SQL Keys

1. Differences between primary key and unique key.
    - Primary key not allowed null values. But unique key allowed null values(typically one)
    - There is only one primary key in a table and it can be multiple unique keys in a table.
    - Primary key used for uniquely identification. But unique key used to ensure that values in a column are unique.
2. Foreign key => a column in one table that matches a primary key in another table.

3. The foreign key can **reference** the unique fields of any table in the database. The table that has the primary key is known as the parent table and the key with the foreign key is known as the child table.

4. Why do we need foreign key =>
    - Foreign key is used to reduce the redundancy(duplicate) in the table.
    - It helps to normalize the data in multiple tables.

5. What are the differences between primary key and foreign key

Primary key	Foreign Key
Always unique	Can be dupplicated
Cannot be null	Can be null
A table can have only one primary key	A table can have multiple foreign keys.

6. Candidate key – A Field/Fields that can uniquely identify each row.
7. Alternate key - A candidate key that is not currently selected as the primary key.
8. All alternate key can be candidate key. But all candidate can not be alternate key. Because primary key considered as candidate key. But it can not considered as a alternate key.
9. Why we need composite key => database table does not have a single column that can uniquely identify each row from the table.
10. Composite key - combination of two or more columns in a table to uniquely identify any record.
11. The individual columns making up the composite key can contain duplicate values, but the combination these columns must be unique. Any column oof the composite key must not contain NULL values.

SQL Indexes

1. SQL Indexes are special lookup tables that are used to speed up the data retrieval operations. It speeds up the performance of data retrieval queries slows down the performance of insert and update queries.
2. An index must be created on column(s) of a large table that are frequently queried for data retrieval.
3. Unique index – it does not allow to insert any duplicate values to table. Unique indexes are not only for performance. But also for data integrity. It is automatically created by PRIMARY and UNIQUE constraints.
4. Single column index – index is created only one table column.
5. Composite indexes - that can be created on two or more columns of a table.
6. Guidelines to use indexes =>
    - Should not be used on small tables.
    - Should not used on columns that contains a high number of Null values.
    - Should not used on tables that have large batch updates or insert operations.
7. We cannot delete the indexes created by PRIMARY KEY or UNIQUE constraints. If you want to delete it you have to drop the entire constraint using alter table statement.
8. Clustered Index -
    - It determines the physical order of data in the table.(Default order is Ascending)
    - A table can have only one clustered index.
    - The actual table data is stored in the index itself.

9.Non Clustered Index –

- It maintain a separate structure of the table with a pointer to the actual data row.
- A table can have multiple non clustered indexes.
- Data is stored separately.

9. The DROP INDEX statement does not drop indexes created by PRIMARY KEY or UNIQUE constraints. To drop indexes associated with them, we need to drop these constraints entirely.
10. Clustered index, physically sorts and stores the table's rows based on the indexed column(s). In non – clustered index, create a separate structure from the table and it contains pointers to the actual data rows.
11. Storage – clustered => stores actual data rows
    Storage – non clustered => stores index + pointer to data
    Therefore non – clustered indexes require less storage space than clustered index

12. A **unique index** can be either **clustered** or **non-clustered –** If your table already has a clustered index, then a new unique index you create is non clustered. If your table does not have a clustered index, and you create a unique index, it may be created a clustered index. Because table can have only one clustered index.

SQL Advanced Concepts

13. What are wildcards in SQL?
    - SQL wildcards are special characters used to match pattern in strings.
    - - => matches one character
    - % => matches any number of characters(zero or more)
14. What are the SQL hosting databases
    - MS SQL server
    - MS Access
    - MySQL
    - Oracle
15. What are the features MS SQL server provide?
    - Easy to use
    - Powerful
    - Minimum scalability and security
16. What are the SQL NULL Functions
    - ISNULL() – replaces the NULL values in a database table with a specific value.
    - COALESCE() – returns first non null expression among it's arguments
    - NULLIF() – if both expressions are the same it returns NULL, otherwise it returns first expression.
17. What is CHECK constraint?
    - It ensures that the data entered into the column meets the specified condition.
18. What are the OLTP and OLAP
    - OLTP (Online Transaction Processing) avoid redundancy and we follow normalization design.
    - OLAP (Online Analytical Processing) improve search performance and we follow denormalization design.
19. What is the difference between char and nchar?
    - Char – it stores only English characters, 1 character = 1 byte
    - Nchar – it stores non English characters also. 1 character = 2 bytes
20. Both clustered and non clustered indexes follows b tree structure. In clustered indexes, leaf nodes points to the actual data. But in non clustered indexes, leaf nodes get help from the clustered indexes.
21. We can not make permanent changes (insert,update,delete) to the database. But we can do that stored procedures.
22. What are @@IDENTITY and SCOPE_IDENTITY()?
    - Both used to retrieve last inserted identity value.
    - @@IDENTITY- Returns the **last identity value** generated **in the current session**, **across all scopes**, including **triggers**. If your insert causes a trigger to insert into another table with an identity column, `@@IDENTITY` will return the **identity from the trigger**, **not your original table**.
    - SCOPE_IDENTITY()- Returns the **last identity value** generated **in the same scope.** It **ignores identity values generated by triggers or other scopes**
    - Therefore SCOPE_IDENTITY() is more safe.

1. What are magic tables in SQL Server?
    - **magic tables** are special, **temporary, read-only tables** that exist only during the execution of a **trigger**.
    - **AFTER INSERT Trigger – only the inserted table has data and deleted is empty.**
    - **AFTER DELETE Trigger – only the deleted table has data and inserted is empty**
    - **AFTER UPDATE Trigger – deleted table contains old value before the update and inserted table contains new value after the update.**
2. DBMS vs RDBMS

DBMS	RDBMS
Stores data as files, hierarchies or other structures.	Stores data in tables.
Limited or no support for relationships between data.	Supports relationships using key (foreign key)
Doesn't support normalization.	Support normalization.

3. What is a trigger?
    - Trigger is an automatic action that execute when a specific event occurs in the table
4. What is the use of IS NULL operator?
    - In SQL IS NULL operator is used to filter data by verifying whether a particular column has a null value. The operator can be used with SQL statements such as SELECT, UPDATE and DELETE.
5. What is the use of IS NOT NULL operator?
    - In SQL IS NOT NULL operator is used to filter data by verifying whether a particular column has a not null value. The operator can be used with SQL statements such as SELECT, UPDATE and DELETE.
6. Why we need NOT NULL constraint?
    - NOT NULL constraint is used to ensure that a column in a table does not contain null. Actually it prevent insert null values during the insertion of the table.
7. What is use of BETWEEN operator?
    - It used to retrieve data within a specific range.
8. What about INTERSECT operator?
    - It is used to retrieve the records that are common between the result sets of two or more tables.
9. What about EXCEPT operator?
    - It will display the distinct records only from the first table which are not in common with the records of the second table.
10. What about TOP clause?
    - It used to restrict the number of rows returned by a SELECT query in SQL Server.
11. What are the advantages of using TOP clause?
    - To implement pagination
    - Sampling data - we can use quickly returned sample data to testing or analyzing a table.
    - Debugging - quickly returned small number of records can be used to test the correctness of the query.
12. What is DISTINCT keyword?

- DISTINCT keyword is associated with SELECT statement and it is used to fetch unique records from single or multiple columns.

```
using System.Net.Http.Json;
using System.Text.Json;
using System.Text.Json.Serialization;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            List<Department> departments = new List<Department>
            {
                new Department
                {
                    DeptId = 1,
                    Name = "IT",
                    Manager = new Employee(100,"Sudam",27),
                    Employees = new List<Employee>
                    {
                        new Employee(100,"Sudam",27),
                        new Employee(101,"Sudila",20),
                        new Employee(102,"Pathum",27),
                    }
                },
                new Department
                {
                    DeptId = 2,
                    Name = "HR",
                    Manager = new Employee(103,"Pasindu",27),
                    Employees = new List<Employee>
                    {
                        new Employee(103,"Pasindu",27),
                        new Employee(104,"Shanaka",35)
                    }
                }
            };
            var options = new JsonSerializerOptions
            {
                WriteIndented = true
            };
            string json = JsonSerializer.Serialize(departments, options);
            Console.WriteLine(json);
            List<Department>? deserialize =
JsonSerializer.Deserialize<List<Department>>(json);

            foreach (var dept in deserialize)
            {
                Console.WriteLine($"Department : {dept.Name} Id : {dept.DeptId}");
                Console.WriteLine($"Manager : {dept.Manager?.Name} Id :
{dept.Manager?.EmpId} Age : {dept.Manager?.Age}");
                Console.WriteLine("Employees");

                foreach (var emps in dept.Employees)
                {
                    Console.WriteLine($"Name : {emps.Name} Id : {emps.EmpId} Age :
{emps.Age}");
                }
            }
        }

    }

    class Employee
    {
        public int? EmpId { get; set; }
        public string? Name { get; set; }
        public int? Age { get; set; }

        public Employee() { }
        public Employee(int? id, string? name, int? age)
        {
            EmpId = id;
            Name = name;
            Age = age;
        }
    }

    class Department
    {
        public int? DeptId { get; set; }
        public string? Name { get; set; }
        public Employee? Manager { get; set; }
        public List<Employee>? Employees { get; set; }
    }
}
```