are defined by users. We have two types of user-defined views.
 - a. Simple View
 - b. Complex View

Q29. What is Simple View and Complex View?

Ans. The simple view and complex views are given below:

- **Simple View** - When we create a view on a single table, it is called a simple view.

```
create VIEW vw_Employee_Test
AS
Select Emp_ID ,Emp_Name ,Emp_Designation
From Employee_Test
```

In a simple view, we can insert, update, and delete data. We can only insert data in a simple view if we have a primary key and all not null fields in the view.

- **Complex View** - When we create a view on more than one table, it is called complex view.

```
-- Now create a view on two tables Employee_Test and Personal_Info
Create VIEW vw_Employee_Personal_Info
As
Select e.Emp_ID, e.Emp_Name,e.Emp_Designation,p.DOB,p.Mobile
From Employee_Test e INNER JOIN Personal_Info p
On e.Emp_Name = p. Emp_Name
```

We can only update data in the complex view. We can't insert, delete data in the complex view.

Q30. What is a transaction?

Ans. A transaction is a set of T-SQL statements that are executed together as a unit like as a single T-SQL statement. If all of these T-SQL statements executed successfully, then a transaction is committed and the changes made by T-SQL statements permanently saved to the database. If any of these T-SQL statements within a transaction fail, then the complete transaction is cancelled/ rolled back.

Q31. Why use transaction in SQL Server?

Ans. We use transaction in that case when we try to modify more than one table or views that are related to each other. Transactions affect SQL Server performance greatly. Since, when a transaction is initiated then it locks all the tables' data that are used in the transaction. Hence during the transaction life cycle, no one can modify these tables' data that are used by the transaction. The reason behind the locking of the data is to maintain Data Integrity.

Q32. What are different types of transaction in SQL Server?

Ans. There are following types of transactions in SQL Server as given below:

- Implicit Transaction
- Explicit Transaction

Q33. What are implicit and explicit transactions in SQL Server?

Ans. The implicit transaction and explicit transactions are explained as:

1. **Implicit Transaction** - Implicit transactions are maintained by SQL Server for each and every DDL (CREATE, ALTER, DROP, TRUNCATE), DML (INSERT, UPDATE, DELETE) statements.

All these T-SQL statements run under the implicit transaction. If there is an error occurs within these statements individually, SQL Server will roll back the complete statement.

2. **Explicit Transaction** - Explicit transactions are defined by programmers. In Explicit transaction, we include the DML statements that need to be executed as a unit.

A SELECT statement does not modify data. Hence generally we don't include only a Select statement in a transaction.

Q34. How to use an explicit transaction in SQL Server?

Ans. An explicit transaction can be used in SQL Server as given below:

For Example:

```
BEGIN TRANSACTION trans
BEGIN TRY
    INSERT INTO Department (DeptID, DeptName, Location) VALUES (2, 'HR', 'Delhi')
    INSERT INTO
Employee (EmpID, Name, Salary, Address, DeptID) VALUES (1, 'Mohan', 18000, 'Delhi', 1)
    IF @@TRANCOUNT > 0
        BEGIN COMMIT TRANSACTION trans
        END
END TRY
BEGIN CATCH
    print 'Error Occured'
    IF @@TRANCOUNT > 0
        BEGIN ROLLBACK TRANSACTION trans
        END
END CATCH
```

3

SQL Queries

Q1. Write SQL Query to calculate running total, the total of a column and row?

Ans. Suppose you have the following table in the database.

	OrderID	Amount	OrderDate
1	1	120.12	2013-09-20 22:59:00
2	2	20.12	2013-09-20 22:59:00
3	3	10.12	2013-09-20 22:59:00
4	4	30.12	2013-09-20 22:59:00
5	5	40.00	2013-09-20 22:59:00

Query for Calculating Running Total:

```
select OrderID, OrderDate, CO.Amount
, (select sum(Amount) from CustomerOrders
where OrderID <= CO.OrderID)
'Running Total'
from CustomerOrders CO
```

	OrderID	OrderDate	Amount	Running Total
1	1	2013-09-20 22:59:00	120.12	120.12
2	2	2013-09-20 22:59:00	20.12	140.24
3	3	2013-09-20 22:59:00	10.12	150.36
4	4	2013-09-20 22:59:00	30.12	180.48
5	5	2013-09-20 22:59:00	40.00	220.48

Query for Calculating Final Total:

```
SELECT OrderID, SUM(Amount) AS Amount
FROM CustomerOrders
GROUP BY OrderID WITH ROLLUP
```

	OrderID	Amount
1	1	120.12
2	2	20.12
3	3	10.12
4	4	30.12
5	5	40.00
6	NULL	220.48

Query for Calculating Total of All Numeric columns in a row

```
SELECT OrderID, Amount, SUM(OrderID+Amount) AS RowNumericColSum
FROM CustomerOrders
GROUP BY OrderID, Amount
ORDER BY OrderID
```

	OrderID	Amount	RowNumericColSum
1	1	120.12	121.12
2	2	20.12	22.12
3	3	10.12	13.12
4	4	30.12	34.12
5	5	40.00	45.00

Q2. How to insert values to an identity column in SQL Server?

Ans. An Identity field is an auto-incremented field which is usually used as a primary key. Normally, it can't be inserted but by setting IDENTITY_INSERT ON you can insert value as shown:

```
SET IDENTITY_INSERT Employee ON

INSERT INTO Employee (ID,Name,Address) VALUES(3,'Mohan','Noida')
INSERT INTO Employee (ID,Name,Address) VALUES(4,'Deepak','Delhi')

SET IDENTITY_INSERT Employee OFF
```

After finishing the Insert operation, set IDENTITY_INSERT OFF.

Q3. How to reseed the identity column in SQL Server?

Ans. You can reseed the identity column value by using *checkident()* method.

Suppose you want to reseed the Employee table ID field from 5 then the new records will be inserted with ID 6,7,8..and so on.

```
--Reseeding the identity
DBCC checkident (Employee, RESEED, 5)

INSERT INTO Employee(Name,Address) VALUES('Rama','Delhi')
```

Q4. Write a query to swap the values of two columns in SQL Server?

Ans. For swapping the values of two database columns both the columns must have the same data type. Moreover, the length of the column should be enough to hold the values of swapped column else data will be truncated. Suppose you have the following table with data as given below-

	CustID	Name	Address	ContactNo
1	2	Rahul	Gurgaon	8797897979
2	3	Hans	Noida	9444444444
3	4	Jeetu	Delhi	9560344074
4	7	Tushar	Delhi	9878908978
5	9	Deepak	Noida	9878909878
6	10	Ram	Noida	9878987890
7	11	Saksham	Noida	8988787878

Query to swap the values:

```
UPDATE Customer SET Name=Address, Address=Name
```

	CustID	Name	Address	ContactNo
1	2	Gurgaon	Rahul	8797897979
2	3	Noida	Hans	9444444444
3	4	Delhi	Jeetu	9560344074
4	7	Delhi	Tushar	9878908978
5	9	Noida	Deepak	9878909878
6	10	Noida	Ram	9878987890
7	11	Noida	Saksham	8988787878

Q5. Write SQL queries to drop all tables, procedure, views and triggers in SQL Server?

Ans. The queries are given below-

```
-- drop all user-defined tables
EXEC sp_MSforeachtable @command1 = "DROP TABLE ?"

-- drop all user defined stored procedures
Declare @procName varchar(500)
Declare cur Cursor For Select [name] From sys.objects where type = 'p'
Open cur
Fetch Next From cur Into @procName
While @@fetch_status = 0
Begin
    Exec('drop procedure ' + @procName)
    Fetch Next From cur Into @procName
End
Close cur
```

```

Deallocate cur

-- drop all user defined views
Declare @viewName varchar(500)
Declare cur Cursor For Select [name] From sys.objects where type = 'v'
Open cur
Fetch Next From cur Into @viewName
While @@fetch_status = 0
Begin
    Exec('drop view ' + @viewName)
    Fetch Next From cur Into @viewName
End
Close cur
Deallocate cur

-- drop all user defined triggers
Declare @trgName varchar(500)
Declare cur Cursor For Select [name] From sys.objects where type = 'tr'
Open cur
Fetch Next From cur Into @trgName
While @@fetch_status = 0
Begin
    Exec('drop trigger ' + @trgName)
    Fetch Next From cur Into @trgName
End
Close cur
Deallocate cur

```

Q6. Write SQL queries to get the nth highest and lowest salary of an employee?

Ans. The queries are given below-

Query to get nth (3rd) highest Salary:

```

Select TOP 1 Salary as '3rd Highest Salary'
from (SELECT DISTINCT TOP 3 Salary from Employee ORDER BY Salary DESC)
a ORDER BY Salary ASC

```

Query to get nth (3rd) lowest Salary:

```

Select TOP 1 Salary as '3rd Lowest Salary'
from (SELECT DISTINCT TOP 3 Salary from Employee ORDER BY Salary ASC)
a ORDER BY Salary DESC

```

Q7. Write SQL query to get field name, data type and size of a database table?

Ans. The query is given below-

```

SELECT column_name as 'Column Name', data_type as 'Data Type',
character_maximum_length as 'Max Length'

```

```
FROM information_schema.columns
WHERE table_name = 'tblUsers'
```

	Column Name	Data Type	Max Length
1	UserNo	int	NULL
2	UserID	varchar	100
3	Password	varchar	50
4	FirstName	varchar	50
5	MiddleName	varchar	50
6	LastName	varchar	50
7	Title	varchar	10
8	EmailID	varchar	100
9	DeptID	int	NULL
10	OrgID	int	NULL
11	LocationID	int	NULL
12	UserCategory	smallint	NULL
13	CreationDate	datetime	NULL
14	IsAdmin	bit	NULL
15	IsActive	bit	NULL
16	VerificationCode	varchar	50
17	Designation	varchar	50
18	AuthenticatCode	varchar	50

Q8. Write a SQL query to remove duplicate records from a table in SQL Server?

Ans. Suppose, you have following data in Employee table then query for removing duplicate records is given below-

	EmpID	Name	Salary	Designation
1	1	Amit	12000.00	SE
2	7	Amit	12000.00	SE
3	2	Mohan	15000.00	SE
4	3	Monu	27000.00	SSE
5	6	Monu	27000.00	SSE
6	4	Riyaz	16000.00	SE
7	5	Riyaz	16000.00	SE

Remove Duplicate Records by using ROW_NUMBER()

```
WITH TempEmp (Name,duplicateRecCount)
AS
(
SELECT Name,ROW_NUMBER() OVER(PARTITION by Name, Salary ORDER BY Name)
AS duplicateRecCount
FROM dbo.Employee
)
```

```
--Now Delete Duplicate Records
DELETE FROM TempEmp
WHERE duplicateRecCount > 1

--See affected table
Select * from Employee
```

	EmpID	Name	Salary	Designation
1	1	Amit	12000.00	SE
2	2	Mohan	15000.00	SE
3	3	Monu	27000.00	SSE
4	4	Riyaz	16000.00	SE

Q9. Write SQL query to create a comma-separated list from the column?

Ans. The query is given below-

```
1 GO
2
3 DECLARE @strEmails VARCHAR(MAX)
4
5 SELECT @strEmails = COALESCE(@strEmails+',', '') + EmailID FROM Employee
6
7 PRINT @strEmails
8
9 GO
```

Messages

mohan.kumar@xyz.com;asif.khan@xyz.com;bkshakya@xyz.com;pavan.kumar@xyz.com

Ans. To stop SQL Server for sending unwanted emails we required to clean the unsent mail from database email queue. We can do this by running below queries.

```
-- To get number of unsent emails
select count(*) from msdb.dbo.sysmail_unsentitems;

-- remove all the unsent emails
delete from msdb.dbo.sysmail_unsentitems;
```

Now, all the unexpected emails have been removed from the SQL Server database mail queue.

4

SQL Statements

Q1. What are SQL Joins?

Ans. SQL joins are used to retrieve data from two or more data tables, based on a join condition. A join condition is a relationship among some columns in the data tables that take part in SQL join.

Q2. What are different types of SQL Joins?

Ans. In SQL Server, there are three types of joins as given below:

- Inner Join
- Outer Join
- Cross Join

Q3. Describe each type of SQL Join?

Ans. The explanation of each type of SQL Join is given below:

1. **Inner Join** - Inner join returns only those records/rows that match/exists in both the tables.

Syntax

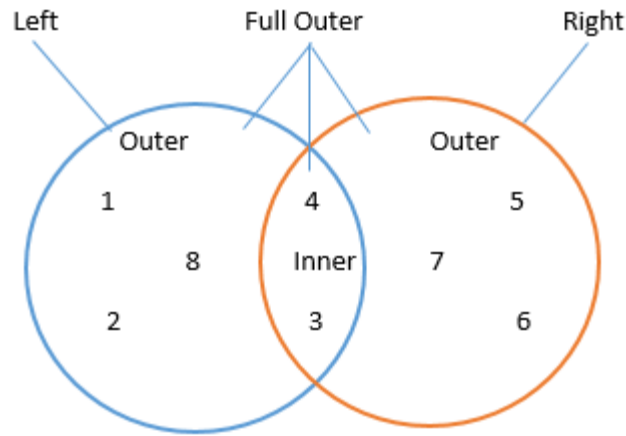
```
Select * from table_1 as t1
inner join table_2 as t2
on t1.IDcol=t2.IDcol
```

2. **Outer Join** – There are three types of Outer join as given below-

- **Left Outer Join** - Left outer join returns all records/rows from the left table and from right table returns only matched records. If there are no columns matching in the right table, it returns NULL values.

Syntax:

```
Select * from table_1 as t1
left outer join table_2 as t2
on t1.IDcol=t2.IDcol
```

Inner Join Result: (4, 3)

Left Outer Join Result: (1, 2, 8, 4, 3)

Right Outer Join Result: (5, 6, 7, 4, 3)

Full Outer Join Result: (1, 2, 8, 4, 3, 5, 6, 7)

- **Right Outer Join** - Right outer join returns all records/rows from the right table and from left table returns only matched records. If there are no columns matching in the left table, it returns NULL values.

Syntax:

```
Select * from table_1 as t1
right outer join table_2 as t2
on t1.IDcol=t2.IDcol
```

- **Full Outer Join** - Full outer join combines left outer join and right outer join. This join returns all records/rows from both the tables. If there are no columns matching in both tables, it returns NULL values.

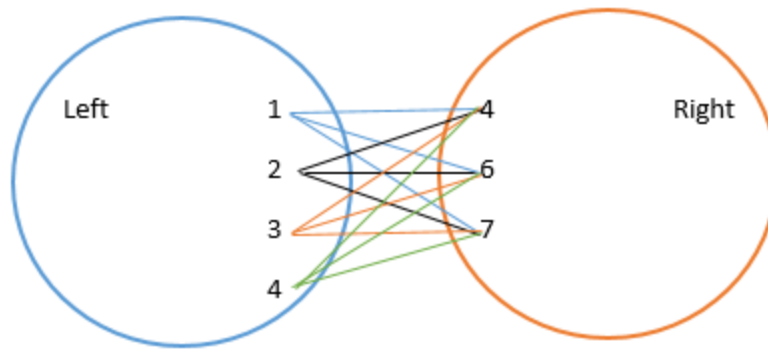
Syntax:

```
Select * from table_1 as t1
full outer join table_2 as t2
on t1.IDcol=t2.IDcol
```

3. **Cross Join** - Cross join is a Cartesian join means the Cartesian product of both the tables. This join does not need any condition to join two tables. This join returns records/rows that are a multiplication of record number from both the tables means each row on the left table will be related to each row of the right table.

Syntax:

```
Select * from table_1
cross join table_2
```



Cross Join Result: ((1, 4), (1, 6), (1, 7), (2, 4), (2, 6), (2, 7),
(3, 4), (3, 6), (3, 7), (4, 4), (4, 6), (4, 7))

Q4. What is Self-Join?

Ans. Self-join is used to join a database table to it, particularly when the table has a foreign key that references its own Primary Key. Typically, there are only three types of joins: Inner join, Outer join and Cross join. You can use any of these three JOINS to join a table to it. Hence Self-join is not a type of SQL join.

Suppose, you have the following table emp and id is primary, supid is a foreign key on the same table *emp*.

	id	name	designation	supid
1	1	mohan	Manger	NULL
2	2	raj kumar	SE	1
3	3	bipul kumar	Manager	NULL
4	4	mrinal kumar	SE	2
5	5	jitendra kumar	SE	2

Now, to get the supervisor name of each employee, you need to write the SQL query as given below-

```
select e.id,e.name,e.supid as managerid, ei.name as managename from emp e
left join emp ei on e.supid=ei.id;
```

Output:

	id	name	managerid	managename
1	1	mohan	NULL	NULL
2	2	raj kumar	1	mohan
3	3	bipul kumar	NULL	NULL
4	4	mrinal kumar	2	raj kumar
5	5	jitendra kumar	2	raj kumar

Q5. What is the difference between inner join and equi-join?

Ans. SQL join clause is used to retrieve data from two or more database tables.

Inner Join- This is the most used join in the SQL. This join returns only those records/rows that match/exists in both the database tables.

For Example:

```
SELECT * FROM tblEmp JOIN tblDept
ON tblEmp.DeptID = tblDept.DeptID
```

tblEmp.Name	tblEmp.DeptID	tblDept.Name	tblDept.DeptID
Ram	1	HR	1
Raju	2	IT	2
Soya	2	IT	2
Sam	3	ADMIN	3

In the join condition, you can also use other operators like <,>, <>.

Equi Join - Equi join is a special type of join in which we use only equality operator. Hence, when you make a query for join using equality operator then that join query comes under Equi join.

For Example:

```
SELECT * FROM tblEmp JOIN tblDept
ON tblEmp.DeptID = tblDept.DeptID;

--Using Clause is not supported by SQL Server
--Oracle and MySQL Query
SELECT * FROM tblEmp INNER JOIN tblDept USING(DeptID)
```

tblEmp.Name	tblEmp.DeptID	tblDept.Name	tblDept.DeptID
Ram	1	HR	1
Raju	2	IT	2
Soya	2	IT	2
Sam	3	ADMIN	3

Note:

- Inner join can have equality (=) and other operators (like <,>, <>) in the join condition.
- Equi join only have equality (=) operator in the join condition.
- Equi join can be an Inner join, Left Outer join, Right Outer join
- The USING clause is not supported by SQL Server and Sybase. This is supported by Oracle and MySQL.

Q6. What is Natural Join?

Ans. Natural join is a type of equi join which occurs implicitly by comparing all the same names columns in both tables. The join result has only one column for each pair of equally named columns.

For Example

```
--Run in Oracle and MySQL
SELECT * FROM tblEmp NATURAL JOIN tblDept
```

DeptID	tblEmp.Name	tblDept.Name
1	Ram	HR
2	Raju	IT
2	Soya	IT
3	Sam	ADMIN

In the above join result, we have only one column "DeptID" for each pair of equally named columns.

Note

- In a Natural join, you can't see what columns from both the tables will be used in the join. In a Natural join, you might not get the desired result what you are expecting.
- Natural join clause is not supported by SQL Server; it is supported by Oracle and MySQL.

Q7. How Group by and Having Clause are related to each other?

Ans. **Group By** - Group by clause is used for grouping the records of the database table(s). This clause creates a single row for each group and this process is called aggregation. To use group by clause we have to use at least one aggregate function in the Select statement. We can use group by clause without where clause.

Syntax for Group by Clause:

```
SELECT Col1, Col2, <Aggregate_function>
FROM Table_Name
WHERE Condition
GROUP BY Col1, Col2
```

For Example:

```
-- Group By clause without where the condition
SELECT st_Name, SUM(st_Marks) AS 'Total Marks'
FROM StudentMarks
GROUP BY st_Name;
```

Having Clause - This clause operates only on group rows of the table(s) and acts as a filter like as where clause. We use having a clause to filter data that we get from the group by clause. To use having a clause, first we need to use group by clause.

```
-- Having clause without where the condition
```

```
SELECT st_Name, SUM(st_Marks) AS 'Students Scored > 205'
FROM StudentMarks
GROUP BY st_Name
HAVING SUM(st_Marks) > 205
```

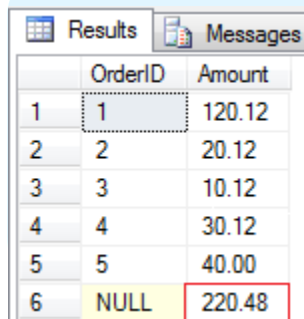
Note

1. To use *Group By* clause, we need to use at least one aggregate function
2. All columns that are not used by aggregate function(s) must be in the *Group By* list
3. We can use *Group By* clause with or without *Where* Clause.
4. To use *Having* Clause, you need to use *Group By* clause since it filters data that you get from *Group By* clause

Q8. What is Rollup operator in SQL Server?

Ans. Rollup operator is used to calculating grand of a column in SQL Server. This operator calculates grand total of the column included in the group by clause as given below-

```
SELECT OrderID, SUM(Amount) AS Amount
FROM CustomerOrders
GROUP BY OrderID WITH ROLLUP
```



	OrderID	Amount
1	1	120.12
2	2	20.12
3	3	10.12
4	4	30.12
5	5	40.00
6	NULL	220.48

Q9. What are SQL injection attacks?

Ans. A SQL Injection attack is an attack mechanism used by hackers to steal sensitive information from a database of an organization. It is the application layer means front-end attack which takes benefit of inappropriate coding of our applications that allows a hacker to insert SQL commands into your code that is using SQL statement.

SQL Injection arises since the fields available for user input allow SQL statements to pass through and query the database directly.

For Example:

Now let's look at the following query string in ASP.net. In this we are passing username from TextBox "txtUserID" and userpwd from TextBox "txtpwd" to check user credential.

```
"SELECT * FROM tbluser WHERE userName = '"+ txtUserID.text +"' and userpwd = '"+
txtPwd.text +'"
```

Now, the hacker will pass the following input to TextBoxes to inject sql attack. What will happen when the below data goes as input?

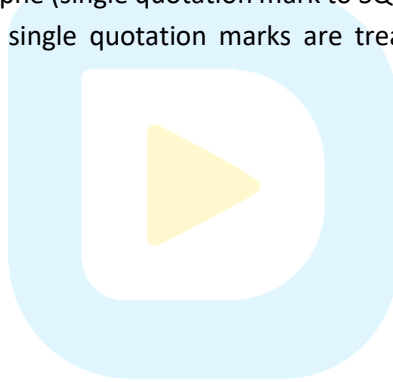
```
"SELECT * FROM tbluser WHERE userName = ';Drop table tblusers --' and userpwd = '123';"
```

Semicolon (;) in above statement will terminate the current SQL. So, "SELECT * FROM tbluser WHERE UserID = "" will become a separate statement and after Semicolon (;) it will start a new SQL statement "Drop table tblusers" that will drop our table tbluser. Hence your user details table has been dropped and your database will be unmanaged.

Q10. What are the recommended solutions to stop SQL Injection Attacks?

Ans. Some of the recommended solutions for SQL injection attacks are given below-

1. In C# or VB.NET Framework during building a SQL Statement, use the SqlParameter to define the Parameter Name, type, and value instead of making a straight command like as above
2. In ASP.NET query specify that CommandType as Text or Stored Procedure.
3. When we use the Parameters Collection, we should use parameters the type and size will also be mention.
4. If we use a stored procedure, instead of directly building by using Exec command, use the sp_executesql command.
5. Another way to stop SQL injection attacks is to filter the user input for SQL characters. Use the REPLACE function to replace any apostrophe (single quotation mark to SQL) with an additional apostrophe. Within a SQL string, two consecutive single quotation marks are treated as an instance of the apostrophe character within the string.



5

Functions, Procedures and Exceptions

Q1. What is SQL function?

Ans. A function is a database object in SQL Server. Basically, it is a set of SQL statements that accept only input parameters, perform actions and return the result. The function can return only a single value or a table. We can't use the function for Insert, Update and Delete records in the database table(s).

Q2. What are the different types of SQL Server functions?

Ans. Typically, there are two types SQL Server functions –

1. System Defined
2. User Defined

Q3. What are the System Defined functions?

Ans. These functions are defined by SQL Server for a different purpose. These can be divided into two categories as given below-

- **Scalar Function** - Scalar functions operate on a single value and return a single value. Below is the list of some useful SQL Server Scalar functions.

Scalar Function	Description
abs(-10.67)	This returns an absolute number of the given number means 10.67.
rand(10)	This will generate a random number of 10 characters.
round(17.56719,3)	This will round off the given number to 3 places of decimal means 17.567
upper('dotnet')	This will return the upper case of given string means 'DOTNET'
lower('DOTNET')	This will return the lower case of given string means 'dotnet'
ltrim(' dotnet')	This will remove the spaces from left-hand side of 'dotnet' string.
convert(int, 15.56)	This will convert the given float value to integer means 15.

- **An aggregate Function-** Aggregate function operates on a collection of values and returns a single value. Below is the list of some useful SQL Server Aggregate functions.

Aggregate Function	Description
max()	This returns maximum value from a collection of values.
min()	This returns minimum value from a collection of values.
avg()	This returns average of all values in a collection.
count()	This returns no of counts from a collection of values.

Q4. What is a User Defined function?

Ans. User-defined functions are created by the user in a system database or in a user-defined database. A user can define three types of user-defined functions as given below -

- **Scalar Function** – A User-defined scalar function returns a single value. It can return any data type value from a function.

```
--Create a function to get emp full name
Create function fnGetEmpFullName
(
    @FirstName varchar(50),
    @LastName varchar(50)
)
returns varchar(101)
As
Begin return (Select @FirstName + ' ' + @LastName);
end

--Calling the above created function
Select dbo.fnGetEmpFullName(FirstName,LastName) as Name, Salary from Employee
```

- **Inline Table-Valued Function** – A User-defined inline table-valued function returns a table variable. The returned table variable value should be derived from a single SELECT statement.

```
--Create a function to get employees
Create function fnGetEmployee()
returns Table
As
return (Select * from Employee)

GO

--Calling the above-created function
Select * from fnGetEmployee()
```


- **Multi-Statement Table-Valued Function** - A User-defined multi-statement table-valued function returns a table variable. In this, a table variable must be declared explicitly and the return value can be derived from multiple SQL statements.

```
--Create function for EmpID, FirstName and Salary of Employee
Create function fnGetMulEmployee()
returns @Emp Table
(
EmpID int,
FirstName varchar(50),
Salary int
)
As
begin
Insert @Emp Select e.EmpID,e.FirstName,e.Salary from Employee e;
--Now update salary of the first employee
update @Emp set Salary=25000 where EmpID=1;
--It will update only in @Emp variable, not in Original Employee table
return
end
```

Q5. What are the limitations of SQL function?

Ans. The limitations of SQL Functions are given below:

- A Function can return an only a single value.
- A Function can accept only input parameters.
- The function cannot be used to Insert, Update, Delete data in a database table(s).
- A Function can be nested up to 32 levels.
- User Defined Function can have up to 1023 input parameters while a Stored Procedure can have up to 2100 input parameters.
- A User Defined Function can't return XML Data Type.
- A User Defined Function doesn't support Exception handling.
- A User Defined Function can call only the Extended Stored Procedure.
- A User Defined Function doesn't support set options like set ROWCOUNT etc.

Q6. What are the stored procedures?

Ans. A stored procedure is a precompiled set of one or more SQL statements that are stored on SQL Server. The benefit of Stored Procedures is that they are executed on the server side and perform a set of actions, before returning the results to the client side. This allows a set of actions to be executed with minimum time and also reduce the network traffic. Hence, stored procedures improve performance to execute SQL statements.

Q7. What are different types of SQL Server stored procedure?

Ans. Typically, there are four types of SQL Server stored procedures as given below-

1. **System Defined Stored Procedure** – These types of stored procedures are already defined within SQL Server. In fact, they are the part of sys schema of each database in SQL Server. This procedure starts with

the *sp_* prefix. So, as a best practice, you should not use an *sp_* prefix when naming a user-defined procedure. Here is a list of some useful system defined procedure.

System Procedure	Description
sp_rename	It is used to rename a database object like stored procedure, views, table etc.
sp_changeowner	It is used to change the owner of a database object.
sp_help	It provides details on any database object.
sp_helpdb	It provides the details of the databases defined in the SQL Server.
sp_helptext	It provides the text of a stored procedure reside in SQL Server
sp_depends	It provides the details of all database objects that depend on the specific database object.

2. **Extended Stored Procedure** – Extended procedures provide an interface to external programs for various maintenance activities. These extended procedures start with the *xp_* prefix and stored in Master Database. Basically, these are used to call programs that reside on the server automatically from a stored procedure or a trigger run by the server.

For Example- Below statements are used to log an event in the NT event log of the server without raising an error on the client application.

```
declare @logmsg varchar(100)
set @logmsg = suser_sname() + ': Tried to access the dotnet system.'
exec xp_logevent 50005, @logmsg
print @logmsg
```

3. **Use Defined Stored Procedure** - These procedures are created by the user for own actions. These can be created in all system databases except the Resource database or in a user-defined database.
4. **CLR Stored Procedure** - CLR stored procedure is introduced with SQL Server 2008 which are based on the CLR in .NET framework. They enable the procedure to be written in one of the .NET languages like C#, Visual Basic and F#.

Q8. What are the limitations of SQL Server stored procedure?

Ans. There are following limitations of SQL Server stored procedure:

- We can nest stored procedures and managed code references in SQL Server up to 32 levels only. This is also applicable for function, trigger and view.
- The current nesting level of the execution of a stored procedure is stored in the @@NESTLEVEL function.
- In SQL Server stored procedure nesting limit is up to 32 levels, but there is no limit on the number of stored procedures that can be invoked within a stored procedure

Q9. What is the difference between Stored Procedure and Function in SQL Server?

Ans. **Stored Procedures** are pre-compiled objects which are compiled for the first time and its compiled format is saved which executes (compiled code) whenever it is called. But **Function** is compiled and executed every time when it is called.

There are following differences between stored procedure and functions as given below –

Stored Procedure	Function
Can return zero or n values.	Must return a value but only one.
Can have input and output parameters.	Can have only input parameters.
Cannot be called from function.	Can be called from another function or a stored procedure.
Allows SELECT as well as DML(INSERT/ UPDATE/ DELETE) statements	Allows only SELECT statement
Cannot be used anywhere in the WHERE/ HAVING/SELECT section of SQL statements	Functions can be used anywhere in the WHERE/ HAVING/SELECT section of SQL statements
Try..Catch block can be used for exception handling	Try..Catch block cannot be used for exception handling
A transaction can be used	A transaction cannot be used

Q10. What is stored procedure plan recompilation and performance tuning?

Ans. The main advantage of the stored procedure is, to execute T-SQL statements in less time than the similar set of T-SQL statements is executed individually. The reason to take less time is that the query execution plan for the stored procedures is already stored in the "sys.procedures" system defined view.

The recompilation process of the stored procedure is like as compilation process and also reduces SQL Server performance. Stored procedure with recompilation option was introduced in SQL Server 2005. We should recompile the stored procedure in the following cases:

1. Changing to the schema (means adding/dropping columns, constraints, rules, index, trigger etc.) of the tables or referenced table(s) or view(s).
2. Updating the statistics used by the execution plan of a stored procedure

Q11. What are various ways to recompile Stored Procedure?

Ans. There are two ways for stored procedure recompilation as given below-

1. **Recompile option at the time of Creation** - In this, we create a stored procedure with RECOMPILE option. When we call this procedure than every time this procedure will be recompiled before executing.

```
CREATE PROCEDURE usp_InsertEmployee
WITH RECOMPILE
@flag bit output, -- return 0 for fail, 1 for success
@EmpID int,
@Name varchar(50),
```

```

@Salary int,
@Address varchar(100)
AS
BEGIN
BEGIN TRANSACTION
BEGIN TRY
    Insert into Employee(EmpID,Name,Salary,Address)
    Values(@EmpID,@Name,@Salary,@Address);

    set @flag=1;
    commit TRANSACTION;

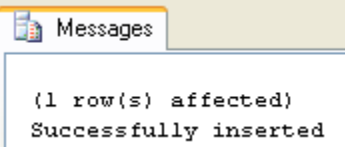
END TRY
BEGIN CATCH

    rollback TRANSACTION;
    set @flag=0;

END CATCH
END

GO
--Calling the procedure
Declare @flag bit
--Now Execute this procedure. Every time this procedure will be recompiled
EXEC usp_InsertEmployee @flag output,1,'Deepak',14000,'Noida'
if @flag=1
    print 'Successfully inserted'
else
    print 'There is some error'

```



Messages

(1 row(s) affected)
Successfully inserted

- 2. Recompile option at the time of Execution-** In this, we call a stored procedure with the RECOMPILE option. Hence this stored procedure will be compiled only when we use the RECOMPILE option at the time of calling. This is the best option for stored procedure recompilation.

```

CREATE PROCEDURE usp_InsertEmployee
@flag bit output,-- return 0 for fail,1 for success
@EmpID int,
@Name varchar(50),
@Salary int,
@Address varchar(100)
AS
BEGIN
BEGIN TRANSACTION
BEGIN TRY

```

```

Insert into Employee(EmpID,Name,Salary,Address)
Values(@EmpID,@Name,@Salary,@Address);
set @flag=1;
commit TRANSACTION;
END TRY
BEGIN CATCH
rollback TRANSACTION;
set @flag=0;
END CATCH
END

GO

--Calling the procedure
Declare @flag bit
--Now Execute this procedure with RECOMPILE option, if you want to recompile its
execution plan
EXEC usp_InsertEmployee @flag output,2,'Jitendra',15000,'Noida' WITH RECOMPILE
if @flag=1
print 'Successfully inserted'
else
print 'There is some error'

```

Note

- Creating the stored procedure by using "WITH RECOMPILE" option force the SQL Server to recompile the stored procedure every time when it is called.
- Call the stored procedure by using "WITH RECOMPILE" option in the EXEC command.
- Altering the procedure will cause the SQL Server to create a new execution plan
- If SQL Server is restarted or stopped then all the execution plans will be flush from server cache and recreated when the stored procedure is executed after restarting the server.
- The "Sp_recompile" system defined stored procedure can be called to refresh the query execution plan for a particular stored procedure.

Q12. What is Exception?

Ans. Exceptions are defined as glitches, unexpected or unseen errors which occur during the execution of a program. These can be caused due to improper user inputs, improper design logic or system errors. Exception handling in SQL Server is like as exception handling in other programming languages.

Q13. What are different types of exception in SQL Server?

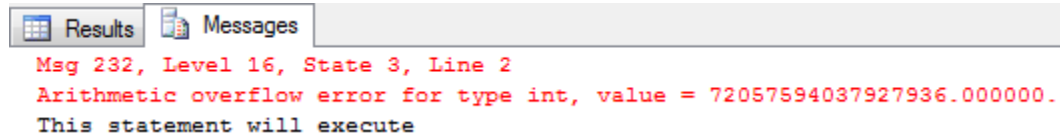
Ans. There are following types of exception in SQL Server as given below:

1. Statement-Level Exception
2. Batch-Level Exception
3. Parsing and Scope-Resolution Exception

Q14. What is Statement-Level exception in SQL Server?

Ans. This type of exception aborts only the current running statement within a batch of T-SQL statements. The rest of the T-SQL statements will execute successfully if they have no exceptions.

```
--Batch
SELECT POWER(4, 28)
PRINT 'This statement will execute'
```

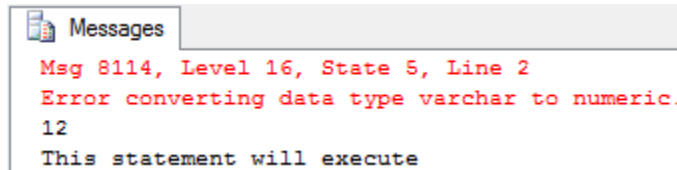


The screenshot shows the 'Messages' window in SQL Server Enterprise Manager. It displays an error message: 'Msg 232, Level 16, State 3, Line 2 Arithmetic overflow error for type int, value = 72057594037927936.000000.' Below the error message, the text 'This statement will execute' is visible, indicating that statements following the error in the same batch continue to execute.

Q15. What is a batch-Level exception in SQL Server?

Ans. This type of exception aborts only the batch in which exception occurs. The rest of the batches will execute successfully if they have no exceptions. The statement in which exception occurs will be aborted and the remaining T-SQL statements within the batch will also stop.

```
--First Batch
DECLARE @var DECIMAL;
set @var= CONVERT(DECIMAL, 'xyz')
PRINT @var
PRINT 'This statement will not execute'
GO
--Second Batch
DECLARE @var DECIMAL;
set @var= CONVERT(DECIMAL, '12.35')
PRINT @var
PRINT 'This statement will execute'
```



The screenshot shows the 'Messages' window in SQL Server Enterprise Manager. It displays an error message: 'Msg 8114, Level 16, State 5, Line 2 Error converting data type varchar to numeric.' Below the error message, the text '12' and 'This statement will execute' are visible, indicating that the batch was aborted at the point of the error.

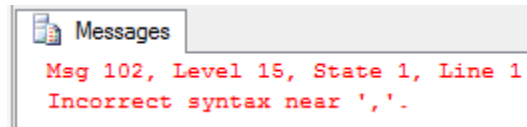
Q16. What is Parsing and Scope-Resolution exception in SQL Server?

Ans. These types of exception occur during the parsing and during the scope-resolution phase of compilation. This exception appears to behave just like batch-level exceptions. However, this has a little different behaviour.

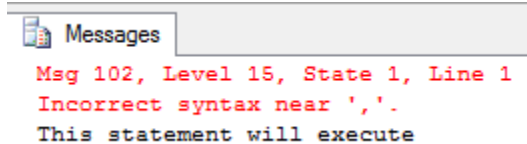
If the exception occurs in the same scope of the batch, it behaves just like a batch-level exception. If the exception occurs in a lower level of scope of the batch, it behaves just like statement-level exception.

Parsing Exception:

```
--Parsing Exception
SELECT EmpID, Name FROM Employee
PRINT 'This statement will execute'
```



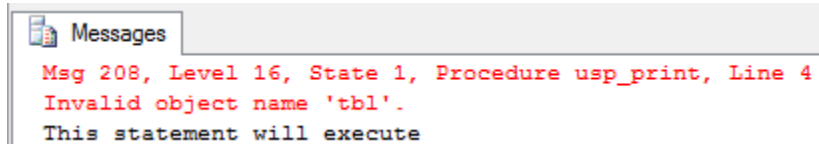
```
--For Successfully execution we need to the executed select statement as dynamic SQL
using the EXEC function
EXEC('SELECT EmpID, Name FROM Employee')
PRINT 'This statement will execute'
```



Scope Resolution Exception

```
--First Create a procedure
CREATE PROCEDURE usp_print
AS
BEGIN
    Select * from table
END
GO

--Now execute the above-created procedure in batch
EXEC usp_print
PRINT 'This statement will execute'
--Since the stored procedure creates a new scope. Hence rest statement will be executed
```



Q17. How to try...catch works in SQL Server?

Ans. Like C#, SQL Server also has TRY...CATCH blocks for handling exceptions. You should wrap your T-SQL statements in a TRY block and to capture and handle exception you should write code in the CATCH block.

```
BEGIN TRY
--T-SQL statements
--or T-SQL statement blocks
END TRY
BEGIN CATCH
--T-SQL statements
--or T-SQL statement blocks
END CATCH
```

Q18. What are the limitations of try...catch in SQL Server?

Ans. There are following limitations of try..catch in SQL Server:

1. A TRY...CATCH block combination catches all the errors that have a severity between 11 and 19.
2. The CATCH block is executed only if there is an error occurs in T-SQL statements within TRY block otherwise the CATCH block is ignored.
3. Each TRY block is associated with only one CATCH block and vice versa
4. TRY and CATCH blocks can't be separated from the GO statement. We need to put both TRY and CATCH blocks within the same batch.
5. TRY...CATCH blocks can be used with transactions. We check the number of open transactions by using the @@TRANCOUNT function in SQL Server.
6. XACT_STATE function within the TRY...CATCH block can be used to check whether an open transaction is committed or not. It will return -1 if the transaction is not committed else returns 1.

Q19. What are different error functions used in SQL Server for error handling?

Ans. There are some important functions for error handling in SQL Server as given below-

- **ERROR_NUMBER()** - This returns the error number and its value is same as for @@ERROR function.
- **ERROR_LINE()** - This returns the line number of T-SQL statement that causes an error.
- **ERROR_SEVERITY()** - This returns the severity level of the error.
- **ERROR_STATE()** - This returns the state number of the error.
- **ERROR_PROCEDURE()** - This returns the name of the stored procedure or trigger in which error occurred.
- **ERROR_MESSAGE()** - This returns the full text of error message.

Q20. What is case expression in SQL Server?

Ans. In SQL Server, case expression is used to just like a switch statement in programming languages. You can use CASE expressions anywhere in the SQL Query. CASE expressions can be used within a SELECT statement, WHERE clauses, Order by clause, HAVING clauses, Insert, UPDATE and DELETE statements.

Q21. What are the different ways to write case expression in SQL Server?

Ans. You can write Case expression in two ways as given below-

- **Simple CASE expression** - In this, an expression is compared w.r.t a set of simple expressions to find the result. If the expression inside the WHEN clause is matched, the T-SQL in the THEN clause will be executed and returned.

Syntax:

```
CASE expression
WHEN exp1 THEN result1
WHEN exp2 THEN result2
ELSE resultN
END
```


- **Searched CASE expressions** - This expression evaluates a set of Boolean expressions to find the result. This expression allows comparison operators, and logical operators AND/OR within each Boolean expression.

Syntax:

```
CASE
WHEN Boolean_expression1 THEN Result1
WHEN Boolean_expression2 THEN Result2
ELSE ResultN
END
```

For Example:

```
--Simple CASE expression:
SELECT FirstName, State=(CASE StateCode
  WHEN 'MP' THEN 'Madhya Pradesh'
  WHEN 'UP' THEN 'Uttar Pradesh'
  WHEN 'DL' THEN 'Delhi'
  ELSE NULL
END), PayRate
FROM dbo.Customer

-- Searched CASE expression:
SELECT FirstName, State=(CASE
  WHEN StateCode = 'MP' THEN 'Madhya Pradesh'
  WHEN StateCode = 'UP' THEN 'Uttar Pradesh'
  WHEN StateCode = 'DL' THEN 'Delhi'
  ELSE NULL
END), PayRate
FROM dbo.Customer
```

Q22. When to use case expression in SQL Server?

Ans. Sometimes, you required to fetch or modify the records based on some conditions. In this case, you may use cursor or loop for modifying your records. In this situation, *Case* expression is the best alternative for Cursor/looping and also provides better performance.

6

Triggers and Tables

Q1. What are triggers?

Ans. Triggers are a database object. Basically, these are a special type of stored procedure that is automatically fired/executed when a DDL or DML command statement related to the trigger is executed.

Triggers are used to assess/evaluate data before or after data modification using DDL and DML statements. These are also used to preserve data integrity, to control server operations, to audit a server and to implement business logic or business rule.

Q2. Write syntax to create a trigger in SQL Server?

Ans. SQL Server trigger syntax is given below:

Syntax:

```
CREATE TRIGGER trigger_name ON {table|view}
[WITH ENCRYPTION|EXECUTE AS]
{FOR|AFTER|INSTEAD OF} {[CREATE|ALTER|DROP|INSERT|UPDATE|DELETE ]}
[NOT FOR REPLICATION]
AS
sql_statement [1...n ]
```

- **trigger_name** - This is the name of the trigger. It should conform to the rules for identifiers in SQL Server.
- **table|view** - This is the table/view on which the trigger is to be created.
- **ENCRYPTION** - This option is optional. If this option is specified, the original text of the CREATE TRIGGER statement will be encrypted.
- **EXECUTE AS** - This option is optional. This option specifies the security context under which the trigger is executed.
- **FOR/AFTER** - FOR/AFTER specifies that the trigger is After Trigger. AFTER is the default, if FOR is the only keyword specified. AFTER triggers cannot be defined on views.
- **INSTEAD OF** - INSTEAD OF specifies that the trigger is Instead of Trigger.
- **CREATE|ALTER|DROP|INSERT|UPDATE|DELETE** - These keywords specify on which action the trigger should be fired. One of these keywords or any combination of these keywords in any order can be used.
- **NOT FOR REPLICATION** - Indicates that the trigger should not be executed when a replication process modifies the table involved in the trigger.
- **AS** - After this, we specify the actions and condition that the trigger performs.

- **sql_statement** - These are the trigger conditions and actions. The trigger actions specified in the T-SQL statements.

Q3. What are different types of triggers in SQL Server?

Ans. There are following types of SQL Server triggers:

- DDL Triggers
- DML Triggers
- CLR Triggers
- LogOn Triggers

Q4. What are DDL triggers in SQL Server?

Ans. When you create a trigger on DDL statements (like CREATE, ALTER, and DROP) or on certain system-defined stored procedures that perform DDL-like operations are known as DDL triggers.

Q5. What are DML triggers in SQL Server?

Ans. These triggers are created on DML statements (like INSERT, UPDATE, and DELETE) or on certain stored procedures that perform DML operations.

Q6. What are different types of DML triggers in SQL Server?

Ans. There are following types of DML Triggers:

1. **After Trigger** – This type of triggers is defined using FOR or AFTER clause. These triggers fire only after successful execution of SQL Server statements on which they are associated.

For Example: If an *after the trigger* is associated with the insert operation on a database table, it will fire after the successful record insertion in that database table. If the record insertion fails because of SQL constraints like primary key, not null etc., it will not fire.

2. **Instead of Trigger** – This type of trigger is defined using INSTEAD OF clause. These triggers fire before executing the SQL Server statements on which they are associated.

For Example: If an *instead of trigger* is associated with the insert operation on a database table, it will fire before executing the insert statement on that database table. It will not wait for statement execution.

Q7. What are CLR triggers in SQL Server?

Ans. CLR triggers are introduced with SQL Server 2008. They allow you to write triggers code using .NET languages like C#, Visual Basic and F#. These triggers are helpful in heavy computations or when you need the references of database objects outside the SQL Server.

Q8. What are Logon triggers in SQL Server?

Ans. Logon triggers are fired a user tries to login into SQL Server. These do not fire if SQL Server authentication fails. These triggers are helpful in auditing and controlling server sessions, such as tracking login activity or limit the number of sessions for a specific login.

Q9. What are the limitations of triggers in SQL Server?

Ans. There are following limitations of triggers in SQL Server:

1. DML trigger can be composed of any T-SQL statements, except CREATE DATABASE, ALTER DATABASE, DROP DATABASE, LOAD DATABASE, LOAD LOG, RECONFIGURE, RESTORE DATABASE, and RESTORE LOG statements.
2. You cannot create triggers against system tables or dynamic management views. Moreover, the TRUNCATE TABLE statement does not fire a trigger because this operation does not log individual row deletions.
3. If you use the DATABASE option, the scope of your DDL trigger will be the current database. If you use the ALL SERVER option, the scope of your DDL triggers to the current server.
4. AFTER triggers cannot be defined on views.
5. AFTER is the default, if FOR is the only keyword specified.

Q10. What are different ways for setting triggers firing order in SQL Server?

Ans. Sometimes we have multiple triggers on the same event(s). In that case, we can't predict the firing order of triggers. Sometimes the firing order of trigger(s) is important for us to implement business logic. To solve this issue, In SQL Server we have the option to set the firing order of triggers on the same event(s).

```
sp_settriggerorder @triggername='trg_name', @order='FIRST|LAST|NONE',
@stmtype='INSERT|UPDATE|DELETE|CREATE_INDEX,ALTER_INDEX',
@namespace='DATABASE|SERVER|NULL'

--Now set triggers firing orders
EXEC sp_settriggerorder 'dbo.trg_i_TriggerOrder1', 'First', 'INSERT'
EXEC sp_settriggerorder 'dbo.trg_i_TriggerOrder2', 'Last', 'INSERT'
--The order of firing the third trigger 'dbo.trg_i_TriggerOrder3' will be between above
two
```

Q11. What are logical tables in SQL Server?

Ans. There are inserted and deleted logical tables in SQL Server. These tables are automatically created and managed by SQL Server internally to hold recently inserted, deleted and updated values during DML operations (Insert, Update, and Delete) on a database table.

Q12. What are inserted and deleted logical tables in SQL Server?

Ans. The inserted and deleted logical tables are explained as:

Inserted Logical Table- Inserted table holds the recently inserted or updated values means new data values. Hence newly added and updated records are inserted into the Inserted table.

Suppose we have Employee table as shown in fig. Now we need to create two triggers to see data within logical tables Inserted and Deleted.

	EmpID	Name	Salary
1	1	Mohan	10000
2	2	Jitendra	13000

```
CREATE TRIGGER trg_Emp_Ins
ON Employee
FOR INSERT
AS
begin
SELECT * FROM INSERTED -- show data in the Inserted logical table
SELECT * FROM DELETED -- show data in the Deleted logical table
End
```

Now insert a new record in the Employee table to see data within the Inserted logical table.

```
--Called Trigger trg_Emp_Ins automatically, when trying to insert data
INSERT INTO Employee(EmpID, Name, Salary) VALUES(3, 'Avin', 23000)

SELECT * FROM Employee
```

	EmpID	Name	Salary
1	3	Avin	23000

Inserted table

	EmpID	Name	Salary
--	-------	------	--------

Deleted table

	EmpID	Name	Salary
1	1	Mohan	10000
2	2	Jitendra	13000
3	3	Avin	23000

Employee table

Deleted Logical Table - The Deleted table holds the recently deleted or updated values means old data values. Hence old updated and deleted records are inserted into the Deleted table.

```
CREATE TRIGGER trg_Emp_Upd
ON Employee
FOR UPDATE
AS
begin
SELECT * FROM INSERTED -- show data in the INSERTED logical table
SELECT * FROM DELETED -- show data in the DELETED logical table
End
```

Now update the record in Employee table to see data within Inserted and Deleted logical tables.

```
--Called Trigger trg_Emp_Upd automatically, when trying to update data
Update Employee set Salary=43000 where EmpID=3

SELECT * FROM Employee
```

Results		Messages	
EmpID	Name	Salary	
1	3	Avin	43000
Inserted table			
EmpID	Name	Salary	
1	3	Avin	23000
Deleted table			
EmpID	Name	Salary	
1	1	Mohan	10000
2	2	Jitendra	13000
3	3	Avin	43000
Employee table			

Q13. When Inserted and Deleted Logical table are used in SQL Server?

Ans. Basically, logical tables are used by triggers for the following purpose:

- To test data manipulation errors and take suitable actions based on the errors.
- To find the difference between the state of a table before and after the data modification and take actions based on that difference.

Except for triggers, when you use the OUTPUT clause in your query, logical tables are automatically created and managed by SQL Server. OUTPUT clause also has access to Inserted and Deleted logical tables just like triggers.

Q14. What is CTE (Common Table Expressions) in SQL Server?

Ans. CTE was introduced with SQL Server 2005 for storing a temporary result set or storing a result of a complex sub-query. It is defined by using WITH statement preceded by a semicolon. Unlike SQL Server temporary table, its life is limited to the current query scope.

A subquery without CTE is given below -

```
SELECT * FROM (
    SELECT Addr.Address, Emp.Name, Emp.Age From Address Addr
    Inner join Employee Emp on Emp.EID = Addr.EID) Temp
WHERE Temp.Age > 50
ORDER BY Temp.NAME
```

By using CTE above query can be re-written as follows-

```

;With CTE1(Address, Name, Age)--Column names for CTE which is optional
AS
(
SELECT Addr.Address, Emp.Name, Emp.Age from Address Addr
INNER JOIN EMP Emp ON Emp.EID = Addr.EID
)
SELECT * FROM Temp --Using CTE
WHERE CTE1.Age > 50
ORDER BY CTE1.NAME

```

Q15. What are temporary tables?

Ans. In SQL Server, temporary tables are created at runtime inside Tempdb database. Just like normal database tables, you can do all the table operations.

Q16. What are different types of temporary tables?

Ans. Temporary tables are of two types as given below-

1. **Local Temp Table** - Local temp tables are created using a single hash (“#”) symbol. These are accessible within a session or connection for a single user i.e. it is scoped to the current query window. These are automatically removed when that user session has been closed.

```

CREATE TABLE #LocalUser
(
    UserID int,
    Name varchar(50),
    Address varchar(150)
)
GO
insert into #LocalUser values ( 1, 'Shailendra', 'Noida');
GO
Select * from #LocalUser

```

Results		Messages	
	UserID	Name	Address
1	1	Shailendra	Noida

2. **Global Temp Table** – Global temp tables are created using double hash (“##”) symbols. These are accessible to all the SQL Server sessions or connections for all users. These can be created by any user and removed automatically when all the SQL Server connections have been closed.

```

CREATE TABLE ##GlobalUser
(
    UserID int,
    Name varchar(50),
    Address varchar(150)
)
GO
insert into ##GlobalUser values ( 1, 'Shailendra', 'Noida');

```

```
GO
Select * from ##GlobalUser
```

Results		Messages	
UserID	Name	Address	
1	1	Shailendra	Noida

Q17. What is the table variable?

Ans. A table variable is just like a variable in a program and exists for a particular batch of query execution. This is created in the *Tempdb* database and enables you to create a primary key or identity column at the time of Table variable declaration.

```
--First batch
GO
DECLARE @tblProducts TABLE
(
  SNo INT IDENTITY(1,1),
  Id INT,
  Name VARCHAR(100),
  Qty INT,
  Total INT
)
--Insert data to Table variable @Product
INSERT INTO @Products(Id) SELECT DISTINCT Id FROM Products ORDER BY Id ASC

--Select data
Select * from @tblProducts

GO
--Next batch
Select * from @tblProducts --gives an error in next batch
```

Q18. What is a local and global variable in SQL Server?

Ans. Local - Local variables are declared by the user and can be used in the functions, procedures and batches of SQL statements to hold information. All local variables are defined using @ symbol. You can create a local variable by using a DECLARE statement.

```
DECLARE @Name VARCHAR(20)
SET @Name='xyz'
```

Global - Global variables are system-defined variables and SQL Server automatically maintains the values of these variables related to the server or a current user session. All global variables start with two @@ symbols.

For example, @@ERROR, @@TRANCOUNT, @@CONNECTIONS, @@ROWCOUNT, @@version

```
SELECT @@version --returns SQL Server version
```


Cursors and Indexes

Q1. What is cursor?

Ans. A cursor is a database object used to retrieve records from a result set one by one. Cursors are useful when you need to modify multiple records in a database table in the record by record.

Q2. Explain the life cycle of the cursor in SQL Server?

Ans. The Cursor lifecycle has the following steps –

1. **Declare Cursor** - A cursor is declared by defining the SQL statement that returns a result set.
2. **Open** - A Cursor is opened and populated by executing the SQL statement defined by the cursor.
3. **Fetch** - When a cursor is opened, records can be fetched from the cursor one by one to perform data manipulation.
4. **Close** - After data manipulation, you should close the cursor explicitly.
5. **Deallocate** - Finally, you need to delete the cursor definition and released all the system resources associated with the cursor.

Syntax

```
DECLARE cursor_name CURSOR
[LOCAL | GLOBAL] --define cursor scope
[FORWARD_ONLY | SCROLL] --define cursor movements (forward/backward)
[STATIC | KEYSET | DYNAMIC | FAST_FORWARD] --basic type of cursor
[READ_ONLY | SCROLL_LOCKS | OPTIMISTIC] --define locks
FOR select_statement --define SQL Select statement
FOR UPDATE [col1,col2,...coln] --define columns that need to be updated
```

- **Syntax to Open A Cursor**- A Cursor can be opened locally or globally. By default, it is opened locally. The basic syntax to open cursor is given below:

```
OPEN [GLOBAL] cursor_name --by default it is local
```

- **Syntax to Fetch A Cursor**- Fetch statement provides the many options to retrieve the rows from the cursor. NEXT is the default option. The basic syntax to fetch cursor is given below:

```

FETCH [NEXT|PRIOR|FIRST|LAST|ABSOLUTE n|RELATIVE n]
FROM [GLOBAL] cursor_name
INTO @Variable_name[1,2,..n]

```

- **Syntax to Close A Cursor-** Close statement closed the cursor explicitly. The basic syntax to close cursor is given below:

```

CLOSE cursor_name --after closing it can be reopened

```

- **Syntax to Deallocate A Cursor-** Deallocate statement delete the cursor definition and free all the system resources associated with the cursor. The basic syntax to close cursor is given below:

```

DEALLOCATE cursor_name --after deallocation, it can't be reopened

```

Q3. What are different types of Cursor in SQL Server?

Ans. There are following types of SQL Server Cursor as given below:

- Static Cursors
- Dynamic Cursors
- Forward Only Cursors
- Keyset-Driven Cursors

Q4. What is Static Cursor in SQL Server?

Ans. A static cursor result set is populated at the time of cursor creation and it is cached for the lifetime of the cursor. A static cursor is slower and uses more memory as compared to another cursor. Hence, you should use it only if scrolling is required.

Q5. How to write a Static Cursor in SQL Server?

Ans. Suppose you have following Employee table as given below and need to write cursor for this table, then you can write this as given below-

	EmpID	Name	Salary	Address
1	1	Mohan	12000	Noida
2	2	Pavan	25000	Delhi
3	3	Amit	22000	Dehradun
4	4	Sonu	22000	Noida
5	5	Deepak	28000	Gurgaon

```

--Static Cursor for Update
SET NOCOUNT ON
DECLARE @Id int
DECLARE @name varchar(50)
DECLARE @salary int

```

```

DECLARE cur_emp CURSOR
STATIC FOR
SELECT EmpID,EmpName,Salary from Employee
OPEN cur_emp
IF @@CURSOR_ROWS > 0
BEGIN
    FETCH NEXT FROM cur_emp INTO @Id,@name,@salary
    WHILE @@Fetch_status = 0
    BEGIN
        PRINT 'ID : ' + convert(varchar(20),@Id)+' , Name : '+@name+ ' , Salary : '
        '+convert(varchar(20),@salary)
        FETCH NEXT FROM cur_emp INTO @Id,@name,@salary
    END
END
CLOSE cur_emp
DEALLOCATE cur_emp
SET NOCOUNT OFF

```

Messages		
ID : 1,	Name : Mohan,	Salary : 12000
ID : 2,	Name : Pavan,	Salary : 25000
ID : 3,	Name : Amit,	Salary : 22000
ID : 4,	Name : Sonu,	Salary : 22000
ID : 5,	Name : Deepak,	Salary : 28000

Q6. What are the limitations of a Static Cursor in SQL Server?

Ans. You can't perform the update and delete operations using a static cursor. Also, it is not sensitive to any changes to the original data source.

Q7. What is Dynamic Cursor in SQL Server?

Ans. A dynamic cursor allows you to see the data updating, deletion and insertion in the data source while the cursor is open. A dynamic cursor is sensitive to any changes to the data source and supports update, delete operations.

Q8. How to write a Dynamic Cursor in SQL Server?

Ans. Suppose you have following Employee table as given below and need to write cursor for this table, then you can write this as given below-

Results		Messages		
	EmpID	Name	Salary	Address
1	1	Mohan	12000	Noida
2	2	Pavan	25000	Delhi
3	3	Amit	22000	Dehradun
4	4	Sonu	22000	Noida
5	5	Deepak	28000	Gurgaon

```

--Dynamic Cursor for Update
SET NOCOUNT ON

```

```

DECLARE @Id int
DECLARE @name varchar(50)
DECLARE Dynamic_cur_empupdate CURSOR
DYNAMIC
FOR
SELECT EmpID, EmpName from Employee ORDER BY EmpName
OPEN Dynamic_cur_empupdate
IF @@CURSOR_ROWS > 0
BEGIN
    FETCH NEXT FROM Dynamic_cur_empupdate INTO @Id,@name
    WHILE @@Fetch_status = 0
    BEGIN
        IF @name='Mohan'
        Update Employee SET Salary=15000 WHERE CURRENT OF Dynamic_cur_empupdate
        FETCH NEXT FROM Dynamic_cur_empupdate INTO @Id,@name
    END
END
CLOSE Dynamic_cur_empupdate
DEALLOCATE Dynamic_cur_empupdate
SET NOCOUNT OFF

Go
Select * from Employee

```

	EmpID	EmpName	Salary	Address
1	1	Mohan	15000	Noida
2	2	Pavan	25000	Delhi
3	3	Amit	22000	Dehradun
4	4	Sonu	22000	Noida
5	5	Deepak	28000	Gurgaon

Q9. What is Forward Only Cursor in SQL Server?

Ans. A forward-only cursor is the fastest cursor among all cursors but it doesn't support backward scrolling. You can update, delete data using a forward-only cursor. It is sensitive to any changes to the original data source.

There are three more types of Forward Only Cursors. Forward_Only KEYSET, FORWARD_ONLY STATIC and FAST_FORWARD.

A FORWARD_ONLY STATIC Cursor is populated at the time of creation and cached the data to the cursor lifetime. It is not sensitive to any changes to the data source.

Q10. How to write a Forward only Cursor in SQL Server?

Ans. Suppose you have following Employee table as given below and need to write cursor for this table, then you can write this as given below-

	EmpID	Name	Salary	Address
1	1	Mohan	12000	Noida
2	2	Pavan	25000	Delhi
3	3	Amit	22000	Dehradun
4	4	Sonu	22000	Noida
5	5	Deepak	28000	Gurgaon

A FAST_FORWARD Cursor is the fastest cursor and it is not sensitive to any changes to the data source.

```
--Forward Only Cursor for Update
SET NOCOUNT ON
DECLARE @Id int
DECLARE @name varchar(50)
DECLARE Forward_cur_empupdate CURSOR
FORWARD_ONLY
FOR
SELECT EmpID, EmpName from Employee ORDER BY EmpName
OPEN Forward_cur_empupdate
IF @@CURSOR_ROWS > 0
BEGIN
    FETCH NEXT FROM Forward_cur_empupdate INTO @Id,@name
    WHILE @@Fetch_status = 0
    BEGIN
        IF @name='Amit'
        Update Employee SET Salary=24000 WHERE CURRENT OF Forward_cur_empupdate
        FETCH NEXT FROM Forward_cur_empupdate INTO @Id,@name
    END
END
CLOSE Forward_cur_empupdate
DEALLOCATE Forward_cur_empupdate
SET NOCOUNT OFF

Go
Select * from Employee
```

	EmpID	EmpName	Salary	Address
1	1	Mohan	15000	Noida
2	2	Pavan	25000	Delhi
3	3	Amit	24000	Dehradun
4	4	Sonu	22000	Noida

Q11. What is Keyset-Driven Cursor in SQL Server?

Ans. A keyset-driven cursor is controlled by a set of unique identifiers as the keys in the keyset. The keyset depends on all the rows that qualified the SELECT statement at the time of the cursor was opened. A keyset-driven cursor is sensitive to any changes to the data source and supports update, delete operations. By default, keyset-driven cursors are scrollable.

Q12. How to write a Keyset-driven Cursor in SQL Server?

Ans. Suppose you have following Employee table as given below and need to write cursor for this table, then you can write this as given below-

	EmpID	Name	Salary	Address
1	1	Mohan	12000	Noida
2	2	Pavan	25000	Delhi
3	3	Amit	22000	Dehradun
4	4	Sonu	22000	Noida
5	5	Deepak	28000	Gurgaon

```
-- Keyset-driven Cursor for Update
SET NOCOUNT ON
DECLARE @Id int
DECLARE @name varchar(50)
DECLARE Keyset_cur_empupdate CURSOR
KEYSET
FOR
SELECT EmpID, EmpName from Employee ORDER BY EmpName
OPEN Keyset_cur_empupdate
IF @@CURSOR_ROWS > 0
BEGIN
    FETCH NEXT FROM Keyset_cur_empupdate INTO @Id,@name
    WHILE @@Fetch_status = 0
    BEGIN
        IF @name='Pavan'
        Update Employee SET Salary=27000 WHERE CURRENT OF Keyset_cur_empupdate
        FETCH NEXT FROM Keyset_cur_empupdate INTO @Id,@name
    END
END
CLOSE Keyset_cur_empupdate
DEALLOCATE Keyset_cur_empupdate
SET NOCOUNT OFF

Go
Select * from Employee
```

	EmpID	EmpName	Salary	Address
1	1	Mohan	15000	Noida
2	2	Pavan	27000	Delhi
3	3	Amit	24000	Dehradun

Q13. What are SQL Server cursor alternatives?

Ans. As we know, the cursors are required when we need to update records in a database table in singleton fashion means row by row. A Cursor also impacts the performance of the SQL Server since it uses the SQL Server instance's memory, reduce concurrency, decrease network bandwidth and lock resources.

You should avoid the use of a cursor. You can use cursor alternatives like as WHILE loop, Temporary tables and Table variables. We should use cursor in that case when there is no option except cursor.

Q14. How XML data type works in SQL Server?

Ans. XML data type was introduced in SQL Server 2005 to work with XML data. Using this data type, we can store XML in its native format and can also query/modify the XML data within the XML. We can use XML data type like as:

- Variable
- Field/Column in a table
- The parameter in the user-defined function (UDF) or stored procedure (SP)
- the return value from a UDF or SP

We can define XML data type field to NOT NULL or we can provide a default value to it.

Q15. What are the limitations of the XML data type in SQL Server?

Ans. There are following limitations of XML data types:

- We can't directly compare an instance of the XML data type to another instance of the XML data type. For equality comparisons, we first need to convert the XML type to a character type.
- We can't use GROUP BY or ORDER BY with an XML data type column.
- We can't use XML data type field as a primary key, Unique key and foreign key.
- We can't define XML data type field with COLLATE keyword.

Q16. What are various ways for querying XML data?

Ans. There are following ways for querying XML data-

AUTO - It generates output with both elements and attributes features in combination with a subquery.

```
SELECT DeptName, EmpID
FROM Employee AS Emp JOIN Department AS Dept
ON Emp.DeptID= Dept.DeptID
FOR XML AUTO;
```

Output:

```
<Dept DeptName="IT">
  <Emp EmpID="1" />
  <Emp EmpID="2" />
</Dept>
<Dept DeptName="HR">
  <Emp EmpID="3" />
</Dept>
<Dept DeptName="Technical">
  <Emp EmpID="4" />
  <Emp EmpID="5" />
</Dept>
```

EXPLICIT - It converts the row set that is the result of the query execution, into an XML document. This mode provides more control over the format of the XML means in which format you want XML you need to define that format in select query.

```
SELECT 1 tag,
      NULL parent,
      EmpID [employee!1!ID],
      EmpName [employee!1!name],
      NULL [order!2!date],
      NULL [department!3!name]
FROM Employee

UNION ALL

SELECT 3, 1,
      EmpID,
      NULL,
      NULL,
      DeptName
FROM Employee e JOIN Department d
ON e.DeptID=d.DeptID
ORDER BY 3, 1
FOR XML EXPLICIT;
```

Output:

```
<employee ID="1" name="Shailendra">
  <department name="IT" />
</employee>
<employee ID="2" name="Mohan">
  <department name="IT" />
</employee>
<employee ID="3" name="Vipul">
  <department name="HR" />
</employee>
<employee ID="4" name="Mrinal">
  <department name="Technical" />
</employee>
<employee ID="5" name="Jitendra">
  <department name="Technical" />
</employee>
```

RAW - It produces a single element or the optionally provided element name for each row in the query result set that is returned by the select statement.

```
SELECT Emp.EmpID, Dept.DeptName
Employee as Emp JOIN Department as Dept
ON Emp.DeptID= Dept.DeptID
FOR XML RAW;
```


Output:

```
<row EmpID="1" DeptName="IT" />
<row EmpID="2" DeptName="IT" />
<row EmpID="3" DeptName="HR" />
<row EmpID="4" DeptName="Technical" />
<row EmpID="5" DeptName="Technical" />
```

Q17. What is an index?

Ans. An index is a database object which is associated with a table or view and used to speed up the retrieval of rows from the table or view. An index contains the keys fields or columns in the table or view. These keys are stored in a B-tree structure that enables SQL Server to find the row(s) associated with the key values quickly and efficiently.

Q18. What are the advantages of the index?

Ans. There are following advantages of index:

1. Allows quick and efficient access to information
2. Enforce uniqueness of records using Primary and Unique constraints

Q19. What are different types of indexes?

Ans. SQL Server supports two types of indexes as given below:

1. Clustered Index
2. Non-Clustered Index

Q20. What is a Clustered index?

Ans. A clustered index affects the way of storing the data in a table (i.e. on disk). Data in the database table is stored in sequential order of clustered index column value and for this reason; a table can contain only one clustered index. By default, primary key acts as clustered index but you can also define another column to clustered index except for the primary key.

You can think of a clustered index as a dictionary in alphabetical order, and a non-clustered index as a book's index.

Q21. What is a Non-Clustered index?

Ans. A non-clustered index does not affect the way of storing the data in a table (i.e. on disk). It creates a new object for the index and stores the column(s) selected for indexing with a pointer back to the row containing the indexed values.

In SQL Server on a table, you can have max 999 non-clustered index having SQL Server 2012 & 20008 and max 249 non-clustered index having SQL Server 2005.

Q22. What is full-text search in SQL Server?

Ans. It was introduced with SQL Server 2008. Full-Text search helps you to search records based on a word or phrase. For this, you need to create a full-text index on a table or indexed view and only one full-text index is allowed per table or view.

This feature works only with RTM (Ready to manufacture) version of SQL Server. It does not work with CTP (Community Technology Preview) version of SQL Server.



References

This book has been written by referring to the following sites:

1. <https://docs.microsoft.com/en-us/sql/sql-server> - Microsoft Docs - SQL Server
2. <https://stackoverflow.com/questions/tagged/sql-server> - Stack Overflow - SQL Server
3. <https://www.dotnettricks.com/learn/sqlserver> - Dot Net Tricks - SQL Server

