What is Data -> Raw Facts (Not Organized)

What is Information -> Organized Format (CSV, TSV, Tables…)

Database: Collection of interrelated information

Advantages:

1. Permanent storage of information
2. Security
3. Multiuser support

Available Databases: SQL Server, Oracle, Teradata, MySQL, MS-Access, DB2, Sybase, Fox Pro, Ingres sql, PostgreSQL, etc.,

Database Objects: Tables, Stored Procedures, Cursors, Views, Triggers, Functions, Indexes, Synonyms, User Defined Types, Cubes, etc.,

Among the above database objects, Table has structure and Data. Remaining are used to implement functionality and business logic.

Database Testing is the act of validating the structure/schema and data.

Authentication: Verifying the user credentials

Authorization: Permissions of the user

SQL: Structured Query Language. SQL is the database language used to perform certain operations on the database.

Query: Request to the database to perform some manipulation on DB

SQL Uses certain commands to carry out the required tasks.

These SQL commands are mainly categorized into following categories.

DDL: Data Definition Language

DML: Data Manipulation Language

DRL: Data Retrieval Language

DCL: Data Control Language

TCL: Transaction Control Language

DDL: Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

List of DDL Commands:

CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).

DROP: This command is used to delete objects from the database.

ALTER: This is used to alter the structure of the database.

TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed.

Rename: This is used to rename an object existing in the database.

DML: The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

List of DML commands:

INSERT: It is used to insert data into a table.

UPDATE: It is used to update existing data within a table.

DELETE: It is used to delete records from a database table.

DRL: DRL is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.

List of DML commands:

SELECT: It is used to retrieve data from the database.

DCL: DCL commands mainly deal with rights, permissions, and other controls of the database system.

List of DCL Commands:

GRANT: This command gives users access privileges to the database.

REVOKE: This command withdraws the user's access privileges given by using the GRANT command.

DENY: The DENY statement prevents users from performing actions.

TCL: TCL commands deal with the transaction within the database.

List of TCL commands:

COMMIT: Commits a Transaction.

ROLLBACK: Rollbacks a transaction in case of any error occurs.

SAVE: Sets a save point within a transaction.

Data Types:

data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

Data Type specifies

1. Type of data
2. Size of the memory it occupies.

Numeric Data Types:

Data Type	Description	Storage
Bit	Integer value 0, 1 or NULL	
Tiny Int	Integer values from 0 to 255	1 byte
Small Int	Values from -32,768 and 32,767	2 bytes
Int	Values from -2,147,483,648 and 2,147,483,647	4 bytes
Big Int	Values from -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
Decimal(p,s)	Fixed Precision and Scale Numbers Allows numbers from -10^38 +1 to 10^38 –1 The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes
numeric(p,s)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 –1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes
Smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
Money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
Float(n)	Floating precision number data from -1.79E + 308 to 1.79E + 308. The n parameter indicates whether the field should hold 4 or 8 bytes. Float (24) holds a 4-byte field and float (53) holds an 8-byte field. Default value of n is 53.	4 or 8 bytes
Real	Floating precision number data from -3.40E + 38 to 3.40E + 38	4 bytes

String Data Types:

Data Type	Description	Max Size	Storage
Char(n)	Fixed width character string	8000 characters	Defined size
varchar(n)	Variable width character string	8000 Characters	2 bytes + number of characters
Varchar(max)	Variable width character string	1,073,741,824 characters	2 bytes + number of characters
Text	Variable width character string	2GB of text data	4 bytes + number of characters
Image	Variable width binary string	2 GB	

Date and Time Data Types:

Data Type	Description	Storage
Datetime	Allows from January 1, 1753 to December 31, 9999	8 bytes
Timestamp	timestamp value is based upon an internal clock and does not correspond to real time	8 bytes
Date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
Time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes

DDL Commands:

Create: This is the command useful to create any database object like Table, Database, Trigger, etc.

Create:

Database Creation syntax: create database <database name>

Here <database name> indicates the user defined database name

Ex: create database Persons

Table creation syntax: create table <table name> (col1 datatype, col2, datatype ,…,coln datatype)

Here <database name> indicates the user defined table name

Col specifies the name of the column and data type specifies the type of data that the column can hold

Ex: create table employee (eid int, ename varchar (20), salary money, age int, gender char (1), did int)

Alter: This is the command useful to modify the database object structure like adding a new column, deleting existing column, changing the datatype of the column.

Adding new column to existing table syntax: alter table <table name> add <col_name> <data type>

Ex: alter table emp add locid int

Removing existing column from table syntax: alter table <table_name> drop <col name>

Ex: alter table emp drop locid

Changing the datatype of existing column syntax: alter table <table name> alter <col name><datatype>

Ex: alter table emp alter salary int

Renaming table name syntax: sp_rename 'emp','employee'

Drop: This statement is used to drop an existing table in a database. This will remove table data as well as table definition

Syntax: Drop table <table name>

Ex: drop table emp

Note: Be careful before dropping a table. Deleting a table will result in loss of complete information stored in the table

DML Commands:

Insert: Insert command is used to insert records into a table. THis can be done in two ways.

- Insert data to all the columns of the table

  Syntax: insert into table_name values(col1,col2,...,coln)

  Ex: insert into emp values(101,'Rama',10)

- Insert data to specific columns of the table

  Syntax: insert into table_name(col1,col2,..,coln) values(col1 data, col2 data,...,coln data)

  Ex: insert into emp(eid, ename) values(102,'Krishna')

  Note: The columns for which data is not available will be inserted with 'NULL'

Update: Update command is used to modify the existing records in a table

Syntax: update table_name set col1 = value1,col2 = value2 where [condition]

Ex: update emp set ename = 'Prasad' where eid = 102

Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If we omit the WHERE clause, all records in the table will be updated

Delete: Delete command is used to delete existing records from the table

Syntax: Delete from table_name where [condition]

Ex: Delete from emp where eid = 102

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If we omit the WHERE clause, all records in the table will be deleted

Truncate: Truncate command is used to data inside the table but not the table definition

Syntax: Truncate table table_name

Ex: Truncate table emp

DRL Commands:

Select: Select command is used to retrieve information from the database objects.

Using select it is possible to retrieve all the columns data or the specific column(s) data

To retrieve data from all the columns

Syntax: select * from table_name

Ex: select * from emp

To retrieve data from specific column(s)

Syntax: select col1, col2,...,coln from table_name

Ex: select eid, ename from emp

Group By: The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns

Syntax: SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s)

Ex: Select count(*) from emp group by did

Having: Having is used to apply condition on the result set of group by query.

Syntax: select * from table_name group by col_name having [condition]

Ex: select count(*) from emp group by did having count(*) > 1

Distinct: Distinct is used to retrieve unique values from the table

Syntax: select distinct col_list from table_name

Ex: select distinct did from emp

Operators: Different operators can be used in sql while retrieving information

Top: Top is used to retrieve top records from the table

Syntax: select top number col_list from table_name

Ex: select top 2 * from emp

Arithmetic Operators: +, -, *, /, %

Ex: select salary+1000 from emp

Comparison Operators: <, <=, >, >=, =, <>, Is

select * from emp where salary <= 5000

Logical Operators: AND, OR, NOT

select * from emp where did = 10 and salary <=5000

List Comparison Operators: IN, ANY, ALL

select * from emp where did in(10,20,30)

Between Operator: To check the range

select * from emp where did between 10 and 30

Like: To retrieve information using pattern

LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are wildcards often used in conjunction with the LIKE operator. They are:

Wildcard Character	Description

%	Represents zero or more characters
_	Represents a single character
[]	Represents any single character within the brackets
^	Represents any character not in the brackets
-	Represents any single character within the specified range

Syntax: Select col_list from table_name where col_name like pattern

Ex: select * from emp where ename like 'a%'

Result: arun, amar, anirudh, akanksha

select * from emp where ename like 'r[ao]m'

Result: ram, rom

Set: UNION, UNIONALL, INTERSECT, EXCEPT

UNION: The UNION operator is used to combine the result-set of two or more SELECT statements.

The UNION operator selects only distinct values by default

- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order

Syntax: SELECT column_name(s) FROM table1

UNION

SELECT column_name(s) FROM table2;

Ex: select eid, ename from emp

UNION

select did, dname from dept

UNION ALL: The UNION ALL operator is used to combine the result-set of two or more SELECT statements same as UNION but includes duplicates.

Syntax: SELECT column_name(s) FROM table1

UNION ALL

SELECT column_name(s) FROM table2;

Ex: select eid, ename from emp

UNION ALL

select did, dname from dept

INTERSECT: The SQL INTERSECT clause/operator is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement. This means INTERSECT returns only common rows returned by the two SELECT statements.

Just as with the UNION operator, the same rules apply when using the INTERSECT operator. MySQL does not support the INTERSECT operator.

Syntax: SELECT column_name(s) FROM table1

INTERSECT

SELECT column_name(s) FROM table2;

Ex: select eid, ename from emp

INTERSECT

select did, dname from dept

EXCEPT: The SQL EXCEPT clause/operator is used to combine two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement. This means EXCEPT returns only rows, which are not available in the second SELECT statement.

Just as with the UNION operator, the same rules apply when using the EXCEPT operator. MySQL does not support the EXCEPT operator.

Syntax: SELECT column_name(s) FROM table1

EXCEPT

SELECT column_name(s) FROM table2;

Ex: select eid, ename from emp

EXCEPT

select did, dname from dept

JOINS:

Join is used to retrieve information from two or more tables based on a common column between them.

Different Types of Joins:

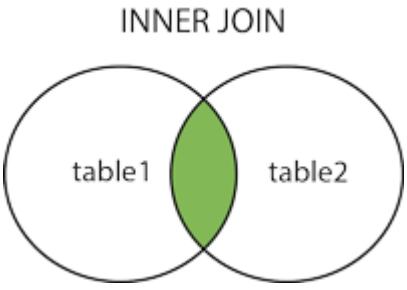
INNER JOIN: Returns records that have matching values in both tables

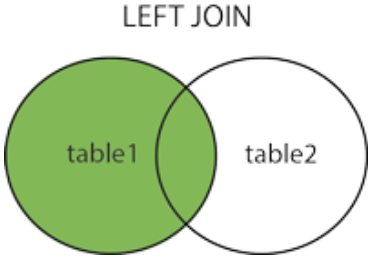
Syntax: select col_list from table1 inner join table2 on table1.colname = table2.colname where [condition]

Ex: select * from emp inner join dept on emp.did = dept.did

LEFT OUTER JOIN: Returns all records from the left table, and the matched records from the right table

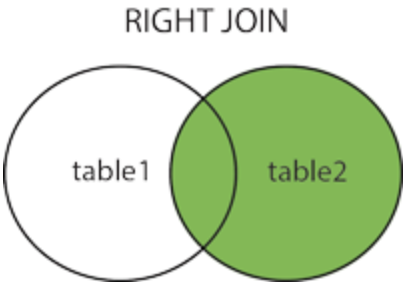
Syntax: select col_list from table1 left join table2 on table1.colname = table2.colname where [condition]





Ex: select * from emp left join dept on emp.did = dept.did

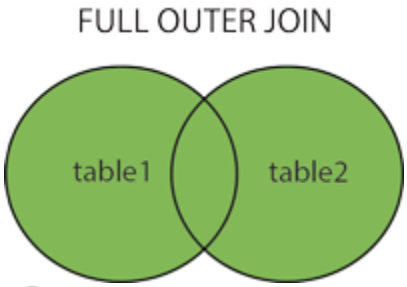
RIGHT OUTER JOIN: Returns all records from the right table, and the matched records from the left table



Syntax: select col_list from table1 right join table2 on table1.colname = table2.colname where [condition]

Ex: select * from emp right join dept on emp.did = dept.did

FULL OUTER JOIN: Returns all records when there is a match in either left or right table



Syntax: select col_list from table1 full join table2 on table1.colname = table2.colname where [condition]

Ex: select * from emp full join dept on emp.did = dept.did

CROSS JOIN: Returns all records with association of each record from left table to right table

Syntax: select col_list from table1 cross join table2

Ex: select * from emp inner join dept on emp.did = dept.did

SELF JOIN: Returns records by joining the table with itself. Alias is used for self-join

Syntax: select col_list from table as X, table as Y where [condition]

Ex: select * from emp x, emp y where x.eid = y.mgrid

Sub Queries: A Subquery or Inner query or a Nested query is a query within another SQL query. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

Ex: select * from emp where did = (select did from dept where dname = 'HR')

Backup Tables:

There are two scenarios to take backup of the table data.

1. Backup table definition is not available, create a table and insert the data
2. Backup table definition is available, insert data into existing backup table

Select * Into: This is used to create the table and insert the data

Syntax: select * into new_table from old_table where [condition]

Ex: select * into emp_temp from emp

Select * into emp_temp from emp where did = 20

Insert into select: This is used to insert data into existing table

Syntax: Insert into table1 select * from table2 where [condition]

Ex: insert into emp_temp select * from emp

insert into emp_temp select * from emp where did = 20

Constraints: Constraints are used to specify rules for data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Below is the list of constraints available in SQL:

1. Primary Key: It will not allow duplicate values, nulls. Only one primary key per table is allowed.

   Syntax: <col name><data type> primary key

2. Unique: It will not allow duplicate values. One Null value

   Syntax: <col name><data type> unique

3. Not Null: It will not allow null values

   Syntax: <col name><data type> not null

4. Check: To apply condition we need to use check

   Syntax: <col name><data type> check(condition)

5. Default: To insert default value into the column

   Syntax: <col name><data type> default <value>

6. Foreign Key:(Referential Integrity) Foreign Key column of primary table should be the primary key column of the referencing table

   Syntax: <col name><data type> foreign key references <table name>(<col name>)

7. Composite Primary Key: Primary key will be applied on the combination of columns.

   Syntax: Primary Key (col list)

   Ex: create table dept (did int primary key, dname varchar(10))

   create table emp

   (eid int primary key,

   ename varchar(20) not null,

   age int check (age>18),

   gender varchar(10) default 'NA',

   email varchar(50) unique,

   did int foreign key references dept(did)

)

Views: view is a virtual table based on the result-set of an SQL statement. View contains rows and columns, similar to a table. A view can be created from one or more tables in a database.

View returns the updated data every time as it recreates the view every execution.

All actions can be performed on views similar to tables.

Creation of a view:

Syntax: create view view_name as select col_list from table_name where [condition]

Ex: create view emp_view as select * from emp where did = 10

Retrieving information from a view:

Syntax: select col_list from view_name

Ex: select * from emp_view

Deleting a view:

Syntax: Drop view view_name

Ex: drop view emp_view

Note: Insert, Update and Delete manipulations can be done on views. The manipulation will be impacted on the original table.

Indexes: Indexes are used to speed up the retrieval of data from the database. Best practice is creating indexes on the columns that will be frequently used from searching. Why because, updating the table with indexes will take more time than updating tables without indexes as the indexes also need to be updated.

Types of Indexes:

1. Unique Index: It will not allow duplicate values

   Syntax: create unique index index_name on table_name(col_name)

   Ex: create unique index i1 on emp(eid)

2. Composite Index: It will create index on two or more columns

   Syntax: create index index_name on table_name (col1, col2..coln)

   Ex: create index i2 on emp(eid, did)

3. Implicit Index: Default index will be created on table columns of primary key and unique
4. Index: It will allow duplicates.

   Syntax: create index index_name on table_name(col_name)

   Ex: create index i3 on emp(eid)

Normalization: Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Types of Normalization Forms:

- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce-Codd Normal Form)
- 4NF (Fourth Normal Form)
- 5NF (Fifth Normal Form)
- 6NF (Sixth Normal Form)

First Normal Form:

- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary key.

Second Normal Form:

- Create separate tables for sets of values that apply to multiple records.
- Relate these tables with a foreign key.

Third Normal Form:

- Eliminate fields that do not depend on the key.

Denormalization: Denormalization is a database optimization technique in which we add redundant data to one or more tables.

Pros of Denormalization:

Retrieving data is faster since we do fewer joins

Queries to retrieve can be simpler (and therefore less likely to have bugs), since we need to look at fewer tables.

Cons of Denormalization:

Updates and inserts are more expensive.

Denormalization can make updating and insert code harder to write.

Data may be inconsistent. Which is the "correct" value for a piece of data?

Data redundancy necessitates more storage.

SQL Interview Questions:

1. Explain authentication modes in Sql Server

   Ans: Windows and Mixed Mode

2. Explain about Joins

3. Is it possible to connect to other server from sql server?

   Ans: SQL server can be connected to any database which has an OLE-DB provider to give a link. Example: Oracle has an OLE-DB provider which has a link to connect with the SQL server group.

4. What is a subquery?

   Ans: A subquery is a query which can be nested inside a main query like Select, Update, Insert or Delete statements. This can be used when expression is allowed.

   Properties of sub query can be defined as

   - A sub query should be placed in the right hand side of the comparison operator of the main query
   - A subquery should be enclosed in parenthesis because it needs to be executed first before the main query
   - More than one subquery can be included

5. What is a Trigger?

   Ans: Triggers are used to execute a batch of SQL code when insert or update or delete commands are executed against a table. Triggers are automatically triggered or executed when the data is modified

6. Types of Triggers

   Ans:

   - Insert
   - Delete
   - Update
   - Instead of

7. What is Identity?

   Ans: IDENTITY column is used in table columns to make that column as Auto increment number

8. What is the difference between UNION and UNION ALL?

9. How can we get count of the number of records in a table

   Ans: select count (*) from table_name

10. What is the difference between COMMIT and ROLLBACK?

    Ans: Every statement between BEGIN and COMMIT becomes persistent to the database when the COMMIT is executed. Every statement between BEGIN and ROLLBACK are reverted to the state when the ROLLBACK was executed

11. How can data be copied from one table to another table?

    Ans: INSERT INTO SELECT: This command is used to insert data into a table which is already created.

    Ex: Insert into <new table name> select * from <old table name>

    SELECT INTO: This command is used to create a new table and its structure and data can be copied from existing table

    Ex: select * into <new table name> from <old table name>

12. How to delete duplicate rows in SQL Server?

    Ans: WITH CTE AS(

    SELECT eid, ename, salary, DOJ, Deptid,

    RN = ROW_NUMBER()OVER(PARTITION BY eid ORDER BY eid)

    FROM emp

)

    Delete FROM CTE WHERE RN > 1

    (OR)

    select distinct * into emp2 from emp

    truncate table emp

    insert into emp select * from emp2

13. What is the query to find 4th highest salary?

    Ans:

    - When salary values are unique: select top 1 * from (select top 4 * from emp order by salary desc) temp order by salary asc
    - When salary values are having duplicates: select * from emp where salary = (select top 1 * from (select top 3 * from (select distinct(salary) from emp) temp1 order by salary desc) temp2 order by salary asc)

14. What Is query?

    Ans: Request to the database to perform some manipulation on DB

15. Explain Constraints

    Ans: SQL constraints are a set of rules or conditions implemented on an RDBMS to specify what data can be inserted, updated, or deleted in its tables. This is done to maintain data integrity and ensure that the information stored in database tables is accurate.

16. What is the purpose of Indexes?

    Ans: An SQL index stores important parts of a database table to allow for a quick and efficient lookup. Rather than searching the entire database, users only have to consult the index during data retrieval. Indexes, therefore, help improve performance in an RDBMS

17. What is alias?

    Ans: Aliases are temporary names given to tables or columns for the duration of a particular SQL query

18. What is the difference between normalization and denormalization?

    Ans: Normalization is the process of dividing data into tables to remove redundant data and improve data integrity.

    Denormalization is used to combine multiple tables in order to reduce the time required to perform queries.

19. How do you select all even or odd numbers in a table?

    Ans:

    For even numbers, use 'MOD (column name, 2) = 1'

    For odd numbers, use 'MOD (column name, 2) = 0'

20. Differences between delete and truncate

Delete	Truncate
The DELETE command is used to delete specific rows(one or more).	While this command is used to delete all the rows from a table.
There may be a WHERE clause in the DELETE command in order to filter the records.	While there will not be a WHERE clause in the TRUNCATE command.
The DELETE statement removes rows one at a time and records an entry in the transaction log for each deleted row.	TRUNCATE TABLE removes the data by deallocating the data pages used to store the table data and records only the page deallocations in the transaction log.
The DELETE command is slower than the TRUNCATE command.	While the TRUNCATE command is faster than the DELETE command.
The delete can be used with indexed views.	Truncate cannot be used with indexed views.

21. Write a query to get the count of employees from each department?

    Ans: select Deptid, count(*) as No_of_Employees from emp group by deptid

22. Write a query to get the duplicate employee records from each department

    Ans: select Deptid, count(*) as No_of_Employees from emp group by deptid having No_of_Employees > 1

23. Write a query to list all the employees whose name starts with 'A'

    Ans: select * from emp where ename like 'A%'

24. write a query to list all the employees whose name starts with 'A' and ends with 'N'

    Ans: select * from emp where ename like 'A%N'

25. Write a query to list all the employees whose name start with 'A' and ends with 'N' and 2 characters in between

    Ans: select * from emp where ename like 'A__N' (Two underscores between A and N)

26. write a query to retrieve 2-4 highest salaries?

    Ans: select * from emp order by salary

    offset 1 rows

    fetch next 4 rows only