# SQL Interview Questions and Answers for Freshers

## 1. Define database.

A database is an organized collection of structured data that can be stored, easily accessed, managed, and retrieved digitally from a remote or local computer system. Databases can be complex and vast and are built with a fixed design and modeling approach. While smaller databases can be stored on a file system, large ones are hosted on computer clusters or cloud storage.

## 2. What is the difference between DBMS and RDBMS?

A database management system or DBMS is system software that can create, retrieve, update, and manage a database. It ensures the consistency of data and sees to it that it is organized and easily accessible by acting as an interface between the database and its end-users or **application software**. DBMS can be classified into four types:

- **Hierarchical Database:** It has a treelike structure with the data being stored in a hierarchical format. A parent in a hierarchical database can have multiple children, but a child can have only one parent.
- **Network Database:** This type of database is presented as a graph that can have many-to-many relationships, allowing entities to have multiple connections.
- **Relational Database:** It is the most widely used and easy-to-use database. It is represented as a table and the values in the columns and rows are related to each other.
- **Object-Oriented Database:** The data values and operations are stored as objects in this type of database, and these objects have multiple relationships among them.

An RDBMS stores data in the form of a collection of tables. The relations are defined between the common fields of these tables. MS SQL Server, MySQL, IBM DB2, Oracle, and Amazon Redshift are all based on RDBMS.

**DBMS Vs. RDBMS**

| Parameters | DBMS | RDBMS |
|---|---|---|
| **Access** | Data elements need to be accessed separately | Multiple data elements can be accessed at the same time |
| **Relationship Between Data** | No relationship between data | Data in tables are related to each other |
| **Normalization** | It is not present | It is present |
| **Distributed Database** | It does not support distributed databases. | It supports distributed database |
| **Data Storage Format** | Data is stored in either a navigational or hierarchical form | Data is stored in a tabular structure with headers being the column names and the rows containing the corresponding values |
| **Amount of Data** | It deals with a small quantity of data | It deals with a larger amount of data |
| **Data Redundancy** | It is prevalent | Keys and indexes do not allow data redundancy |
| **Number of Users** | It supports a single user | It supports multiple users |
| **Data Fetching** | It is slower for large amounts of data | It is speedy due to the relational approach |
| **Data Security** | Low-security levels when it comes to data manipulation | Multiple levels of data security exist |
| **Software and Hardware Requirements** | Low | High |
| **Examples** | XML, Window Registry, etc. | MySQL, SQL Server, Oracle, Microsoft Access, PostgreSQL, etc. |

## 3. What is SQL?

SQL stands for Structured Query Language. It is the standard language for RDBMS and is useful in handling organized data with entities or variables with relations between them. SQL is used for communicating with databases.

According to ANSI, SQL is used for maintaining RDBMS and for performing different operations of data manipulation on different types of data by using the features of SQL. It is a database language that is used for the creation and deletion of databases. It can also be used, among other things, to fetch and modify the rows of a table.

## 4. What is normalization and what are its types?

Normalization is used to reduce data redundancy and dependency by organizing fields and tables in databases. It involves constructing tables and setting up relationships between those tables according to certain rules. The redundancy and inconsistent dependency can be removed using these rules to make normalization more flexible.

The different forms of normalization are as follows:

- **First Normal Form:** If every attribute in a relation is single-valued, then it is in the first normal form. If it contains a composite or multi-valued attribute, it violates the first normal form.
- **Second Normal Form:** A relation is said to be in the second normal form if it has met the conditions for the first normal form and does not have any partial dependency, i.e., it does not have a non-prime attribute that relies on any proper subset of any candidate key of the table. Often, the solution to this problem is to specify a single-column primary key.
- **Third Normal Form:** A relation is in the third normal form when it meets the conditions for the second normal form and there is not any transitive dependency between the non-prime attributes, i.e., all the

non-prime attributes are decided only by the candidate keys of the relation and not by other non-prime attributes.
- **Boyce-Codd Normal Form:** A relation is in the Boyce-Codd normal form or BCNF if it meets the conditions of the third normal form, and for every functional dependency, the left-hand side is a super key. A relation is in BCNF if and only if X is a super key for every non-trivial functional dependency in form X –> Y.

## 5. What is denormalization?

Denormalization is the opposite of normalization; redundant data is added to speed up complex queries that have multiple tables that need to be joined. Optimization of the read performance of a database is attempted by adding or grouping redundant copies of data.

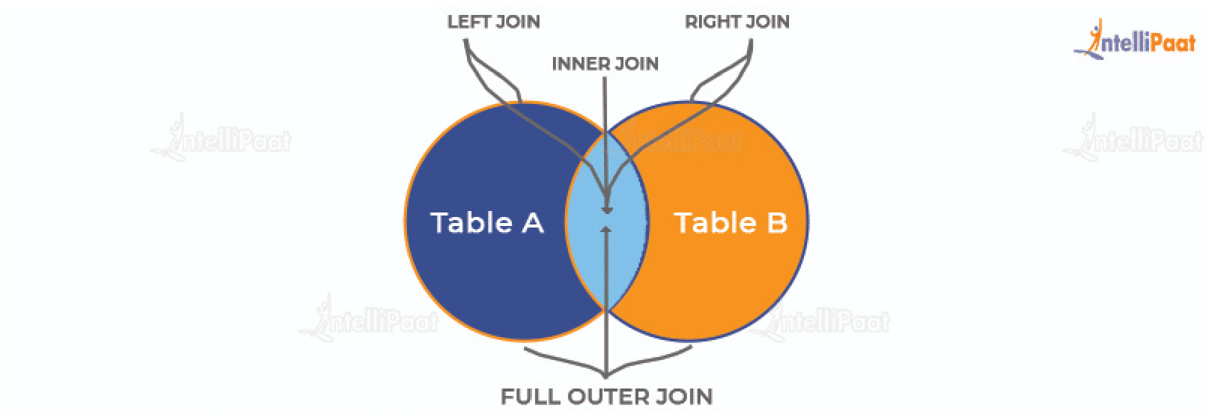
## 6. What are Joins in SQL?

JOINS in SQL is used to combine rows from two or more tables based on a related column between them. Various types of JOINS can be used to retrieve data, depending on the relationship between tables.

There are four types of Joins:

- **Inner Join**
- **Left Join**
- **Right Join**
- **Full Join**

## 7. Explain the types of SQL joins.



There are four different types of SQL Joins:

- 
    - **(Inner) Join:** An inner join is used to retrieve the records that have matching values in tables involved in the join. It combines rows from two tables based on a related column and returns only the matching record. Inner Join is mostly used to join queries.

```
SELECT *
FROM Table_A
JOIN Table_B;
SELECT *
FROM Table_A
INNER JOIN Table_B;
```

- 
    - **Left (Outer) Join:** The use of left join is to retrieve all the records or rows from the left and the matched ones from the right.

```
SELECT *
FROM Table_A A
LEFT JOIN Table_B B
```

```
ON A.col = B.col;
```

-

  - **Right (Outer) Join:** The use of Right join is to retrieve all the records or rows from the right and the matched ones from the left.

```
SELECT *
FROM Table_A A
RIGHT JOIN Table_B B
ON A.col = B.col;
```

- **Full (Outer) Join:** The use of Full join is to retrieve the records that have a match either in the left table or the right table.

```
SELECT *
FROM Table_A A
FULL JOIN Table_B B
ON A.col = B.col;
```

## 8. What are the subsets of SQL?

SQL queries are divided into four main categories:

- **Data Definition Language (DDL)**
  DDL queries are made up of SQL commands that can be used to define the structure of the database and modify it.
  - **CREATE:** Creates databases, tables, schema, etc.
  - **DROP:** Drops tables and other database objects
  - **DROP COLUMN:** Drops a column from any table structure

    - **ALTER:** Alters the definition of database objects
    - **TRUNCATE:** Removes tables, views, procedures, and other database objects
    - **ADD COLUMN:** Adds any column to the table schema
- **Data Manipulation Language (DML)**
  These SQL queries are used to manipulate data in a database.
    - **SELECT INTO:** Selects data from one table and inserts it into another
    - **INSERT:** Inserts data or records into a table
    - **UPDATE:** Updates the value of any record in the database
    - **DELETE:** Deletes records from a table
- **Data Control Language (DCL)**
  These SQL queries manage the access rights and permission control of the database.
    - **GRANT:** Grants access rights to database objects
    - **REVOKE:** Withdraws permission from database objects
- **Transaction Control Language (TCL)**
  TCL is a set of commands that essentially manages the transactions in a database and the changes made by the DML statements. TCL allows statements to be grouped together into logical transactions.
    - **COMMIT:**
    - Commits an irreversible transaction, i.e., the previous image of the database before the transaction cannot be retrieved. When a transaction is committed using the COMMIT statement in SQL, it permanently saves the changes made within the transaction to the database. Once committed, the changes cannot be rolled back or undone, and the previous state of the database before the transaction cannot be retrieved without restoring from a backup or utilizing other data recovery methods.
    - **ROLLBACK:** Reverts the steps in a transaction in case of an error

  - **SAVEPOINT:** Sets a savepoint in the transaction to which rollback can be executed
  - **SET TRANSACTION:** Sets the characteristics of the transaction

## 9. What are the applications of SQL?

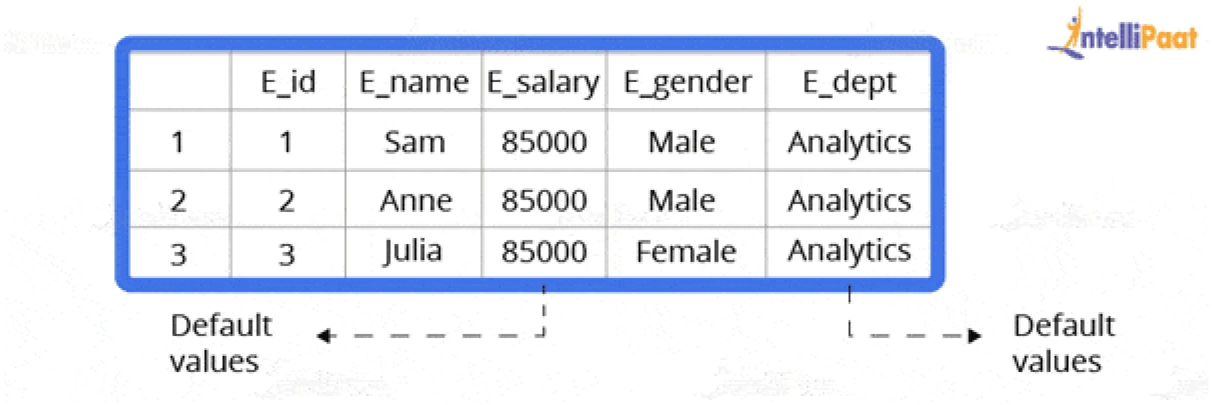
The major applications of SQL are listed below:

- Writing data integration scripts
- Setting and running analytical queries
- Retrieving subsets of information within a database for analytics applications and transaction processing
- Adding, updating, and deleting rows and columns of data in a database

## 10. What is a DEFAULT constraint?

Constraints in SQL are used to specify some sort of rules for processing data and limiting the type of data that can go into a table. Now, let us understand what a default constraint is.

A default constraint is used to define a default value for a column so that it is added to all new records if no other value is specified. For example, if we assign a default constraint for the E_salary column in the following table and set the default value to 85000, all the entries in this column will have the default value of 85000, unless no other value has been assigned during the insertion.



| | E_id | E_name | E_salary | E_gender | E_dept |
|---|---|---|---|---|---|
| 1 | 1 | Sam | 85000 | Male | Analytics |
| 2 | 2 | Anne | 85000 | Male | Analytics |
| 3 | 3 | Julia | 85000 | Female | Analytics |

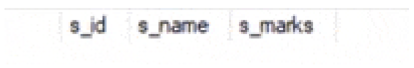
Default values
Default values

Now, let us go through how to set a default constraint. We will start by creating a new table and adding a default constraint to one of its columns.

**Code:**

```
create table stu1(s_id int, s_name varchar(20), s_marks int default
50)
select *stu1
```

**Output:**



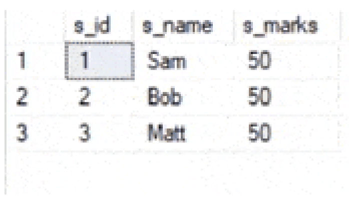
```
s_id  s_name  s_marks
```

Now, we will insert the records.

**Code:**

```
insert into stu1(s_id,s_name) values(1,’Sam’)
insert into stu1(s_id,s_name) values(2,’Bob’)
insert into stu1(s_id,s_name) values(3,’Matt’)
select *from stu1
```

**Output:**

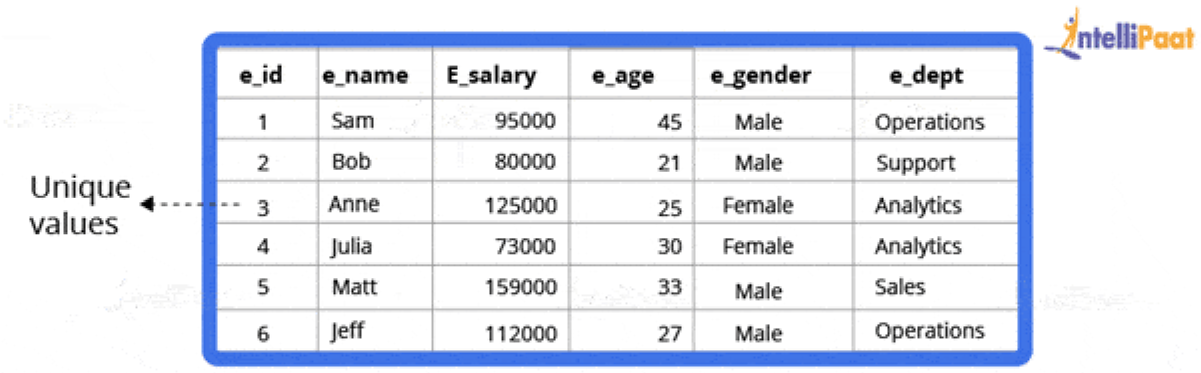


```
   s_id  s_name  s_marks
1  1     Sam     50
2  2     Bob     50
3  3     Matt    50
```

***Also, learn from our blog on [MySQL Interview Questions and Answers] to crack any Interview.***

## 11. What is a UNIQUE constraint?

Unique constraints ensure that all the values in a column are different. For example, if we assign a unique constraint to the e_name column in the following table, then every entry in this column should have a unique value.


Unique values
IntelliPaat

| e_id | e_name | E_salary | e_age | e_gender | e_dept |
|---|---|---|---|---|---|
| 1 | Sam | 95000 | 45 | Male | Operations |
| 2 | Bob | 80000 | 21 | Male | Support |
| 3 | Anne | 125000 | 25 | Female | Analytics |
| 4 | Julia | 73000 | 30 | Female | Analytics |
| 5 | Matt | 159000 | 33 | Male | Sales |
| 6 | Jeff | 112000 | 27 | Male | Operations |

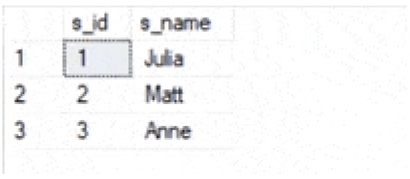
First, we will create a table.

```
create table stu2(s_id int unique, s_name varchar(20))
```

Now, we will insert the records.

```
insert into stu2 values(1,’Julia’)
insert into stu2 values(2,’Matt’)
insert into stu2 values(3,’Anne’)
```

**Output:**



|  | s_id | s_name |
|---|---|---|
| 1 | 1 | Julia |
| 2 | 2 | Matt |
| 3 | 3 | Anne |

A PRIMARY KEY constraint will automatically have a UNIQUE constraint. However, unlike a PRIMARY KEY, multiple UNIQUE constraints are allowed per table.

## 12. What is meant by table and field in SQL?

An organized data in the form of rows and columns is said to be a table. Simply put, it is a collection of related data in a table format.

Here rows and columns are referred to as tuples and attributes, and the number of columns in a table is referred to as a field. In the record, fields represent the characteristics and attributes and contain specific information about the data.

***Prepare yourself for PostgreSQL from our PostgreSQL interview questions blog***

## 13. What is a primary key?

A primary key is used to uniquely identify all table records. It cannot have NULL values and must contain unique values. Only one primary key can exist in one table, and it may have single or multiple fields, making it a composite key.

Now, we will write a query to demonstrate the use of a primary key for the employee table:

```
//
CREATE TABLE Employee (
ID int NOT NULL,
Employee_name varchar(255) NOT NULL,
Employee_designation varchar(255),
Employee_Age int,
PRIMARY KEY (ID)
);
```

## 14. What is a unique key?

A key that can accept only a null value and cannot accept duplicate values is called a unique key. The role of a unique key is to make sure that all columns and rows are unique.

The syntax for a unique key will be the same as the primary key. So, the query using a unique key for the employee table will be as follows:

```
//
CREATE TABLE Employee (
ID int NOT NULL,
Employee_name varchar(255) NOT NULL,
Employee_designation varchar(255),
Employee_Age int,
UNIQUE(ID)
);
```

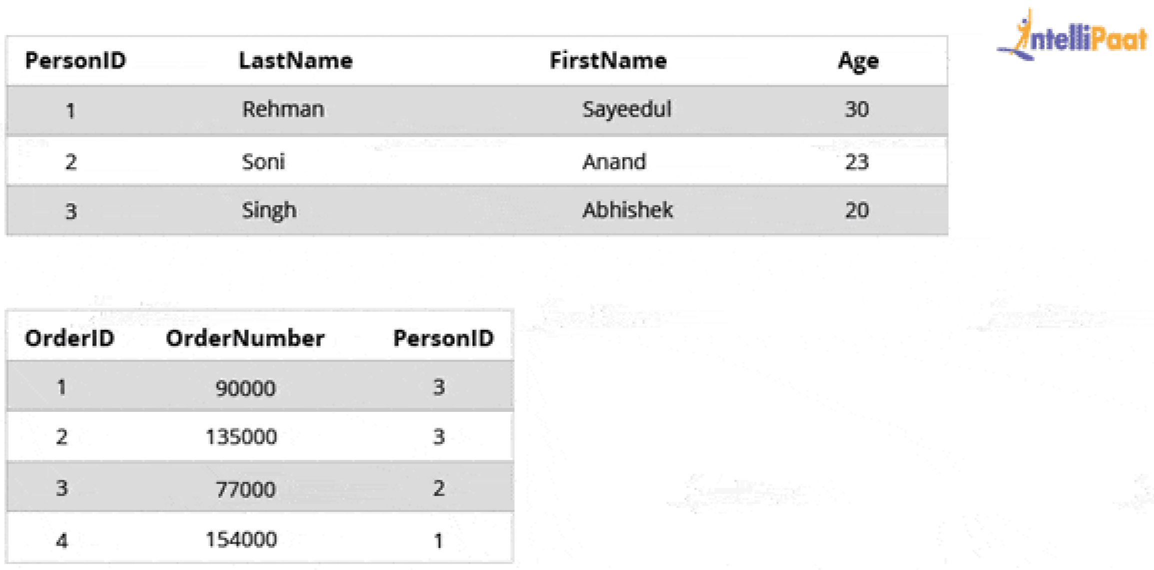
## 15. What is the difference between a primary key and a unique key?

Both primary and unique keys carry unique values but a primary key cannot have a null value, while a unique key can. In a table, there cannot be more than one primary key, but there can be multiple unique keys.

## 16. What is a foreign key?

A foreign key is an attribute or a set of attributes that reference the primary key of some other table. Basically, a foreign key is used to link together two tables together.

Let us create a foreign key for the following table:



| PersonID | LastName | FirstName | Age |
|---|---|---|---|
| 1 | Rehman | Sayeedul | 30 |
| 2 | Soni | Anand | 23 |
| 3 | Singh | Abhishek | 20 |

| OrderID | OrderNumber | PersonID |
|---|---|---|
| 1 | 90000 | 3 |
| 2 | 135000 | 3 |
| 3 | 77000 | 2 |
| 4 | 154000 | 1 |

```
CREATE TABLE Orders (
OrderID int NOT NULL,
OrderNumber int NOT NULL,
PersonID int,
PRIMARY KEY (OrderID),
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
)
```

## 17. What are the benefits of SQL database over NoSQL database?

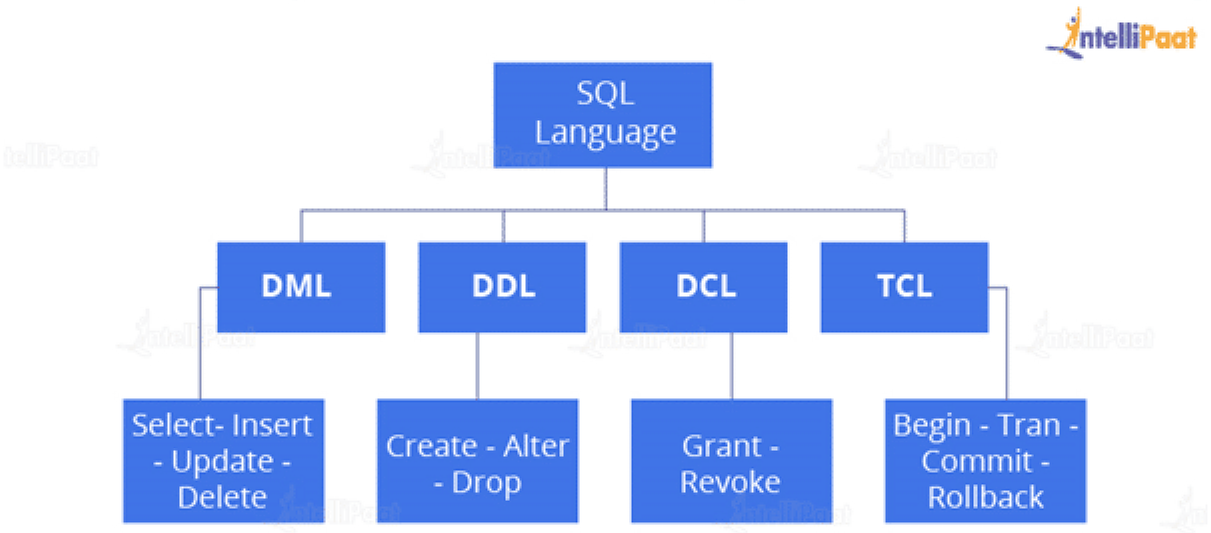
SQL (Structured Query Language) and NoSQL (Not Only SQL) are different database management systems, each offering unique advantages. Here, we will discuss the benefits of SQL over NoSQL in a comprehensive manner.

1. **Schema and Data Consistency:** SQL databases enforce a predefined schema, ensuring the data is structured and follows specific rules. It results in a higher level of data consistency, reducing the risk of data corruption or inconsistencies. On the other hand, NoSQL databases are schema-less, allowing for more flexibility but potentially sacrificing consistency.
2. **ACID Compliance:** SQL databases are designed to follow the ACID (Atomicity, Consistency, Isolation, Durability) principles. These properties guarantee transactional integrity, ensuring that database operations are reliable and maintain data integrity. NoSQL databases may sacrifice some of these properties for scalability and performance advantages.
3. **Advanced Querying Capabilities:** SQL databases provide a rich set of powerful querying capabilities through SQL. It allows for complex joins, aggregations, filtering, and data sorting. SQL queries are expressive and can handle complex relationships between tables efficiently. NoSQL databases may have limited querying capabilities, often requiring additional programming logic to achieve similar results.
4. **Data Integrity and Validation:** SQL databases offer built-in mechanisms for data integrity and validation through constraints, such as unique keys, foreign key relationships, and check constraints. These features ensure data quality and prevent invalid or inconsistent

data insertion. NoSQL databases may require additional application-level logic to enforce data integrity.

5. **Mature Ecosystem and Tooling:** SQL databases have been around for decades and have a well-established ecosystem with many tools, libraries, and frameworks. It includes robust backup and recovery solutions, monitoring tools, and mature integration options. Being relatively newer, NoSQL databases may have a more limited tooling and ecosystem.

## 18. Explain the different types of SQL commands.



- **DDL:** DDL is that part of SQL that defines the data structure of the database in the initial stage when the database is about to be created. It is mainly used to create and restructure database objects. Commands in DDL are:
    - Create table
    - Alter table
    - Drop table
- **DML:** DML is used to manipulate already existing data in a database, i.e., it helps users to retrieve and manipulate data. It is used to perform operations such as inserting data into the database through

the insert command, updating data with the update command, and deleting data from the database through the delete command.

- **DCL:** DCL is used to control access to the data in the database. DCL commands are normally used to create objects related to user access and to control the distribution of privileges among users. The commands that are used in DCL are Grant and Revoke.
- **TCL:** TCL is used to control the changes made by DML commands. It also authorizes the statements to assemble in conjunction with logical transactions. The commands that are used in TCL are Commit, Rollback, Savepoint, Begin, and Transaction.

Also, Have a look at SQL Command Cheatsheet.

## 19. What are the uses of SQL?

The following operations can be performed by using a SQL database:

- Creating new databases
- Inserting new data
- Deleting existing data
- Updating records
- Retrieving the data
- Creating and dropping tables
- Creating functions and views
- Converting data types

## 20. What is an index?

Indexes help speed up searching in a database. If there is no index on a column in the WHERE clause, then the SQL Server has to skim through the entire table and check each and every row to find matches, which may result in slow operations with large data.

Indexes are used to find all rows matching with some columns and then to skim through only those subsets of the data to find the matches.

**Syntax:**

```
CREATE INDEX INDEX_NAME ON TABLE_NAME (COLUMN)
```

## 21. Explain the types of indexes.

Single-Column Indexes: A single-column index is created for only one column of a table.

**Syntax:**

```
CREATE INDEX index_name
ON table_name(column_name);
```

Composite-Column Indexes: A composite-column index is created for two or more columns of a table.

**Syntax:**

```
CREATE INDEX index_name
ON table_name (column1, column2)
```

Unique Indexes: A unique index is used to maintain the data integrity of a table. A unique index does not allow multiple values to be inserted into the table.

**Syntax:**

```
CREATE UNIQUE INDEX index
ON table_name(column_name)
```

## 22. What are entities and relationships?

**Entities:** An entity can be a person, place, thing, or any identifiable object for which data can be stored in a database.

For example, in a company's database, employees, projects, salaries, etc., can be referred to as entities.

**Relationships:** A relationship between entities can be referred to as a connection between two tables or entities.

For example, in a college database, the student entity and the department entity are associated with each other.

That ends the section of basic interview questions. Let us now move on to the next section of intermediate interview questions.

## Intermediate SQL Interview Questions and Answers

### 23. What are SQL operators?

SQL operators are the special keywords or characters that perform specific operations. They are also used in SQL queries. These operators can be used within the WHERE clause of SQL commands. Based on the specified condition, SQL operators filter the data.

The SQL operators can be categorized into the following types:

- **Arithmetic Operators:** For mathematical operations on numerical data
  - addition (+)
  - subtraction (-)
  - multiplication (*)
  - division (/)
  - remainder/modulus (%)
- **Logical Operators:** For evaluating the expressions and returning results in True or False
  - ALL
  - AND
  - ANY

- ISNULL
- EXISTS
- BETWEEN
- IN
- LIKE
- NOT
- OR
- UNIQUE

- **Comparison Operators:** For comparisons of two values and checking whether they are the same or not
  - equal to (=)
  - not equal to (!= or <>)
  - less than (<),
  - greater than (>)
  - less than or equal to (<=)
  - greater than or equal to (>=)
  - not less than (!<)
  - not greater than (!>)

- **Bitwise Operators:** For bit manipulations between two expressions of integer type. It first performs the conversion of integers into binary bits and then applied operators
  - AND (& symbol)
  - OR (|, ^)
  - NOT (~)

- **Compound Operators:** For operations on a variable before setting the variable's result to the operation's result
  - Add equals (+=)
  - subtract equals (-=)
  - multiply equals (*=)
  - divide equals (/=)
  - modulo equals (%=)

- **String Operators:** For concatenation and pattern matching of strings
    - + (String concatenation)
    - += (String concatenation assignment)
    - % (Wildcard)
    - [] (Character(s) matches)
    - [^] (Character(s) not to match)
    - _ (Wildcard match one character)

## 24. What do you mean by data integrity?

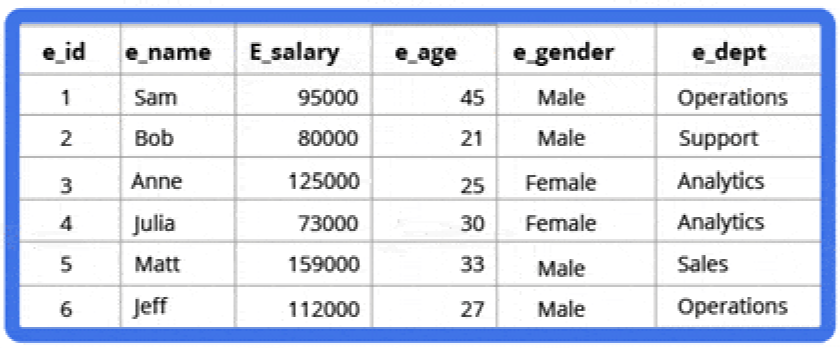
Data integrity is the assurance of the accuracy and consistency of data over its whole life cycle. It is a critical aspect of the design, implementation, and usage of systems that store, process, or retrieve data.

Data integrity also defines integrity constraints for enforcing business rules on data when it is entered into a database or application.

## 25. What is a data warehouse?

A data warehouse is a large store of accumulated data, from a wide range of sources, within an organization. The data helps drive business decisions.

## 26. How would you find the second highest salary from the following table?

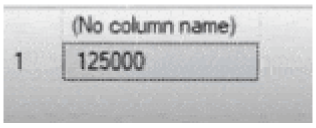


| e_id | e_name | E_salary | e_age | e_gender | e_dept |
|---|---|---|---|---|---|
| 1 | Sam | 95000 | 45 | Male | Operations |
| 2 | Bob | 80000 | 21 | Male | Support |
| 3 | Anne | 125000 | 25 | Female | Analytics |
| 4 | Julia | 73000 | 30 | Female | Analytics |
| 5 | Matt | 159000 | 33 | Male | Sales |
| 6 | Jeff | 112000 | 27 | Male | Operations |

**Code:**

```
select * from employee
select max(e_salary) from employee where e_salary not in (select
max(e_salary) from employee)
```

**Output:**



```
   (No column name)
1  125000
```

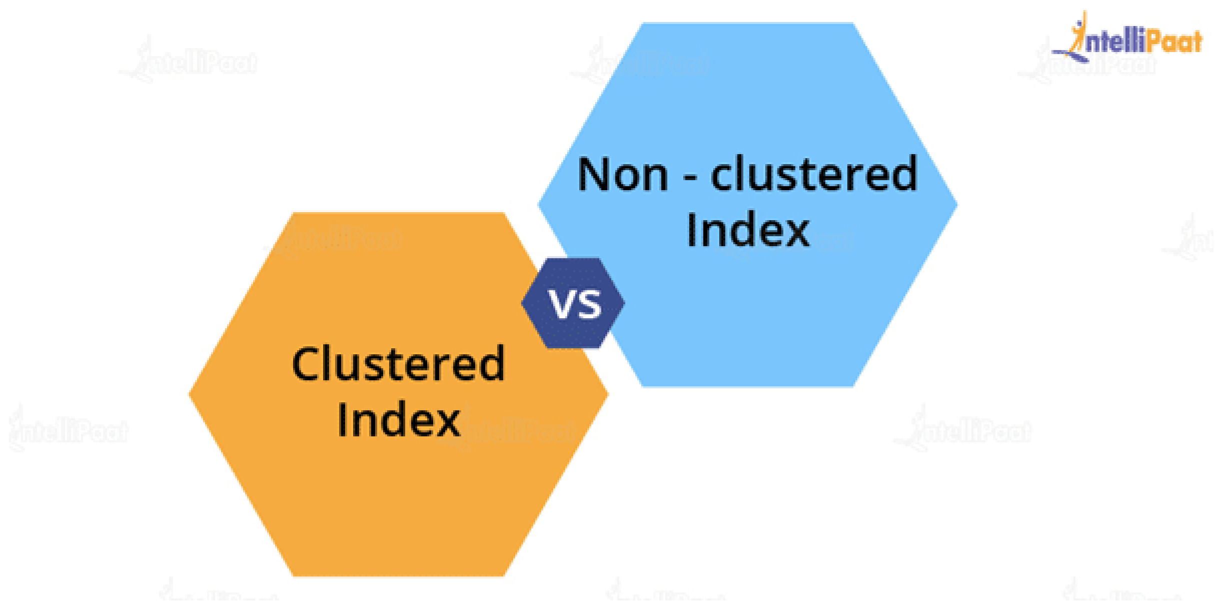
## 27. Why is the FLOOR function used in SQL Server?

The FLOOR() function helps to find the largest integer value for a given number, which can be an equal or lesser number.

***Are you planning to learn SQL Server? Here is the Microsoft SQL Certification training. Enroll now!***

## 28. State the differences between clustered and non-clustered indexes

- **Clustered Index:** It is used to sort the rows of data by their key values. A clustered index is like the contents of a phone book. We can directly open the book on David's index (for "David, Thompson") and find information for all Davids right next to each other. Since the data are located next to each other, it helps a lot in fetching the data based on range-based queries. A clustered index is actually related to how the data is stored; only one clustered index is possible per table.
- **Non-Clustered Index:** It stores data at one location and indexes at another location. The index has pointers that point to the location of the data. As the indexes in a non-clustered index are stored in a different place, there can be many non-clustered indexes for a table.



Now, we will see the major differences between clustered and non-clustered indexes:

| Parameters | Clustered Index | Non-Clustered Index |
|---|---|---|
| **Used For** | Sorting and storing records physically in memory | Creating a logical order for data rows; pointers are used for physical data files |
| **Methods for Storing** | Stores data in the leaf nodes of the index | Never stores data in the leaf nodes of the index |
| **Size** | Quite large | Comparatively, small |
| **Data Accessing** | Fast | Slow |
| **Additional Disk Space** | Not required | Required to store indexes separately |
| **Type of Key** | By default, the primary key of a table is a clustered index | It can be used with the unique constraint on the table that acts as a composite key |
| **Main Feature** | Improves the performance of data retrieval | Should be created on columns used in Joins |

## 29. What do you know about CDC in SQL Server?

CDC refers to change data capture. It captures recent INSERT, DELETE, and UPDATE activity applied to SQL Server tables. It records changes to SQL Server tables in a compatible format.

## 30. What is the difference between SQL and MySQL?

Now Let's compare the difference between SQL and MySQL.

| SQL | MySQL |
|---|---|
| It is a structured query language used in a database | It is a database management system |
| It is used for query and operating database systems, | It allows data handling, storing, and modification in an organized manner |
| It is always the same | It keeps updating |
| It supports only a single storage engine | It supports multiple storage engines |
| The server is independent | During backup sessions, the server blocks the database |

## 31. State the differences between SQL and PL/SQL

| SQL | PL/SQL |
|---|---|
| It is a database structured query language | It is a programming language for a database that uses SQL |
| It is an individual query that is used to execute DML and DDL commands | It is a block of codes used to write the entire procedure or a function |
| It is a declarative and data-oriented language | It is a procedural and application-oriented language |
| It is mainly used for data manipulation | It is used for creating applications |
| It provides interaction with the database server | It does not provide interaction with the database server |
| It cannot contain PL/SQL code | It can contain SQL because it is an extension of SQL |

## 32. What is the ACID property in a database?

The full form of ACID is atomicity, consistency, isolation, and durability. ACID properties are used to check the reliability of transactions.

- 
    - Atomicity refers to completed or failed transactions, where a transaction refers to a single logical operation on data. This implies that if any aspect of a transaction fails, the whole transaction fails and the database state remains unchanged.
- Consistency means that the data meets all validity guidelines. The transaction never leaves the database without finishing its state.
- Concurrency management is the primary objective of isolation.
- Durability ensures that once a transaction is committed, it will occur regardless of what happens in between such as a power outage, fire, or some other kind of disturbance.

## 33. What is the need for group functions in SQL?

Group functions operate on a series of rows and return a single result for each group. COUNT(), MAX(), MIN(), SUM(), AVG(), and VARIANCE() are some of the most widely used group functions.

## 34. What do you understand about a character manipulation function?

Character manipulation functions are used for the manipulation of character data types.
Some of the character manipulation functions are as follows:

**UPPER:** It returns the string in uppercase.

**Syntax:**

```
UPPER(‘ string’)
```

**Example:**

```
SELECT UPPER(‘demo string’) from String;
```

**Output:**

```
DEMO STRING
```

**LOWER:** It returns the string in lowercase.

**Syntax:**

```
LOWER(‘STRING’)
```

**Example:**

```
SELECT LOWER (‘DEMO STRING’) from String
```

**Output:**

```
demo string
```

**INITCAP:** It converts the first letter of the string to uppercase and retains others in lowercase.

**Syntax:**

```
Initcap(‘sTRING’)
```

**Example:**

```
SELECT Initcap(‘dATASET’) from String
```

**Output:**

```
Dataset
```

**CONCAT:** It is used to concatenate two strings.

**Syntax:**

```
CONCAT(‘str1’,’str2’)
```

**Example:**

```
SELECT CONCAT(‘Data’,’Science’) from String
```

**Output:**

```
Data Science
```

**LENGTH:** It is used to get the length of a string.

**Syntax:**

```
LENGTH(‘String’)
```

**Example:**

```
SELECT LENGTH(‘Hello World’) from String
```

**Output:**

```
11
```

## 35. What is AUTO_INCREMENT?

AUTO_INCREMENT is used in SQL to automatically generate a unique number whenever a new record is inserted into a table.

Since the primary key is unique for each record, this primary field is added as the AUTO_INCREMENT field so that it is incremented when a new record is inserted.

The AUTO-INCREMENT value starts at 1 and is incremented by 1 whenever a new record is inserted.

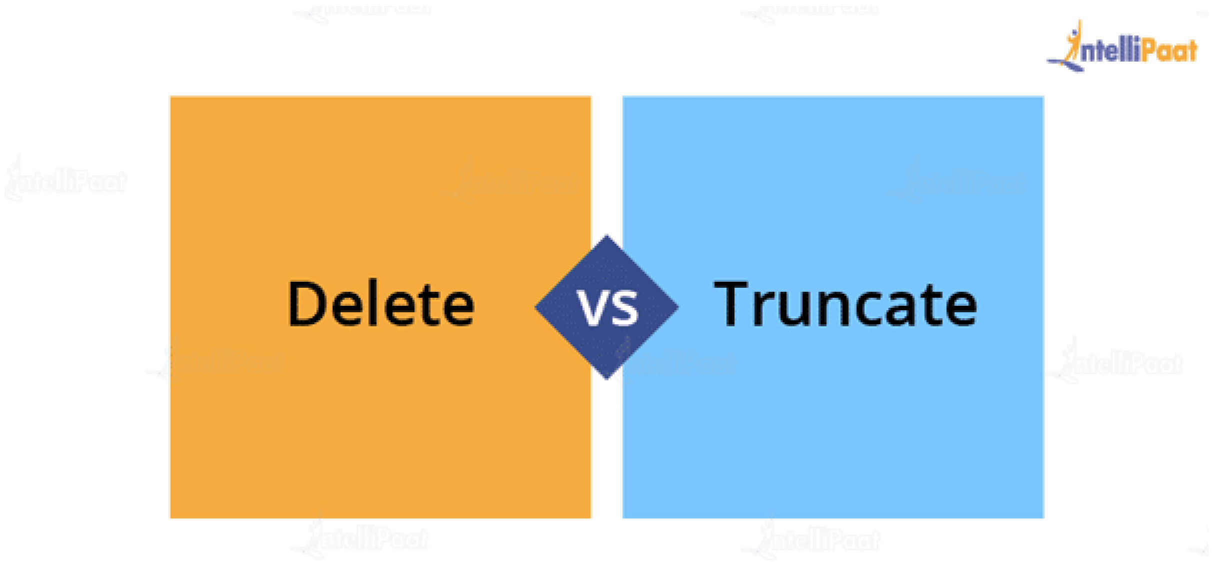
Syntax:

```
CREATE TABLE Employee(
Employee_id int NOT NULL AUTO-INCREMENT,
Employee_name varchar(255) NOT NULL,
Employee_designation varchar(255)
Age int,
PRIMARY KEY (Employee_id)
)
```

***Check out our Blog on [PL/SQL Interview Questions] to crack your SQL Interview.***

## 36. What is the difference between the DELETE and TRUNCATE commands?

- **DELETE:** This query is used to delete or remove one or more existing tables.
- **TRUNCATE:**  This statement deletes all the data inside a table.



The differences between DELETE and TRUNCATE commands are the following:

- TRUNCATE is a DDL command, and DELETE is a DML command.
- With TRUNCATE, we cannot really execute and trigger, while with DELETE, we can accomplish a trigger.
- If a table is referenced by foreign key constraints, then TRUNCATE will not work. So, if we have a foreign key, we have to use the DELETE command.

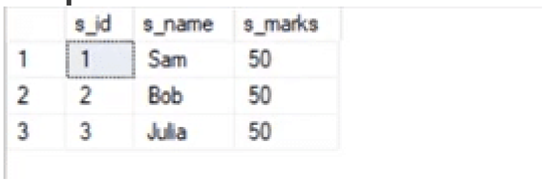
The syntax for the DELETE command is as follows:

```
DELETE FROM table_name
[WHERE condition];
```

**Example:**

```
select * from stu
```

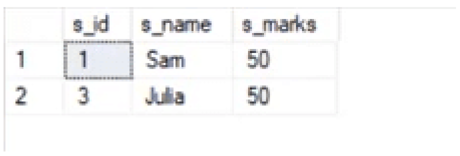
**Output:**



|   | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 1 | Sam | 50 |
| 2 | 2 | Bob | 50 |
| 3 | 3 | Julia | 50 |

```
delete from stu where s_name='Bob'
```

**Output:**



|   | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 1 | Sam | 50 |
| 2 | 3 | Julia | 50 |

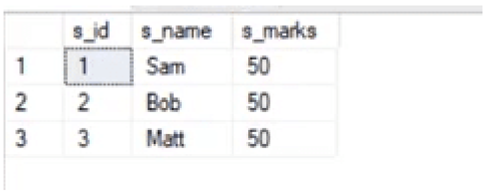
The syntax for the TRUNCATE command:

```
TRUNCATE TABLE
Table_name;
```

**Example:**

```
select * from stu1
```

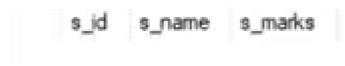
**Output:**



|   | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 1 | Sam | 50 |
| 2 | 2 | Bob | 50 |
| 3 | 3 | Matt | 50 |

```
truncate table stu1
```

**Output:**



|   | s_id | s_name | s_marks |
|---|---|---|---|

This deletes all the records from a table.

## 37. What is the difference between the DROP and TRUNCATE commands?

If a table is dropped, all things associated with that table are dropped as well. This includes the relationships defined on the table with other tables, access privileges, and grants that the table has, as well as the integrity checks and constraints.

To create and use the table again in its original form, all the elements associated with the table need to be redefined.

However, if a table is truncated, there are no such problems as mentioned above. The table retains its original structure.

## 38. What is a “TRIGGER” in SQL?

The trigger can be defined as an automatic process that happens when an event occurs in the database server. It helps to maintain the integrity of the table. The trigger is activated when the commands, such as insert, update, and delete, are given.

The syntax used to generate the trigger function is as follows:

```
CREATE TRIGGER trigger_name
```

## 39. Where are usernames and passwords stored in SQL Server?

In SQL Server, usernames and passwords are stored in the main database in the sysxlogins table.

## 40. What are the types of relationships in SQL Server databases?

Relationships are developed by interlinking the columns of one table with the column of another table. There are three different types of relationships, which are as follows:

- One-to-one relationship
- Many-to-one relationship
- Many-to-many relationship

## 41. What are the third-party tools that are used in SQL Server?

The following is the list of third-party tools that are used in SQL Server:

- SQL CHECK
- SQL DOC 2
- SQL Backup 5
- SQL Prompt
- Litespeed 5.0

## 42. How can you handle expectations in SQL Server?

TRY and CATCH blocks handle exceptions in SQL Server. Put the SQL statement in the TRY block and write the code in the CATCH block to handle expectations. If there is an error in the code in the TRY block, then the control will automatically move to the CATCH block.

## 43. How many authentication modes are there in SQL Server? What are they?

Two authentication modes are available in SQL Server. They are as follows:

- **Windows Authentication Mode:** It allows authentication for Windows but not for SQL Server.
- **Mixed Mode:** It allows both types of authentication—Windows and SQL Server.

## 44. What is a function in SQL Server?

A function is an SQL Server database object. It is basically a set of SQL statements that allow input parameters, perform processing, and return results only. A function can only return a single value or table. The ability to insert, update, and delete records in database tables is not available.

## 45. Mention different types of replication in SQL Server?

In SQL Server, three different types of replications are available:

- Snapshot replication
- Transactional replication
- Merge replication

## 46. Which command is used to find out the SQL Server version?

The following command is used to identify the version of SQL Server:

```
Select SERVERPROPERTY('productversion')
```

## 47. What is the COALESCE function?

The COALESCE function takes a set of inputs and returns the first non-null value.

**Syntax:**

```
COALESCE(val1,val2,val3,……,nth val)
```

**Example:**

```
SELECT COALESCE(NULL, 1, 2, ‘MYSQL’)
```

**Output:**

```
1
```

## 48. Can we link SQL Server with others?

Yes, SQL Server can be linked with other database systems using various methods. One common method is through the use of linked servers. Linked servers allow SQL Server to establish connections and access data from other database platforms. By configuring appropriate settings and creating the necessary connections, SQL Server can interact with databases such as MySQL, Oracle, PostgreSQL, and more, enabling data integration and querying across multiple systems.

## 49. What is SQL Server Agent?

SQL Server Agent plays an important role in the daily work of SQL Server administrators or DBAs. This is one of the important parts of SQL Server. The aim of the server agent is to easily implement tasks using a scheduler engine that enables the tasks to be performed at scheduled times. SQL Server Agent uses SQL Server to store scheduled management task information.

## 50. What do you know about magic tables in SQL Server?

A magic table can be defined as a provisional logical table that is developed by an SQL Server for tasks such as insert, delete, or update (DML) operations. The operations recently performed on the rows are automatically stored in magic tables. Magic tables are not physical tables; they are just temporary internal tables.

## 51. What are some common clauses used with SELECT queries in SQL?

There are many SELECT statement clauses in SQL. Some of the most commonly used clauses with SELECT queries are as follows:

- **FROM**
  The FROM clause defines the tables and views from which data can be interpreted. The tables and views listed must exist at the time the question is given.

- **WHERE**
  The WHERE clause defines the parameters that are used to limit the contents of the results table. You can test for basic relationships or for relationships between a column and a series of columns using subselects.
- **GROUP BY**
  The GROUP BY clause is commonly used for aggregate functions to produce a single outcome row for each set of unique values in a set of columns or expressions.
- **ORDER BY**
  The ORDER BY clause helps in choosing the columns on which the table's result should be sorted.
- **HAVING**
  The HAVING clause filters the results of the GROUP BY clause by using an aggregate function.

## 52. What is wrong with the following SQL query?

```
SELECT gender, AVG(age) FROM employee WHERE AVG(age)>30 GROUP BY
gender
```

When this command is executed, it gives the following error:

```
Msg 147, Level 16, State 1, Line 1
```

Aggregation may not appear in the WHERE clause unless it is in a subquery contained in the HAVING clause or a select list; the column being aggregated is an outer reference.

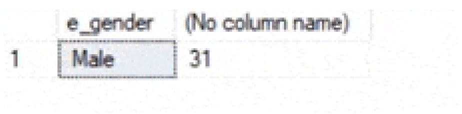
```
Msg 147, Level 16, State 1, Line 1
Invalid column name ‘gender’.
```

This means that whenever we work with aggregate functions and use the GROUP BY clause, we cannot use the WHERE clause. Therefore, instead of the WHERE clause, we should use the HAVING clause.

When we use the HAVING clause, the GROUP BY clause should come first, followed by the HAVING clause.

```
select e_gender, avg(e_age) from employee group by e_gender having
avg(e_age)>30
```

**Output:**



```
   e_gender  (No column name)
1  Male      31
```

## 53. What do you know about the stuff() function?

The stuff() function deletes a part of the string and then inserts another part into the string, starting at a specified position.

**Syntax:**

```
STUFF(String1, Position, Length, String2)
```

Here, String1 is the one that will be overwritten. The position indicates the starting location for overwriting the string. Length is the length of the substitute string, and String2 is the string that will overwrite String1.

**Example:**

```
select stuff('SQL Tutorial',1,3,'Python')
```

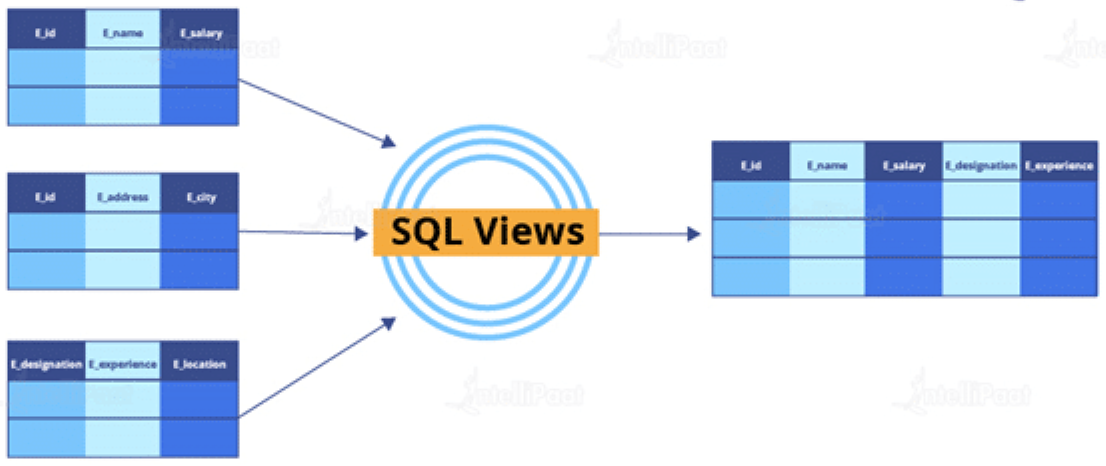
This will change 'SQL Tutorial' to 'Python Tutorial'

**Output:**

```
Python Tutorial
```

## 54. What are views? Give an example.

Views are virtual tables used to limit the tables that we want to display. Views are nothing but the result of an SQL statement that has a name associated with it. Since views are not physically present, they take less space to store.

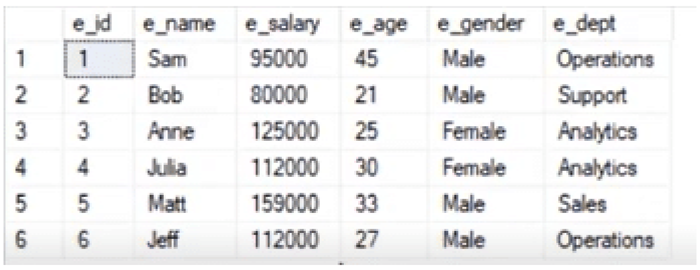


Let us consider an example. In the following employee table, say we want to perform multiple operations on the records with the gender “Female”. We can create a view-only table for the female employees from the entire employee table.

Now, let us implement it on SQL Server.

This is the employee table:

```
select * from employee
```



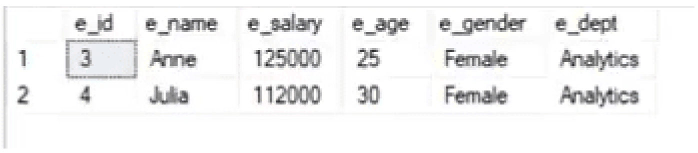
| | e_id | e_name | e_salary | e_age | e_gender | e_dept |
|---|---|---|---|---|---|---|
| 1 | 1 | Sam | 95000 | 45 | Male | Operations |
| 2 | 2 | Bob | 80000 | 21 | Male | Support |
| 3 | 3 | Anne | 125000 | 25 | Female | Analytics |
| 4 | 4 | Julia | 112000 | 30 | Female | Analytics |
| 5 | 5 | Matt | 159000 | 33 | Male | Sales |
| 6 | 6 | Jeff | 112000 | 27 | Male | Operations |

Now, we will write the syntax for the view.

**Syntax:**

```
create view female_employee as select * from employee where
e_gender=’Female’
select * from female_employee
```

**Output:**



| | e_id | e_name | e_salary | e_age | e_gender | e_dept |
|---|---|---|---|---|---|---|
| 1 | 3 | Anne | 125000 | 25 | Female | Analytics |
| 2 | 4 | Julia | 112000 | 30 | Female | Analytics |

### 55. What are the types of views in SQL?

In SQL, the views are classified into four types. They are the following:

- **Simple View:** It is a view based on a single table and does not have a GROUP BY clause or other features.
- **Complex View:** It is a view built from several tables and includes a GROUP BY clause as well as functions.
- **Inline View:** It is a view built on a subquery in the FROM clause, which provides a temporary table and simplifies a complicated query.
- **Materialized View:** It is a view that saves both the definition and the details. It builds data replicas by physically preserving them.

## Advanced SQL Interview Questions and Answers For Experienced

### 56. What is a stored procedure? Give an example.

A stored procedure is a prepared SQL code that can be saved and reused. In other words, we can consider a stored procedure to be a function consisting of many SQL statements to access the database system. We can consolidate several SQL statements into a stored procedure and execute them whenever and wherever required.

A stored procedure can be used as a means of modular programming, i.e., we can create a stored procedure once, store it, and call it multiple times as required. This also supports faster execution when compared to executing multiple queries.

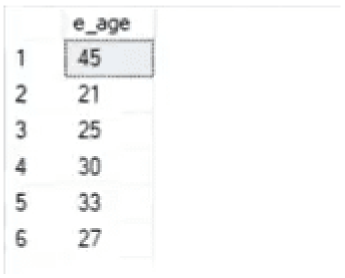
**Syntax:**

```
CREATE PROCEDURE procedure_name
AS
Sql_statement
GO;
To execute we will use this:
EXEC procedure_name
```

**Example:**

```
We are going to create a stored procedure that will help us extract
the age of the employees.
create procedure employee_age
as
select e_age from employee
go
Now, we will execute it.
exec employee_age
```

**Output:**



```
   e_age
1  45
2  21
3  25
4  30
5  33
6  27
```

## 57. Explain Inner Join with an example.

Inner Join gives us those records that have matching values in two tables.

Let us assume that we have two tables: Table A and Table B. When we apply Inner Join to these two tables, we will get only records common to both Table A and Table B.

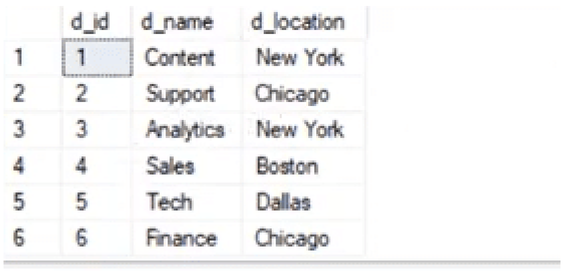
**Syntax:**

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column_x=table2.column_y;
```

**Example:**

```
select * from employee
select * from department
```

**Output:**



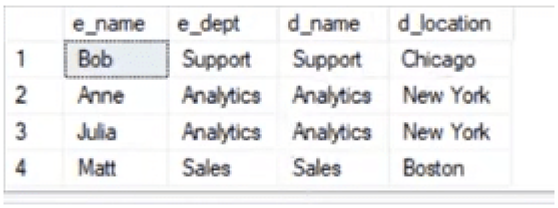
```
   d_id  d_name     d_location
1  1     Content    New York
2  2     Support    Chicago
3  3     Analytics  New York
4  4     Sales      Boston
5  5     Tech       Dallas
6  6     Finance    Chicago
```

Now, we will apply Inner Join to both these tables, where the e_dept column in the employee table is equal to the d_name column of the department table.

**Syntax:**

```
select employee.e_name, employee.e_dept, department.d_name,
department.d_location
from employee inner join department
on
employee.e_dept=department.d_name
```

**Output:**



```
   e_name  e_dept     d_name     d_location
1  Bob     Support    Support    Chicago
2  Anne    Analytics  Analytics  New York
3  Julia   Analytics  Analytics  New York
4  Matt    Sales      Sales      Boston
```

After applying Inner Join, we have only those records where the departments match in both tables. As we can see, the matched departments are Support, Analytics, and Sales.

## 58. State the differences between views and tables.

| Views | Tables |
|---|---|
| A view is a virtual table that is extracted from a database. | A table is structured with a set number of columns and a boundless number of rows. |
| A view does not hold the data itself. | A table contains data and stores it in |

| | |
|---|---|
| | databases. |
| A view is utilized to query certain information contained in a few distinct tables. | A table holds fundamental client information and cases of a characterized object. |
| In a view, we will get frequently queried information. | In a table, changing the information in the database changes the information that appears in the view. |

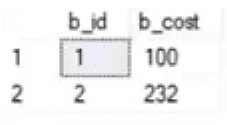
## 59. What do you understand about a temporary table? Write a query to create a temporary table

A temporary table helps us store and process intermediate results. Temporary tables are created and can be automatically deleted when they are no longer used. They are very useful in places where temporary data needs to be stored.

**Syntax:**

```
CREATE TABLE #table_name();
The below query will create a temporary table:
create table #book(b_id int, b_cost int)
Now, we will insert the records.
insert into #book values(1,100)
insert into #book values(2,232)
select * from #book
```

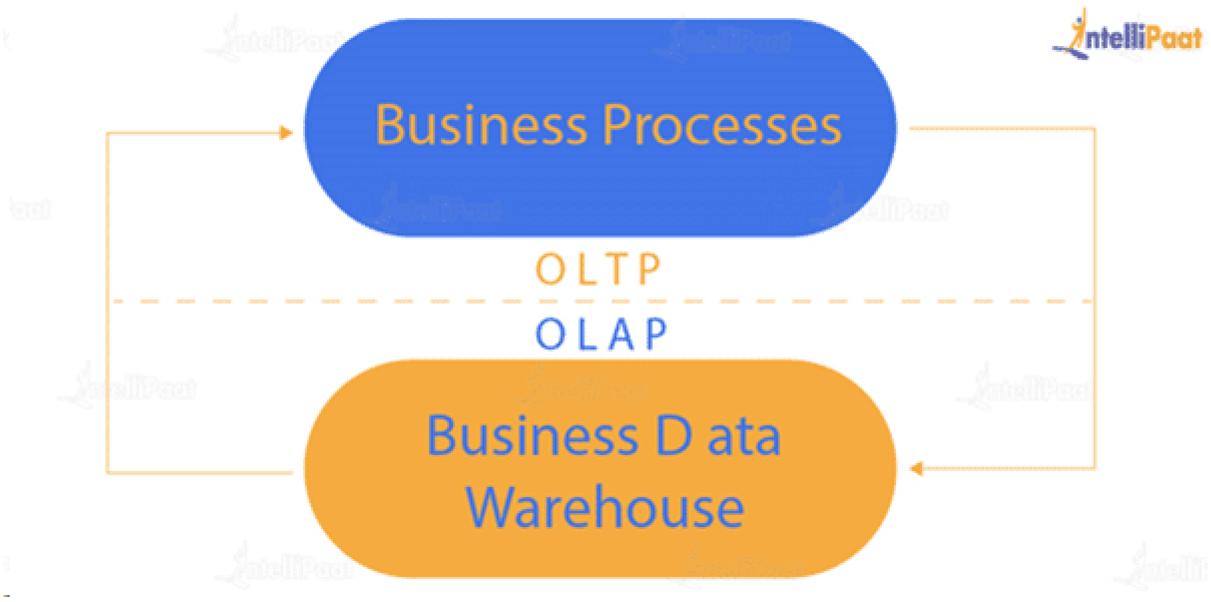
**Output:**



```
   b_id  b_cost
1  1     100
2  2     232
```

## 60. Explain the difference between OLTP and OLAP.

**OLTP:** It stands for online transaction processing, and we can consider it a category of software applications that are efficient for supporting transaction-oriented programs. One of the important attributes of the OLTP system is its potential to maintain consistency. The OLTP system often follows decentralized planning to avoid single points of failure. This system is generally designed for a large audience of end users to perform short transactions. The queries involved in such databases are generally simple, require a fast response time, and, in comparison, return only a few records. So, the number of transactions per second acts as an effective measure for those systems.



**OLAP:** It stands for online analytical processing, and it is a category of software programs that are identified by a comparatively lower frequency of online transactions. For OLAP systems, the efficiency of computing depends heavily on the response time. Hence, such systems are generally used for data mining or maintaining aggregated historical data, and they are usually used in multidimensional schemas.

## 61. What is Hybrid OLAP?

Hybrid OLAP (HOLAP) uses a combination of multidimensional data structures and relational database tables to store multidimensional data. The aggregations for a HOLAP partition are stored by analysis services in a multidimensional structure. The facts are stored in a relational database.

## 62. What do you understand by Self Join? Explain using an example

Self Join in SQL is used for joining a table with itself. Here, depending on some conditions, each row of the table is joined with itself and with other rows of the table.

**Syntax:**

```
SELECT a.column_name, b.column_name
FROM table a, table b
WHERE condition
```

**Example:**

Consider the customer table given below.

**Example:**

Consider the customer table given below.

| ID | Name | Age | Address | Salary |
|---|---|---|---|---|
| 1 | Anand | 32 | Ahmedabad | 2,000.00 |
| 2 | Abhishek | 25 | Delhi | 1,500.00 |
| 3 | Shivam | 23 | Kota | 2,000.00 |
| 4 | Vishal | 25 | Mumbai | 6,500.00 |
| 5 | Sayeedul | 27 | Bhopal | 8,500.00 |
| 6 | Amir | 22 | MP | 4,500.00 |
| 7 | Arpit | 24 | Indore | 10,000.00 |

We will now join the table using Self Join:

```
SQL> SELECT a.ID, b.NAME, a.SALARY
FROM CUSTOMERS a, CUSTOMERS b
WHERE a.SALARY < b.SALARY;
```

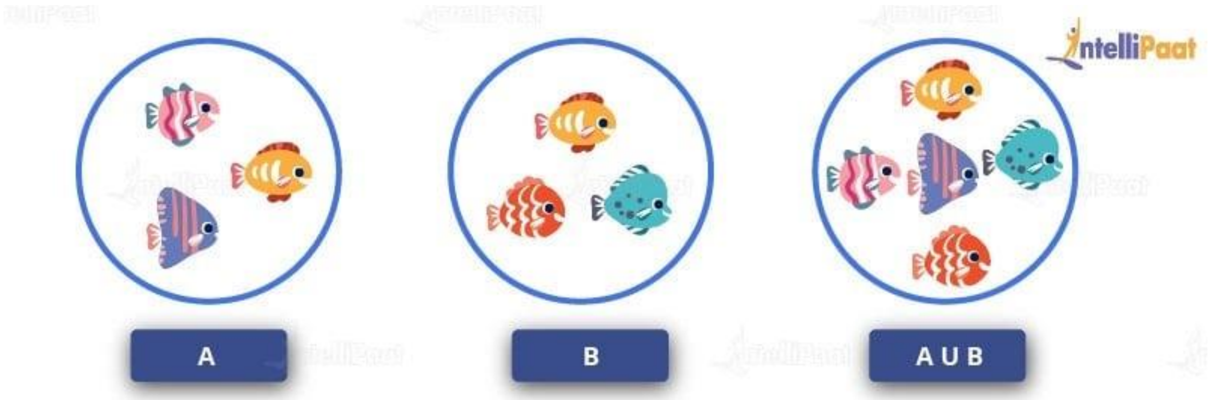
**Output:**

| ID | Name | Salary |
|---|---|---|
| 2 | Anand | 1,500.00 |
| 2 | Abhishek | 1,500.00 |
| 1 | Vishal | 2,000.00 |
| 2 | Vishal | 1,500.00 |
| 3 | Vishal | 2,000.00 |

| | | |
|---|---|---|
| 6 | Vishal | 4,500.00 |
| 1 | Sayeedul | 2,000.00 |
| 2 | Sayeedul | 1,500.00 |
| 3 | Sayeedul | 2,000.00 |
| 4 | Sayeedul | 6,500.00 |
| 6 | Sayeedul | 4,500.00 |
| 1 | Amir | 2,000.00 |
| 2 | Amir | 1,500.00 |
| 3 | Amir | 2,000.00 |
| 1 | Arpit | 2,000.00 |
| 2 | Arpit | 1,500.00 |
| 3 | Arpit | 2,000.00 |
| 4 | Arpit | 6,500.00 |
| 5 | Arpit | 8,500.00 |
| 6 | Arpit | 4,500.00 |

## 63. What is the difference between Union and Union All operators?

The union operator is used to combine the result set of two or more select statements. For example, the first select statement returns the fish shown in Image A, and the second statement returns the fish shown in Image B. The Union operator will then return the result of the two select statements as shown in Image A U B. If there is a record present in both tables, then we will get only one of them in the final result.



**Syntax:**

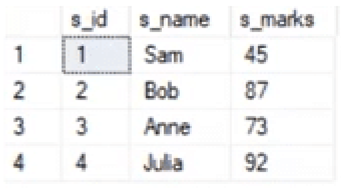
```
SELECT column_list FROM table1
```

**Union:**

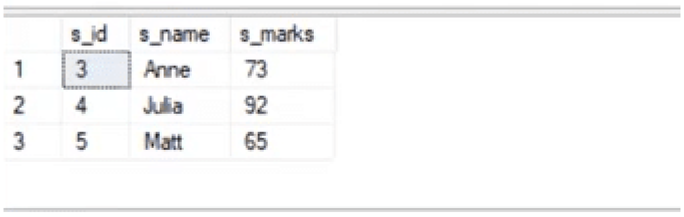
```
SELECT column_list FROM table2
```

Now, we will execute it in the SQL Server.

These are the two tables in which we will use the Union operator.



| | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 1 | Sam | 45 |
| 2 | 2 | Bob | 87 |
| 3 | 3 | Anne | 73 |
| 4 | 4 | Julia | 92 |



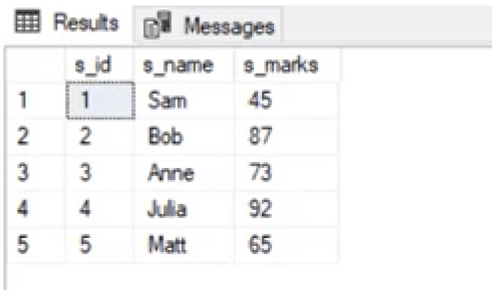
| | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 3 | Anne | 73 |
| 2 | 4 | Julia | 92 |
| 3 | 5 | Matt | 65 |

```
select * from student_details1
```

**Union:**

```
select * from student_details2
```

**Output:**



| | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 1 | Sam | 45 |
| 2 | 2 | Bob | 87 |
| 3 | 3 | Anne | 73 |
| 4 | 4 | Julia | 92 |
| 5 | 5 | Matt | 65 |

The Union All operator gives all the records from both tables including the duplicates.

A
B
A union all B

Let us implement it in the SQL Server.

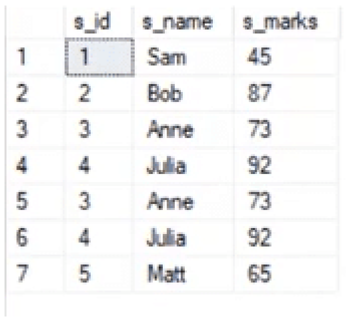
**Syntax:**

```
select * from student_details1
```

**Union All:**

```
select * from student_details2
```

**Output:**



| | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 1 | Sam | 45 |
| 2 | 2 | Bob | 87 |
| 3 | 3 | Anne | 73 |
| 4 | 4 | Julia | 92 |
| 5 | 3 | Anne | 73 |
| 6 | 4 | Julia | 92 |
| 7 | 5 | Matt | 65 |

## 64. What is a database cursor? How to use a database cursor?

A database cursor is a control that allows you to navigate around a table's rows or documents. It can be referred to as a pointer for a row in a set of rows. Cursors are extremely useful for database traversal operations such as extraction, insertion, and elimination.

- After any variable declaration, DECLARE a cursor. A SELECT statement must always be aligned with the cursor declaration.

- To initialize the result set, OPEN statements must be called before fetching the rows from the result table.
- To grab and switch to the next row in the result set, use the FETCH statement.
- To deactivate the cursor, use the CLOSE expression.
- Finally, use the DEALLOCATE clause to uninstall the cursor description and clear all the resources associated with it.

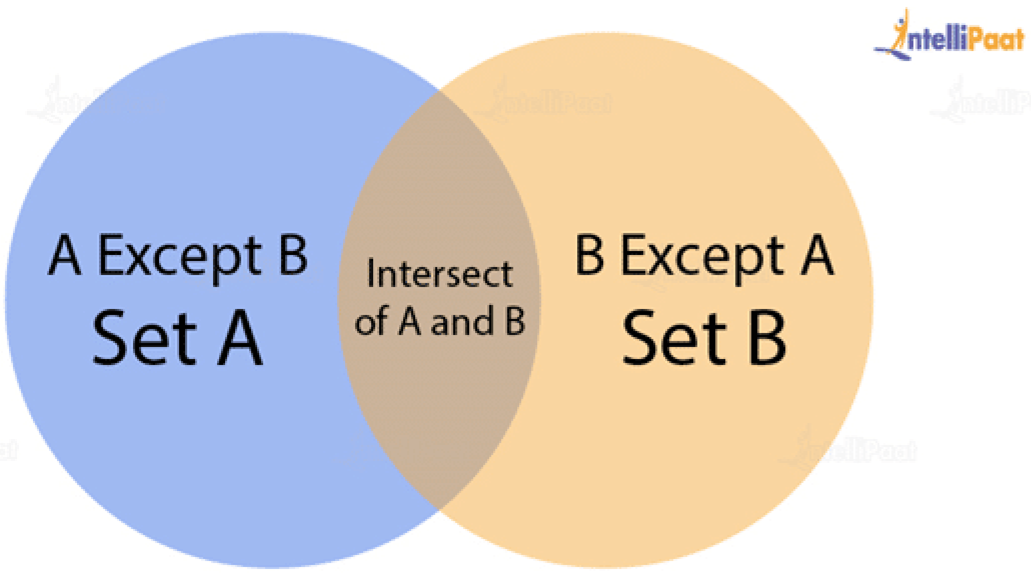
Here is an example SQL cursor:

```
DECLARE @name VARCHAR(50)

DECLARE db_cursor CURSOR FOR
SELECT name
From myDB.company
WHERE employee_name IN (‘Jay’, ‘Shyam’)
OPEN db_cursor
FETCH next
FROM db_cursor
Into @name
Close db_cursor
DEALLOCATE db_cursor
```

## 65. What is the use of the INTERSECT operator?

The INTERSECT operator helps combine two select statements and returns only those records that are common to both select statements. After we get Table A and Table B over here, and if we apply the INTERSECT operator on these two tables, then we will get only those records that are common to the result of the select statements of these two tables.



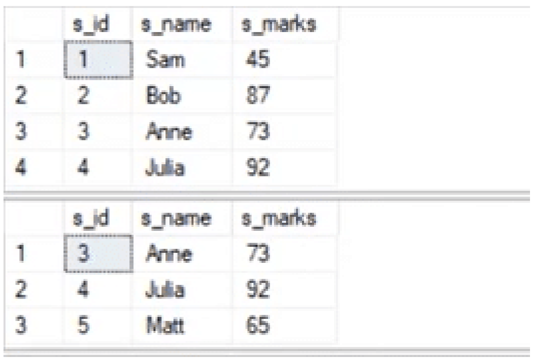
**Syntax:**

```
SELECT column_list FROM table1
INTERSECT
SELECT column_list FROM table2
```

Now, let us take a look at an example of the INTERSECT operator.

```
select * from student_details1
select * from student_details1
```

**Output:**

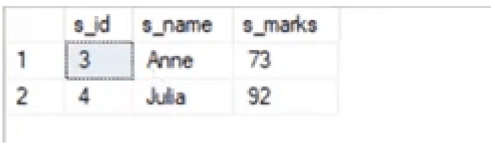


| | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 1 | Sam | 45 |
| 2 | 2 | Bob | 87 |
| 3 | 3 | Anne | 73 |
| 4 | 4 | Julia | 92 |

| | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 3 | Anne | 73 |
| 2 | 4 | Julia | 92 |
| 3 | 5 | Matt | 65 |

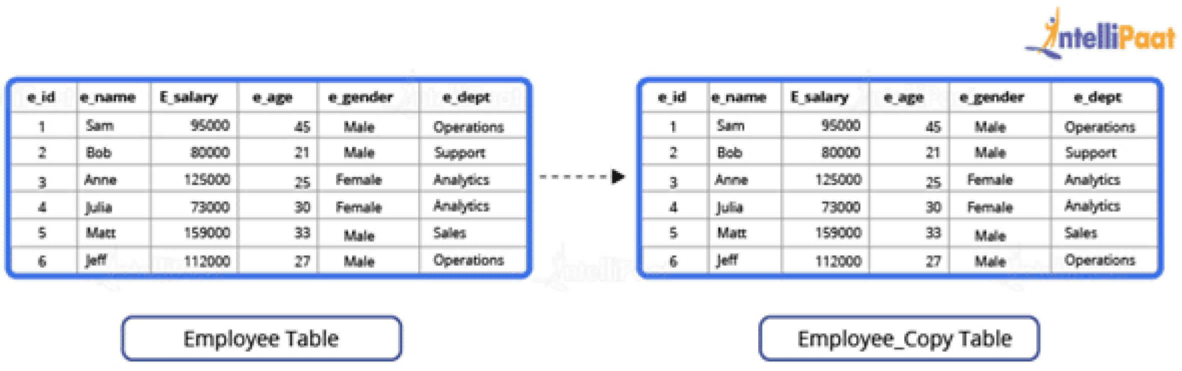
```
select * from student_details1
intersect
select * from student_details2
```

**Output:**



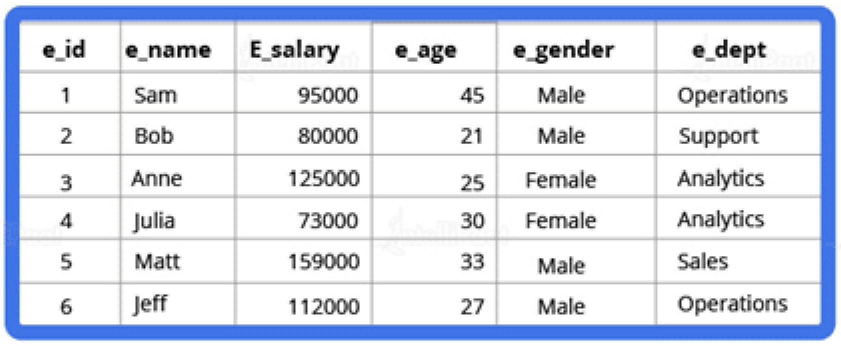
| | s_id | s_name | s_marks |
|---|---|---|---|
| 1 | 3 | Anne | 73 |
| 2 | 4 | Julia | 92 |

## 66. How can you copy data from one table to another table?



| e_id | e_name | E_salary | e_age | e_gender | e_dept |
|---|---|---|---|---|---|
| 1 | Sam | 95000 | 45 | Male | Operations |
| 2 | Bob | 80000 | 21 | Male | Support |
| 3 | Anne | 125000 | 25 | Female | Analytics |
| 4 | Julia | 73000 | 30 | Female | Analytics |
| 5 | Matt | 159000 | 33 | Male | Sales |
| 6 | Jeff | 112000 | 27 | Male | Operations |

Here, we have our employee table.



| e_id | e_name | E_salary | e_age | e_gender | e_dept |
|---|---|---|---|---|---|
| 1 | Sam | 95000 | 45 | Male | Operations |
| 2 | Bob | 80000 | 21 | Male | Support |
| 3 | Anne | 125000 | 25 | Female | Analytics |
| 4 | Julia | 73000 | 30 | Female | Analytics |
| 5 | Matt | 159000 | 33 | Male | Sales |
| 6 | Jeff | 112000 | 27 | Male | Operations |

We have to copy this data into another table. For this purpose, we can use the INSERT INTO SELECT operator. Before we go ahead and do that, we will have to create another table that will have the same structure as the above-given table.

**Syntax:**

```
create table employee_duplicate(
e_id int,
e_name varchar(20),
e_salary int,
e_age int,
e_gender varchar(20)
e_dept varchar(20)
)
```

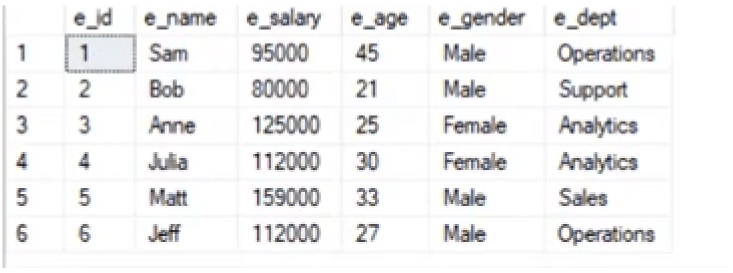
For copying the data, we will use the following query:

```
insert into employee_duplicate select * from employees
```

Let us take a look at the copied table.

```
select * from employee_duplicate
```

**Output:**



| | e_id | e_name | e_salary | e_age | e_gender | e_dept |
|---|---|---|---|---|---|---|
| 1 | 1 | Sam | 95000 | 45 | Male | Operations |
| 2 | 2 | Bob | 80000 | 21 | Male | Support |
| 3 | 3 | Anne | 125000 | 25 | Female | Analytics |
| 4 | 4 | Julia | 112000 | 30 | Female | Analytics |
| 5 | 5 | Matt | 159000 | 33 | Male | Sales |
| 6 | 6 | Jeff | 112000 | 27 | Male | Operations |

## 67. What is the difference between BETWEEN and IN operators in SQL?

The BETWEEN operator is employed to identify rows that fall within a specified range of values, encompassing numerical, textual, or date values. It returns the count of values that exist between the two defined boundaries.

On the other hand, the IN operator serves as a condition operator utilized for searching values within a predetermined range. When multiple values are available for selection, the IN operator is utilized.

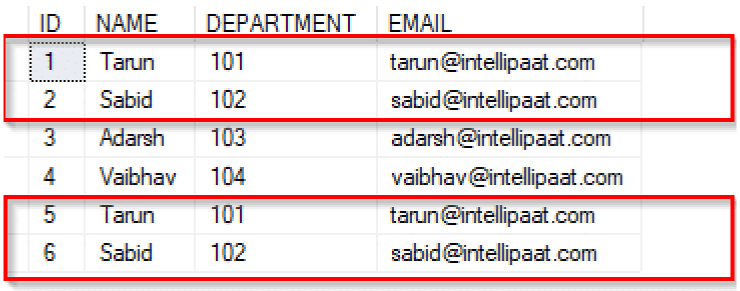
***Check out how to use IN and BETWEEN Operators in SQL with examples.***

## 68. Describe how to delete duplicate rows using a single statement but without any table creation.

Let us create an employee table where the column names are ID, NAME, DEPARTMENT, and EMAIL. Below are the SQL scripts for generating the sample data:

```
CREATE TABLE EMPLOYEE
(
ID INT,
NAME Varchar(100),
DEPARTMENT INT,
EMAIL Varchar(100)
)

INSERT INTO EMPLOYEE VALUES (1,'Tarun',101,'tarun@intellipaat.com')
INSERT INTO EMPLOYEE VALUES (2,'Sabid',102,'sabid@intellipaat.com')
INSERT INTO EMPLOYEE VALUES
(3,'Adarsh',103,'adarsh@intellipaat.com')
INSERT INTO EMPLOYEE VALUES
(4,'Vaibhav',104,'vaibhav@intellipaat.com')
-These are the duplicate rows:
INSERT INTO EMPLOYEE VALUES (5,'Tarun',101,'tarun@intellipaat.com')
INSERT INTO EMPLOYEE VALUES (6,'Sabid',102,'sabid@intellipaat.com')
```

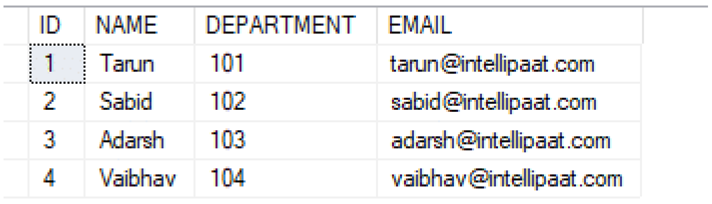


| ID | NAME | DEPARTMENT | EMAIL |
|---|---|---|---|
| 1 | Tarun | 101 | tarun@intellipaat.com |
| 2 | Sabid | 102 | sabid@intellipaat.com |
| 3 | Adarsh | 103 | adarsh@intellipaat.com |
| 4 | Vaibhav | 104 | vaibhav@intellipaat.com |
| 5 | Tarun | 101 | tarun@intellipaat.com |
| 6 | Sabid | 102 | sabid@intellipaat.com |

We can see the duplicate rows in the above table.

```
DELETE e1 FROM EMPLOYEE e1, EMPLOYEE e2 WHERE e1.name = e2.name AND
e1.id > e2.id
```

The SQL query above will delete the rows, where the name fields are duplicated, and it will retain only those unique rows in which the names are unique and the ID fields are the lowest, i.e., the rows with IDs 5 and 6 are deleted, while the rows with IDs 1 and 2 are retained.



| ID | NAME | DEPARTMENT | EMAIL |
|---|---|---|---|
| 1 | Tarun | 101 | tarun@intellipaat.com |
| 2 | Sabid | 102 | sabid@intellipaat.com |
| 3 | Adarsh | 103 | adarsh@intellipaat.com |
| 4 | Vaibhav | 104 | vaibhav@intellipaat.com |

## 69. Can you identify the employee who has the third-highest salary from the given employee table (with salary-related data)?

Consider the following employee table. In the table, Sabid has the third-highest salary (60,000).

| Name | Salary |
|---|---|
| Tarun | 70,000 |
| Sabid | 60,000 |
| Adarsh | 30,000 |
| Vaibhav | 80,000 |

Below is a simple query to find out which employee who has the third-highest salary. The functions RANK, DENSE RANK, and ROW NUMBER are used to obtain the increasing integer value by imposing the ORDER BY clause in the SELECT statement, based on the ordering of the rows. The ORDER BY clause is necessary when the RANK, DENSE RANK, or ROW NUMBER functions are used. On the other hand, the PARTITION BY clause is optional.

```
WITH CTE AS
(
    SELECT Name, Salary, RN = ROW_NUMBER() OVER (ORDER BY Salary
DESC) FROM EMPLOYEE
)
SELECT Name, Salary FROM CTE WHERE RN =3
```

## 70. What is the difference between HAVING and WHERE clauses?

The main difference between the 'HAVING' and 'WHERE' clauses in SQL is that the 'WHERE' clause operates on individual rows of data, while the 'HAVING' clause is used to filter aggregated data. The **'WHERE' clause** cannot be used with aggregate functions, whereas the 'HAVING' clause specifically filters results based on aggregate conditions.

Let us consider the employee table below.

| Name | Department | Salary |
|---|---|---|
| Tarun | Production | 50,000 |
| Tarun | Testing | 60,000 |
| Sabid | Marketing | 70,000 |
| Adarsh | Production | 80,000 |
| Vaibhav | Testing | 90,000 |

The following will select the data on a row-by-row basis:

```
SELECT Name, Salary FROM Employee WHERE Salary >=50000
```

**Output:**

| Name | Salary |
|---|---|
| Tarun | 50,000 |
| Tarun | 60,000 |
| Sabid | 70,000 |
| Adarsh | 80,000 |
| Vaibhav | 90,000 |

The HAVING clause, on the other hand, operates on the aggregated results.

```
SELECT Department, SUM(Salary) AS total FROM Employee GROUP BY
Department
```

**Output:**

| Department | Total |
|---|---|
| Marketing | 70,000 |
| Production | 130,000 |
| Testing | 150,000 |

Now, let us see the output when we apply HAVING to the above query.

```
SELECT Department, SUM(Salary) AS total FROM Employee GROUP BY
Department HAVING SUM(Salary)>70000
```

**Output:**

| Department | Total |
|---|---|
| Production | 130,000 |
| Testing | 150,000 |

## 71. Explain database white box testing and black box testing.

The white box testing method mainly deals with the internal structure of a particular database, where users hide specification details. The white box testing method involves the following:

- As the coding error can be detected by testing the white box, it can eliminate internal errors.
- To check for the consistency of the database, it selects the default table values.
- This method verifies the referential integrity rule.
- It helps perform the module testing of database functions, triggers, views, and SQL queries.

The black box testing method generally involves interface testing and database integration. The black box testing method involves the following:

- Mapping details
- Verification of incoming data
- Verification of outgoing data from the other query functions

## 72. How can you create empty tables with the same structure as another table?

This can be achieved by fetching the records of one table into a new table using the INTO operator while fixing a WHERE clause to be false for all records. In this way, SQL prepares the new table with a duplicate structure to accept the fetched records. However, there are no records that will get fetched due to the WHERE clause in action. Therefore, nothing is inserted into the new table, thus creating an empty table.

```
SELECT * INTO Students_copy
FROM Students WHERE 1 = 2;
```