1. DDL – Data Definition Language

DDL is used to define the structure that holds the data. For example, Create, Alter, Drop and Truncate table.

2. DML – Data Manipulation Language

DML is used for manipulation of the data itself. Typical operations are Insert, Delete, Update and retrieving the data from the table. The Select statement is considered as a limited version of the DML, since it can't change the data in the database. But it can perform operations on data retrieved from the DBMS, before the results are returned to the calling function.  
3. DCL – Data Control Language   
DCL is used to control the visibility of data like granting database access and set privileges to create tables, etc. Example - Grant, Revoke access permission to the user to access data in the database.

**Get employee details from employee table whose joining year is “2013”**

SQL Queries in Oracle, Select \* from EMPLOYEE where to\_char(joining\_date,'YYYY')='2013'  
  
SQL Queries in SQL Server, Select \* from EMPLOYEE where SUBSTRING(convert(varchar,joining\_date,103),7,4)='2013'  
  
SQL Queries in MySQL, Select \* from EMPLOYEE where year(joining\_date)='2013'

**Question 1: SQL Query to find second highest salary of Employee**

Answer: There are many ways to find second highest salary of Employee in SQL, you can either use SQL Join or Subquery to solve this problem. Here is SQL query using Subquery:

select MAX(Salary) from Employee WHERE Salary NOT IN (select MAX(

**Question 2: SQL Query to find Max Salary from each department.**

Answer: You can find the maximum salary for each department by grouping all records by DeptId and then using MAX() function to calculate maximum salary in each group or each department.

SELECT DeptID, MAX(Salary) FROM Employee  GROUP BY DeptID.

These questions become more interesting if Interviewer will ask you to print department name instead of department id, in that case, you need to join Employee table with Department using foreign key DeptID, make sure you do LEFT or RIGHT OUTER JOIN to include departments without any employee as well.  Here is the query

SELECT DeptName, MAX(Salary) FROM Employee e RIGHT JOIN Department d ON e.DeptId = d.DeptID GROUP BY DeptName;

In this query, we have used RIGHT OUTER JOIN because we need the name of the department from Department table which is on the right side of JOIN clause, even if there is no reference of dept\_id on Employee table.

**Write an SQL Query find number of employees according to gender  whose DOB is between 01/01/1960 to 31/12/1975.**

SELECT COUNT(\*), sex from Employees WHERE DOB BETWEEN '01/01/1960' AND '31/12/1975' GROUP BY sex;

**Question 11: Write SQL Query to find duplicate rows in a database? and then write SQL query to delete them?**  
Answer: You can use the following query to select distinct records:

SELECT \* FROM emp a WHERE rowid = (SELECT MAX(rowid) FROM EMP b WHERE a.empno=b.empno)

to Delete:

DELETE FROM emp a WHERE rowid != (SELECT MAX(rowid) FROM emp b WHERE a.empno=b.empno);

**Question 13: How do you find all employees which are also manager? .**  
You have given a standard employee table with an additional column mgr\_id, which contains employee id of the manager.

Answer: You need to know about self-join to solve this problem. In Self Join, you can join two instances of the same table to find out additional details as shown below

SELECT e.name, m.name FROM Employee e, Employee m WHERE e.mgr\_id = m.emp\_id;

One follow-up is to modify this query to include employees which don't have a manager. To solve that, instead of using the inner join, just use left outer join, this will also include employees without managers.

**How to delete duplicate rows in a table?**delete from emp a where rowid != (select max(rowid) from emp b where  a.empno=b.empno);

List all the Employees who have at least one person reporting to them.  
SELECT ENAME FROM EMPLOYEE WHERE EMPNO IN (SELECT MGR FROM EMPLOYEE);

What Is Normalization?

Normalization is the process of minimizing redundancy and dependency by organizing fields and table of a database. The main aim of Normalization is to add, delete or modify field that can be made in a single table.

The normal forms can be divided into 5 forms, and they are explained below :  
**First Normal Form (1NF):** This should remove all the duplicate columns from the table. Creation of tables for the related data and identification of unique columns.  
**Second Normal Form (2NF):** Meeting all requirements of the first normal form. Placing the subsets of data in separate tables and Creation of relationships between the tables using primary keys.  
**Third Normal Form (3NF):** This should meet all requirements of 2NF. Removing the columns which are not dependent on primary key constraints.  
**Fourth Normal Form (3NF):** Meeting all the requirements of third normal form and it should not have multi- valued dependencies..

Question #8) When do we use triggers**?**

**Answer:** The word ‘Trigger’ means to activate. In PL/SQL, the trigger is a stored procedure that defines an action taken by the database when database related event is performed. Triggers are mainly required for the following purposes

To maintain complex integrity constraints

Auditing table information by recording the changes

Signaling other program actions when changes are made to table

Enforcing complex business rules

Preventing invalid transactions

[What is difference between a PROCEDURE & FUNCTION ?](http://www.geekinterview.com/question_details/4662" \o "What is difference between a PROCEDURE & FUNCTION ?)

. Function is mainly used in the case where it must return a value. Where as a procedure may or may not return a value or may return more than one value using the OUT parameter.  
2. Function can be called from SQL statements where as procedure can not be called from the sql statements   
  
3. Functions are normally used for computations where as procedures are normally used for executing business logic.   
  
4. You can have DML (insert,update, delete) statements in a function. But, you cannot call such a function in a SQL query.  
  
5.?Function returns 1 value only. Procedure can return multiple values (max 1024).   
  
6.Stored Procedure: supports deferred name resolution. Example while writing a stored procedure that uses table named tabl1 and tabl2 etc..but actually not exists in database is allowed only in during creation but runtime throws error Function wont support deferred name resolution.   
  
7.Stored procedure returns always integer value by default zero. where as function return type could be scalar or table or table values  
  
8. Stored procedure is precompiled execution plan where as functions are not.  
?  
9.A procedure may modify an object where a function can only return a value The RETURN statement immediately completes the execution of a subprogram and returns control to the caller.

**Explain the uses of database trigger?**

A PL/SQL program unit associated with a particular database table is called a database trigger. It is used for :

Audit data modifications.

Log events transparently.

Enforce complex business rules.

Maintain replica tables

Derive column values

Implement Complex security authorizations

Show some predefined exceptions.

DUP\_VAL\_ON\_INDEX

ZERO\_DIVIDE

NO\_DATA\_FOUND

TOO\_MANY\_ROWS

CURSOR\_ALREADY\_OPEN

INVALID\_NUMBER

INVALID\_CURSOR

PROGRAM\_ERROR

TIMEOUT \_ON\_RESOURCE

STORAGE\_ERROR

LOGON\_DENIED

VALUE\_ERROR

**show the cursor attributes of PL/SQL.**

%ISOPEN : Checks if the cursor is open or not

%ROWCOUNT : The number of rows that are updated, deleted or fetched.

%FOUND : Checks if the cursor has fetched any row. It is true if rows are fetched

%NOT FOUND : Checks if the cursor has fetched any row. It is True if rows are not fetched.

**Trigger**

Triggers provide a way of executing PL/SQL code on the occurrence of specific database events. For example, you can maintain an audit log by setting triggers to fire when insert or update operations are carried out on a table. The insert and update triggers add an entry to an audit table whenever the table is altered.  
  
The actions that Informix Dynamic Server triggers perform are constrained to multiple insert, update, delete, and execute procedure clauses; whereas, Oracle allows triggers to execute arbitrary PL/SQL code. Oracle triggers are similar to stored procedures in that they can contain declarative, execution, and exception handling code blocks.

**Stored procedure**

Stored procedures provide a powerful way to code application logic that can be stored on the server. Informix Dynamic Server and Oracle both use stored procedures. Oracle also uses an additional type of subprogram called a function.  
The language used to code stored procedures is a database-specific procedural extension of SQL. In Oracle it is PL/SQL and in Informix Dynamic Server it is Informix Dynamic Server Stored Procedure Language (SPL). These languages differ considerably. However, most of the individual SQL statements and the procedural constructs, such as if-then-else, are similar in both languages.

A stored procedure is a user defined piece of code written in the local version of PL/SQL, which may return a value (making it a function) that is invoked by calling it explicitly.

A trigger is a stored procedure that runs automatically when various events happen (eg update, insert, delete).

Indexing is a data structure technique used to find data more quickly and efficiently in a table.

An index is a small table having only two columns. The first column contains a copy of the primary or candidate key of a table and the second column contains a set of pointers holding the address of the disk block where that particular key value can be found. The users cannot see the indexes, they are just used to speed up searches/queries.

Indexes are of 3 types:

1. Primary indexes : In primary index, there is a one-to-one relationship between the entries in the index table and the records in the main table. Primary index is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation.

2. Secondary indexes : Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values .While creating the index, generally the index table is kept in the primary memory (RAM) and the main table, because of its size is kept in the secondary memory (Hard Disk).

3. Clustering indexes : Clustering index is defined on an ordered data file. The data file is ordered on a non-key field e.g. if we are asked to create an index on a non-unique key, such as Dept-id. There could be several employees in each department. Here we use a clustering index, where all employees belonging to the same Dept-id are considered to be within a single cluster, and the index pointers point to the cluster as a whole.

**Left Outer Join and Right Outer Join**

In left outer join, all the tuples of left relation are covered, filling NULL entries in the fields of right relation if there is no match.

In left outer join, all the tuples of right relation are covered, filling NULL entries in the fields of left relation if there is no match.

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2 ON table1.column\_name = table2.column\_name;

**Explain Weak Entity Types.**

**http://practice.geeksforgeeks.org/problems/explain-weak-entity-types**

The entity sets which do not have sufficient attributes to form a primary key are known as weak entity sets and the entity sets which have a primary key are known as strong entity sets.

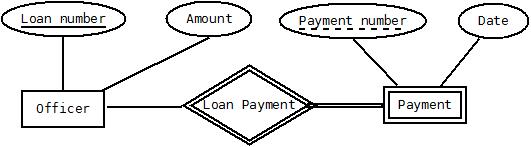
As the weak entities do not have any primary key, they cannot be identified on their own, so they depend on some other entity (known as owner entity). The weak entities have total participation constraint (existence dependency) in its identifying relationship with owner identity.

For Total Participation Constraint, you may refer http://practice.geeksforgeeks.org/problemSub.php?pid=1400407

Weak entity types have partial keys. Partial Keys are set of attributes with the help of which the tuples of the weak entities can be distinguished and identified.

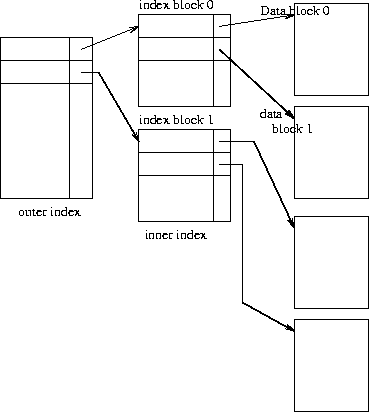
Weak entities are represented with double rectangular box in the ER Diagram and the identifying relationships are represented with double diamond. Partial Key attributes are represented with dotted lines.

In the below ER Diagram , ‘Payment’ is the weak entity. ‘Loan Payment’ is the identifying relationship and ‘Payment Number’ is the partial key. Primary Key of the Loan along with the partial key would be used to identify the records.



The entity sets that do not have primary key of their own. They depend on primary key of another entity set called the strong entity set.

Indexing is done in order to do searching faster and index table must reside in the main memory. In some of the case index table size becomes large and it is difficult to put index table in main memory. If single-level index is used, then a large size index cannot be kept in memory which leads to multiple disk accesses.Multi-level Index breaks down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory.



**What is de-normalization and why it is required?**

Denormalization:-

It is a process of adding redundant data to the database in order to improve the performance and optimizing the database.

The number of redundancies reduces the number of joins required in a query resulting in improving performance.

The goal of using denormalization is to decrease run time of select queries by making data more accessible.

Methods of Denormalization:-

1) Entity Inheritance:- When name of one entity is same as another

2) Role expansion is used when it is known that two entities are related.

3) Look up Entity:- When entity depends on look up table

Requirement of Denormalization:-

1)  Maintaining history

2) Improving query performance.

3) Speeding up the process.

4) When certain columns are queried large number of time.

5) When some primary key exists and  when related with foreign key,consumes more disk space.

6) When many calculations are done in query.

 7) When some queries exists that depends on more than one tables.

Example before denormalization:-

**What are joins and its types?**

**JOIN:**- It is used to combine two or more tables based on column having similar values. Join indicates joining two or more tables. Data is fetched from two or more tables based on some conditions. Join condition tells about relationship between two tables.

Types of Joins:-

1**) Self Join:-** It is used to join table to itself.It is used to retrieve the records having some relation or similarity with other records in the same table.In this we need to create aliases to satisfy the condition.It is used especially when the table has a foreign key references which references its own primary key.

Table:-



Query:- SELECT a.emp\_id AS "Emp\_ID",a.emp\_name AS "Employee Name",  b.emp\_id AS "Supervisor ID",b.emp\_name AS "Supervisor Name"  FROM employee a, employee b WHERE a.emp\_supv = b.emp\_id;

2.**Inner Join:-**

It selects the data from both the tables as long as there is match between the tables.

select e1.Username,e1.FirstName,e1.LastName,e2.DepartmentName from Employee e1 inner join Departments e2 on e1.DepartID=e2.id;

3**.EQUIJOIN:-**

 An equal sign (=) is used as comparison operator in the where clause to refer equality.

You may also perform EQUI JOIN by using JOIN keyword followed by ON keyword and then specifying names of the columns along with their associated tables to check equality.

SELECT \* FROM Employee e1 JOIN Departments e2 ON e1.DepartID = e2.id;

4.**Outer join**:-

a) Left outer join:-

It displays all the data of first table and matched data of second table.

Syntax:

 SELECT \* FROM Employee e1 LEFT OUTER JOIN Departments e2  
ON e1.DepartID = e2.id ;

b) Right outer Join:-

It displays all data of second table and matched data of first table

SELECT \* FROM Employee e1 RIGHT OUTER JOIN Departments e2  
ON e1.DepartID = e2.id ;

c) Full outer join:-

Full outer join returns all the rows from both tables whether it has been matched or not.

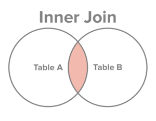
SELECT \* FROM Employee e1 FULL OUTER JOIN Departments e2  
ON e1.DepartID = e2.id

**5.Cross Join:**-

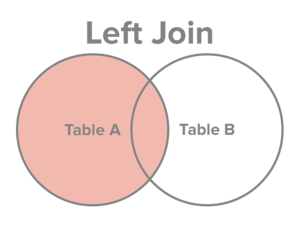
It gives the cross product between two tables. No of rows is equal to product of number of rows of first table (m) and second table(n) then output is m\*n

SELECT \* FROM Employee cross join Departments e2;

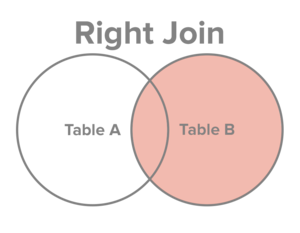
 As employee table has 8 records and department table has 6 records so 8\*6=48 records will be there in cross join



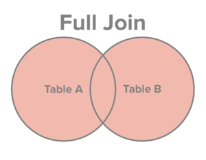
Select all records from Table A and Table B, where the join condition is met.



Select all records from Table A, along with records from Table B for which the join condition is met (if at all).



Select all records from Table B, along with records from Table A for which the join condition is met (if at all).



Select all records from Table A and Table B, regardless of whether the join condition is met or not.

for more information refer: http://www.dofactory.com/sql/join

**What are Triggers?**

Triggers are stored program or procedure which automatically get executed when some event occurs.They can be executed in response to:-DDL:(update,insert, delete), DML:-(create,alter,drop) and  Database Operation(logon,start up, shut down)

Benefits of using Triggers:-

Prevent invalid transactions

Provides security and authentication

Syntax:CREATE [OR REPLACE ] TRIGGER trigger\_name {INSERT [OR] | UPDATE [OR] | DELETE} [OF col\_name] ON table\_name [REFERENCING OLD AS o NEW AS n] [FOR EACH ROW] WHEN (condition) DECLARE Declaration-statements BEGIN Executable-statements EXCEPTION Exception-handling-statements END;

Each trigger is enclosed in Begin and end tags

A **database trigger** is [procedural code](https://en.wikipedia.org/wiki/Procedural_code) that is automatically executed in response to certain [events](https://en.wikipedia.org/wiki/Event_%28computing%29) on a particular [table](https://en.wikipedia.org/wiki/Table_%28database%29) or [view](https://en.wikipedia.org/wiki/View_%28database%29) in a [database](https://en.wikipedia.org/wiki/Database). The trigger is mostly used for maintaining the [integrity](https://en.wikipedia.org/wiki/Database_integrity) of the information on the database. For example, when a new record (representing a new worker) is added to the employees table, new records should also be created in the tables of the taxes, vacations and salaries.

The four main types of triggers are:

Row Level Trigger: This gets executed before or after any column value of a row changes

Column Level Trigger: This gets executed before or after the specified column changes

For Each Row Type: This trigger gets executed once for each row of the result set affected by an insert/update/delete

For Each Statement Type: This trigger gets executed only once for the entire result set, but also fires each time the statement is executed.

**How to handle NULL values in aggregate functions?**

Aggregate Functions are:- SUM,COUNT,MAX,MIN,SUM,AVG.

All the aggregate functions except COUNT and grouping ignore NULL values.So solution to this is that we can use NVL function and substitute in the aggregate function to get the desired result.For all the remaining aggregate functions, if the data set contains no rows, or contains only rows with nulls as arguments to the aggregate function, then the function returns null.

Example: NVL(col1,col2.....)

All aggregate functions except COUNT(\*) and GROUPING ignore nulls. You can use the NVL function in the argument to an aggregate function to substitute a value for a null. COUNT never returns null, but returns either a number or zero. For all the remaining aggregate functions, if the data set contains no rows, or contains only rows with nulls as arguments to the aggregate function, then the function returns null.

This means that, given a table with values like this

GROUP\_KEY VAL

---------- ----------

Group-1 (null)

Group-1 (null)

Group-2 a

Group-2 a

Group-2 z

Group-2 z

Group-2 (null)

Group-3 A

Group-3 A

Group-3 Z

aggregate functions like [MAX](http://www.oracle.com/pls/db102/to_toc?pathname=server.102/b14200/functions085.htm) , [MIN](http://www.oracle.com/pls/db102/to_toc?pathname=server.102/b14200/functions087.htm) , and [COUNT](http://www.oracle.com/pls/db102/to_toc?pathname=server.102/b14200/functions032.htm) will return values that for the most part ignore nulls, like these.

select

group\_key ,

MAX( VAL ) max\_val ,

MIN( VAL ) min\_val ,

COUNT( \* ) count\_all\_rows ,

COUNT( VAL ) count\_val ,

COUNT( DISTINCT VAL ) count\_distinct\_val

from t1 group by group\_key order by group\_key ;

GROUP\_KEY MAX\_VAL MIN\_VAL COUNT\_ALL\_ROWS COUNT\_VAL COUNT\_DISTINCT\_VAL

---------- ---------- ---------- -------------- ---------- ------------------

Group-1 (null) (null) 2 0 0

Group-2 z a 5 4 2

Group-3 Z A 3 3 2

**Write a query to find the name and department of the employee with highest salary**

deptno: department number, ename: employee name, sal:- salary, loc : location

Select d.deptno, e.ename, d.dname, e.sal, d.loc  
from   
(  
  select e.ename,    e.sal,     e.deptno,  
    dense\_rank() over (partition by deptno order by sal desc) as salRank  
  from employees e  
) e  inner join departments d  
  on e.deptno = d.deptno  
where salRank = 1;

SELECT name ,department from employee WHERE salary =(SELECT max(salary) from employee);

select name,department from employee where salary in (select max(salary) from employee)

**How would you Select LAST n records from a table?**

SELECT TOP n \* FROM  table\_name order by id DESC;

SELECT \* FROM table\_name order by id DESC LIMIT N;

select \* from emp minus select \* from emp where rownum <= (select count(\*) - &n from emp);

select \* from emp minus select \* from emp where rownum <= (select count(\*) - &n from emp);

let us suppose the records are sorted in ascending order by supplier name then

select \* from (select \* from suppliers ORDER BY supplier\_name DESC) suppliers2 where rownum<=n ORDERBY rownum DESC

**How would you Select FIRST n records from a table?**

select top n \* from employee

select \* from emp where rownum <= &n;

select \* from table order by table\_id limit n;

**How to get nth max salaries?**

select min(t.salary) from (select \* from Emp order by salary desc limit n) as t;

SELECT MIN(SAL) FROM (SELECT DISTINCT(SAL) FROM EMP ORDER BY SAL)

WHERE ROWNUM<=N;

select \* from employee emp1 where (N-1)=(select  count(Distinct (emp2.salary)) from employee emp2 where emp2.salary>emp1.salary

select distinct hiredate from emp a where &n =  (select count(distinct sal) from emp b where a.sal >= b.sal);

select max(t.salary) from (select \* from Emp order by salary desc limit n) as t;

reference:hac\_123

**Difference between varchar and varchar2 data types?**

**Varchar**

 1.varchar can store upto 2000 bytes.

  2. varchar occupies space for the NULL values.

  3.varchar is an ANSI standard

**Varchar2 :**

 1.varchar2 can store upto 4000 bytes.

2. varchar2 is not occupying space for the NULL values.

3.varchar2 is an oracle standard.

**Varchar**can store upto 2000 bytes and **varchar2**can store upto 4000 bytes. Varchar will occupy space for NULL values and Varchar2 will not occupy any space. Both are differed with respect to space.

**Difference between delete and truncate in SQL?**

Truncate:-

1) TRUNCATE is a DDL command

2) TRUNCATE is executed using a table lock and whole table is locked for remove all records.

3) We cannot use Where clause with TRUNCATE.

4) TRUNCATE removes all rows from a table.

5) Minimal logging in transaction log, so it is performance wise faster.

6) TRUNCATE TABLE removes the data by de-allocating the data pages used to store the table data and records only the page de-allocations in the transaction log.

7) Identify column is reset to its seed value if table contains any identity column.

8) To use Truncate on a table you need at least ALTER permission on the table.

9) Truncate uses the less transaction space than Delete statement.

10)  Truncate cannot be used with indexed views

DELETE:-

1) DELETE is a DML command

2) DELETE is executed using a row lock, each row in the table is locked for deletion.

3) We can use where clause with DELETE to filter & delete specific records.

4) The DELETE command is used to remove rows from a table based on WHERE condition.

5) It maintain the log, so it slower than TRUNCATE.

6) The DELETE statement removes rows one at a time and records an entry in the transaction log for each deleted row

7) Identity of column keep DELETE retain the identity

8) To use Delete you need DELETE permission on the table.

9) Delete uses the more transaction space than Truncate statement.

10) Delete can be used with indexed views

**Difference between function and stored procedure.**

Functions must return some value whereas for stored procedure it is optional

In functions, only SELECT statement is allowed but in case of stored procedure SELECT, INSERT, UPDATE, DELETE are allowed. It means in function you can not change existing data. But data alteration is allowed for stored procedures

You can call function inside stored procedure but vice versa is not true

To call the function you can use SELECT statement. e.g. SELECT <<function\_name>> <<parameters>> but same is not allowed for stored procedure. To call / execute procedure you have to use syntax as exec <<procedure\_name>>  <<parameters>>

You can grant users permission to execute a Stored Procedure independently of underlying table permissions.

Functions can be used with JOIN statements also but procedures can not be used for JOIN purpose

**What are the advantages and disadvantages of views in a database?**

**Advantages**:

1. Views don't store data in a physical location.

2. The view can be used to hide some of the columns

from the table.

3. Views can provide Access Restriction,

since data insertion, update and deletion

is not possible with the view.

**Disadvantages**:

1. When a table is dropped, associated view become irrelevant.

2. Since the view is created when a query requesting data from view is triggered, its a bit slow.

3. When views are created for large tables, it occupies more memory.

As a developer, we know any SQL query can be written in multiple ways but we should follow best practices/ techniques to achieve better query performance. *I’m highlighting some of them below*:

1. Use EXISTS instead of IN to check existence of data.  
2. Avoid \* in SELECT statement. Give the name of columns which you require.  
3. Choose appropriate Data Type. E.g. To store strings use varchar in place of text data type. Use text data type, whenever you need to store large data (more than 8000 characters).  
4. Avoid nchar and nvarchar if possible since both the data types takes just double memory as char and varchar.  
5. Avoid NULL in fixed-length field. In case of requirement of NULL, use variable-length (varchar) field that takes less space for NULL.  
6. Avoid Having Clause. Having clause is required if you further wish to filter the result of an aggregations.  
7. Create Clustered and Non-Clustered Indexes.  
8. Keep clustered index small since the fields used in clustered index may also used in non-clustered index.  
9. Most selective columns should be placed leftmost in the key of a non-clustered index.  
10. Drop unused Indexes.  
11. Better to create indexes on columns that have integer values instead of characters. Integer values use less overhead than character values.  
12. Use joins instead of sub-queries.  
13. Use WHERE expressions to limit the size of result tables that are created with joins.  
14. Use TABLOCKX while inserting into a table and TABLOCK while merging.  
15. Use WITH (NOLOCK) while querying the data from any table.  
16. Use SET NOCOUNT ON and use TRY- CATCH to avoid deadlock condition.  
17. Avoid Cursors since cursor are very slow in performance.  
18. Use Table variable in place of Temp table. Use of Temp tables required interaction with TempDb database which is a time taking task.  
19. Use UNION ALL in place of UNION if possible.  
20. Use Schema name before SQL objects name.  
21. Use Stored Procedure for frequently used data and more complex queries.  
22. Keep transaction as small as possible since transaction lock the processing tables data and may results into deadlocks.  
23. Avoid prefix “sp\_” with user defined stored procedure name because SQL server first search the user defined procedure in the master database and after that in the current session database.  
24. Avoid use of Non-correlated Scalar Sub Query. Use this query as a separate query instead of part of the main query and store the output in a variable, which can be referred to in the main query or later part of the batch.  
25. Avoid Multi-statement Table Valued Functions (TVFs). Multi-statement TVFs are more costly than inline TVFs.