MY SQL Tutorial

Define the following terms.

Data

- Fact that can be recorded or stored.
- E.g. Person Name, Age, Gender and Weight etc.

Information

- When data is processed, organized, structured or presented in a given context so as to make it useful, it is called information.

Database

- A Database is a collection of inter-related (logically-related) data.
- E.g. Books Database in Library, Student Database in University etc.

DBMS (Database Management System)

- A database management system is a collection of inter-related data and set of programs to manipulate those data.
- DBMS = Database + Set of programs
- E.g. MS SQL Server, Oracle, My SQL, SQLite, MongoDB etc.

Field

- A field is a character or group of characters that have a specific meaning.
- It is also called a data item. It is represented in the database by a value.
- For Example customer id, name, society and city are all fields for customer Data.

Record

- A record is a collection of logically related fields.
- For examples, collection of fields (id, name, address & city) forms a record for customer.

MySQL

- MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by **Oracle Company**.
- Our MySQL tutorial includes all topics of MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc.

What is MySQL?

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications.

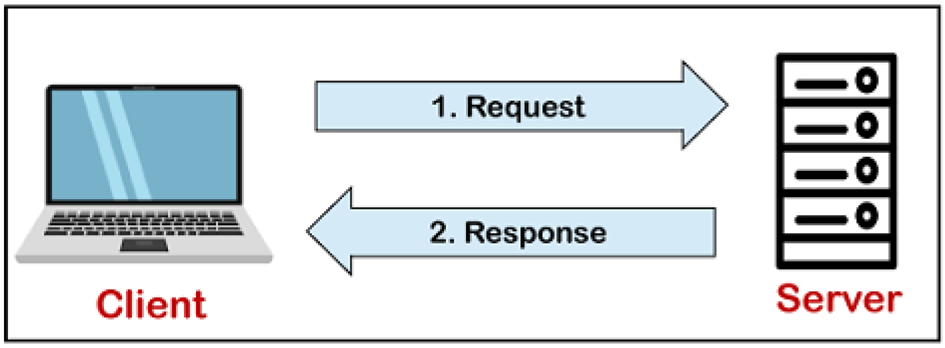
It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
- It provides the Referential Integrity between rows or columns of various tables.
- It allows us to updates the table indexes automatically.
- It uses many SQL queries and combines useful information from multiple tables for the end-users.

How MySQL Works?

MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services. Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched. The process of MySQL environment is the same as the client-server model.



The core of the MySQL database is the MySQL Server. This server is available as a separate program and responsible for handling all the database instructions, statements, or commands. The working of MySQL database with MySQL Server are as follows:

1. MySQL creates a database that allows you to build many tables to store and manipulate data and defining the relationship between each table.
2. Clients make requests through the GUI screen or command prompt by using specific SQL expressions on MySQL.
3. Finally, the server application will respond with the requested expressions and produce the desired result on the client-side.

A client can use any MySQL GUI. But, it is making sure that your GUI should be lighter and user-friendly to make your data management activities faster and easier. Some of the most widely used MySQL GUIs are MySQL Workbench, SequelPro, DBVisualizer, and the Navicat DB Admin Tool. Some GUIs are commercial, while some are free with limited functionality, and some are only compatible with MacOS. Thus, you can choose the GUI according to your needs.

Explain different database users.

There are four different database users.

Application programmers

- These users are computer professionals who write application programs using some tools. E.g. Software developers

Sophisticated users

- These users interact with system without writing program. They form their request in a database query language. E.g. Analyst.

Specialized users

- These users write specialized database applications that do not fit into the traditional data processing framework. E.g. Database Administrator.

Naive users

- These users are unsophisticated users who have very less knowledge of database system.

- These users interact with the system by using one of the application programs that have been written previously.
- Examples, e.g. Clerk in bank

MySQL Features

MySQL is a relational database management system (RDBMS) based on the SQL (Structured Query Language) queries. It is one of the most popular languages for accessing and managing the records in the table. MySQL is open-source and free software under the GNU license. Oracle Company supports it.

The following are the most important features of MySQL:

1. **Relational Database Management System (RDBMS)**

   MySQL is a relational database management system. This database language is based on the SQL queries to access and manage the records of the table.

2. **Easy to use**

   MySQL is easy to use. We have to get only the basic knowledge of SQL. We can build and interact with MySQL by using only a few simple SQL statements.

3. **It is secure**

   MySQL consists of a solid data security layer that protects sensitive data from intruders. Also, passwords are encrypted in MySQL.

4. **Client/ Server Architecture**

   MySQL follows the working of a client/server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they can query data, save changes, etc.

5. **Free to download**

   MySQL is free to use so that we can download it from MySQL official website without any cost.

6. **It is scalable**

   MySQL supports multi-threading that makes it easily scalable. It can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, we can increase this number to a theoretical limit of 8 TB of data.

7. **Speed**

   MySQL is considered one of the very fast database languages, backed by a large number of the benchmark test.

8. **High Flexibility**

   MySQL supports a large number of embedded applications, which makes MySQL very flexible.

9. **Compatible on many operating systems**

   MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

10. **Allows roll-back**

    MySQL allows transactions to be rolled back, commit, and crash recovery.

11. **Memory efficiency**

    Its efficiency is high because it has a very low memory leakage problem.

12. **High Performance**

    MySQL is faster, more reliable, and cheaper because of its unique storage engine architecture. It provides very high-performance results in comparison to other databases without losing an essential functionality of the software. It has fast loading utilities because of the different cache memory.

13. **High Productivity**

MySQL uses Triggers, Stored procedures, and views that allow the developer to give higher productivity.

14. **Platform Independent**

    It can download, install, and execute on most of the available operating systems.

15. **Partitioning**

    This feature improves the performance and provides fast management of the large database.

16. **GUI Support**

    MySQL provides a unified visual database graphical user interface tool named "**MySQL Workbench**" to work with database architects, developers, and Database Administrators. MySQL Workbench provides SQL development, data modeling, data migration, and comprehensive administration tools for server configuration, user administration, backup, and many more. MySQL has a fully GUI supports from MySQL Server version 5.6 and higher.

17. **Dual Password Support**

    MySQL version 8.0 provides support for dual passwords: one is the current password, and another is a secondary password, which allows us to transition to the new password.

Disadvantages / Drawbacks of MySQL

Following are the few disadvantages of MySQL:

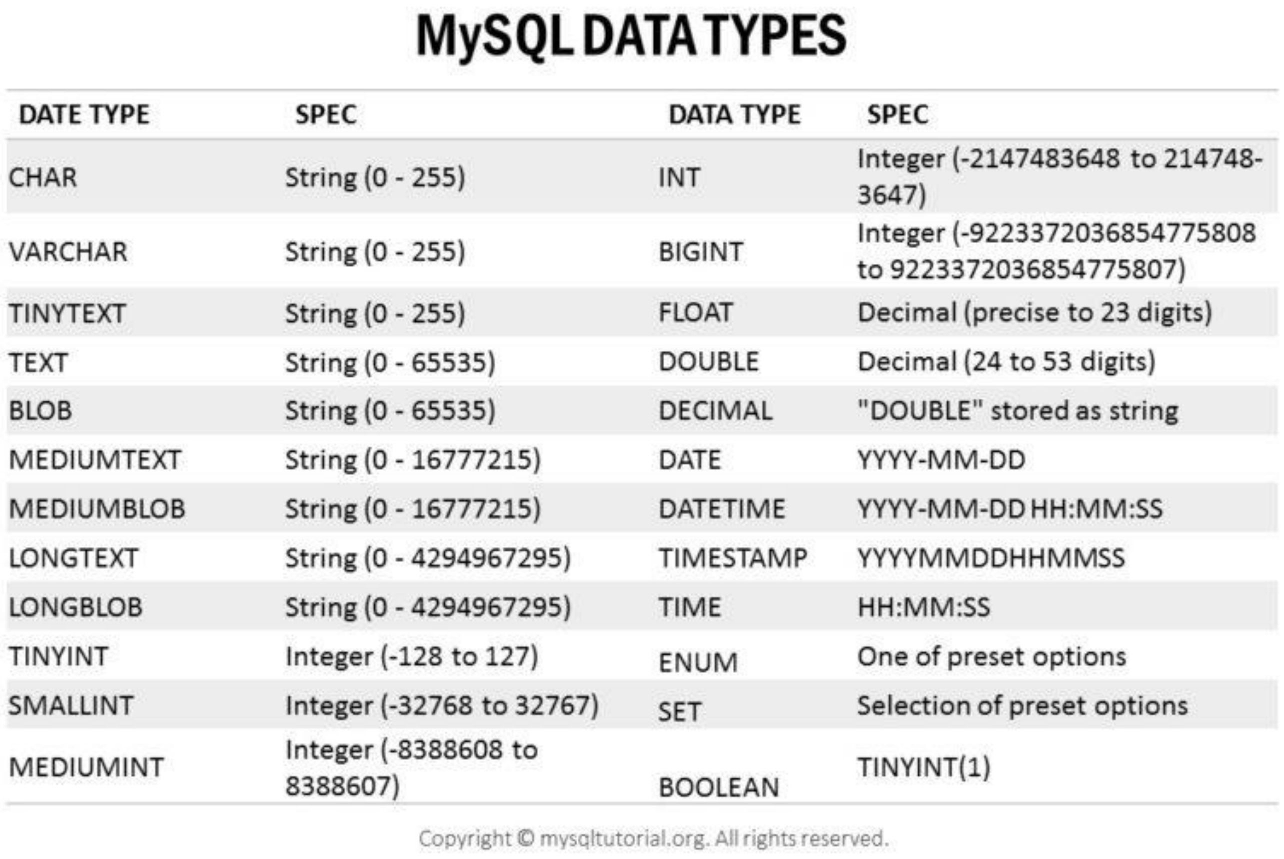
- MySQL version less than 5.0 doesn't support ROLE, COMMIT, and stored procedure.
- MySQL does not support a very large database size as efficiently.
- MySQL doesn't handle transactions very efficiently, and it is prone to data corruption.
- MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
- MySQL doesn't support SQL check constraints.

MySQL Data Types

A Data Type specifies a particular type of data, like integer, floating points, Boolean, etc. It also identifies the possible values for that type, the operations that can be performed on that type, and the way the values of that type are stored. In MySQL, each database table has many columns and contains specific data types for each column.

Determining the data type in MySQL with the following characteristics:

- The type of values (fixed or variable) it represents.
- The storage space it takes is based on whether the values are a fixed-length or variable length.
- Its values can be indexed or not.
- How MySQL performs a comparison of values of a particular data type.



MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

Categorizing MySQL Data Types

MySQL numeric data types

In MySQL, you can find all SQL standard numeric types including exact number data type and approximate numeric data types including integer, fixed-point and floating-point. In addition, MySQL also has BIT data type for storing bit values. Numeric types can be signed or unsigned except for the BIT type.

The following table shows the summary of numeric types in MySQL:

Numeric Types	Description
TINYINT	A very small integer
SMALLINT	A small integer
MEDIUMINT	A medium-sized integer
INT	A standard integer
BIGINT	A large integer
DECIMAL	A fixed-point number
FLOAT	A single-precision floating point number
DOUBLE	A double-precision floating point number
BIT	A bit field

MySQL Boolean data type

MySQL does not have the built-in BOOLEAN or BOOL data type. To represent 8oolean values, MySQL uses the smallest integer type which is TINYINT(1). In other words, BOOLEAN and BOOL are synonyms for TINYINT(1).

MySQL String data types

In MySQL, a string can hold anything from plain text to binary data such as images or files. Strings can be compared and searched based on pattern matching by using the LIKE operator, regular expression, and full-text search.

The following table shows the string data types in MySQL:

String Types	Description
CHAR	A fixed-length nonbinary (character) string
VARCHAR	A variable-length non-binary string
BINARY	A fixed-length binary string
VARBINARY	A variable-length binary string
TINYBLOB	A very small BLOB (binary large object)
BLOB	A small BLOB
MEDIUMBLOB	A medium-sized BLOB
LONGBLOB	A large BLOB
TINYTEXT	A very small non-binary string
TEXT	A small non-binary string
MEDIUMTEXT	A medium-sized non-binary string
LONGTEXT	A large non-binary string
ENUM	An enumeration; each column value may be assigned one enumeration member
SET	A set; each column value may be assigned zero or more SET members

MySQL date and time data types

MySQL provides types for date and time as well as the combination of date and time. In addition, MySQL supports the timestamp data type for tracking the changes in a row of a table. If you just want to store years without dates and months, you can use the YEAR data type.

The following table illustrates the MySQL date and time data types:

Date and Time Types	Description
DATE	A date value in CCYY-MM-DD format

Date and Time Types	Description
TIME	A time value in hh:mm:ss format
DATETIME	A date and time value inCCYY-MM-DD hh:mm:ssformat
TIMESTAMP	A timestamp value in CCYY-MM-DD hh:mm:ss format
YEAR	A year value in CCYY or YY format

Note: MySQL also supports Spatial & JSON Data Type

What is Integrity Constraints?

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.
- Various Integrity Constraints are:

1. Check
   - ✓ This constraint defines a business rule on a column. All the rows in that column must satisfy this rule.
   - ✓ Limits the data values of variables to a specific set, range, or list ofvalues.
   - ✓ The constraint can be applied for a single column or a group of columns.
   - ✓ E.g. value of SPI should be between 0 to 10.
2. Not null
   - ✓ This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.
   - ✓ E.g. name column should have some value.
3. Unique
   - ✓ This constraint ensures that a column or a group of columns in each row have a distinct (unique) value.
   - ✓ A column(s) can have a null value but the values cannot be duplicated.
   - ✓ E.g enrollmentno column should have unique value.
4. Primary key
   - ✓ This constraint defines a column or combination of columns which uniquely identifies each row in thetable.

- ✓ Primary key = Unique key + Not null
- ✓ E.g enrollmentno column should have unique value as well as can’t be null.

5. Foreign key
   - ✓ A referential integrity constraint (foreign key) is specified between two tables.
   - ✓ In the referential integrity constraints, if a foreign key column in table 1 refers to the primary key column of table 2, then every value of the foreign key column in table 1 must be null or be available in primary key column of table 2.

Explain keys.

Super key

- A super key is a set of one or more attributes (columns) that allow us to identify each tuple (records) uniquely in a relation (table).
- For example, the enrollment_no, roll_no, semester with department_name of a student is sufficient to distinguish one student tuple from another. So {enrollment_no} and {roll_no, semester, department_name} both are super key.

Candidate key

- Candidate key is a super key for which no proper subset is a super key.
- For example, combination of roll_no, semester and department_name is sufficient to distinguish one student tuple from another. But either roll_no or semester or department_name alone or combination of any two columns is not sufficient to distinguish one student tuple from another. So {roll_no, semester, department_name} is candidate key.
- Every candidate key is super key but every super key may not candidate key.

Primary key

- A Primary key is a candidate key that is chosen by database designer to identify tuples uniquely in a relation.

Alternate key

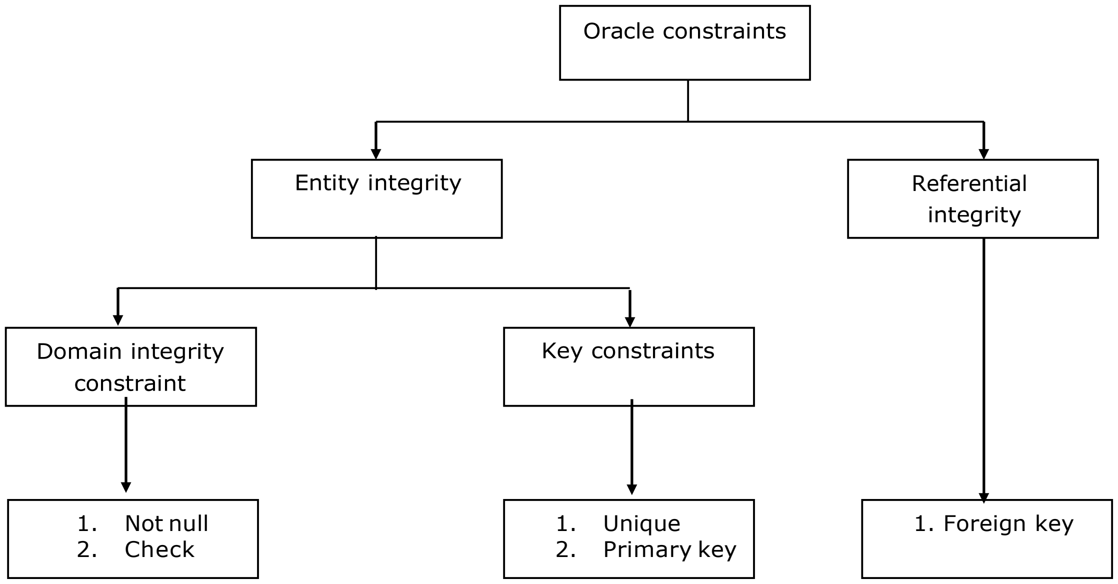
- An Alternate key is a candidate key that is not chosen by database designer to identify tuples uniquely in a relation.

Foreign key

- A foreign key is a set of one or more attributes whose values are derived from the primary key attribute of another relation.

What is constraint? Explain types of constraints. *OR* What are integrity constraints? Explain various types of integrity constraints with suitable example.

- A constraint is a rule that restricts the values that may be present in the database.
- Constraints can be mainly classified in to two categories.
    1. Entity integrity constraints
    2. Referential integrity constraints



Not Null constraint

- A null value indicates 'not applicable', 'missing', or 'not known'.
- A null value is distinct from zero or blank space.
- A column, defined as a not null, cannot have a null value.
- Such a column becomes a mandatory (compulsory) column and cannot be left empty for any record.
- **Syntax** : ColumnName datatype (size) NOT NULL
- **Example** :

```
create table Account (ano int,
                      Balance decimal(8,2) NOT NULL,
                      Branch varchar(10));
```

- Now, if we insert NULL value to Balance column then it will generate an error.

Check constraint

- The check constraint is used to implement business rule. So, it is also called business rule constraint.
- Example of business rule: A balance in any account should not be negative.
- Business rule define a domain for a particular column.

- The check constraint is bound to a particular column.
- Once check constraint is implemented, any insert or update operation on that table must follow this constraint.
- If any operation violates condition, it will be rejected.
- **Syntax** *:* ColumnName datatype (size) check(condition)
- **Example** :

```
create table Account (ano int,
                      Balance decimal(8,2) CHECK (balance >= 0),
                      Branch varchar(10));
```

- Any business rule validations can be applied using this constraint.
- A condition must be some valid logical expression.
- A check constraint takes longer time to execute.
- On violation of this constraint, oracle display error message like – "check constraint violated".

Unique Constraint

- Sometime there may be requirement that column cannot contain duplicate values.
- A column, defined as a unique, cannot have duplicate values across all records.
- **Syntax** *:* ColumnName datatype (size) UNIQUE
- **Example** :

```
create table Account (ano int UNIQUE,
                      Balance decimal(8,2),
                      Branch varchar(10));
```

- Though, a unique constraint does not allow duplicate values, NULL values can be duplicated in a column defined as a UNIQUE column.
- A table can have more than one column defined as a unique column.
- If multiple columns need to be defined as composite unique column, then only table level definition is applicable.
- Maximum 16 columns can be combined as a composite unique key in a table.

Primary key Constraint

- A primary key is a set of one or more columns used to identify each record uniquely in a column.
- A single column primary key is called a simple key, while a multi-column primary key is called composite primary key.
- Oracle provides a primary key constraint to define primary key for a table.
- A column, defined as a primary key, cannot have duplicate values across all records and cannot have a null value.
- **Syntax** *:* ColumnName datatype (size) primary key

- **Example** :

```
create table Account (ano int NOT NULL PRIMARY KEY,
                      Balance decimal(8,2),
                      Branch varchar(10));
```

- A primary key constraint is combination of UNIQUE constraint and NOT NULL constraint.
- A table cannot have more than one primary key.
- If multiple columns need to be defined as primary key column, then only table level definition is applicable.
- Maximum 16 columns can be combined as a composite primary key in a table.

Foreign Key Constraint

- A foreign key constraint is also called referential integrity constraint, is specified between two tables.
- This constraint is used to ensure consistency among records of the two tables.
- The table, in which a foreign key is defined, is called a foreign table, detail table or child table.
- The table, of which primary key or unique key is defined, is called a primary table, master table or parent table.
- Restriction on child table :
  - ✓ Child table contains a foreign key. And, it is related to master table.
  - ✓ Insert or update operation involving value of foreign key is not allowed, if corresponding value does not exist in a master table.
- Restriction on master table :
  - ✓ Master table contains a primary key, which is referred by foreign key in child table.
  - ✓ Delete or update operation on records in master table are not allowed, if corresponding records are present in child table.
- **Syntax** *:* ColumnName datatype (size) REFERENCES TableName (ColumnName)
- **Example** :

```
create table Account (ano int,
                      Balance decimal(8,2),
                      BranchID varchar(10) REFERENCES branchs(branchID));
```

- Master table must be exist before creating child table.

Explain DDL, DML, DCL and DQL. *OR* Describe component of SQL.

DDL (Data Definition Language)

- It is a set of SQL commands used to create, modify and delete database objects such as tables, views, indices, etc.
- It is normally used by DBA and database designers.
- It provides commands like:
  - ✓ CREATE: to create objects in a database.
  - ✓ ALTER: to alter the schema, or logical structure of the database.
  - ✓ DROP: to delete objects from the database.
  - ✓ TRUNCATE: to remove all records from the table**.**

CREATE: Create is used to create the database or its objects like table, view, index etc.

- **Create Table**

  The CREATE TABLE statement is used to create a new table in a database.

  Syntax:

```
CREATE TABLE table_name
(
    Column1 Datatype(Size) [ NULL | NOT NULL ],
    Column2 Datatype(Size) [ NULL | NOT NULL ],
  …
);
```

  Example:

```
CREATE TABLE Students
(
    Roll_No int(3) NOT NULL,
    Name varchar(20),
    Subject varchar(20)
);
```

Explanation**:**

- The column should either be defined as NULL or NOT NULL. By default, a column can hold NULL values.
- The NOT NULL constraint enforces a column to NOT accept NULL values. This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

ALTER: ALTER TABLE statement is used to add, modify, or drop columns in a table.

- **Add Column**
  The ALTER TABLE statement in SQL to add new columns in a table.
  Syntax:
  ```
  ALTER TABLE table_name
  ADD Column1 Datatype(Size), Column2 Datatype(Size), … ;
  ```
  Example:
  ```
  ALTER TABLE Students
  ADD Marks int;
  ```

- **Drop Column**
  The ALTER TABLE statement in SQL to drop a column in a table.
  Syntax:
  ```
  ALTER TABLE table_name
  DROP COLUMN column_name;
  ```
  Example:
  ```
  ALTER TABLE Students
  DROP COLUMN Subject;
  ```

- **Modify Column**
  The ALTER TABLE statement in SQL to change the data type/size of a column in a table.
  Syntax:
  ```
  ALTER TABLE table_name
  ALTER COLUMN column_name datatype(size);
  ```
  Example:
  ```
  ALTER TABLE Students
  ALTER COLUMN Roll_No float;
  ```

DROP: Drop is used to drop the database or its objects like table, view, index etc.

- **Drop Table**
  The DROP TABLE statement is used to drop an existing table in a database.
  Syntax:
  ```
  DROP TABLE table_name;
  ```
  Example:
  ```
  DROP TABLE Students;
  ```

TRUNCATE: Truncate is used to remove all records from the table.

Syntax:
```
TRUNCATE TABLE table_name;
```
Example:
```
TRUNCATE TABLE Students;
```

DML (Data Manipulation Language)

- It is a set of SQL commands used to insert, modify and delete data in a database.
- It is normally used by general users who are accessing database via pre-developed applications.
- It provides commands like:
  - ✓ INSERT: to insert data into a table.
  - ✓ UPDATE: to modify existing data in a table.
  - ✓ DELETE: to delete records from a table.

INSERT: The INSERT STATEMENT is used to insert data into a table.

Syntax:

```
INSERT INTO table_name (column1, column2, column3, …)
VALUES (value1, value2, value3, …);
                    OR
INSERT INTO table_name
VALUES (value1, value2, value3, …);
```

Example:

```
INSERT INTO Students (Roll_No,Name,Subject)
VALUES (1,'anil','Maths');
```

UPDATE: The UPDATE STATEMENT is used to modify existing data in a table.

Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2, …
WHERE condition;
```

Example:

```
UPDATE Students
SET Name= 'Mahesh'
WHERE Roll_No=1;
```

DELETE: The DELETE STATEMENT is used to delete records from a table.

Syntax:

```
DELETE FROM table_name
WHERE condition;
```

Example:

```
DELETE FROM Students
WHERE  Subject='Maths';
```

DQL (Data Query Language)

- It is a component of SQL that allows data retrieval from the database.
- It provides command like SELECT. This command is a heart of SQL, and allows data retrieval in different ways.

SELECT: The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

Syntax:

```
SELECT column1, column2, …
FROM table_name
WHERE condition;
            OR
SELECT *
FROM table_name
WHERE condition;
```

Example:

```
SELECT *
FROM Students
WHERE Roll_No>=1;
```

DCL (Data Control Language)

- It is set of SQL commands used to control access to data and database. Occasionally DCL commands are grouped with DML commands.
- It provides commands like:
  - ✓ GRANT: to give access privileges to users on the database.
  - ✓ REVOKE: to withdraw access privileges given to users on the database.

TCL (Transaction Control Language)

- TCL is abbreviation of Transactional Control Language. It is used to manage different transactions occurring within a database.
  - ✓ COMMIT – Saves work done in transactions
  - ✓ ROLLBACK – Restores database to original state since the last COMMIT command in transactions
  - ✓ SAVE TRANSACTION – Sets a save point within a transaction.

Explain Transaction Control Commands. *OR* Explain commit, rollback and savepoint command.

- Transaction Control Command (TCL) is a set of SQL commands that are used to control transactional processing in a database.
- A transaction is logical unit of work that consists of one or more SQL statements, usually a group of Data Manipulation Language (DML) statements.

- Transaction Control Commands (TCL) commands include

1. **COMMIT:**
   - COMMIT command is used to permanently save any transaction into the database.
   - When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.
   - To avoid that, COMMIT is used to mark the changes as permanent.

   Example: Consider following table

Student		
Rollno	**Name**	**SPI**
1	Raju	8
2	Hari	9
3	Mahesh	7

```
BEGIN TRANSACTION t1
DELETE FROM STUDENT WHERE SPI <8;
COMMIT TRANSACTION t1;
```

Output:

Student		
Rollno	**Name**	**SPI**
1	Raju	8
2	Hari	9

Note:
- A transaction is a set of operations performed so that all operations are guaranteed to succeed or fail as one unit.
- If you place the BEGIN TRANSACTION before your SQL statement, the transaction will automatically turn into the explicit transaction and it will lock the table until the transaction is committed or rolled back.

2. **ROLLBACK:**
   - The ROLLBACK command is the transactional control command used to undo transactions that have not already been saved to the database.
   - Rollbacks a transaction to the beginning of the transaction.
   - It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.

- You can use ROLLBACK TRANSACTION to erase all data modifications made from the start of the transaction or to a savepoint.

  Example: Consider following table

Student		
Rollno	**Name**	**SPI**
1	Raju	8
2	Hari	9
3	Mahesh	7

```
BEGIN TRANSACTION t1
DELETE FROM STUDENT WHERE SPI <8;
ROLLBACK TRANSACTION t1;
```

Output:

Student		
Rollno	**Name**	**SPI**
1	Raju	8
2	Hari	9
3	Mahesh	7

3. SAVEPOINT:

- A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.
- The ROLLBACK command is used to undo a group of transactions.

  Example: Consider following table

Student		
Rollno	**Name**	**SPI**
1	Raju	8
2	Hari	9
3	Mahesh	7

```
BEGIN TRANSACTION t1
SAVE TRANSACTION s1
INSERT INTO Student Values (4,’Anil’,6);

SAVE TRANSACTION s2
```

```
INSERT INTO Student Values (5, Gita',9);

ROLLBACK TRANSACTION s2;
```

Output:

Student		
Rollno	**Name**	**SPI**
1	Raju	8
2	Hari	9
3	Mahesh	7
4	Anil	6

Explain Security Privileged Commands. *OR* Explain Grant and Revoke command. *OR* Explain Data Control Language (DCL) Commands.

- DCL commands are used to enforce (implement) database security in a multiple user database environment.
- These commands are used to give or take back permission on any object to/from any user.
- Two types of DCL commands are GRANT and REVOKE.
- Only Database Administrator or owner of the database object can provide/remove privileges on a database object.

1. **GRANT**: GRANT is a command used to provide access or privileges or rights on the database objects to the users.

   Syntax:

   ```
   GRANT privilege_name
   ON object_name
   TO {user_name |PUBLIC}
   [WITH GRANT OPTION];
   ```

   Explanation:

   - privilege_name is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
   - object_name is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
   - user_name is the name of the user to whom an access right is being granted.
   - PUBLIC is used to grant access rights to all users.
   - WITH GRANT OPTION – allows a user to grant (give) access rights to other users.

2. **REVOKE**: The REVOKE is a command used to take back access or privileges or

rights on the database objects from the users.

Syntax:

```
REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC }
```

Explanation:

- privilege_name is the access right or privilege want to take back from the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- object_name is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
- user_name is the name of the user from whom an access right is being taken back.
- PUBLIC is used to take back access rights to all users.

Explain "on delete cascade" with example.

- A foreign key with cascade delete means that if a record in the Parent (Master) table is deleted, then the corresponding records in the Child (Foreign) table with automatically be deleted. This is called a cascade delete in Oracle.
- A foreign key with a cascade delete can be defined in either a CREATE TABLE statement or an ALTER TABLE statement.
- Syntax (Create table)

```
CREATE TABLE table_name
(
  column1 datatype  null/not null,
  column2 datatype  null/not null,
  ...

CONSTRAINT fk_column
        FOREIGN KEY (column1, column2,  ... column_n)
        REFERENCES parent_table (column1, column2, ... column_n)
        ON DELETE CASCADE
);
```

- Syntax (Alter table)

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name
FOREIGN KEY (column1, column2, ... column_n)
        REFERENCES parent_table (column1, column2, ... column_n)
        ON DELETE CASCADE;
```

Describe the following SQL functions.

SQL Function	Description	SQL Query Example
Math function		
Abs(n)	Returns the absolute value of n.	Select Abs(-15); O/P : 15
Sign(n)	Returns the sign of x as -1,0,1	Select Sign(-15); O/P : -1
Ceiling(n)	Returns the smallest integer value that is smaller than or equal to a number.	Select ceil(24.8); O/P : 25
Floor(n)	Returns the largest integervalue that is greater than or equal to a number.	Select Floor(24.8); O/P : 24
Power (m,n)	Returns m raised to nth power.	Select power(3,2); O/P : 9
Round (n,m)	Returns n rounded to m places the right of decimal point.	Select round(15.91,1); O/P : 15.9
Square(n)	Returns square of n.	Select Square(4); O/P : 16
Sqrt(n)	Returns square root of n.	Select sqrt(25); O/P : 5
Exp(n)	Returns e raised to the nth power, e=2.17828183.	Select exp(1); O/P : 1
X%Y	Returns reminder of X modulus Y.	Select 5%3; O/P : 2
Pi()	Returns the value of pi.	Select Pi(); O/P : 3.14159265358979
Sin(n)	Returns the sine value ofn.	Select Sin(0); O/P : 0
Cos(n)	Returns the cosine value of n.	Select Cos(0); O/P : 1
Tan(n)	Returns the tangent value of n.	Select Tan(0); O/P : 0
Rand(n)	Returns a random decimal number between 0 and 1.	Select Rand(); O/P : 0.845610816728278
Log(n)	Returns the log value of n having base e.	Select Log(1); O/P : 0
Log10(n)	Returns the log value of n having base 10.	Select Log10(1000); O/P : 3
String function		
ASCII(x)	Returns ASCII value of character.	Select ASCII('A'); O/P : 65
Char(x)	Returns a character of int ASCII code.	Select CHAR(65); O/P : A

Concat()	Concatenates two strings.	Select CONCAT('great','DIET'); O/P : greatDIET
Len ()	Returns the number of character in x.	Select LEN('DIET'); O/P : 4
Lower()	Converts the string to lower case.	Select LOWER('DIET'); O/P : diet
Upper()	Converts the string to upper case.	Select UPPER('diet'); O/P : DIET
Ltrim()	Trim blanks from the left of x.	Select LTRIM(' diet '); O/P : diet

Rtrim()	Trim blanks from the right of x.	Select RTRIM(' diet '); O/P : diet
Replace()	Looks for the string and replace the string every time it occurs.	Select Replace('this is college','is','may be'); O/P :thmay be may be college
Substring()	Returns the part of string	Select Substring('this is college',6,7); O/P : is my c
Left()	Returns the specified number of characters from the left.	Select Left('this is college',7); O/P : this is
Right()	Returns the specified number of characters from the right.	Select Right('this is college',7); O/P : college
Reverse()	Returns the reversed string.	Select Reverse('DIET'); O/P : TEID
Space()	Returns n number of spaces	Select Space(10); O/P :
Replicate()	Returns repeated string for n number of times.	Select Replicate('DIET',2); O/P :DIETDIET
Date function		
Date()	Returns current date and time.	Select date(); O/P : 2018-09-08 10:42:02.113
Day()	Returns day of a given date.	Select Day('23/JAN/2018'); O/P : 23
Month()	Returns month of a given date.	Select Month('23/JAN/2018'); O/P : 1
Year()	Returns year of a given date.	Select Year('23/JAN/2018'); O/P : 2018

AddDate()	It is used to get the date in which some time/date intervals are added.	select adddate(date, days);
		Select adddate('23-1-2018',2);
		O/P : 25-1-2018

Date_Format()	It is used to get the date in specified format.	Select Date_Format('2018-09-25', '%M %d %Y'); O/P : September 25 2018
Datediff()	Returns the difference between two date values, based on the interval specified.	Select Datediff(day,5, '23/JAN/2018','23/FEB/2018'); O/P : 31

CurDate() / Current_Date()	It is used to get the current day.	select curdate();
		select current_date();

		O/P : 2018-01-31

CurTime() / Current_Time()	It is used to get the current time.	select curtime();
		select current_time();
		O/P : 22:37:26

adddata()	The adddata() function is used to get the date in which some time/date intervals are added.	last_day()	The last_day() function is used to get the last date of the given month on the date.
day()	The day() function is used to get the day from the given date.	localtime()	The localtime() function is used to get the current date and time.
dayname()	The dayname() function is used to get the name of the day from the given date.	localtimestamp()	The localtimestamp() function is used to get the current date and time.
dayofmonth()	The dayofmonth() function is used to get the day for the specified date.	makedate()	The makedate() function is used to make the date from the given year and number of days.
dayofweek()	The dayofweek() function is used to get the day of the week in numeric.	maketime()	The maketime() function is used to make the time from given hour, minute and second.
dayofyear()	The dayofyear() function is used to get the number of day in the year.	microsecond()	The microsecond() function is used

			to get the value of the microsecond from the given datetime or time.
from_days()	The from_days() function is used to get the date of the given number of days.	minute()	The minute() function is used to get the value of month for the specified datetime or time.
hour()	The hour() function is used to get the hour from the given datetime.	month()	The month() function is used to get the value of month from given datetime or time.
addtime()	The addtime() function is used to get the time/datetime value in which some time intervals are added.	monthname()	The monthname() function is used to get the full month name.
current_timestamp()	The current_timestamp() function is used to get the current date and time.	now() The now()	function is used to get the current date and time.
weekday()	The weekday() function is used to get the index for a date	period_add()	The period_add() function adds the given number of month in the given period in the format

			YYMM or YYYYMM.
week()	The week() function is used to get the week portion for the specified date.	period_diff()	The period_diff() function is used to get the difference between the given two periods.
weekofyear()	The weekofyear() function is used to get the week of the given date.	quater()	The quarter() function is used to get the quarter portion of the specified date/datetime.
time()	The time() function is used to get the time for the given time/datetime.	sec_to_time()	The sec_to_time() function is used to convert the specified second into time.
time_format()	The time_format() function is used to format the time in specified format_mask.	second()	The second() function is used to get the second portion from the specified date/datetime.
time_to_sec()	The time_to_sec() function is used to convert the time into seconds.	str_to_date()	The str_to_date() function is used to convert the string into the given format_mask.
timediff()	The timediff() function is	subdate()	The subdate()

	used to get the difference for the given two time/datetime.		function is used to get the date which is subtracted by given intervals.
timestamp()	The timestamp() function is used to convert the expression into datetime/time.	subtime()	The subtime() function is used to get the time/datetime which is subtracted by certain intervals.
to_day()	The to_day() function is used to convert the date into numeric number of days.	sysdate()	The sysdate() function is used to get the system date.

Discuss aggregate functions with example(s).

- Aggregate functions perform a calculation on a set of values and return a single value.
- Aggregate functions ignore NULL values except COUNT(*).
- It is used with the GROUP BY clause of the SELECT statement.

Example: Consider following table

Student		
Rollno	**Name**	**SPI**
1	Raju	8
2	Hari	9
3	Mahesh	7
4	NULL	9
5	Anil	5

1. Avg() : It returns the average of the data values.
   Select Avg(SPI) FROM Student;
   Output: 7
2. Sum() : It returns the addition of the data values.
   Select Sum(SPI) FROM Student;
   Output: 38
3. Max() : It returns maximum value for a column.
   Select Max(SPI) FROM Student;

Output: 9

4. Min() : It returns Minimum value for a column.

   Select Min(SPI) FROM Student;

   Output: 5
5. Count() : It returns total number of values in a given column.

   Select Count(Name) FROM Student;

   Output: 4
6. Count(*) : It returns the number of rows in a table.

   Select Count(*) FROM Student;

   Output: 5

What is join? List and explain various types of joins.

- A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them.
- Different types of Joins are:
  - ✓ Inner Join
  - ✓ Outer Join
    1. Left Outer Join
    2. Right Outer Join
    3. Full Outer Join
  - ✓ Cross join
  - ✓ Self Join

INNER JOIN

- It returns records that have matching values in both tables.

Syntax:

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

Example:

Consider the following tables:

Student		
RNO	**Name**	**Branch**
101	Raju	CE
102	Amit	CE
103	Sanjay	ME
104	Neha	EC
105	Meera	EE
106	Mahesh	ME

Result	
RNO	**SPI**
101	8.8
102	9.2
104	8.2
105	7
107	8.9

```
SELECT Student.RNO, Student.Name, Student.Branch, Result.SPI
FROM Student
INNER JOIN Result
ON Student.RNO = Result.RNO;
```

Output:

Inner Join			
RNO	**Name**	**Branch**	**SPI**
101	Raju	CE	8.8
102	Amit	CE	9.2
104	Neha	EC	8.2
105	Meera	EE	7

LEFT OUTER JOIN

- The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2).
- The result is NULL from the right side, if there is no match.

Syntax:

```
SELECT columns
FROM table1 LEFT OUTER JOIN table2
ON table1.column = table2.column;
```

Example:

Consider the Student and result tables:

```
SELECT Student.RNO, Student.Name, Student.Branch, Result.SPI
FROM Student LEFT OUTER JOIN Result
ON Student.RNO = Result.RNO;
```

Output:

Left Join			
RNO	**Name**	**Branch**	**SPI**
101	Raju	CE	8.8
102	Amit	CE	9.2
103	Sanjay	ME	NULL
104	Neha	EC	8.2
105	Meera	EE	7
106	Mahesh	ME	NULL

RIGHT OUTER JOIN

- The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1).
- The result is NULL from the left side, if there is no match.

Syntax:

```
SELECT columns
FROM table1 RIGHT OUTER JOIN table2
ON table1.column = table2.column;
```

Example:

Consider the Student and result tables:

```
SELECT Student.RNO, Student.Name, Student.Branch, Result.SPI
FROM Student RIGHT OUTER JOIN Result
ON Student.RNO = Result.RNO;
```

Output:

Right Join			
RNO	**Name**	**Branch**	**SPI**
101	Raju	CE	8.8
102	Amit	CE	9.2
104	Neha	EC	8.2
105	Meera	EE	7
NULL	NULL	NULL	8.9

FULL OUTER JOIN

- The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

Syntax:

```
SELECT columns
FROM table1 FULL OUTER JOIN table2
ON table1.column = table2.column;
```

Example:

Consider the Student and result tables:

```
SELECT Student.RNO, Student.Name, Student.Branch, Result.SPI
FROM Student FULL OUTER JOIN Result
ON Student.RNO = Result.RNO;
```

Output:

Full Join			
RNO	**Name**	**Branch**	**SPI**
101	Raju	CE	8.8
102	Amit	CE	9.2
103	Sanjay	ME	NULL
104	Neha	EC	8.2
105	Meera	EE	7
106	Mahesh	ME	NULL
NULL	NULL	NULL	8.9

CROSS JOIN

- When each row of first table is combined with each row from the second table, known as Cartesian join or cross join.
- SQL CROSS JOIN returns the number of rows in first table multiplied by the number of rows in second table.

Syntax:

```
SELECT columns
FROM table1 CROSS JOIN table2;
```

Example:

Consider the following tables:

Color	
Code	**Name**
1	Red
2	Blue

Size
Amount
Small
Large

```
SELECT Color.Code, Color.Name, Size.Amount
FROM Color CROSS JOIN Size;
```

Output:

Cross Join		
Code	**Name**	**Amount**
1	Red	Small
2	Blue	Small
1	Red	Large
2	Blue	Large

SELF JOIN

- A self join is a regular join, but the table is joined with itself.
- Here, we need to use aliases for the same table to set a self join between single table.

Syntax:

```
SELECT a.column, b.column
FROM tablename a CROSS JOIN tablename b
WHERE a.column=b.column;
```

Example:

Consider the following table:

Employee		
EmpNo	**Name**	**MngrNo**
E01	Tarun	E02
E02	Rohan	E05
E03	Priya	E04
E04	Milan	NULL
E05	Jay	NULL

E06	Anjana	E03

```
SELECT e.Name as Employee, m.Name as Employee
FROM Employee e INNER JOIN Employee m
ON e.MngrNo=m.EmpNo;
```

Output:

Employee	
Employee	**Manager**
Tarun	Rohan
Rohan	Jay
Priya	Milan
Anjana	Priya

Define view. What are the types of view? Write syntax to create view of each type. Give an example of view.

- Views are virtual tables that are compiled at runtime.
- The data associated with views are not physically stored in the view, but it is stored in the base tables of the view.
- A view can be made over one or more database tables.
- Generally, we put those columns in view that we need to retrieve/query again and again.
- Once you have created the view, you can query view like as table.

TYPES OF VIEW

1. Simple View
2. Complex View

Syntax:

```
CREATE VIEW view_name
AS
SELECT column1, column2...
FROM table_name
[WHERE condition];
```

Simple View

- When we create a view on a single table, it is called simple view.
- In a simple view we can delete, update and Insert data and that changes are applied on base table.
- Insert operation are perform on simple view only if we have primary key and all not null fields in the view.

Example:

Consider following table:

Employee			
Eid	**Ename**	**Salary**	**Department**
101	Raju	5000	Admin
102	Amit	8000	HR
103	Sanjay	3000	IT
104	Neha	7000	Sales

--Create View

```
CREATE VIEW EmpSelect
AS
SELECT Eid, Ename, Department
FROM Employee;
```

--Display View

```
Select * from EmpSelect;
```

Output		
Eid	**Ename**	**Department**
101	Raju	Admin
102	Amit	HR
103	Sanjay	IT
104	Neha	Sales

Complex View

- When we create a view on more than one table, it is called complex view.
- We can only update data in complex view.
- You can't insert data in complex view.
- In particular, complex views can contain: join conditions, a group by clause, an order by clause etc.

Example:

Consider following table:

Employee			
Eid	**Ename**	**Salary**	**Department**
101	Raju	5000	Admin
102	Amit	8000	HR
103	Sanjay	3000	IT
104	Neha	7000	Sales

ContactDetails		
Eid	**City**	**Mobile**
101	Rajkot	1234567890
102	Ahmedabad	2345678901
103	Baroda	3456789120
104	Rajkot	4567891230

--Create View

```
        CREATE VIEW Empview
        AS
        SELECT Employee.Eid, Employee.Ename, ConcactDetails.City
        FROM Employee Inner Join ConcactDetails
        On Employee.Eid= ConcactDetails.Eid;

--Display View
        Select * from Empview;
```

Output		
Eid	**Ename**	**City**
101	Raju	Rajkot
102	Amit	Ahmedabad
103	Sanjay	Baroda
104	Neha	Rajkot