What is SQL?

SQL is a database computer language designed for storing, manipulating and retrieving data stored in a relational database. SQL stands for Structured Query Language.

SQL is a language to operate databases; it includes database creation, deletion, fetching rows, modifying rows, etc. SQL is an ANSI (American National Standards Institute) standard language, but there are many different versions of the SQL language.

Why to Learn SQL?

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.
SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Also, they are using different languages, such as:
MS SQL Server using T-SQL,
Oracle using PL/SQL,
MS Access version of SQL is called JET SQL (native format) etc.

Applications of SQL

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

What is Database ?

The Database is an essential part of our life. As we encounter several activities that involve our interaction with databases, for example in the bank, in the railway station, in school, in a grocery store, etc. These are the instances where we need to store a large amount of data in one place and fetch these data easily.

A database is a collection of data that is organized, which is also called structured data. It can be accessed or stored in a computer system. It can be managed through a Database Management System (DBMS), a software used to manage data. Database refers to related data in a structured form.

In a database, data is organized into tables consisting of rows and columns and it is indexed so data can be updated, expanded, and deleted easily. Computer databases typically contain file records data like transactions money in one bank account to another bank account, sales and customer details, fee details of students, and product details. There are different kinds of databases, ranging from the most prevalent approach, the relational database, to a distributed database, cloud database, and NoSQL databases.

Relational Database:
A relational database is made up of a set of tables with data that fits into a predefined category.

Distributed Database
A distributed database is a database in which portions of the database are stored in multiple physical locations, and in which processing is dispersed or replicated among different points in a network.

Cloud Database
A cloud database is a database that typically runs on a cloud computing platform. Database service provides access to the database. Database services make the underlying software-stack transparent to the user.
These interactions are the example of a traditional database where data is of one type-that is textual. In advancement of technology has led to new applications of database systems. New media technology has made it possible to store images, video clips. These essential features are making multimedia databases.

NoSQL Database
NoSQL is an approach to databases that represents a shift away from traditional relational database management systems (RDBMS). ... Relational databases rely on tables, columns, rows, or schemas to organize and retrieve data. In contrast, NoSQL databases do not rely on these structures and use more flexible data models.

A Brief History of SQL

1970 − Dr. Edgar F. "Ted" Codd of IBM is known as the father of relational databases. He described a relational model for databases.

1974 − Structured Query Language appeared.

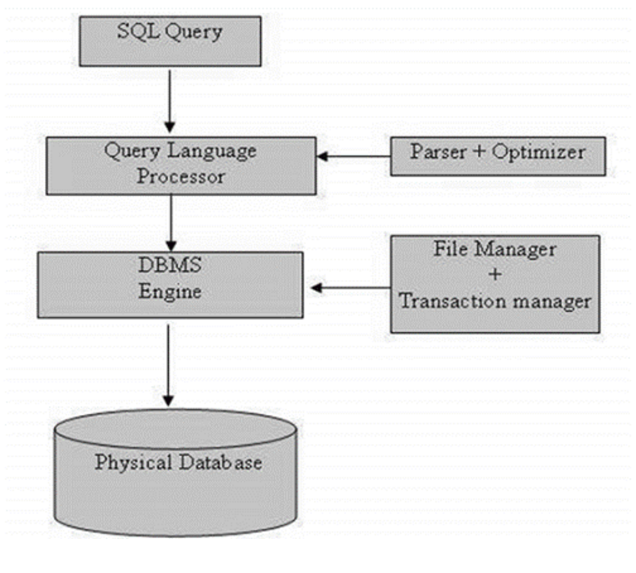
1978 − IBM worked to develop Codd's ideas and released a product named System/R.

1986 − IBM developed the first prototype of relational database and standardized by ANSI. The first relational database was released by Relational Software which later came to be known as Oracle.

SQL Process

When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

There are various components included in this process, these components are:
Query Dispatcher, Optimization Engines, Classic Query Engine, SQL Query Engine, etc.
A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.
Following is a simple diagram showing the SQL Architecture



SQL - RDBMS Databases

There are many popular RDBMS available to work with. Here are some of the most popular RDBMS's. This would help you to compare their basic features.

MySQL

MySQL is an open source SQL database, which is developed by a Swedish company – MySQL AB. MySQL is pronounced as "my ess-que-ell," in contrast with SQL, pronounced "sequel."

MySQL is supporting many different platforms including Microsoft Windows, the major Linux distributions, UNIX, and Mac OS X.

MySQL has free and paid versions, depending on its usage (non-commercial/commercial) and features. MySQL comes with a very fast, multi-threaded, multi-user and robust SQL database server.

History

Development of MySQL by Michael Widenius & David Axmark beginning in 1994.

First internal release on 23rd May 1995.

Windows Version was released on the 8th January 1998 for Windows 95 and NT.

Version 3.23: beta from June 2000, production release January 2001.

Version 4.0: beta from August 2002, production release March 2003 (unions).

Version 4.1: beta from June 2004, production release October 2004.

Version 5.0: beta from March 2005, production release October 2005.

Sun Microsystems acquired MySQL AB on the 26th February 2008.

Version 5.1: production release 27th November 2008.

Features:

- High Performance.
- High Availability.
- Scalability and Flexibility Run anything.
- Robust Transactional Support.
- Web and Data Warehouse Strengths.
- Strong Data Protection.
- Comprehensive Application Development.
- Management Ease.
- Open Source Freedom and 24 x 7 Support.
- Lowest Total Cost of Ownership.

MS SQL Server

MS SQL Server is a Relational Database Management System developed by Microsoft Inc. Its primary query languages are:
T-SQL
ANSI SQL

History

1987 - Sybase releases SQL Server for UNIX.

1988 - Microsoft, Sybase, and Aston-Tate port SQL Server to OS/2.

1989 - Microsoft, Sybase, and Aston-Tate release SQL Server 1.0 for OS/2.

1990 - SQL Server 1.1 is released with support for Windows 3.0 clients.

Aston - Tate drops out of SQL Server development.

2000 - Microsoft releases SQL Server 2000.

2001 - Microsoft releases XML for SQL Server Web Release 1 (download).

2002 - Microsoft releases SQLXML 2.0 (renamed from XML for SQL Server).

2002 - Microsoft releases SQLXML 3.0.

2005 - Microsoft releases SQL Server 2005 on November 7th, 2005.

Features

- High Performance
- High Availability
- Database mirroring
- Database snapshots
- CLR integration
- Service Broker
- DDL triggers
- Ranking functions
- Row version-based isolation levels
- XML integration
- TRY...CATCH
- Database Mail

ORACLE

It is a very large multi-user based database management system. Oracle is a relational database management system developed by 'Oracle Corporation'.

Oracle works to efficiently manage its resources, a database of information among the multiple clients requesting and sending data in the network.

It is an excellent database server choice for client/server computing. Oracle supports all major operating systems for both clients and servers, including MSDOS, NetWare, UnixWare, OS/2 and most UNIX flavors.

History

Oracle began in 1977 and celebrating its 32 wonderful years in the industry (from 1977 to 2009).

1977 - Larry Ellison, Bob Miner and Ed Oates founded Software Development Laboratories to undertake development work.

1979 - Version 2.0 of Oracle was released and it became first commercial relational database and first SQL database. The company changed its name to Relational Software Inc. (RSI).

1981 - RSI started developing tools for Oracle.

1982 - RSI was renamed to Oracle Corporation.

1983 - Oracle released version 3.0, rewritten in C language and ran on multiple platforms.

1984 - Oracle version 4.0 was released. It contained features like concurrency control - multi-version read consistency, etc.

1985 - Oracle version 4.0 was released. It contained features like concurrency control - multi-version read consistency, etc.

2007 - Oracle released Oracle11g. The new version focused on better partitioning, easy migration, etc.

Features

- Concurrency
- Read Consistency
- Locking Mechanisms
- Quiesce Database
- Portability
- Self-managing database
- SQL*Plus
- ASM
- Scheduler
- Resource Manager
- Data Warehousing
- Materialized views
- Bitmap indexes
- Table compression
- Parallel Execution
- Analytic SQL
- Data mining
- Partitioning

MS ACCESS

This is one of the most popular Microsoft products. Microsoft Access is an entry-level database management software. MS Access database is not only inexpensive but also a powerful database for small-scale projects.

MS Access uses the Jet database engine, which utilizes a specific SQL language dialect (sometimes referred to as Jet SQL).

MS Access comes with the professional edition of MS Office package. MS Access has easyto-use intuitive graphical interface.

1992 - Access version 1.0 was released.

1993 - Access 1.1 released to improve compatibility with inclusion the Access Basic programming language.

The most significant transition was from Access 97 to Access 2000.

2007 - Access 2007, a new database format was introduced ACCDB which supports complex data types such as multi valued and attachment fields.

Features

Users can create tables, queries, forms and reports and connect them together with macros.

Option of importing and exporting the data to many formats including Excel, Outlook, ASCII, dBase, Paradox, FoxPro, SQL Server, Oracle, ODBC, etc.

There is also the Jet Database format (MDB or ACCDB in Access 2007), which can contain the application and data in one file. This makes it very convenient to distribute the entire application to another user, who can run it in disconnected environments.

Microsoft Access offers parameterized queries. These queries and Access tables can be referenced from other programs like VB6 and .NET through DAO or ADO.

The desktop editions of Microsoft SQL Server can be used with Access as an alternative to the Jet Database Engine.

Microsoft Access is a file server-based database. Unlike the client-server relational database management systems (RDBMS), Microsoft Access does not implement database triggers, stored procedures or transaction logging.

What is RDBMS?

RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.

What is a table?

The data in an RDBMS is stored in database objects which are called as tables. This table is basically a collection of related data entries and it consists of numerous columns and rows.

Remember, a table is the most common and simplest form of data storage in a relational database. The following program is an example of a CUSTOMERS table.

ID	NAME	AGE	SALARY
101	MD KALEEM	35	25000
102	SYED JUNAID	32	22000
103	RAHUL	34	25000
105	ANTHONY	40	15000

What is a field?

Every table is broken up into smaller entities called fields. The fields in the CUSTOMERS table consist of ID, NAME, AGE and SALARY.
A field is a column in a table that is designed to maintain specific information about every record in the table.

What is a Record or a Row?

A record is also called as a row of data is each individual entry that exists in a table. For example, there are 4 records in the above CUSTOMERS table.
Following is a single row of data or record in the CUSTOMERS table

101	MD KALEEM	35	25000

A record is a horizontal entity in a table.

What is a column?

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

For example, a column in the CUSTOMERS table is ID, which represents ID description and would be as shown below:

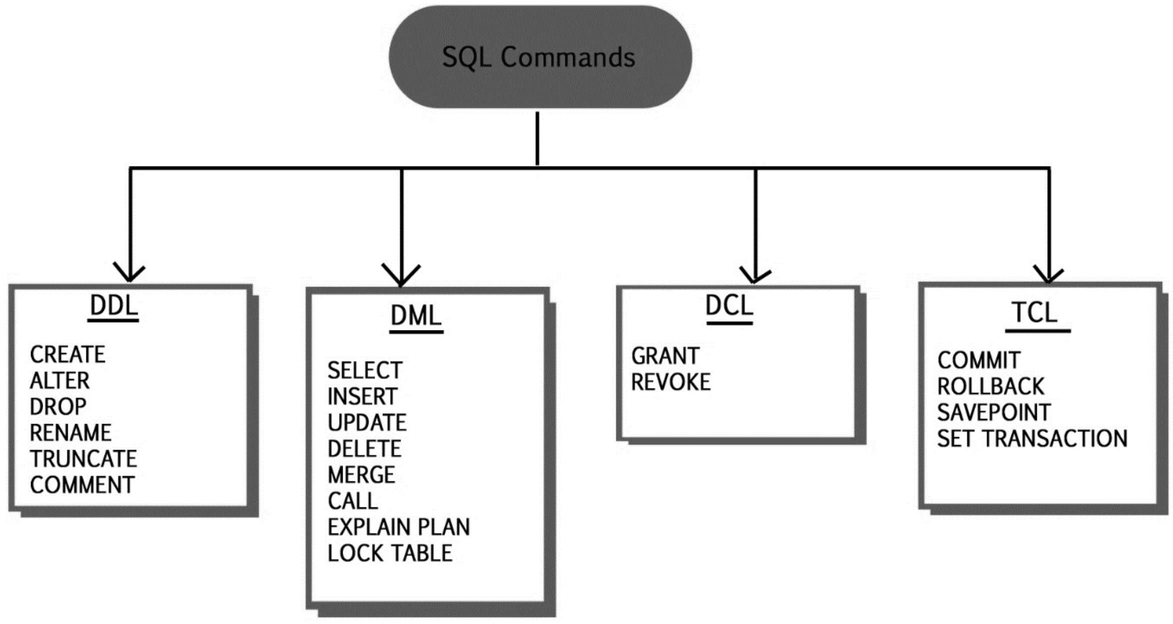
ID
101
102
103
105

What is a NULL value?
A NULL value in a table is a value in a field that appears to be blank, which means a field with a NULL value is a field with no value.
It is very important to understand that a NULL value is different than a zero value or a field that contains spaces. A field with a NULL value is the one that has been left blank during a record creation.

SQL Commands

SQL commands are the instructions used to communicate with a database to perform tasks, functions, and queries with data. SQL commands can be used to search the database and to do other functions like creating tables, adding data to tables, modifying data, and dropping tables.

These SQL commands are mainly categorized into four categories as:
DDL – Data Definition Language
DQl – Data Query Language
DML – Data Manipulation Language
DCL – Data Control Language
Though many resources claim there to be another category of SQL clauses **TCL – Transaction Control Language.**



DDL - Data Definition Language

Data Definition Language is used to define the database structure or schema. DDL is also used to specify additional properties of the data. The storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language. These statements define the implementation details of the database schema, which are usually hidden from the users. The data values stored in the database must satisfy certain consistency constraints.

For example, suppose the university requires that the account balance of a department must never be negative. The DDL provides facilities to specify such constraints. The database system checks these constraints every time the database is updated. In general, a constraint can be an arbitrary predicate pertaining to the database. However, arbitrary predicates may be costly to the test. Thus, the database system implements integrity constraints that can be tested with minimal overhead.

Domain Constraints : A domain of possible values must be associated with every attribute (for example, integer types, character types, date/time types). Declaring an attribute to be of a particular domain acts as the constraints on the values that it can take.

Referential Integrity : There are cases where we wish to ensure that a value appears in one relation for a given set of attributes also appear in a certain set of attributes in another relation i.e. Referential Integrity. For example, the department listed for each course must be one that actually exists.

Assertions : An assertion is any condition that the database must always satisfy. Domain constraints and Integrity constraints are special form of assertions.

Authorization : We may want to differentiate among the users as far as the type of access they are permitted on various data values in database. These differentiation are expressed in terms of Authorization.
The most common being :
read authorization – which allows reading but not modification of data ;
insert authorization – which allow insertion of new data but not modification of existing data
update authorization – which allows modification, but not deletion.

Some Commands :
1) **CREATE** Creates a new table, a view of a table, or other object in the database.
2) **ALTER** Modifies an existing database object, such as a table.
3) **DROP** Deletes an entire table, a view of a table or other objects in the database.

DML - Data Manipulation Language

DML statements are used for managing data with in schema objects.
DML are of two types :
Procedural DMLs : require a user to specify what data are needed and how to get those data.
Declerative DMLs (also referred as Non-procedural DMLs) : require a user to specify what data are needed without specifying how to get those data.
Declarative DMLs are usually easier to learn and use than procedural DMLs. However, since a user does not have to specify how to get the data, the database system has to figure out an efficient means of accessing data.

Some Commands :
1) **SELECT** Retrieves certain records from one or more tables.
2) **INSERT** Creates a record.
3) **UPDATE** Modifies records.
4) **DELETE** Deletes records.

DCL - Data Control Language:

A Data Control Language is a syntax similar to a computer programming language used to control access to data stored in a database (Authorization). In particular, it is a component of Structured Query Language (SQL).

Some Commands :
1) **GRANT** Gives a privilege to user.
2) **REVOKE** Takes back privileges granted from user.

TCL (Transaction Control Language) :

Transaction Control Language commands are used to manage transactions in the database. These are used to manage the changes made by DML-statements. It also allows statements to be grouped together into logical transactions.

Some Commands :

1) **COMMIT:** Commit command is used to permanently save any transaction into the database.
2) **ROLLBACK:** This command restores the database to last committed state. It is also used with savepoint command to jump to a savepoint in a transaction.
3) **SAVEPOINT:** Savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

SQL - Syntax

SQL is followed by a unique set of rules and guidelines called Syntax.

All the SQL statements start with any of the keywords like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW and all the statements end with a semicolon (;).

The most important point to be noted here is that SQL is case insensitive, which means SELECT and select have same meaning in SQL statements. Whereas, MySQL makes difference in table names. So, if you are working with MySQL, then you need to give table names as they exist in the database.

All the examples given in this tutorial have been tested with a MySQL server.

SQL Data Types for MySQL, SQL Server

The data type of a column defines what value the column can hold: integer, character, money, date and time, binary, and so on.

SQL Data Types

Each column in a database table is required to have a name and a data type.

An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

Note: Data types might have different names in different database. And even if the name is the same, the size and other details may be different! Always check the documentation!

MySQL Data Types (Version 8.0)

In MySQL there are three main data types: string, numeric, and date and time.

String Data Types

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The size parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data

MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

Numeric Data Types

Data type	**Description**
BIT(size)	A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1.
TINYINT(size)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The size parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(size)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The size parameter specifies the maximum display width (which is 255)
MEDIUMINT(size)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The size parameter specifies the maximum display width (which is 255)
INT(size)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255)
INTEGER(size)	Equal to INT(size)
BIGINT(size)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The size parameter specifies the maximum display width (which is 255)
FLOAT(size, d)	A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(p)	A floating point number. MySQL uses the p value to determine whether to use FLOAT or DOUBLE for the resulting data type. If p is from 0 to 24, the data type becomes FLOAT(). If p is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(size, d)	A normal-size floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter
DECIMAL(size, d)	An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0.
DEC(size, d)	Equal to DECIMAL(size,d)

Note: All the numeric data types may have an extra option: UNSIGNED or ZEROFILL. If you add the UNSIGNED option, MySQL disallows negative values for the column. If you add the ZEROFILL option, MySQL automatically also adds the UNSIGNED attribute to the column.

Date and Time Data Types

Data type	**Description**
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(fsp)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(fsp)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(fsp)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

SQL Server Data Types

String Data Types

Data type	**Description**	**Max size**	**Storage**
char(n)	Fixed width character string	8,000 characters	Defined width
varchar(n)	Variable width character string	8,000 characters	2 bytes + number of chars
varchar(max)	Variable width character string	1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string	2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string	4,000 characters	Defined width x 2
nvarchar	Variable width Unicode string	4,000 characters	
nvarchar(max)	Variable width Unicode string	536,870,912 characters	
ntext	Variable width Unicode string	2GB of text data	
binary(n)	Fixed width binary string	8,000 bytes	
varbinary	Variable width binary string	8,000 bytes	
varbinary(max)	Variable width binary string	2GB	
image	Variable width binary string	2GB	

Numeric Data Types

Data type	**Description**	**Storage**
bit	Integer that can be 0, 1, or NULL	
tinyint	Allows whole numbers from 0 to 255	1 byte
smallint	Allows whole numbers between -32,768 and 32,767	2 bytes
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
decimal(p,s)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 –1.	5-17 bytes

	The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0.	
numeric(p,s)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 –1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0.	5-17 bytes
smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
float(n)	Floating precision number data from -1.79E + 308 to 1.79E + 308. The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53.	4 or 8 bytes
real	Floating precision number data from -3.40E + 38 to 3.40E + 38	4 bytes

Date and Time Data Types

Data type	Description	Storage
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	6-8 bytes
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 bytes
date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset	8-10 bytes
timestamp	Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable	

Other Data Types

Data type	Description
sql_variant	Stores up to 8,000 bytes of data of various data types, except text, ntext, and timestamp
uniqueidentifier	Stores a globally unique identifier (GUID)
xml	Stores XML formatted data. Maximum 2GB
cursor	Stores a reference to a cursor used for database operations
table	Stores a result-set for later processing

SQL - Operators

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

- Arithmetic operators
- Comparison operators
- Logical operators
- Operators used to negate conditions

Arithmetic Operators

Assume 'variable a' holds 10 and 'variable b' holds 20, then:

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	a + b will give 30
- (Subtraction)	Subtracts right hand operand from left hand operand.	a - b will give -10
* (Multiplication)	Multiplies values on either side of the operator.	a * b will give 200
/ (Division)	Divides left hand operand by right hand operand.	b / a will give 2
% (Modulus)	Divides left hand operand by right hand operand and returns remainder.	b % a will give 0

Comparison Operators

Assume 'variable a' holds 10 and 'variable b' holds 20, then −

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(a = b) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a != b) is true.
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a <> b) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.	(a !< b) is false.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.	(a !> b) is true.

SQL Logical Operators

Operator	Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list as per the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	The EXISTS operator is used to search for the presence of a row in a specified table that meets a certain criterion.
IN	The IN operator is used to compare a value to a list of literal values that have been specified.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard operators.

NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. This is a negate operator.
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
IS NULL	The NULL operator is used to compare a value with a NULL value.
UNIQUE	The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

SQL | CREATE

There are two CREATE statements available in SQL:

- CREATE DATABASE
- CREATE TABLE

CREATE DATABASE

A Database is defined as a structured set of data. So, in SQL the very first step to store the data in a well-structured manner is to create a database. The CREATE DATABASE statement is used to create a new database in SQL.

Syntax:

```
CREATE DATABASE database_name;
```

database_name: name of the database.

Example:

```
CREATE DATABASE my_database;
```

CREATE TABLE

To store the data we need a table to do that. The CREATE TABLE statement is used to create a table in SQL. Table comprises of rows and columns. So while creating tables we have to provide all the information to SQL about the names of the columns, type of data to be stored in columns, size of the data etc.

Syntax:

```
CREATE TABLE table_name
(
column1 data_type(size),
column2 data_type(size),
column3 data_type(size),
....
);
```

table_name: name of the table.

column1: name of the first column.

data_type: Type of data we want to store in the particular column.

size: Size of the data we can store in a particular column. For example if for a column we specify the data_type as int and size as 5 then this column can store an integer number of maximum 5 digits.

Example:

```
CREATE TABLE Students(ADMNO int(3), NAME varchar(20), TMARKS int(3));
```

This query will create a table named Students with three columns, ADMNO, NAME and TMARKS.

SQL - ALTER TABLE Command

The SQL ALTER TABLE command is used to add, delete or modify columns in an existing table. You should also use the ALTER TABLE command to add and drop various constraints on an existing table.

Syntax:

To add a New Column in an existing table:
ALTER TABLE table_name ADD column_name datatype;

To DROP COLUMN in an existing table :
ALTER TABLE table_name DROP COLUMN column_name;

To change the DATA TYPE of a column in a table :
ALTER TABLE table_name MODIFY COLUMN column_name datatype;

To add a NOT NULL constraint to a column in a table :
ALTER TABLE table_name MODIFY column_name datatype NOT NULL;

To ADD UNIQUE CONSTRAINT to a table :
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);

To ADD CHECK CONSTRAINT to a table :
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);

To ADD PRIMARY KEY constraint to a table :
ALTER TABLE table_name
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);

To DROP CONSTRAINT from a table :
ALTER TABLE table_name DROP PRIMARY KEY;
alter table sdata2 drop class, drop sec;

SQL | DROP, TRUNCATE

DROP is used to delete a whole database or just a table.The DROP statement destroys the objects like an existing database, table, index, or view.
A DROP statement in SQL removes a component from a relational database management system (RDBMS).

Syntax:
DROP object object_name

Examples:
DROP TABLE table_name;
table_name: Name of the table to be deleted.

DROP DATABASE database_name;
database_name: Name of the database to be deleted.

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extents of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table, typically bypassing a number of integrity enforcing mechanisms. It was officially introduced in the SQL:2008 standard.
The TRUNCATE TABLE mytable statement is logically (though not physically) equivalent to the DELETE FROM mytable statement (without a WHERE clause).

Syntax:
TRUNCATE TABLE table_name;
table_name: Name of the table to be truncated.
DATABASE name - student_data

DROP vs TRUNCATE

Truncate is normally ultra-fast and its ideal for deleting data from a temporary table.
Truncate preserves the structure of the table for future use, unlike drop table where the table is deleted with its full structure.
Table or Database deletion using DROP statement cannot be rolled back, so it must be used wisely.

To delete the whole database.
DROP DATABASE student_data;
After running the above query whole database will be deleted.

To truncate Student_details table from student_data database.
TRUNCATE TABLE Student_details;
After running the above query Student_details table will be truncated, i.e, the data will be deleted but the structure will remain in the memory for further operations.

DROP or DELETE Database

The SQL DROP DATABASE statement is used to drop an existing database in SQL schema.

Syntax
The basic syntax of DROP DATABASE statement is as follows:
DROP DATABASE DatabaseName;

Example
SQL> DROP DATABASE testDB;
NOTE: Be careful before using this operation because by deleting an existing database would result in loss of complete information stored in the database.

SQL | SELECT Query

The SELECT Statement in SQL is used to retrieve or fetch data from a database. We can fetch either the entire table or according to some specified rules. The data returned is stored in a result table. This result table is also called result-set.

With the SELECT clause of a SELECT command statement, we specify the columns that we want to be displayed in the query result and, optionally, which column headings we prefer to see above the result table.

The select clause is the first clause and is one of the last clauses of the select statement that the database server evaluates. The reason for this is that before we can determine what to include in the final result set, we need to know all of the possible columns that could be included in the final result set.

Basic Syntax:
To fetch the entire table or all the fields in the table:
SELECT * FROM table_name;

To fetch all the rows in the table with fields column1 , column2,……
SELECT column1,column2,……. FROM table_name
column1 , column2,…..: names of the fields of the table

To fetch desired/filtered rows in the table with fields column1 , column2,……
The WHERE clause is used to filter records, It is used to extract only those records that fulfill a specified condition.
SELECT column1, column2, ...
FROM table_name WHERE condition;

Note: The WHERE clause is not only used in SELECT statements, it is also used in UPDATE, DELETE, etc.

SQL - WHERE Clause

The SQL WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table. You should use the WHERE clause to filter the records and fetching only the necessary records.

The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc., which we would examine in the subsequent chapters.
Syntax

The basic syntax of the SELECT statement with the WHERE clause is as shown below.

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

You can specify a condition using the comparison or logical operators like >, <, =, LIKE, NOT, etc. The following examples would make this concept clear.

SQL - AND and OR Conjunctive Operators

The SQL AND & OR operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called as the conjunctive operators.

These operators provide a means to make multiple comparisons with different operators in the same SQL statement.

The AND Operator

The AND operator allows the existence of multiple conditions in an SQL statement's with WHERE clause.
The basic syntax of the AND operator with a WHERE clause is as follows:

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition1] AND [condition2]...AND [conditionN];
```

You can combine N number of conditions using the AND operator. For an action to be taken by the SQL statement, whether it be a transaction or a query, all conditions separated by the AND must be TRUE

The OR Operator

The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
The basic syntax of the OR operator with a WHERE clause is as follows:

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition1] OR [condition2]...OR [conditionN]
```

You can combine N number of conditions using the OR operator. For an action to be taken by the SQL statement, whether it be a transaction or query, the only any ONE of the conditions separated by the OR must be TRUE.

SQL - LIKE Clause

The SQL LIKE clause is used to compare a value to similar values using wildcard operators. **There are two wildcards used in conjunction with the LIKE operator.**

The percent sign (%)
The underscore (_)

The percent sign represents zero, one or multiple characters. The underscore represents a single number or character. These symbols can be used in combinations.

The basic syntax of % and _ is as follows:

SELECT FROM table_name
WHERE column LIKE 'XXXX%'

SQL | INSERT INTO Statement

The INSERT INTO statement of SQL is used to insert a new row in a table. There are two ways of using INSERT INTO statement for inserting rows:

Only values: First method is to specify only the value of data to be inserted without the column names.
INSERT INTO table_name VALUES (value1, value2, value3,…);
table_name: name of the table.
value1, value2,.. : value of first column, second column,… for the new record

Column names and values both: In the second method we will specify both the columns which we want to fill and their corresponding values as shown below:
INSERT INTO table_name (column1, column2, column3,..) VALUES (value1, value2, value3,..);
table_name: name of the table.
column1: name of first column, second column …
value1, value2, value3 : value of first column, second column,… for the new record

Queries:
Method 1 (Inserting only values) :
INSERT INTO Student VALUES (‘5’,’HARSH’,’WEST BENGAL’,’XXXXXXXXXX’,’19’);

Method 2 (Inserting values in only specified columns):
INSERT INTO Student (ROLL_NO, NAME, Age) VALUES (‘5’,’PRATIK’,’19’);

Notice that the columns for which the values are not provided are filled by null. Which is the default values for those columns.

Using SELECT in INSERT INTO Statement

We can use the SELECT statement with INSERT INTO statement to copy rows from one table and insert them into another table. The use of this statement is similar to that of INSERT INTO statement. The difference is that the SELECT statement is used here to select data from a different table. The different ways of using INSERT INTO SELECT statement are shown below:
Inserting all columns of a table: We can copy all the data of a table and insert into in a different table.
INSERT INTO first_table SELECT * FROM second_table;
first_table: name of first table.
second_table: name of second table.

We have used the SELECT statement to copy the data from one table and INSERT INTO statement to insert in a different table.

Inserting specific columns of a table: We can copy only those columns of a table which we want to insert into in a different table.
Syntax:
INSERT INTO first_table(names_of_columns1) SELECT names_of_columns2 FROM second_table;
first_table: name of first table.
second_table: name of second table.
names of columns1: name of columns separated by comma(,) for table 1.
names of columns2: name of columns separated by comma(,) for table 2.

We have used the SELECT statement to copy the data of the selected columns only from the second table and INSERT INTO statement to insert in first table.

Copying specific rows from a table: We can copy specific rows from a table to insert into another table by using WHERE clause with the SELECT statement. We have to provide appropriate condition in the WHERE clause to select specific rows.
INSERT INTO table1 SELECT * FROM table2 WHERE condition;
first_table: name of first table.
second_table: name of second table.
condition: condition to select specific rows.

Queries:
Method 1(Inserting all rows and columns):
INSERT INTO Student SELECT * FROM LateralStudent;

Method 2(Inserting specific columns):
INSERT INTO Student(ROLL_NO,NAME,Age) SELECT ROLL_NO, NAME, Age FROM LateralStudent;

To insert multiple rows in a table using Single SQL Statement

```
INSERT INTO table_name(Column1,Column2,Column3,.......)
VALUES (Value1, Value2,Value3,.....),
       (Value1, Value2,Value3,.....),
        (Value1, Value2,Value3,.....),
        ............................ ;
```

table_name: name of the table
Column1: name of first column, second column …
Value1, Value2, Value3 : value of first column, second column,… for each new row inserted
You need To provide Multiple lists of values where each list is separated by ",". Every list of value corresponds to values to be inserted in each new row of the table.
Values in the next list tells values to be inserted in the next Row of the table.

The following SQL statement insert multiple rows in Student Table.

```
INSERT INTO STUDENT(ID, NAME,AGE,GRADE,CITY) VALUES(1,"AMIT
KUMAR",15,10,"DELHI"),
                                       (2,"GAURI RAO",18,12,"BANGALORE"),
                                       (3,"MANAV BHATT",17,11,"NEW DELHI"),

                                      (4,"RIYA KAPOOR",10,5,"UDAIPUR");
```

SQL | UPDATE Statement

The UPDATE statement in SQL is used to update the data of an existing table in database. We can update single columns as well as multiple columns using UPDATE statement as per our requirement.

Basic Syntax
UPDATE table_name SET column1 = value1, column2 = value2,...
WHERE condition;
table_name: name of the table
column1: name of first , second, third column....
value1: new value for first, second, third column....
condition: condition to select the rows for which the values of columns needs to beupdated.

NOTE: In the above query the SET statement is used to set new values to the particular column and the WHERE clause is used to select the rows for which the columns are needed to be updated. If we have not used the WHERE clause then the columns in all the rows will be updated. So the WHERE clause is used to choose the particular rows.
table1

Example Queries
Updating single column: Update the column NAME and set the value to 'PRATIK' in all the rows where Age is 20.
UPDATE Student SET NAME = 'PRATIK' WHERE Age = 20;

Updating multiple columns: Update the columns NAME to 'PRATIK' and ADDRESS to 'SIKKIM' where ROLL_NO is 1.
UPDATE Student SET NAME = 'PRATIK', ADDRESS = 'SIKKIM' WHERE ROLL_NO = 1;

Note: For updating multiple columns use comma(,) to separate the names and values of two columns.
Omitting WHERE clause: If we omit the WHERE clause from the update query then all of the rows will get updated.
UPDATE Student SET NAME = 'PRATIK';

SQL | DELETE Statement

The DELETE Statement in SQL is used to delete existing records from a table. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

Basic Syntax:
DELETE FROM table_name WHERE some_condition;
table_name: name of the table
some_condition: condition to choose particular record.
Note: We can delete single as well as multiple records depending on the condition we provide in WHERE clause. If we omit the WHERE clause then all of the records will be deleted and the table will be empty.
Example Queries:
Deleting single record: Delete the rows where NAME = 'Raheem'. This will delete only the first row.
DELETE FROM Student WHERE NAME = 'Raheem';

Deleting multiple records: Delete the rows from the table Student where Age is 20. This will delete 2 rows(third row and fifth row).
DELETE FROM Student WHERE Age = 20;

To Delete all of the records: There are two queries to do this as shown below,
query1: DELETE FROM Student;
query2: DELETE * FROM Student;

All of the records in the table will be deleted.

SQL GRANT REVOKE Commands

DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrator's or owner's of the database object can provide/remove privileges on a database object.

SQL GRANT Command

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

Syntax:

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

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 PROCEDURE 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.
ROLES are a set of privileges grouped together.
WITH GRANT OPTION - allows a user to grant access rights to other users.

For Example: GRANT SELECT ON employee TO user1; This command grants a SELECT permission on employee table to user1.You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on employee from user1, still user2 will have SELECT privilege on employee table.

SQL REVOKE Command:

The REVOKE command removes user access rights or privileges to the database objects.

Syntax:

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

For Example: REVOKE SELECT ON employee FROM user1;This command will REVOKE a SELECT privilege on employee table from user1.When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one users, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

Privileges and Roles:

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.
1) **System privileges -** This allows the user to CREATE, ALTER, or DROP database objects.
2) **Object privileges -** This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

System Privileges	Description
CREATE object	allows users to create the specified object in their own schema.
CREATE ANY object	allows users to create the specified object in any schema.

The above rules also apply for ALTER and DROP system privileges.

Few of the object privileges are listed below:

Object Privileges	Description
INSERT	allows users to insert rows into a table.
SELECT	allows users to select data from a database object.
UPDATE	allows user to update data in a table.
EXECUTE	allows user to execute a stored procedure or a function.

Roles: Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users, thereby automatically granting or revoking privileges. You can either create Roles or use the system roles pre-defined by oracle.

Some of the privileges granted to the system roles are as given below:

System Role	Privileges Granted to the Role
CONNECT	CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE SESSION etc.
RESOURCE	CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER etc. The primary usage of the RESOURCE role is to restrict access to database objects.
DBA	ALL SYSTEM PRIVILEGES

Creating Roles:
Syntax:
CREATE ROLE role_name
[IDENTIFIED BY password];

For Example: To create a role called "developer" with password as "pwd",the code will be as follows:
CREATE ROLE testing
[IDENTIFIED BY pwd];

It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user. If a role is identified by a password, then, when you GRANT or REVOKE privileges to the role, you definitely have to identify it with the password.

We can GRANT or REVOKE privilege to a role as below:

For example: To grant CREATE TABLE privilege to a user by creating a testing role:

First, create a testing Role
CREATE ROLE testing

Second, grant a CREATE TABLE privilege to the ROLE testing. You can add more privileges to the ROLE.
GRANT CREATE TABLE TO testing;

Third, grant the role to a user.
GRANT testing TO user1;

To revoke a CREATE TABLE privilege from testing ROLE, you can write:
REVOKE CREATE TABLE FROM testing;

The Syntax to drop a role from the database is as below:
DROP ROLE role_name;

For example: To drop a role called developer, you can write:
DROP ROLE testing;

Differences between Grant and Revoke commands:

S.NO	Grant	Revoke
1	This DCL command grants permissions to the user on the database objects.	This DCL command removes Permissions if any granted to the users on database objects.
2	It assigns access rights to users.	It revokes the useraccess rights of users.
3	For each user you need to specify the permissions.	If access for one user is removed; all the particular permissions provided by that users to others will be removed.
4	When the access is decentralized granting permissions will be easy.	If decentralized access removing the granted permissions is difficult.

SQL COMMIT, ROLLBACK SAVEPOINT Commands:
COMMIT: This command is used to save the data permanently.
Whenever we perform any of the DDL command like -INSERT, DELETE or UPDATE, these can be rollback if the data is not stored permanently. So in order to be at the safer side COMMIT command is used.
Syntax: commit;

ROLLBACK : This command is used to get the data or restore the data to the last savepoint or last committed state. If due to some reasons the data inserted, deleted or updated is not correct, you can rollback the data to a particular savepoint or if savepoint is not done, then to the last committed state.
Syntax: rollback;

SAVEPOINT : This command is used to save the data at a particular point temporarily, so that whenever needed can be rollback to that particular point.
Syntax: Savepoint A;

Consider the following Table Student:

Name	Marks
John	79
Jolly	65
Shuzan	70

UPDATE STUDENT SET NAME = ‘Sherlock’ WHERE NAME = ‘Jolly’;
By using this command you can update the record and save it permanently by using COMMIT command.

COMMIT;
Now after COMMIT :

Name	Marks
John	79
Sherlock	65
Shuzan	70

If commit was not performed then the changes made by the update command can be rollback.

Now if no COMMIT is performed.

UPDATE STUDENT SET NAME = ‘Sherlock’ WHERE STUDENT_NAME = ‘Jolly’;

After update command the table will be:

Name	Marks
John	79
Sherlock	65
Shuzan	70

Now if ROLLBACK is performed on the above table:

rollback;

After Rollback:

Name	Marks
John	79
Jolly	65
Shuzan	70

If on the above table savepoint is performed:

INSERT into STUDENT VALUES ('Jack', 95);

Commit;

UPDATE NAME SET NAME= ‘Rossie’ WHERE marks= 70;

SAVEPOINT A;

INSERT INTO STUDENT VALUES (‘Zack’, 76);

Savepoint B;

INSERT INTO STUDENT VALUES (‘Bruno’, 85);

Savepoint C;

SELECT * FROM STUDENT;

Name	Marks
John	79
Jolly	65
Rossie	70
Jack	95
Zack	76
Bruno	85

Now if we Rollback to Savepoint B:

Rollback to B;

The resulting Table will be:

Name	Marks
John	79
Jolly	65
Rossie	70
Jack	95
Zack	76

Now if we Rollback to Savepoint A:

Rollback to A;

The resulting Table will be:

Name	Marks
John	79
Jolly	65
Rossie	70
Jack	95

SQL | Views

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

CREATING VIEWS

We can create View using CREATE VIEW statement. A View can be created from a single table or multiple tables.

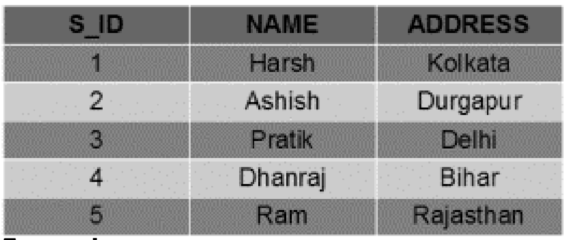
Syntax:

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

view_name: Name for the View
table_name: Name of the table
condition: Condition to select rows

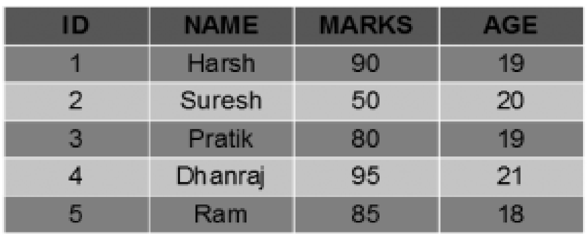
Sample Tables:

StudentDetails



S_ID	NAME	ADDRESS
1	Harsh	Kolkata
2	Ashish	Durgapur
3	Pratik	Delhi
4	Dhanraj	Bihar
5	Ram	Rajasthan

StudentMarks



ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

Examples:

Creating View from a single table:

In this example we will create a View named DetailsView from the table StudentDetails.

Query:

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM StudentDetails
WHERE S_ID < 5;
```

To see the data in the View, we can query the view in the same manner as we query a table.

```
SELECT * FROM DetailsView;
```

In this example, we will create a view named StudentNames from the table StudentDetails.

Query:

```
CREATE VIEW StudentNames AS
SELECT S_ID, NAME
FROM StudentDetails
ORDER BY NAME;
```

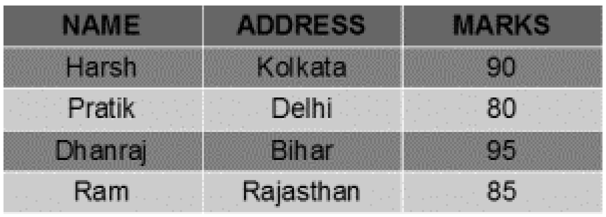
Creating View from multiple tables:

In this example we will create a View named MarksView from two tables StudentDetails and StudentMarks. To create a View from multiple tables we can simply include multiple tables in the SELECT statement.

Query:

```
CREATE VIEW MarksView AS
SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

MarksView



NAME	ADDRESS	MARKS
Harsh	Kolkata	90
Pratik	Delhi	80
Dhanraj	Bihar	95
Ram	Rajasthan	85

DELETING VIEWS

To delete or drop a View using the DROP statement.

Syntax:

```
DROP VIEW view_name;
```

view_name: Name of the View which we want to delete.

Example: DROP VIEW MarksView;

UPDATING VIEWS

There are certain conditions needed to be satisfied to update a view. If any one of these conditions is not met, then we will not be allowed to update the view.

- The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
- The SELECT statement should not have the DISTINCT keyword.
- The View should have all NOT NULL values.
- The view should not be created using nested queries or complex queries.
- The view should be created from a single table. If the view is created using multiple tables then it will not be allow to update the view.

We can use the CREATE OR REPLACE VIEW statement to add or remove fields from a view.

Syntax:

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1,coulmn2,..
FROM table_name
WHERE condition;
```

For example, **if we want to update the view MarksView and add the field AGE to this View from StudentMarks Table, we can do this as:**

```
CREATE OR REPLACE VIEW MarksView AS
SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS,
StudentMarks.AGE FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

Inserting a row in a view:

We can insert a row in a View in a same way as we do in a table. We can use the INSERT INTO statement of SQL to insert a row in a View.

Syntax:

```
INSERT INTO view_name(column1, column2 , column3,..)
VALUES(value1, value2, value3..);
```

view_name: Name of the View

Example:

In the below example we will insert a new row in the View DetailsView which we have created above in the example of “creating views from a single table”.

```
INSERT INTO DetailsView(NAME, ADDRESS)
VALUES("Suresh","Gurgaon");
```

Deleting a row from a View:

Deleting rows from a view is also as simple as deleting rows from a table. We can use the DELETE statement of SQL to delete rows from a view. Also deleting a row from a view first delete the row from the actual table and the change is then reflected in the view.

Syntax:

```
DELETE FROM view_name
WHERE condition;
view_name:Name of view from where we want to delete rows
condition: Condition to select rows
```

Example:

In this example we will delete the last row from the view DetailsView which we just added in the above example of inserting rows.

```
DELETE FROM DetailsView
WHERE NAME="Suresh";
```

WITH CHECK OPTION

The WITH CHECK OPTION clause in SQL is a very useful clause for views. It is applicable to a updatable view. If the view is not updatable, then there is no meaning of including this clause in the CREATE VIEW statement.

The WITH CHECK OPTION clause is used to prevent the insertion of rows in the view where the condition in the WHERE clause in CREATE VIEW statement is not satisfied.
If we have used the WITH CHECK OPTION clause in the CREATE VIEW statement, and if the UPDATE or INSERT clause does not satisfy the conditions then they will return an error.

Example:

In the below example we are creating a View SampleView from StudentDetails Table with WITH CHECK OPTION clause.

```
CREATE VIEW SampleView AS
SELECT S_ID, NAME
FROM  StudentDetails
WHERE NAME IS NOT NULL
WITH CHECK OPTION;
```

In this View if we try to insert a new row with null value in the NAME column then it will give an error because the view is created with the condition for NAME column as NOT NULL.
For example,though the View is updatable but then also the below query for this View is not valid:

```
INSERT INTO SampleView(S_ID) VALUES(6);
```

NOTE: The default value of NAME column is null.

Uses of a View

A good database should contain views due to the given reasons:
Restricting data access: Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table.

Hiding data complexity: A view can hide the complexity that exists in a multiple table join.

Simplify commands for the user: Views allows the user to select information from multiple tables without requiring the users to actually know how to perform a join.

Store complex queries: Views can be used to store complex queries.

Rename Columns: Views can also be used to rename the columns without affecting the base tables provided the number of columns in view must match the number of columns specified in select statement. Thus, renaming helps to to hide the names of the columns of the base tables.

Multiple view facility: Different views can be created on the same table for different users.

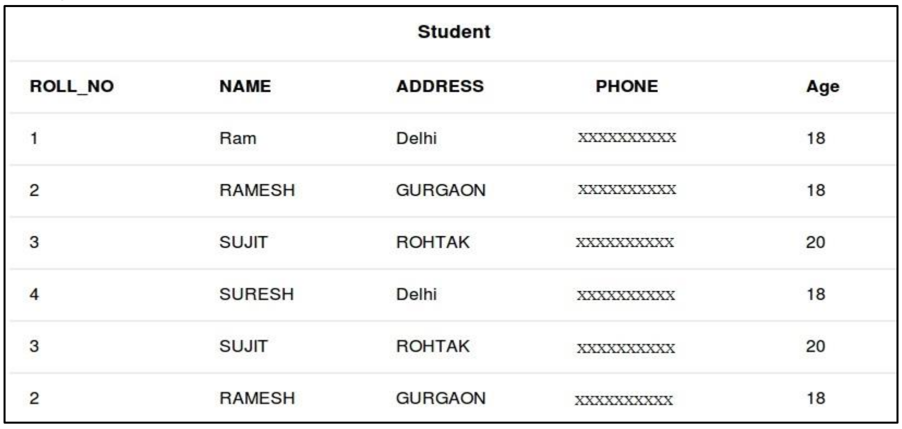
SQL | Wildcard operators

Wildcard operators are used with LIKE operator, there are four basic operators:

Operator	**Description**
%	It is used in substitute of zero or more characters.
_	It is used in substitute of one character.
___	It is used to substitute a range of characters.
[range_of_characters]	It is used to fetch matching set or range of characters specified inside the brackets.
[^range_of_characters] or [!range of characters]	It is used to fetch non-matching set or range of characters specified inside the brackets.

Basic syntax:
SELECT column1,column2 FROM table_name WHERE column LIKE wildcard_operator;
column1 , column2: fields in the table
table_name: name of table
column: name of field used for filtering data

Sample Table



Student				
ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18
2	RAMESH	GURGAON	XXXXXXXXXX	18
3	SUJIT	ROHTAK	XXXXXXXXXX	20
4	SURESH	Delhi	XXXXXXXXXX	18
3	SUJIT	ROHTAK	XXXXXXXXXX	20
2	RAMESH	GURGAON	XXXXXXXXXX	18

To fetch records from Student table with NAME ending with letter 'T'.
SELECT * FROM Student WHERE NAME LIKE '%T';

ROLL_NO	NAME	ADDRESS	PHONE	Age
3	SUJIT	ROHTAK	XXXXXXXXXX	20
3	SUJIT	ROHTAK	XXXXXXXXXX	20

To fetch records from Student table with NAME ending any letter but starting from 'RAMES'.
SELECT * FROM Student WHERE NAME LIKE 'RAMES_';

ROLL_NO	NAME	ADDRESS	PHONE	Age
2	RAMESH	GURGAON	XXXXXXXXXX	18

To fetch records from Student table with address containing letters 'a', 'b', or 'c'.
SELECT * FROM Student WHERE ADDRESS LIKE '%[A-C]%';

ROLL_NO	NAME	ADDRESS	PHONE	Age
2	RAMESH	GURGAON	XXXXXXXXXX	18

2	RAMESH	GURGAON	XXXXXXXXXX	18
3	SUJIT	ROHTAK	XXXXXXXXXX	20
3	SUJIT	ROHTAK	XXXXXXXXXX	20

To fetch records from Student table with ADDRESS not containing letters ‘a’, ‘b’, or ‘c’.

```
SELECT * FROM Student WHERE ADDRESS LIKE '%[^A-C]%';
```

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18
4	SURESH	Delhi	XXXXXXXXXX	18

To fetch records from Student table with PHONE field having a ‘9’ in 1st position and a ‘5’ in 4th position.

```
SELECT * FROM Student WHERE PHONE LIKE '9__5%';
```

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18

To fetch records from Student table with ADDRESS containing total of 6 characters.

```
SELECT * FROM Student WHERE ADDRESS LIKE '______';
```

ROLL_NO	NAME	ADDRESS	PHONE	Age
3	SUJIT	ROHTAK	XXXXXXXXXX	20
3	SUJIT	ROHTAK	XXXXXXXXXX	20

To fetch records from Student table with ADDRESS containing ‘OH’ at any position, and the result set should not contain duplicate data.

```
SELECT DISTINCT * FROM Student WHERE ADDRESS LIKE '%OH%';
```

ROLL_NO	NAME	ADDRESS	PHONE	Age
3	SUJIT	ROHTAK	XXXXXXXXXX	20