For More Details Call +91 8368 979712, 63804 86914
EmailID–a1projecttraining@gmail.com

SQL Server Interview Questions & Answers

Q1) What is the Difference between MySql Vs SQL Server Performance?

Ans.

MySQl Server Vs SQL Server		
Function	**MS SQL Server**	**MySql**
Developer	Microsoft	Oracle
License	Commercial	OpenSource
Cloud-based	No	No
Implementation Language	C++	C & C++
XML Support	Yes	Yes
Supported Programming Lang	It supports C#, PHP, Python, Ruby, R, Visual Basic, Java etc	It supports c++, C#, Java, PHP, Perl, Python, Ruby, Tcl, Delphi, D etc
Server-Side Scripting.	Net & TransaScripting.	Yes, it supports
Concurreny	It supports ,yes	It supports ,yes
Consistency Concept's	It enables immediate consistency	It enables immediate consistency
Transaction concepts	It supports ACID	It supports ACID

Q2) What is normalization? Explain different levels of normalization?

Ans. It is the way to eliminate redundant data

1. Reduces null value
2. Enables efficient indexing
3. 1NF – Removes duplicated attributes, Attribute data should be atomic, and attribute should be same kind.
4. 2NF – Should be in 1NF and each non-key is fully dependent on the primary key.
5. 3NF – Should be in 2NF and all the non-key attributes which are not dependent on the primary key should be removed. All the attributes which are dependent on the other non-key attributes should also be removed. Normalization is done in OLTP.

Q3) What is denormalization and when would you go for it?

Ans. It is the reverse process of normalization. It increases the query performance by reducing the joins. It is used for OLAP applications.

Q4) How do you implement one-to-one, one-to-many and many-to-many relationships while designing tables?

Ans. Relationships in sql server are explained below

1. One to One –It can be implemented as a single table. Rarely it is implemented in two tables. For each instance in the first entity there is one and only one in the second entity and vice versa.
2. One to Many –For each instance in the first entity there can be one or more in the second entity. For each instance in the second entity there can be one and only one instance in the first entity.
3. Many to Many –For each instance in the first entity there can be one or more instance in the second entity and moreover, for each instance in the second entity there can be one or more instance in the first entity.

Q5) Difference between Primary key and Unique key.

Ans. **Primary Key**

- 1.Enforces uniqueness of the column in a table
- 2.Default clustered index
- 3.Does not Allow nulls

Unique Key

- 1.Enforces uniqueness of the column in a table.
- 2.Default non-clustered index.
- 3.Allows one null value

Q6) Define following keys:

Ans. Candidate key, Alternate key, Composite key.

- 1.Candidate key –Key which can uniquely identify a row in table.
- 2.Alternate key –If the table has more than one candidate keys and when one becomes a primary key the rest becomes alternate keys.
- 3.Composite key –More than one key uniquely identify a row in a table.

Q7) What are defaults? Is there a column to which a default can't be bound?

Ans. 1.It is a value that will be used by a column if no value is supplied to that column while inserting data.

2.I can't be assigned for identity and timestamp values.

Q8) What are user defined data types and when you should go for them?

Ans. Lets you extend the base SQL server data types by providing a descriptive name and format to the database.

E.g. Flight_num appears in many tables and all these tables have varchar(8)

Create a user defined data-type

Q9) What is a transaction and what are ACID properties?

Ans. A transaction is a logical unit of work in which, all the steps must be performed or none. ACID stands for Atomicity, Consistency, Isolation, and Durability. These are the properties of a transaction.

Q10) What part does database design have to play in the performance of a SQL Server-based application?

Ans. It plays a very major part. When building a new system, or adding to an existing system, it is crucial that the design is correct. Ensuring that the correct data is captured and is placed in the appropriate tables, that the right relationships exist between the tables and that data redundancy is eliminated is an ultimate goal when considering performance. Planning a design should be an iterative process, and constantly reviewed as an application is developed. It is rare, although it should be the point that everyone tries to achieve, when the initial design and system goals are not altered, no matter how slightly. Therefore, a designer has to be on top of this and ensure that the design of the database remains efficient..

Q11) What can a developer do during the logical and physical design of a database in order to help ensure that their database and SQL Server-based application will perform well?

Ans. A developer must investigate volumes of data (capacity planning), what types of information will be stored, and how that data will be accessed. If you are dealing with an upgrade to an existing system, analyzing the present data and where existing data volumes occur, how that data is accessed and where the current response bottlenecks are occurring, can help you search for problem areas in the design.

A new system would require a thorough investigation of what data will be captured, and looking at volumes of data held in other formats also will aid design. Knowing your data is just as important as knowing the constituents of your data. Also, constantly revisit your design. As your system is built, check relationships, volumes of data, and indexes to ensure that the physical design is still at its optimum. Always be ready to check your system by using tools like the SQL Server Profiler.

Q12) What are the main steps in Data Modeling?

Ans.

- 1.Logical – Planning, Analysis and Design
- 2.Physical – Design, Implementation and Maintenance

DATABASE DEVELOPMENT / PROGRAMMING

Q13) What are cursors? Explain different types of cursors. What are the disadvantages of cursors? How can you avoid cursors?

Ans. Cursors allow row-by-row processing of the result sets.

Types of cursors:

Static – Makes a temporary copy of the data and stores in tempdb and any modifications on the base table does not reflected in data returned by fetches made by the cursor.

Dynamic – Reflects all changes in the base table.

Forward-only – specifies that cursor can only fetch sequentially from first to last.

Keyset-driven – Keyset is the set of keys that uniquely identifies a row is built in a tempdb.

Disadvantages of cursors:

Each time you fetch a row from the cursor, it results in a network roundtrip, whereas a normal SELECT query makes only one roundtrip, however large the result set is. Cursors are also costly because they require more resources and temporary storage (results in more IO operations). Further, there are restrictions on the SELECT statements that can be used with some types of cursors.

Most of the times set-based operations can be used instead of cursors.

Here is an example:

If you have to give a flat hike to your employees using the following criteria:

Salary between 30000 and 40000 — 5000 hike

Salary between 40000 and 55000 — 7000 hike

Salary between 55000 and 65000 — 9000 hike

In this situation, many developers tend to use a cursor, determine each employee's salary and update his salary according to the above formula. But the same can be achieved by multiple update statements or can be combined in a single UPDATE statement as shown below:

```
UPDATE tbl_emp SET salary =
CASE
WHEN salary BETWEEN 30000 AND 40000 THEN salary + 5000
WHEN salary BETWEEN 40000 AND 55000 THEN salary + 7000
WHEN salary BETWEEN 55000 AND 65000 THEN salary + 10000
END
```

another situation in which developers tend to use cursors: You need to call a stored procedure when a column in a particular row meets certain condition. You don't have to use cursors for this. This can be achieved using WHILE loop, as long as there is a unique key to identify each row.

Q14) Write down the general syntax for a SELECT statement covering all the options.

Ans. Here's the basic syntax: (Also checkout SELECT in books online for advanced syntax).

```
SELECT select_list
[HDEV:INTO new_table_]
FROM table_source
[HDEV:WHERE search_condition]
[HDEV:GROUP BY group_by_expression]
```

```
5   [HDEV:HAVING search_condition]
6   [ORDER BY order_expression [ASC | HDEV:DESC] ]
7
```

Q15) What is a Join and explain different types of Joins?

Ans. Joins are used in queries to explain how different tables are related. Joins also let you select data from a table depending upon data from another table.
Types of joins: INNER JOINs, OUTER JOINs, CROSS JOINs. OUTER JOINs are further classified as LEFT OUTER JOINS, RIGHT OUTER JOINS and FULL OUTER JOINS.

Q16) Can you have a nested transaction?

Ans. Yes, very much. Check out BEGIN TRAN, COMMIT, ROLLBACK, SAVE TRAN and @@TRANCOUNT

Q17) What is an extended stored procedure? Can you instantiate a COM object by using T-SQL?

Ans. An extended stored procedure is a function within a DLL (written in a programming language like C, C++ using Open Data Services (ODS) API) that can be called from T-SQL, just the way we call normal stored procedures using the EXEC statement.
Yes, you can instantiate a COM (written in languages like VB, VC++) object from T-SQL by using sp_OACreate stored procedure. Also see books online for sp_OAMethod, sp_OAGetProperty, sp_OASetProperty, sp_OADestroy.

Q18) What is the system function to get the current user's userid?

Ans. USER_ID(). Also check out other system functions like USER_NAME(), SYSTEM_USER, SESSION_USER, CURRENT_USER, USER, SUSER_SID(), HOST_NAME().

Q19) What are triggers? How many triggers you can have on a table? How to invoke a trigger on demand?

Ans. Triggers are special kind of stored procedures that get executed automatically when an INSERT, UPDATE or DELETE operation takes place on a table. In SQL Server 6.5 you could define only 3 triggers per table, one for INSERT, one for UPDATE and one for DELETE. From SQL Server 7.0 onwards, this restriction is gone, and you could create multiple triggers per each action. But in 7.0 there's no way to control the order in which the triggers fire. In SQL Server 2000 you could specify which trigger fires first or fires last using sp_settriggerorder.
Triggers can't be invoked on demand. They get triggered only when an associated action (INSERT, UPDATE, DELETE) happens on the table on which they are defined. Triggers are generally used to implement business rules, auditing. Triggers can also be used to extend the referential integrity checks, but wherever possible, use constraints for this

purpose, instead of triggers, as constraints are much faster.
Till SQL Server 7.0, triggers fire only after the data modification operation happens. So in a way, they are called post triggers. But in SQL Server 2000 you could create pre triggers also – INSTEAD OF triggers.
Virtual tables – Inserted and Deleted form the basis of trigger architecture.

Q20) What is a self join? Explain it with an example.

Ans. Self join is just like any other join, except that two instances of the same table will be joined in the query. Here is an example: Employees table which contains rows for normal employees as well as managers. So, to find out the managers of all the employees, you need a self join.

```
CREATE TABLE emp
(
empid int,
mgrid int,
empname char(10)
)
INSERT emp SELECT 1,2,'Vyas'
INSERT emp SELECT 2,3,'Mohan'
INSERT emp SELECT 3,NULL,'Shobha'
INSERT emp SELECT 4,2,'Shridhar'
INSERT emp SELECT 5,2,'Sourabh'
SELECT t1.empname [HDEV:Employee], t2.empname [HDEV:Manager]
FROM emp t1, emp t2
WHERE t1.mgrid = t2.empid
```

Here's an advanced query using a LEFT OUTER JOIN that even returns the employees without managers (super bosses)

```
SELECT t1.empname [HDEV:Employee], COALESCE(t2.empname, 'No manager') [HDEV:Manager]
FROM emp t1
LEFT OUTER JOIN
emp t2
ON
t1.mgrid = t2.empid
```

Q21) Write a SQL Query to find first Week Day of month?

```
SELECT DATENAME(dw, DATEADD(dd, - DATEPART(dd, GETDATE()) + 1, GETDATE())) AS
FirstDay
```

Q22) How to find 6th highest salary from Employee table?

```
SELECT TOP 1 salary FROM (SELECT DISTINCT TOP 6 salary FROM employee ORDER BY
salary DESC) a ORDER BY salary
```

Q23) How can I enforce to use particular index?

You can use index hint (index=index_name) after the table name. SELECT au_lname FROM authors (index=aunmind)

Q24) What is ORDER BY and how is it different than clustered index?

The ORDER BY clause sorts query results by one or more columns up to 8,060 bytes. This will happen by the time when we retrieve data from database. Clustered indexes will physically sort data, while inserting/updating the table.

Q25) What is the difference between a UNION and a JOIN?

A JOIN selects columns from 2 or more tables. A UNION selects rows.

Q26) What is the Referential Integrity?

Referential integrity refers to the consistency that must be maintained between primary and foreign keys, i.e. every foreign key value must have a corresponding primary key value

Q27) What is the purpose of UPDATE STATISTICS?

It updates information about the distribution of key values for one or more statistics groups (collections) in the specified table or indexed view.

Q28) What is the use of SCOPE_IDENTITY() function?

It returns the most recently created identity value for the tables in the current execution scope.

Q29) What do you consider are the best reasons to use stored procedures in your application instead of passing Transact-SQL code directly to SQL Server?

First and foremost, a stored procedure is a compiled set of code, where passing T-SQL through languages such as VB, Visual FoxPro, etc., means that the set of code needs to be compiled first. Although T-SQL within VB, etc., can be prepared before running, this is still slower than using a stored procedure. Then, of course, there is the security aspect, where, by building a stored procedure, you can place a great deal of security around it. When dealing with sensitive data, you can use an encrypted stored procedure to hide sensitive columns, calculations, and so on. Finally, by using a stored procedure, I feel that transactional processing becomes a great deal easier and, in fact, using nested transactions become more insular and secure. Having to deal with transactions within code that may have front end code, will slow up a transaction and therefore a lock will be held for longer than necessary.

Q30) What are some techniques for writing fast performing stored procedures?

Fast performing stored procedures are like several other areas within T-SQL. Revisiting stored procedures every six months or so, to ensure that they are still running at their optimum performance is essential. However, actual techniques themselves include working with as short a transaction area as possible, as lock contention will certainly impact performance. Recompiling your stored procedures after index additions if you are unable or not wishing to restart SQL Server, will also ensure that a procedure is using the correct index, if that stored procedure is accessing the table which has received the new index. If you have a T-SQL command that joins several tables, and it takes a long time to return a value, first of all check out the indexes. But what you may find tends to help, is to break down the code and try to determine which join it is that is causing the performance problem. Then analyze this specific join and see why it is a problem. Always check out a stored procedure's performance as you build it up by using the SHOWPLAN commands.
Also, try to use EXISTS, rather than a JOIN statement. An EXISTS statement will only join on a table until one record is found, rather than joining all the records . Also, try to look at using sub queries when you are trying to find a handful of values in the sub query statement, and there is no key on the column you are looking up on.

Q31) When should SQL Server-based cursors be used, and not be used?

SQL Server cursors are perfect when you want to work one record at a time, rather than taking all the data from a table as a single bulk. However, they should be used with care as they can affect performance, especially when the volume of data increases. From a beginner's viewpoint, I really do feel that cursors should be avoided every time because if they are badly written, or deal with too much data, they really will impact a system's performance. There will be times when it is not possible to avoid cursors, and I doubt if many systems exist without them. If you do find you need to use them, try to reduce the number of records to process by using a temporary table first, and then building the cursor from this. The lower the number of records to process, the faster the cursor will finish. Always try to think "out of the envelope".

Q32) What alternatives do developers have over using SQL Server-based cursors? In other words, how can developers perform the same function as a cursor without using a cursor?

Perhaps one of the performance gains least utilized by developers starting out in SQL Server are temporary tables. For example, using one or more temporary tables to break down a problem in to several areas could allow blocks of data to be processed in their own individual way, and then at the end of the process, the information within the

temporary tables merged and applied to the underlying data. The main area of your focus should be, is there an alternative way of doing things? Even if I have to break this down into several chunks of work, can I do this work without using cursors, and so result in faster performance. Another area that you can look at is the use of CASE statements within your query. By using a CASE statement, you can check the value within a column and make decisions and operations based on what you have found. Although you will still be working on a whole set of data, rather than a subset found in a cursor, you can use CASE to leave values, or records as they are, if they do not meet the right criteria. Care should be taken here though, to make sure that by looking at all the data, you will not be creating a large performance impact. Again, look at using a subset of the data by building a temporary table first, and then merging the results in afterwards. However, don't get caught out with these recommendations and do any of them in every case. Cursors can be faster if you are dealing with small amounts of data. However, what I have found, to be rule number one, is get as little data in to your cursor as is needed.

Q33) If you have no choice but to use a SQL Server-based cursor, what tips do you have in order to optimize them?

Perhaps the best performance gain is when you can create a cursor asynchronously rather than needing the whole population operation to be completed before further processing can continue. Then, by checking specific global variables settings, you can tell when there is no further processing to take place. However, even here, care has to be taken. Asynchronous population should only occur on large record sets rather than those that only deal with a small number of rows. Use the smallest set of data possible. Break out of the cursor loop as soon as you can. If you find that a problem has occurred, or processing has ended before the full cursor has been processed, then exit. If you are using the same cursor more than once in a batch of work, and this could mean within more than one stored procedure, then define the cursor as a global cursor by using the GLOBAL keyword, and not closing or deallocating the cursor until the whole process is finished. A fair amount of time will be saved, as the cursor and the data contained will already be defined, ready for you to use.

DATABASE PERFORMANCE OPTIMIZATION / TUNING

Q34) What are the steps you will take to improve performance of a poor performing query?

This is a very open ended question and there could be lot of reasons behind the poor performance of a query. But some general issues that you could talk about would be:

1. No indexes
2. No Table scans

3. Missing or out of date statistics
4. Blocking
5. Excess recompilations of stored procedures

Q35) What is an ER Diagram?

An ER diagram or Entity-Relationship diagram is a special picture used to represent the requirements and assumptions in a system from a top down perspective. It shows the relations between entities (tables) in a database.

Q36) What is a prime attribute?

A prime attribute is an attribute that is part of a candidate key.

Q37) What are the properties of a transaction?

The ACID properties. Atomicity, Consistency, Isolation, and Durability.

Q38) What is a non-prime attribute?

A non-prime attribute is an attribute that is not a part of a candidate key.

Q39) What is Atomicity?

This means the transaction finish completely, or it will not occur at all.

Q40) What is Consistency?

Consistency means that the transaction will repeat in a predictable way each time it is performed.

Q41) What is Isolation?

The data the transactions are independent of each other. The success of one transaction doesn't depend on the success of another.

Q42) What is Durability?

Guarantees that the database will keep track of pending changes so that the server will be able to recover if an error occurs.

Q43) What is a DBMS?

A DBMS is a set of software programs used to manage and interact with databases.

Q44) What is a RDBMS?

It is a set of software programs used to interact with and manage relational databases. Relational databases are databases that contain tables.

Q45) What is business intelligence?

Refers to computer-based techniques used in identifying, extracting, and analyzing business data, such as sales revenue by products and/or departments, or by associated costs and incomes.

Q46) What is normalization?

Database normalization is the process of organizing the fields and tables of a relational database to minimize redundancy and dependency.

Q47) What is a relationship?

The way in which two or more concepts/entities are connected, or the state of being connected.

Q48) What are the different types of relationships?

One to one, one to many, many to many, many to fixed cardinality.

Q49) What is the difference between a OLTP and database?

An OLTP is the process of gathering the data from the users, and a database is the initial information.

Q50) What are the different kinds of relationships?

Identifying and non-identifying.

Q51) What is an entity?

Something that exists by itself, although it need not be of material existence.

Q52) What is a conjunction table?

A table that is composed of foreign keys that points to other tables.

Q53) What is a relational attribute?

An attribute that would not exist if it were not for the existence of a relation.

Q54) What are associative entities?

An associative entity is a conceptual concept. An associative entity can be thought of as both an entity and a relationship since it encapsulates properties from both. It is a relationship since it is serving to join two or more entities together, but it is also an entity since it may have its own properties.

Q55) What is the difference between a derived attribute, derivedpersistent attribute, and computed column?

A derived attribute is a attribute that is obtained from the values of other existing columns and does not exist on it's own. A derived persistent attribute is a derived attribute that is stored. A computed attribute is a attribute that is computed from internal system values.

Q56) What are the types of attributes?

Simple, composite (split into columns), multi-valued (becomes a separate table), derived, computed, derived persistent.

Q57) Is the relationship between a strong and weak entity always identifying?

Yes, this is the requirement.

Q58) Do stand alone tables have cardinality?

No.

Q59) What is a simple key?

It is a key that in composed of one attribute.
Give/ recite the types of UDF functions.
Scalar, In-line, Multi

Q60) Describe what you know about PK, FK, and UK.

Primary keys – Unique clustered index by default, doesn't accept null values, only one primary key per table.
Foreign Key – References a primary key column. Can have null values. Enforces referential integrity.
Unique key – Can have more than one per table. Can have null values. Cannot have repeating values. Maximum of 999 clustered indexes per table.

Q61) What do you mean by CTEs? How will you use it?

CTEs also known as common table expressions are used to create a temporary table that will only exist for the duration of a query. They are used to create a temporary table whose content you can reference in order to simplify a queries structure.

Q62) What is a sparse column?

It is a column that is optimized for holding null values.

Q63) What would the command: DENY CREATE TABLE TO Peter do?

It wouldn't allow the user Peter to perform the operation CREATE TABLE regardless of his role.

Q64) What does the command: GRANT SELECT ON project TO Peter do?

It will allow the SELECT operation on the table 'project' by Peter.

Q65) What does the command: REVOKE GRANT SELECT ON project TO Peter do?

It will revoke the permission granted on that table to Peter.

Q66) New commands in SQL 2008?

Database encryption, CDCs tables – For on the fly auditing of tables, Merge operation, INSERT INTO – To bulk insert into a table from another table, Hierarchy attributes, Filter indexes, C like operations for numbers, resource management, Intellisense – For making programming easier in SSMS, Execution Plan Freezing – To freeze in place how a query is executed.
What is new in SQL 2008 R2?
PowerPivot, maps, sparklines, data bars, and indicators to depict data.

Q67) What is faster? A table variable or temporary table?

A table variable is faster in most cases since it is held in memory while a temporary table is stored on disk. However, when the table variable's size exceeds memory size the two table types tend to perform similarly.

Q68) How big is a tinyint, smallint, int, and bigint?

1 byte, 2 bytes, 4 bytes, and 8 bytes.

Q69) What does @@trancount do?

It will give you the number of active transactions for the current user.

Q70) What are the drawbacks of CTEs?

It is query bound.

Q71) What is the transaction log?

It keeps a record of all activities that occur during a transaction and is used to roll back changes.

Q72) What are before images, after images, undo activities and redo activities in relation to transactions?

Before images refers to the changes that are rolled back on if a transaction is rolled back. After images are used to roll forward and enforce a transaction. Using the before images is called the undo activity. Using after images is called the redo activity.

Q73) What are shared, exclusive and update locks?

A shared lock, locks a row so that it can only be read. An exclusive lock locks a row so that only one operation can be performed on it at a time. An update lock basically has the ability to convert a shared lock into an exclusive lock.

Q74) What does WITH TIES do?

If you use TOP 3 WITH TIES *, it will return the rows, that have a similarity in each of their columns with any of the column values from the returned result set.

Q75) How can you get a deadlock in SQL?

By concurrently running the same resources that access the same information in a transaction.

Q76) What is LOCK_TIMEOUT used for?

It is used for determining the amount of time that the system will wait for a lock to be released.

Q77) What is the ANY predicate used for?

```
1  SELECT * FROM emp_table WHERE enter_date > ANY (SELECT enter_date FROM works_on)
```

Q78) What is the ALL predicate used for?

```
1   SELECT * FROM emp_table WHERE enter_date > ALL (SELECT enter_date FROM works_on)
```

Q79) What are some control flow statements in SQL?

while, if, case, for each etc..

Q80) What is the EXISTS function used for?

It is used to determine whether a query returns one or more rows. If it does, the EXIST function returns TRUE, otherwise, it will return FALSE.

SQL Query Interview Questions with Answers

Inner Join: It is used to retrieve matching records from both the tables

Department:

Department_No	Department_Name
10	ECE
20	ECE
30	CSE
40	IT

Employee Details:

Employee_No	Emp_Name	Address	Age	Department_No	Salary
1	Anil	Hyderabad	23	10	20000
2	Sunil	Hyderabad	22	10	21000
3	Ajay	Chennai	24	20	23000
4	Vijay	Chennai	25	30	22000
5	James	Hyderabad	24	50	230000

Q1) Write a Query to display employee details who are working in ECE department?

```
1   SELECT employee.employee_name, employee.address, employee.salary, employee.age,
2   FROM Department D
3   INNER JOIN Employees E
4   ON department.D_no=employee.D_no WHERE department.D_name= 'ECE'
```

Q2) Write a Query to display employee details?

```
1   SELECT * FROM employee;
```

Q3) Write a Query to display employee details along with department_name?

```
SELECT employee.employee_no, employee.employee_name, employee.address, employee.salary
department.department_name
FROM department D
INNER JOIN employee E
ON department.D_no=employee.D_no
```

Q4) Write a Query to display employee details whose sal>20000 and who is working in ECE department?

```
SELECT employee.employee_no, employee.employee_name, employee.address,
employee.salary, employee.age
FROM department D
INNER JOIN employee E
ON dept.D_no=emp.D_no
WHERE dept.D_name='ECE' and E.salary>20000
```

5) Write a Query to display employee details along with department_name and who is working in ECE department, whose name starts with a?

```
SELECT emp.e_no, emp.e_name, emp.address, emp.salary, emp.age, dept.dname
FROM department D
INNER JOIN employee E
ON dept.D_no=emp.D_no
WHERE dept.D_name='ECE' and emp.E_name like 'a%'
```

Q6) Write a Query to display employee details along with department_name and whose age between 20 and 24?

```
SELECT emp.e_no, emp.e_name, emp.address, emp.salary, emp.age, dept.d_name
FROM department D
INNER JOIN employee E
ON dept.D_no=emp.D_no
WHERE E.age between 20 and 24
```

Q7) Write a Query to display employee details along with department_name and who are staying in hyderabad?

```
SELECT emp.e_no, emp.e_name, emp.address, emp.salary, emp.age, dept.d_name
FROM department D
INNER JOIN employee E
ON dept.D_no=emp.D_no
WHERE E.address='hyd'
```

Q8) Write a Query to display employee details whose salary>20000 and whose age>20 & who is working in ECE department?

```
SELECT emp.e_no, emp.e_name, emp.address, emp.salary, emp.age, dept.d_name
FROM department D
INNER JOIN employee E
ON dept.D_no=emp.D_no
WHERE E.age>20 and E.salary>20000 and dept.D_name='ECE'
```

State Table:

State ID	State Name
S1	Telangana
S2	AP
S3	Tamil Nadu
S4	Karnataka
S5	Kerala

City

City ID	City Name	State ID
1	Hyderabad	S1
2	Vizag	S2
3	Vijayawada	S2
4	Chennai	S3
5	Madhurai	S3
6	Bangalore	S4

Blood Group Details

Blood Group ID	Blood Group
B1	A+ve
B2	B+ve
B3	AB +ve
B4	A -ve
B5	O +ve

Donor Details

Donor ID	Donor Name	Phone Number	City ID	Blood Group ID
D1	Anil	9999	1	B1
D2	Sunil	8888	1	B1
D3	Ajay	7777	2	B1
D4	John	6666	4	B3
D5	James	5555	4	B5

Q9) Write a Query to display city names belongs to AP?

```
SELECT C.City_Name
FROM State S
INNER JOIN City C
ON S.State_ID
WHERE S.State_Name 'AP'
```

Q10) Write a Query to display Donor_ID, Donor_Name, Phone No, City?

```
SELECT D.Donor_ID, D_Name, D_Phone No, C.City_Name
FROM Donor D
INNER JOIN City C
ON D.City_ID=C.City_ID
```

Q11) Write a Query to display Donor_ID, Donor_Name, Phone No, Blood Group?

```
SELECT D.Donor_ID, D_Name, D_Phone No, B.Blood_Group
FROM Donor D
INNER JOIN Blood B
ON D.Blood_ID=B.Blood_ID;
```

Q12) Write a Query to display Donor_ID, Donor_Name, Phone No and who are staying in hyderabad?

```
SELECT D.Donor_ID, D_Name, D_Phone No, C.City_Name
FROM Donor D
INNER JOIN City C
ON C.City_ID=D.City_ID
WHERE C.City_Name='hyderabad'
```

Q13) Write a Query to display donor details whose blood group is A +ve?

```
SELECT D.Donor_ID, D_Name, D_Phone No
FROM Donor D
INNER JOIN Blood B
ON D.Donor_ID=B.Blood_ID
WHERE B.Blood_Group='A+ve'
```

Q14) Write a Query to display Donor_ID, Donor_Name, Phone No, City, Blood Group?

```
SELECT D.Donor_ID, D_Name, D_Phone No, C.City_Name B.Blood_Group
FROM Blood B
INNER JOIN Donor D
ON D.Blood_ID=B.Donor_Name
INNER JOIN City C
ON D.City_ID=C.City_ID
```

Q15) Write a Query to display Donor_Name, Phone No, Blood Group of the donors who is staying in hyderabad and whose blood group is A+ve?

```
SELECT D.Donor_Name, D. Phone_Number, B.Blood_Group
FROM Donor D
INNER JOIN Blood B
```

```
ON D.Blood_ID=B.Blood_ID
INNER JOIN City C
ON D.City_ID=C.City_ID
WHERE C.City_Name='hyderabad' and B.Blood_Group='A+ve'
```

Outer Join A join that includes rows even if they do not have related rows in the joined table is an Outer Join.. You can create three different outer join to specify the unmatched rows to be included:

- Left Outer Join
- Right Outer Join
- Full Outer Join

Employee Details Table

Employee_No	Employee_Name	Dept_No
101	Anil	10
102	Sunil	20
103	Ajay	30
104	Vijay	40
105	Null	Null

Department Details Table

Dept_No	Depat_Name
10	EEE
20	EEE
30	CSE
Null	Null
50	IT

Q16) Write a Query to display only left records?

```
SELECT e.*
FROM Employee E
LEFT OUTER JOIN Department D
ON E.D_no
WHERE D.D_No IS NULL
```

Q17) Write a Query to display employee details where employee no is 101?

```
SELECT *
FROM Employee E
WHERE E_No=101
```

Q18) Write a Query to display employee details where employee number is null?

```
SELECT *
FROM Employee E
WHERE E_No IS NULL
```

Q19) Write a Query to display only right records?

```
SELECT D.*
FROM Employee E
RIGHT OUTER JOIN Department D
ON E.D.No=D.D_No
WHERE E.D_No IS NULL
```

Q20) Write a Query to display all the records from the table except matching records?

```
SELECT E.*, D.*
FROM Employee E
FULL JOIN Department D
ON E.D_No=D.D_No
WHERE E.D_No IS NULL or D.D_No IS NULL
```

Department Details Table

Dept_No	Dept_Name
1	ECE
2	CSE
3	EEE

Course Details Table

Course_ID	Course_Name	Cr
1	EDC	4
2	PDC	4
3	SS	4
4	DAA	4
5	OS	4

Student Details Table

Student_No	Student_Name
101	Anil
102	Sunil
103	Ajay
104	Vijay

105	John

Enroll Details Table

Enroll_Date	Student_No	Dpet_No	S_ID
1/2/2014	101	10	S1
3/2/2016	102	10	S1
3/2/2016	103	10`	S1
3/2/2016	104	20	S2
3/2/2016	105	20	S2

Address Table

Emp_No	Address
E1	Hyderabad
E2	Vizag
E3	Hyderabad
E4	Bangalore
E5	Hyderabad

Employee Details Table

Emp_No	Emp_Name
E1	Arun
E2	Kiran
E3	Kumar
E4	Anus
E5	James

Semester Details Table

Semester	Sn
S1	1
S2	2-1
S3	2-2
S4	3-1
S5	3-2
S6	4-1

S7	4-2

Course Department Details

Dept_No	Course_ID
10	1
10	2
10	3
20	4
20	5

Syllabus Table

Dept_No	Course_ID	S_ID
10	1	S1
10	2	S1
10	3	S1
20	4	S2
20	5	S2

Instructor Details Table

Emp_No	Dept_No
E1	10
E2	10
E3	10
E4	20
E5	30

Course Instructor Table

Course_ID	Emp_No	S_ID	Dept_No
1	E1	S1	10
1	E1	S1	20
1	E2	S1	30
2	E3	S1	10
4	E4	S2	20
5	E4	S2	20

5	E5	S1	10

Q) Write a query to display Student No, Student Name, Enroll Date, Department Name?

```
SELECT S.Student_No,   S.Student_Name, S.Enroll_Date, D.Dept_Name
FROM Student S
INNER JOIN Enroll E
ON S.Student_No=E.Student_No
INNER JOIN Department D
ON D.Dept_No=E.Dept_No
```

Q) Write a query to display Employee Number, Employee Name and address, department name?

```
SELECT E.Emp_No, E.Emp_Name, A.Address, D.Dept_Name
FROM Employee E
INNER JOIN Address A
ON E.Emp_No=A.Emp_No
INNER JOIN Instructor I
ON A.Emp_No=I.Emp_No
INNER JOIN Department D
ON I.Dept_No=D.Dept_No
```

Q) Write a query to display course name belongs to ECE department?

```
SELECT C.Course_Name
FROM Department D
INNER JOIN Course Department CD
ON D.Dept_NO=CD.Dept_NO
INNER JOIN Course C
ON CD.CourseDept_ID=C.Course_ID
WHERE D.Dept_Name='ECE'
```

Q)) Write a query to display student number, student name, enroll date, dept name, semester name?

```
SELECT S.Student_No, S.Student_Name, S.Enroll_Date, D.Dpet_Name, Sem.Student_Name
FROM Enroll E
INNER JOIN Student S
ON S.Student_No=E.Student_No
INNER JOIN Deprtment D
ON E.Dept_No=D.Dept_No
INNER JOIN Semester SE
ON E.Student_ID=Sem.Student_ID
```

Q) Write a query to display the syllabus of ECE department 1st year?

```
SELECT C.Course_Name
FROM Department D
INNER JOIN Syllabus Sy
ON D.Dept_No=Sy.Dept_No
INNER JOIN Course C
ON Sy.Course_ID=C.Course_ID
INNER JOIN Semester Se
```

```
ON Syllabus_Sy_ID=Se_Sy_ID
WHERE D.Dept_Name='ECE' and Se.Semester='1'
```

Q) Write a query to display the employee names and faculty names of ECE dept 1st year?

```
SELECT E.Emp_Name
FROM Employee E
INNER JOIN Course Instructor Ci
ON E.Emp_No=Ci.Emp_No
INNER JOIN Semester Se
ON Se.Student_ID=Ci.Student_ID
INNER JOIN Dept D
ON Ci.Dept_No=D.Dept_No
WHERE D.Dept_Name='ECE' and Se.Student_Name='1'
```

Q)) Write a query to display student details who enrolled for ECE department?

```
SELECT S.Student_NO, S.Student_Name, S.Enroll_Date
FROM Student S
INNER JOIN Enroll E
ON S.Student_No=E.Student_No
INNER JOIN Department D
ON E.Dept_No=D.Dept_No
WHERE D.Dept_Name='ECE'
```

Q)) Write a query to display student details along with dept name who are enrolled in ECE department first year?

```
SELECT S.Student_No, S.Student_Name, S.Enroll_Date, D.Dept_Name
FROM Student S
INNER JOIN Enrollment E
ON S.Student_No=E.Student_No
INNER JOIN Department D
ON D.Dept_No=E.Dept_No
INNER JOIN Semester Se
ON E.Student_ID=Se.Student_ID
WHERE D.Dept_Name='ECE' and Se.Student_Name='1'
```

Q)) Write a query to display employee name who is teaching EDC?

```
SELECT E.Emp_Name
FROM Employee E
INNER JOIN Course Instructor Ci
ON E.Emp_No=Ci.Emp_No
INNER JOIN Course C
ON Ci.Course_ID=C.Course_ID
WHERE C.Course_Name='EDC'
```

Q)) Write a query to display employee details along with dept name who are staying in Hyderabad?

```
SELECT E.Emp_No, Emp_Name, D.Dept_Name
FROM Employee E
```

```
INNER JOIN Address A
ON E.Emp_No=A.Emp_No
INNER JOIN Instructor I
ON A.Emp_No=I.Emp_No
INNER JOIN Department D
ON I.Dept_No=D.Dept_No
WHERE A.Address='hyderabad'
```

Emp_No	Emp_Name	Salary	Age	Dept_Name
101	Anil	20,000	22	ECE
102	Sunil	23000	23	EEE
103	Vijay	32000	24	CSE

Using Range Operator:: BETWEEN, NOT BETWEEN

Q) Write a Query to display employee details whose salary > 20000 and whose age >23?

```
SELECT * FROM Employee
WHERE Salary>20000 AND Age>23;
```

Q) Write a Query to display employee details whose salary >20000 and who is working in ECE department?

```
SELECT * FROM Employee
WHERE Salary>20000 AND Dept_Name='ECE'
```

Q) Write a Query to display employee details whose age is BETWEEN 18 and 22?

```
SELECT * FROM Employee Details
WHERE Age BETWEEN 18 AND 22;
```

Q) Write a Query to display employee details whose salary range BETWEEN 20000 and 23000?

```
SELECT * FROM Employee
WHERE Salary BETWEEN 20000 AND 23000;
```

Q) Write a Query to display employee details whose age is NOT BETWEEN 18 & 22?

```
SELECT * FROM Employee
WHERE Age NOT BETWEEN 18 AND 22;
```

Using String Operators:: LIKE, NOT LIKE

Q) Write a Query to display employee details whose name starts with a?

```
SELECT * FROM Employee
WHERE Emp_Name LIKE 'a%'
```

a% ----> starts with a

%a ----> ends with a

Q) Write a Query to display employee details and whose age>20 & whose name starts with a?

```
SELECT * FROM Employee
WHERE Salary>20000 AND Age>20 AND Emp_Name LIKE 'a%'
```

Q) Write a Query to display employee details whose name not starts with a?

```
SELECT * FROM employee
WHERE Emp_Name NOT LIKE 'a%'
```