**What is SQL?**

Structured Query Language is a database tool which is used to create and access database to support software application.

**What are different types of statements supported by SQL?**

**There are 3 types of SQL statements**

1. DDL (Data Definition Language): It is used to define the database structure such as tables. It includes three statements such as **Create**, **Alter**, and **Drop**.
2. DML (Data Manipulation Language): These statements are used to manipulate the data in records. Commonly used DML statements are **Insert**, **Update**, and **Delete**.
3. DCL (Data Control Language): These statements are used to set privileges such as **Grant** and **Revoke** database access permission to the specific user.

**What are different Clauses used in SQL?**

**WHERE Clause:** This clause is used to define the condition, extract and display only those records which fulfill the given condition

Syntax: *SELECT column\_name(s)*

*FROM table\_name*

*WHERE condition;*

**GROUP BY Clause:** It is used with SELECT statement to group the result of the executed query using the value specified in it. It matches the value with the column name in tables and groups the end result accordingly.

Syntax: *SELECT column\_name(s)*

*FROM table\_name*

*GROUP BY column\_name;*

**HAVING clause:** This clause is used in association with GROUP BY clause. It is applied to the each group of result or the entire result as single group and much similar as WHERE clause, the only difference is you cannot use it without GROUP BY clause

Syntax: *SELECT column\_name(s)*

*FROM table\_name*

*GROUP BY column\_name*

*HAVING condition;*

**ORDER BY clause**: This clause is to define the order of the query output either in ascending (ASC) or in descending (DESC) order. Ascending (ASC) is the default one but descending (DESC) is set explicitly.

Syntax: *SELECT column\_name(s)*

*FROM table\_name*

*WHERE condition*

*ORDER BY column\_name ASC|DESC;*

**USING clause:** USING clause comes in use while working with SQL Joins. It is used to check equality based on columns when tables are joined. It can be used instead ON clause in Joins.

Syntax: *SELECT column\_name(s)*

*FROM table\_name*

*JOIN table\_name*

*USING (column\_name);*

**Why do we use SQL constraints? Which constraints we can use while creating database in SQL?**

Constraints are used to set the rules for all records in the table. If any constraints get violated then it can abort the action that caused it.

Constraints are defined while creating the database itself with CREATE TABLE statement or even after the table is created once with ALTER TABLE statement.

There are 5 major constraints are used in SQL, such as

* **NOT NULL**: That indicates that the column must have some value and cannot be left null

*CREATE TABLE Student(s\_id int NOT NULL, Name varchar(60), Age int);*

* **UNIQUE**: This constraint is used to ensure that each row and column has unique value and no value is being repeated in any other row or column.

*CREATE TABLE Student(s\_id int NOT NULL UNIQUE, Name varchar(60), Age int);*

OR

*ALTER TABLE Student ADD UNIQUE(s\_id);*

* **PRIMARY KEY**: This constraint is used in association with NOT NULL and UNIQUE constraints such as on one or the combination of more than one columns to identify the particular record with a unique identity.

*CREATE table Student (s\_id int PRIMARY KEY, Name varchar(60) NOT NULL, Age int);*

OR

*ALTER table Student ADD PRIMARY KEY (s\_id);*

* **FOREIGN KEY**: It is used to ensure the referential integrity of data in the table and also matches the value in one table with another using Primary Key.

*CREATE table Order\_Detail(*

*order\_id int PRIMARY KEY,*

*order\_name varchar(60) NOT NULL,*

*c\_id int FOREIGN KEY REFERENCES Customer\_Detail(c\_id)*

*);*

OR

*ALTER table Order\_Detail ADD FOREIGN KEY (c\_id) REFERENCES Customer\_Detail(c\_id);*

* **CHECK**: It is used to ensure whether the value in columns fulfills the specified condition.

*CREATE table Student(*

*s\_id int NOT NULL CHECK(s\_id > 0),*

*Name varchar(60) NOT NULL,*

*Age int*

*);*

OR

*ALTER table Student ADD CHECK(s\_id > 0);*

**What are different JOINS used in SQL?**

There are 4 major types of joins made to use while working on multiple tables in SQL databases

* **INNER JOIN**: It is also known as SIMPLE JOIN which returns all rows from BOTH tables when it has at least one column matched

Syntax: SELECT column\_name(s) FROM table\_name1 INNER JOIN table\_name2 ON column\_name1=column\_name2;

* **CROSS JOIN**: Cross join defines as Cartesian product where number of rows in the first table multiplied by number of rows in the second table. If suppose, WHERE clause is used in cross join then the query will work like an INNER JOIN.

Example:

select \* from table1 cross join table2 where table1.id = table2.fk\_id

* **LEFT JOIN (LEFT OUTER JOIN):** This join returns all rows from a LEFT table and its matched

rows from a RIGHT table.

Syntax: SELECT column\_name(s) FROM table\_name1 LEFT JOIN table\_name2 ON column\_name1=column\_name2;

* **RIGHT JOIN (RIGHT OUTER JOIN):** This joins returns all rows from the RIGHT table and its matched rows from a LEFT table.

Syntax: SELECT column\_name(s) FROM table\_name1 RIGHT JOIN table\_name2 ON column\_name1=column\_name2;

* **FULL JOIN (FULL OUTER JOIN):** This joins returns all when there is a match either in the RIGHT table or in the LEFT table.

Syntax: SELECT column\_name(s) FROM table\_name1 FULL OUTER JOIN table\_name2 ON column\_name1=column\_name2;

* **SELF JOIN:** Self-join is set to be query used to compare to itself. This is used to compare values in a column with other values in the same column in the same table. ALIAS ES can be used for the same table comparison.

Syntax: SELECT column-names FROM table-name T1 JOIN table-name T2 WHERE condition

Examples: SELECT B.FirstName AS FirstName1, B.LastName AS LastName1, A.FirstName AS FirstName2, A.LastName AS LastName2,

B.City, B.Country FROM Customer A, Customer B WHERE A.Id <> B.Id AND A.City = B.City AND A.Country = B.Country

ORDER BY A.Country

**What are transaction and its controls?**

A transaction can be defined as the sequence task that is performed on databases in a logical manner to gain certain results. Operations performed like Creating, updating, deleting records in the database comes from transactions.

In simple word, we can say that a transaction means a group of SQL queries executed on database records.

There are 4 transaction controls such as

* **COMMIT**: It is used to save all changes made through the transaction
* **ROLLBACK**: It is used to roll back the transaction such as all changes made by the transaction are reverted back and database remains as before
* **SET TRANSACTION**: Set the name of transaction
* **SAVEPOINT**: It is used to set the point from where the transaction is to be rolled back

**What are Indexes in SQL?**

The index can be defined as the way to retrieve the data more quickly. We can define indexes using CREATE statements.

Syntax: CREATE INDEX index\_name

ON table\_name (column\_name)

There are three types of indexes -

**Unique Index**: This indexing does not allow the field to have duplicate values if the column is unique indexed. Unique index can be applied automatically when primary key is defined.

Syntax: CREATE UNIQUE INDEX index\_name

ON table\_name (column\_name)

**Clustered Index:** This type of index reorders the physical order of the table and search based on the key values. Each table can have only one clustered index.

**NonClustered Index**: NonClustered Index does not alter the physical order of the table and maintains logical order of data. Each table can have 999 nonclustered indexes.

**What is the difference between clustered and non-clustered indexes?**

* One table can have only one clustered index but multiple nonclustered indexes.
* Clustered indexes can be read rapidly rather than non-clustered indexes.
* Clustered indexes store data physically in the table or view and non-clustered indexes do not store data in table as it has separate structure from data row

**What is the difference between DELETE and TRUNCATE?**

* The basic difference in both is DELETE is DML command and TRUNCATE is DDL
* DELETE is used to delete a specific row from the table whereas TRUNCATE is used to remove all rows from the table
* We can use DELETE with WHERE clause but cannot use TRUNCATE with it

**What is the difference between DROP and TRUNCATE?**

**TRUNCATE** removes all rows from the table which cannot be retrieved back, **DROP** removes the entire table from the database and it cannot be retrieved back.

**SET Operations in SQL**

We will cover 4 different types of SET operations, along with example:

**UNION** is used to combine the results of two or more SELECT statements. However it will eliminate duplicate rows from its resultset. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.

Syntax:

SELECT \* FROM First

UNION

SELECT \* FROM Second;

**UNION ALL**: This operation is similar to Union. But it also shows the duplicate rows.

Syntax:

SELECT \* FROM First

UNION ALL

SELECT \* FROM Second;

**INTERSECT**: Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of Intersect the number of columns and datatype must be same.

Syntax:

SELECT \* FROM First

INTERSECT

SELECT \* FROM Second;

MINUS: The Minus operation combines results of two SELECT statements and return only those in the final result, which belongs to the first set of the result.

Syntax:

SELECT \* FROM First

MINUS

SELECT \* FROM Second;

**What Is the Difference Between a Join and UNION?**

* The columns of joining tables may be different in JOIN but in UNION the number of columns and order of columns of all queries must be same.
* The UNION puts rows from queries after each other( puts vertically ) but JOIN puts the column from queries after each other (puts horizontally)
* Joins combine data into new columns. If two tables are joined together, then the data from the first table is shown in one set of column alongside the second table’s column in the same row.
* Unions combine data into new rows. If two tables are “unioned” together, then the data from the first table is in one set of rows, and the data from the second table in another set. The rows are in the same result.

**What are aggregate and scalar functions?**

Aggregate functions are used to evaluate mathematical calculation and return single values. This can be calculated from the columns in a table. Scalar functions return a single value based on the input value.

Example -.

Aggregate – max(), count – Calculated with respect to numeric.

Scalar – UCASE(), NOW() – Calculated with respect to strings.

**What is a primary key?**

A Primary key is a column whose values uniquely identify every row in a table. Primary key values can never be reused.

**What is a composite primary key?**

Primary key created on more than one column is called composite primary key.

**What are foreign keys?**

When a one table’s primary key field is added to related tables in order to create the common field which relates the two tables, it called a foreign key in other tables.

**What is CHECK Constraint?**

A CHECK constraint is used to limit the values or type of data that can be stored in a column. They are used to enforce domain integrity.

**What is a stored procedure?**

A stored procedure is a set of SQL queries which can take input and send back output.

**What is Normalization?**

The process of table design to minimize the data redundancy is called normalization. We need to divide a database into two or more table and define relationships between them.

**What is Trigger?**

Trigger allows us to execute a batch of SQL code when a table event occurs (Insert, update or delete command executed against a specific table)

Syntax: CREATE TRIGGER name {BEFORE|AFTER} (event [OR..]}

ON table\_name [FOR [EACH] {ROW|STATEMENT}]

EXECUTE PROCEDURE functionname {arguments}

**What is the use of NVL function?**

NVL function is used to convert the null value to its actual value.

**How can you create an empty table from an existing table?**

Example:

Select \* into studentcopy from student where 1=2

Here, we are copying student table to another table with the same structure with no rows copied.

**How to fetch common records from two tables?**

Example:

Select studentID from student

INTERSECT

Select StudentID from Exam

**How to fetch alternate records from a table?**

To display even numbers:

Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=0

To display odd numbers:

Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1

**How to select unique records from a table?**

Select DISTINCT StudentID, StudentName from Student

**What is the command used to fetch first 5 characters of the string?**

There are many ways to fetch first 5 characters of the string –

Select SUBSTRING(StudentName,1,5) as studentname from student

Or

Select RIGHT(Studentname,5) as studentname from student

**Which operator is used in query for pattern matching?**

**LIKE operator is used for pattern matching, and it can be used as -.**

* **% – Matches zero or more characters.**
* **\_(Underscore) – Matching exactly one character.**

Example:

Select \* from Student where studentname like ‘a%’

Select \* from Student where studentname like ‘ami\_’

[**How**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records) **print row numer in existing table 1st column:**

Syntax:

SELECT ROWNUM,e.\* FROM EN\_ROW\_VOICE e

**How can I create a copy of an Oracle table without copying the data?**

**create table structure from another table in oracle?**

Syntax:

CREATE TABLE EN\_ROW\_VOICE2 AS SELECT \* from EN\_ROW\_VOICE WHERE 1=0

**How to create the Student\_1 table, which is exact replica of Student table?**

Syntax:

Create Table Student\_1 as select \* from Student;

**How to create Round Up, Round Down & Round in query?**

Syntax:

SELECT DURATION, DURATION/60 AS DR,CEIL(DURATION/60) AS RoundUP,FLOOR(DURATION/60) AS RoundDown, ROUND(DURATION/60) AS Round FROM EN\_ROW\_VOICE;

[**Oracle SELECT TOP 10 records**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records)**:**

Syntax:

SELECT DURATION,DURATION/60 AS DR,CEIL(DURATION/60) AS RoundUP,FLOOR(DURATION/60) AS RundDown, ROUND(DURATION/60) AS Round FROM EN\_ROW\_VOICE **WHERE ROWNUM <10**;

[**Oracle SELECT Last 10 records**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records)**:**

SELECT DURATION,DURATION/60 AS DR,CEIL(DURATION/60) AS RoundUP,FLOOR(DURATION/60) AS RoundDown, ROUND(DURATION/60) AS Round FROM EN\_ROW\_VOICE **WHERE ROWNUM <10 ORDER BY ROWNUM DESC;**

[**How**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records) **to login User Name:**

Syntax:

select user from dual;

[**How**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records) **to get default password user name:**

Syntax:

select username from SYS.DBA\_USERS\_WITH\_DEFPWD;

[**How**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records) **to delete particular row:**

Syntax:

DELETE LIVE\_VAS\_MASTER\_DOT where CIRCLE='CIRCLE'

COMMIT

[**How**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records) **to find all User and Password:**

Syntax:

select username, password from DBA\_USERS;

[**How**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records) **to create forget User password:**

Syntax:

alter user SCOTT IDENTIFIED by aircel;

[**How**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records) **to find out MAX Sal from EMP:**

Syntax:

select \* from emp where sal=(select max(sal) from emp);

[**How**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records) **to find out 2nd highest Sal from EMP:**

Syntax:

select \* from emp where sal=(select max(sal) from emp where sal <(select max(sal) from emp));

[**How**](https://stackoverflow.com/questions/2498035/oracle-select-top-10-records) **to find out 3RD highest Sal from EMP:**

Syntax:

select \* from emp where sal=(select max(sal) from emp where sal <(select max(sal) from emp where sal<(select max(sal) from emp)));

**How to find TOP 5 Sal** **from EMP:**

Syntax:

SELECT \* FROM (SELECT \* FROM EMP ORDER BY SAL DESC) WHERE ROWNUM <=5;

**How to Selecting Duplicate rows from table?**

Syntax:

Select rollno FROM Student WHERE ROWID <>

(Select max (rowid) from Student b where rollno=b.rollno);

**How to Delete duplicate rows?**

Syntax:

Delete FROM Student WHERE ROWID <>

(Select max (rowid) from Student b where rollno=b.rollno);

**How to find count of duplicate rows?**

Syntax:

Select rollno, count (rollno) from Student

Group by rollno

Having count (rollno)>1

Order by count (rollno) desc;

**How to Show the Max marks and min marks together from student table?**

Syntax:

Select max (marks) from Student

Union

Select min (marks) from Student;

**How to fetch common records from two different tables which has not any joining condition.**

Syntax:

Select \* from Table1

Intersect

Select \* from Table2;

**How to select random rows from a table?**

Using SAMPLE clause we can select random rows.

Example:

SELECT \* FROM table\_name SAMPLE(10);

**Can we rename a column in the output of SQL query?**

Yes using the following syntax we can do this.

SELECT column\_name AS new\_name FROM table\_name;

**How can you select all the even number records from a table? All the odd number records?**

To select all the **even number** records from a table:

Select \* from table where id % 2 = 0

To select all the **odd number** records from a table:

Select \* from table where id % 2 != 0

**How to find a duplicate record?**

1. **duplicate records with one field**
2. **duplicate records with more than one field**
3. duplicate records with one field

SELECT name, COUNT(email)

FROM users

GROUP BY email

HAVING COUNT(email) > 1

1. duplicate records with more than one field

SELECT name, email, COUNT(\*)

FROM users

GROUP BY name, email

HAVING COUNT(\*) > 1

**Insert value into only specific columns**

Example:

INSERT INTO student(id, name) values(102, 'Alex');

**Insert NULL value to a column**

Example:

INSERT INTO student(id, name) values(102, 'Alex');

OR

INSERT INTO Student VALUES(102,'Alex', null);

**Insert Default value to a column**

INSERT INTO Student VALUES(103,'Chris', default)

**UPDATE command**

UPDATE table\_name SET column\_name = new\_value WHERE some\_condition;

UPDATE student SET age=18 WHERE student\_id=102;

**Updating Multiple Columns**

UPDATE student SET name='Abhi', age=17 where s\_id=103;

**UPDATE Command: Incrementing Integer Value**

UPDATE student SET age = age+1;

**What is the difference between FUNCTION, PROCEDURE AND PACKAGE in PL/SQL?**

**Function:** The main purpose of a PL/SQL function is generally to compute and return a single value. A function has a return type in its specification and must return a value specified in that type.

**Procedure:** A procedure does not have a return type and should not return any value but it can have a return statement that simply stops its execution and returns to the caller. A procedure is used to return multiple values otherwise it is generally similar to a function.

**Package:** A package is schema object which groups logically related PL/SQL types , items and subprograms. You can also say that it is a group of functions, procedure, variables and record type statement. It provides modularity, due to this facility it aids application development. It is used to hide information from unauthorized users.

**What are the data types available in PL/SQL?**

Data types define the ways to identify the type of data and their associated operations. There are 4 types of predefined data types explained as follows

* **Scalar Data Types**: A scalar data type is an atomic data type that does not have any internal components.

For example

* **CHAR** (fixed length character value between 1 and 32,767 characters)
* **VARCHAR2** (variable length character value between 1 and 32,767 characters)
* **NUMBER** ( fixed-decimal, floating-decimal or integer values)
* **BOOLEAN** ( logical data type for TRUE FALSE or NULL values)
* **DATE** (stores date and time information)
* **LONG** (character data of variable length)
* **Composite Data Types:** A composite data type is made up of other data types and internal components that can be easily used and manipulated. For example RECORD, TABLE, and VARRAY.
* **Reference Data Types:** A reference data types holds values, called pointers that designate to other program items or data items. For example REF CURSOR.
* **Large Object Data Types:** A Large Object datatype holds values, called locators, that defines the location of large objects( such as video clips, graphic image, etc) stored out of line.

For example:

* **BFILE** (Binary file)
* **BLOB** (Binary large object)
* **CLOB** ( Character large object)
* **NCLOB**( NCHAR type large object)

**What is an exception in PL/SQL? What are the two types of exceptions?**

Exceptions are manageable errors in a program. This means that errors handled by exceptions are within the bounds of the programmer to repair and PL/SQL provides catch features to encapsulate these errors to enable debugging and preventing the program to stop working.

There are two main types of exceptions – System Exceptions and User-Defined Exceptions. **System Exceptions** are such as no\_data\_found or too\_many\_rows and already defined by PL SQL. **User-Defined Exceptions** are exceptions defined by the user to handle particular errors.

**Explain the purpose of %TYPE and %ROWTYPE data types with the example?**

PL/SQL uses **%TYPE** declaration attribute for anchoring. This attribute provides the datatype of a variable, constant or column. %TYPE attribute is useful while declaring a variable that has the same datatype as a table column.

For example, the variable m\_empno has the same data type and size as the column empno in table emp.

m\_empno emp.empno%TYPE;

Syntax:

DECLARE

name VARCHAR(50);

firstName name%TYPE;

lastName name%TYPE;

province name%TYPE;

nationality name%TYPE;

**%ROWTYPE** attribute is used to declare a variable to be a record having the same structure as a row in a table. The row is defined as a record and its fields have the same names and data types as the columns in the table or view.

For example: dept\_rec dept%ROWTYPE;

This declares a record that can store an entire row for DEPT table.

Syntax:

emp employees\_table%ROWTYPE;

BEGIN

Execution section;

END;

**How does ROWID help in running a query faster?**

**ROWID** is the logical address of a row, it is not a physical column. It composes of data block number, file number and row number in the data block. Thus, I/O time gets minimized retrieving the row, and results in a faster query.

**Differentiate between SGA and PGA.**

SGA stands for System Global Area whereas PGA stands for Program or Process Global Area. PGA is only allocated 10% RAM size, but SGA is given 40% RAM size.

**Explain SPOOL**

Spool command can print the output of sql statements in a file.

Syntax:

spool/tmp/sql\_out.txt

select smp\_name, smp\_id from smp where dept=’accounts’;

spool off;