**SQL Question**

# Q. TRUNCATE TABLE ?

# ANS. SQL statement is used to remove all rows (complete data) from a table. It is similar to the DELETE statement with no WHERE clause.

#### Q. TRUNCATE TABLE Vs DELETE TABLE

Truncate table is faster and uses lesser resources than DELETE TABLE command.

#### Q..TRUNCATE TABLE Vs DROP TABLE

Drop table command can also be used to delete complete table but it deletes table structure too. TRUNCATE TABLE doesn't delete the structure of the table.

# Q.SELECT

SQL command is used to query the database and retrieve selected data that follow the conditions we want.

In simple words, we can say that the select statement used to query or retrieve data from a table in the database.

Let's see the syntax of select statement.

**SELECT** expressions  **FROM** tables  **WHERE** conditions;

# Q.SQL SELECT UNIQUE

Actually, there is no difference between DISTINCT and UNIQUE.

In simple words, we can say that SELECT UNIQUE statement is used to retrieve a unique or distinct element from the table.

**SELECT** **UNIQUE** column\_name  **FROM** table\_name;

**SELECT** **DISTINCT** column\_name ,column\_name  **FROM**  table\_name;

**Q.**SQL WITH AND AS CLAUSE

|  |  |
| --- | --- |
| This syntax is for SQL WITH clause using a single sub-query alias.  **WITH** <alias\_name> **AS** (sql\_sub-query\_statement)   1. **SELECT** column\_list **FROM** <alias\_name> [**table** **name**]  Q.SQL ORDER BY Clause The SQL ORDER BY clause is used for sorting data in ascending and descending order based on one or more columns.  Some databases sort query results in ascending order by default.  SQL ORDER BY syntax:   1. **SELECT** expressions  **FROM** tables  **WHERE** conditions  **ORDER** **BY** expression [**ASC** | **DESC**];   5.6M Q.SQL JOIN As the name shows, JOIN means *to combine something*. In case of SQL, JOIN means **"to combine two or more tables"**.  **ANSI standard SQL** defines five types of JOIN :   1. inner join, 2. left outer join, 3. right outer join, 4. full outer join, and 5. cross join.   **Q.**SQL KEY  PRIMARY   |  | | --- | | A column or columns is called **primary key (PK)** that *uniquely identifies each row in the table*.  If you want to create a primary key, you should define a PRIMARY KEY constraint when you create or modify a table.  When multiple columns are used as a primary key, it is known as **composite primary key**.  In designing the composite primary key, you should use as few columns as possible. It is good for storage and performance both, the more columns you use for primary key the more storage space you require.  Inn terms of performance, less data means the database can process faster. Points to remember for primary key:  * Primary key enforces the entity integrity of the table. * Primary key always has unique data. * A primary key length cannot be exceeded than 900 bytes. * A primary key cannot have null value. * There can be no duplicate value for a primary key. * A table can contain only one primary key constraint. **MySQL:** |   Composite Key A composite key is a combination of two or more columns in a table that can be used to uniquely identify each row in the table when the columns are combined uniqueness is guaranteed, but when it taken individually it does not guarantee uniqueness.  Sometimes more than one attributes are needed to uniquely identify an entity. A primary key that is made by the combination of more than one attribute is known as a composite key.  In other words we can say that:   1. **CREATE** **TABLE** SAMPLE\_TABLE 2. (COL1 **integer**, 3. COL2 **varchar**(30), 4. COL3 **varchar**(50), 5. **PRIMARY** **KEY** (COL1, COL2));   **Ne** Unique Key in SQL A unique key is a set of one or more than one fields/columns of a table that uniquely identify a record in a database table.  You can say that it is little like primary key but it can accept only one null value and it cannot have duplicate values.  The unique key and primary key both provide a guarantee for uniqueness for a column or a set of columns. Alternate Key in SQL Alternate key is a secondary key it can be simple to understand by an example:  Let's take an example of student it can contain NAME, ROLL NO., ID and CLASS.  Here ROLL NO. is primary key and rest of all columns like NAME, ID and CLASS are alternate keys.  If a table has more than one candidate key, one of them will become the primary key and rest of all  are called alternate keys. IN vs. EXISTS This operator allows us to specify multiple values along with the [WHERE clause](https://www.javatpoint.com/sql-where). It reduces the use of **multiple OR conditions** in [SELECT](https://www.javatpoint.com/sql-select), [INSERT](https://www.javatpoint.com/sql-insert), [UPDATE](https://www.javatpoint.com/sql-update), and [DELETE](https://www.javatpoint.com/sql-delete) queries; that's why it is also called the shorthand for multiple OR conditions.  In this operator, the inner query executes first, and the result obtained is used by the outer query to display the output. It should be remembered that the inner query is executed only once. The **IN operator** has the following syntax:  **SELECT** column\_name(s)   **FROM** table\_name   **WHERE** column\_name IN (value1, value2, - - - - );  **xt**  **St** |
|  |

## EXISTS Operator

EXISTS is a Boolean operator which **checks the subquery result and returns an either TRUE or FALSE value**. It is used in combination with subquery and checks whether a row is returned through this subquery or not. This operator returns **TRUE** if the subquery returns single or multiple records. Otherwise, it gives a **FALSE** result when no records are returned.

When the EXISTS operator detects the first true event, it automatically terminates for further processing. This feature enhances the query's efficiency. We can use the EXISTS operator with SELECT, UPDATE, DELETE, and INSERT statements. The following is the **syntax of EXISTS operator**:

**SELECT** col\_names  **FROM** tab\_name    **WHERE** [NOT] EXISTS (

**SELECT** col\_names    **FROM** tab\_name     **WHERE** condition  );

## HAVING Clause

HAVING clause in MySQL **used in conjunction with GROUP BY** clause enables us to specify conditions that filter which group results appear in the result. It returns only those values from the groups in the final result that fulfills certain conditions. We can also use the WHERE and HAVING clause together during selection. In this case, WHERE clause first filters the individual rows, then rows are grouped, performs aggregate calculations, and at last HAVING clause filter the groups.

This clause places conditions on groups created by the GROUP BY clause. It behaves like the WHERE clause when the SQL statement does not use the GROUP BY keyword. We can use the aggregate (group) functions such as [SUM](https://www.javatpoint.com/sql-select-sum), MIN, MAX, AVG, and [COUNT](https://www.javatpoint.com/sql-select-count) only with two clauses: SELECT and HAVING.

SQL-SYNTEXT **SELECT** column\_lists,   aggregate\_function (expression)  **FROM** table\_name  **WHERE**  conditions  **GROUP** **BY** column\_lists   **HAVING** condition;

### Q.What is an Index in SQL?

An index is a disc structure associated with a table or view that speeds up row retrieval. t is used to increase the performance and allow faster retrieval of records from the table. Indexing reduces the number of data pages we need to visit to find a particular data page. It also has a unique value meaning that the index cannot be duplicated. An index creates an entry for each value which makes it faster to retrieve data.

**For example:** Suppose we have a book which carries the details of the countries. If you want to find out information about India, why will you go through every page of that book? You could directly go to the index. Then from the index, you can go to that particular page where all the information about India is given.

[MySQL](https://www.javatpoint.com/mysql-tutorial) allowed index type **BTREE**

### Q.Different types of indexes in SQL?

### clustered index- A clustered index is actually a table where the data for the rows are stored. It defines the order of the table data based on the key values that can be sorted in only one way. In the database, each table can have only one clustered index. In a relational database, if the table column contains a primary key or unique key, [MySQL](https://www.javatpoint.com/mysql-tutorial) allows you to create a clustered index named ****PRIMARY**** based on that specific column.

### unique index -UNIQUE INDEX is used to enforce the uniqueness of values in single or multiple columns. We can create more than one unique index in a single table. For creating a unique index, the user has to check the data in the column because the unique indexes are used when any column of the table has unique values. This indexing does not allow the field to have duplicate values if the column is unique indexed. A unique index can be applied automatically when a primary key is defined.

### non-clustered index -The indexes other than PRIMARY indexes (clustered indexes) are called non-clustered indexes. We know that clustered indexes are created automatically when primary keys are generated, and non-clustered indexes are created when multiple joins conditions and various filters are used in the query. The non-clustered index and table data are both stored in different places. It cannot be able to alter the physical order of the table and maintains the logical order of data.

The purpose of creating a non-clustered index is for searching the data. Its best example is a book where the content is written in one place, and the index is at a different place. We can create 0 to 249 non-clustered indexes in each table. The non-clustered indexing improves the performance of the queries which use keys without assigning the primary key.

### Q. What is the difference between clustered and non-clustered indexes in SQL?

Indexing is a method to get the requested data very fast. There are mainly two types of indexes in SQL, clustered index and non-clustered index. The differences between these two indexes are very important from an SQL performance perspective. The following comparison chart explains their main differences:

|  |  |
| --- | --- |
| **Clustered Index** | **Non-Clustered Index** |
| A clustered index is a table or view where the data for the rows are stored. In a relational database, if the table column contains a primary key, MySQL automatically creates a clustered index named PRIMARY. | The indexes other than PRIMARY indexes (clustered indexes) are called non-clustered indexes. It has a structure separate from the data row. The non-clustered indexes are also known as secondary indexes. |
| Clustered indexes store the data information and the data itself. | Non-clustered indexes stores only the information, and then it will refer you to the data stored in clustered data. |
| There can only be one clustered index per table. | There can be one or more non-clustered indexes in a table. |
| A clustered index determines how data is stored physically in the table. Therefore, reading from a clustered index is faster. | It creates a logical ordering of data rows and uses pointers for accessing the physical data files. Therefore, reading from a clustered index is slower. |
| A clustered index always contains an index id of 0. | A non-clustered index always contains an index id>0. |

### Q.What are the different types of joins in SQL?

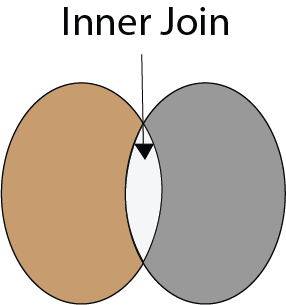
|\\Joins are used to merge two tables or retrieve data from tables. It depends on the relationship between tables. According to the ANSI standard, the following are the different types of joins used in SQL:

* INNER JOIN
* SELF JOIN
* LEFT OUTER JOIN
* RIGHT OUTER JOIN
* FULL OUTER JOIN
* CROSS JOIN

### Q.INNER JOIN in SQL?

### Inner join returns only those records from the tables that match the specified condition and hides other rows and columns. In simple words, it fetches rows when there is at least one match of rows between the tables is found. INNER JOIN keyword joins the matching records from two tables. It is assumed as a default join, so it is optional to use the INNER keyword with the query.

The below visual representation explain this join more clearly:

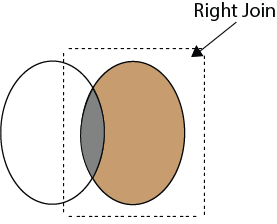


1. Exp SELECT column\_lists  FROM table1    INNER JOIN table2 ON join\_condition1    INNER JOIN table3 ON join\_condition2

### What is the Right JOIN in SQL?

The Right join is used to retrieve all rows from the right-hand table and only those rows from the other table that fulfilled the join condition. It returns all the rows from the right-hand side table even though there are no matches in the left-hand side table. If it finds unmatched records from the left side table, it returns a Null value. This join is also known as Right Outer Join.

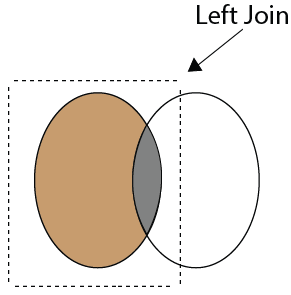
The below visual representation explain this join more clearly:



### Q.Left Join in SQL?

The Left Join is used to fetch all rows from the left-hand table and common records between the specified tables. It returns all the rows from the left-hand side table even though there are no matches on the right-hand side table. If it will not find any matching record from the right side table, then it returns null. This join can also be called a Left Outer Join.

The following visual representation explains it more clearly:



### Q.self-join and what is the requirement of self-join?

A SELF JOIN is used to join a table with itself. This join can be performed using table aliases, which allow us to avoid repeating the same table name in a single sentence. It will throw an error if we use the same table name more than once in a single query without using table aliases.

A SELF JOIN is required when we want to combine data with other data in the same table itself. It is often very useful to convert a hierarchical structure to a flat structure.

Exp a table ki primay ki a tbale me other key me use ho rahi ho;

|  |  |  |
| --- | --- | --- |
| Empid | Name | ManageriD |
| 1 | Rahul | 3 |
| 2 | Jay | 3 |
| 3 | Sonam | 4 |
| 4 | Kunal | 5 |
| 5 | Ram | 6 |
| 6 | Rani | Null |
| 7 | Veeru | 6 |

### Q."TRIGGER" in SQL?

**Trigger:** A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

**Syntax:**

create trigger [trigger\_name]

[before | after]

{insert | update | delete}

on [table\_name]

[for each row]

[trigger\_body]

**Explanation of syntax:**

1. create trigger [trigger\_name]: Creates or replaces an existing trigger with the trigger\_name.
2. [before | after]: This specifies when the trigger will be executed.
3. {insert | update | delete}: This specifies the DML operation.
4. on [table\_name]: This specifies the name of the table associated with the trigger.
5. [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.
6. [trigger\_body]: This provides the operation to be performed as trigger is fired

**BEFORE and AFTER of Trigger:**  
BEFORE triggers run the trigger action before the triggering statement is run.  
AFTER triggers run the trigger action after the triggering statement is run.

**Example:**  
Given Student Report Database, in which student marks assessment is recorded. In such schema, create a trigger so that the total and average of specified marks is automatically inserted whenever a record is insert.

Here, as trigger will invoke before record is inserted so, BEFORE Tag can be used.

**Suppose the database Schema –**

mysql> desc Student;

+-------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+----------------+

| tid | int(4) | NO | PRI | NULL | auto\_increment |

| name | varchar(30) | YES | | NULL | |

| subj1 | int(2) | YES | | NULL | |

| subj2 | int(2) | YES | | NULL | |

| subj3 | int(2) | YES | | NULL | |

| total | int(3) | YES | | NULL | |

| per | int(3) | YES | | NULL | |

+-------+-------------+------+-----+---------+----------------+

SQL Trigger to problem statement.

create trigger stud\_marks

before INSERT

on

Student

for each row

set Student.total = Student.subj1 + Student.subj2 + Student.subj3, Student.per = Student.total \* 60 / 100;

Above SQL statement will create a trigger in the student database in which whenever subjects marks are entered, before inserting this data into the database, trigger will compute those two values and insert with the entered values. i.e.,

mysql> insert into Student values(0, "ABCDE", 20, 20, 20, 0, 0);

Query OK, 1 row affected (0.09 sec)

mysql> select \* from Student;

+-----+-------+-------+-------+-------+-------+------+

| tid | name | subj1 | subj2 | subj3 | total | per |

+-----+-------+-------+-------+-------+-------+------+

| 100 | ABCDE | 20 | 20 | 20 | 60 | 36 |

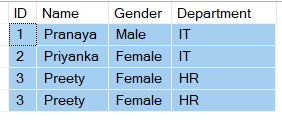
+-----+-------+-------+-------+-------+-------+------+

### Q.What are the set operators in SQL?

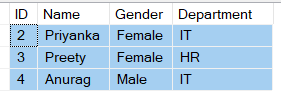
We use the set operators to merge data from one or more tables of the same kind. Although the set operators are like SQL joins, there is a significant distinction. SQL joins combine columns from separate tables, whereas SQL set operators combine rows from different queries. SQL queries that contain set operations are called compound queries. The set operators in SQL are categories into four different types:

1. **UNION**: Combine two or more result sets into a single set, without duplicates.
2. **UNION ALL**: Combine two or more result sets into a single set, including all duplicates.
3. **INTERSECT**: Takes the data from both result sets which are in common.
4. **EXCEPT**: Takes the data from the first result set, but not in the second result set (i.e. no matching to each other) unique data access exp : jo data a table me h but b table me nahi h.

**SELECT \* FROM TableA**

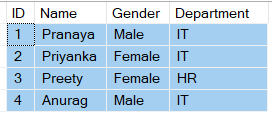


**SELECT \* FROM TableB**



UNION

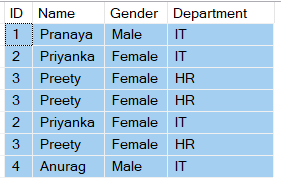
**SELECT ID, Name, Gender, Department FROM TableA**  
**UNION**  
**SELECT ID, Name, Gender, Department FROM TableB**

**Result:**   


UNION ALL

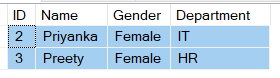
**SELECT ID, Name, Gender, Department FROM TableA**  
**UNION ALL**  
**SELECT ID, Name, Gender, Department FROM TableB**

**Result:**



##### **INTERSECT**

**SELECT ID, Name, Gender, Department FROM TableA**  
**INTERSECT**  
**SELECT ID, Name, Gender, Department FROM TableB**

**Result:**   


##### **EXCEPT**

**SELECT ID, Name, Gender, Department FROM TableA**  
**EXCEPT**  
**SELECT ID, Name, Gender, Department FROM TableB**

**Result:**   
Differences between UNION EXCEPT and INTERSECT Operators    
If you want the rows that are present in Table B but not in Table A, reverse the queries.

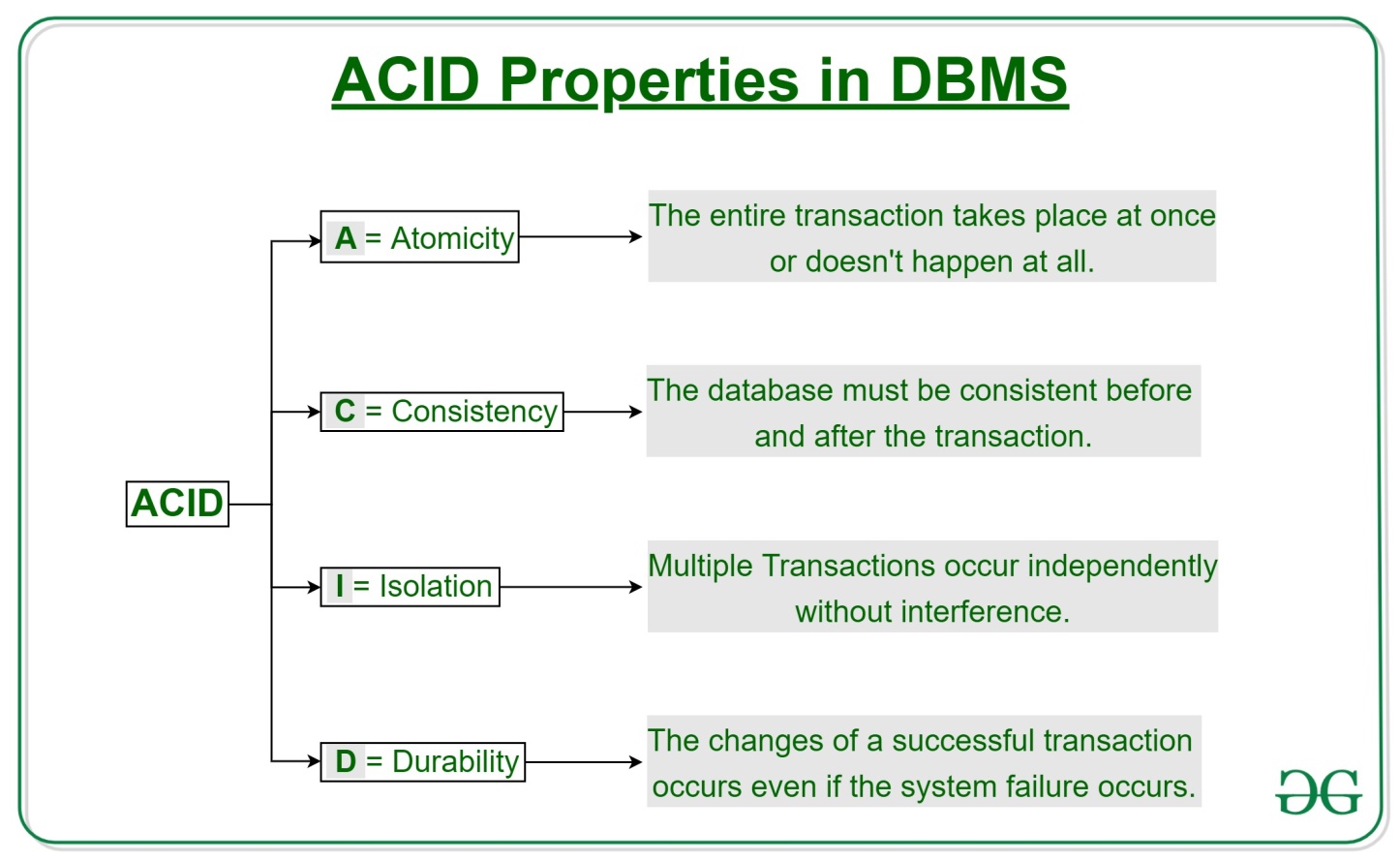
**SELECT ID, Name, Gender, Department FROM TableB**  
**EXCEPT**  
**SELECT ID, Name, Gender, Department FROM TableA**

###### **Result:  Differences between UNION EXCEPT and INTERSECT Operators**

### Q.Constraint

simpaly condition on the colum is constraint. Like as unique ,not null , primary, forgein key, check ,default

### Q.ACID property in a database?



ACID properties are meant for the transaction that goes through a different group of tasks.

**Atomicity:** It ensures that all statements or operations within the transaction unit must be executed successfully. If one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. Its main features are COMMIT, ROLLBACK, and AUTO-COMMIT.

**Consistency: start to end data dekhan .**

**Exp a(100) b (300) total (400). Txn end a (50) or b (350) total 400 this is consistency;**

**Isolation:** This property guarantees that the concurrent property of execution in the transaction unit must be operated independently.

All transaction working independently.

**Durability:** This property guarantees that once a transaction has been committed, it persists permanently even if the system crashes, power loss, or failed.

### Q.What is the usage of the NVL() function?

The function returns the value of the second parameter if the first parameter is NULL. If the first parameter is anything other than NULL, it is left unchanged. This function is used in Oracle, not in SQL and MySQL. Instead of NVL() function, MySQL have IFNULL() and SQL Server have ISNULL() function.

### Q.DISTINCT statement? What is its use?

The DISTINCT keyword is used to ensure that the fetched value always has unique values. It does not allow to have duplicate values. The DISTINCT keyword is used with the SELECT statement and retrieves different values from the table's column. We can use it with the help of the following syntax:

1. SELECT DISTINCT column\_lists FROM table\_name WHERE [condition];