Unit III

3 Basics of SQL

3.1 DDL, DML, DCL Structure

Q.1 What is SQL ? What are the characteristics and advantages of SQL ?

Ans. : SQL : SQL stands for Structured Query Language.

- SQL makes use of query. A Query is a set of instruction given to the database management system. It tells any database what information we would like to get from the database.

Characteristics and Advantages :

1) SQL is a standard computer language for creating and manipulating databases.
2) SQL is very simple and easy to learn.
3) SQL allows the users to create, update, delete and retrieve data from the database.
4) SQL is used to create view, stored procedures and functions in a database.
5) SQL allows the users to set the permissions on the tables, procedures and views in the database.

Q.2 Explain DDL, DML and DCL structure.

Ans. : Data Definition Language

Sr. No.	Command	Purpose
1.	CREATE	This command is used to create database, tables, views or any other database objects.

2.	ALTER	It modifies the existing tables.
3.	DROP	This command deletes complete table, view or any other database object.

Data Manipulation Language

Sr. No.	Command	Purpose
1.	SELECT	This command is used to retrieve either all or desired records from one or more tables.
2.	INSERT	For inserting the records in the table, this command is used.
3.	UPDATE	For updating one or more fields of the table, this command is used.
4.	DELETE	This command is used for deleting the desired record.

Data Control Language

Sr. No.	Command	Purpose
1.	GRANT	This command is used to give access rights or privileges to the database.
2.	INVOKE	The revoke command removes user access rights or privileges to the database objects.

Q.3 What is the difference between DDL and DML ?

Ans. :

Difference between DDL and DML

DDL	DML
DDL stands for Data Definition Language.	DML stands for Data Manipulation Language.
DDL commands are used to define	DML commands are used for

database structure.	managing data within the database.
It works on whole table.	It works on one or more rows.
It cannot be classified further.	It can be classified as - procedural and non-procedural language.

3.2 Creation and Alteration

Q.4 Explain creation of Table using SQL.

Ans. :

- A database can be considered as a container for tables and a table is a grid with rows and columns to hold data.
- Individual statements in SQL are called **queries.**
- We can execute SQL queries for various tasks such as creation of tables, insertion of data into the tables, deletion of record from table and so on.

In this section we will discuss how to create a table.

Step 1 : We normally create a database using following SQL statement

Syntax

```
CREATE DATABASE database_name;
```

Example

```
CREATE DATABASE Person_DB
```

Step 2 : The table can be created inside the database as follows -

```
CREATE TABLE table_name (
    col1_name datatype,
    col2_name datatype,
    ...
    coln_name datatype
);
```

Example

```
CREATE TAELE person_details (
        AadharNo int,
        FirstName VARCHAR(20),
        MiddleName VARCHAR(20),
        LastName VARCHAR(20),
        Address VARCHAR(30),
        City VARCHAR(10)
)
```

Q.5 Give the syntax and example of insertion of data into the table using SQL.

Ans. :

- We can insert data into the table using INSERT statement.

Syntax

```
INSERT INTO table_name (col1,col2,....,coln)
VALUES (value1,value2,....,valuen)
```

Example

```
INSERT INTO person_details (AadharNo, FirstName, MiddleName,
LastName, Address, City)
VALUES (111,'AAA','BBB','CCC','M.G. Road','Pune')
```

The above query will result into –

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune

Q.6 How to delete a record from a table ? Explain it with SQL syntax and example.

Ans. :

- We can delete one or more records based on some condition. The syntax is as follows -

Syntax

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

Example

```
DELETE FROM person_details
WHERE AadharNo=333
```


We can delete all the records from table. But in this deletion, all the records get deleted without deleting table. For that purpose the SQL statement will be

```
DELETE FROM person_details;
```

Q.7 Explain the use of DISTINCT keyword in SQL.

Ans. :

- The keyword DISTINCT is used along with the SELECT statements.
- It is used to obtain unique values from the table. This query does not allow duplication of element.

Syntax :

```
SELECT DISTICT Column-name FROM table-name;
```

Example :

Consider following database table **Student**

Roll No	Name	City
1	Ankita	Pune
2.	Rohit	Hyderabad
3.	Prajkta	Chennai
4.	Sunil	Pune
5.	Sharda	Chennai

SQL Statement

```
SELECT DISTICT City
FROM Student;
```

This will result into

Pune
Hyderabad
Chennai

3.3 : Defining Constraints

Q.8 How to apply primary key constraints ? Explain it with suitable example.

Ans. :

(1) Primary key : The primary key constraint is defined to uniquely identify the records from the table.

The primary key must contain unique values. Hence database designer should choose primary key very carefully.

For example

Consider that we have to create a **perons_details** table with AdharNo, FirstName, MiddleName, LastName, Address and City.

Now making AdharNo as a primary key is helpful here as using this field it becomes easy to identify the records correctly.

The result will be

```
CREATE TABLE person_details (
AadharNo int,
FirstName VARCHAR(20),
MiddleName VARCHAR(20),
LastName VARCHAR(20),
Address VARCHAR(30),
City VARCHAR(10),
PRIMARY KEY(AadharNo)
);
```

We can create a composite key as a primary key using CONSTRAINT keyword. For example

```
CREATE TABLE person_details (
AadharNo int NOT NULL,
FirstName VARCHAR(20),
MiddleName VARCHAR(20),
LastName VARCHAR(20) NOT NULL,
```



```
Address VARCHAR(30),
City VARCHAR(10),
CONSTRAINT PK_person_details PRIMARY
KEY(AadharNo, LastName)
);
```

Q.9 Explain the use of IN operator with suitable example.

Ans. :

The IN operator is just similar to OR operator.

It allows to specify multiple values in WHERE clause.

Syntax

```
SELECT col1,col2,...
FROM table_name
WHERE column-name IN (value1, value2, ...);
```

Example

Consider following table

Employee

empID	empName	Salary	DeptID
1	AAA	1000	D101
2	BBB	2000	D102
3	CCC	3000	D103
4	DDD	4000	D104
5	EEE	5000	D105

```
SELECT * FROM Employee
WHERE empID IN (1, 3);
```

The result will be

empID	empName	Salary	DeptID
1	AAA	1000	D101
2	BBB	2000	D102
3	CCC	3000	D103

SELECT col1, col2, ... coln
from table-name
ORDER BY col1, col2, ... ASC|DESC

3.4 Use of Group By, Having and Order By Clause

Q.10 Explain Group By, Order By and Having clause with suitable example.

Ans. :

(1) Order By

- For getting the sorted records in the table we use **ORDER BY** command.
- The ORDER BY keyword sorts the records in ascending order by default.

Syntax

```
SELECT col1, col2,...,coln
FROM table_name
ORDER BY col1,col2,... ASC|DESC
```

Here ASC is for ascending order display and DESC is for descending order display.

```
SELECT *
FROM person_details
ORDER BY AadharNo DESC;
```

(2) Group By

- The GROUP BY clause is a SQL command that is used to group rows that have the same values.
- The GROUP BY clause is used in the SELECT statement.
- This query returns a single row for every grouped item.
- **Syntax :**

SELE SELECT column-names
from table-name
where condition
group by column-names

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
```

Query : Find the total marks of each student in each city

```
SELECT SUM(marks), city
FROM Student
GROUP BY city
```


(3) Having

- HAVING filters records that work on summarized GROUP BY results.
- HAVING applies to summarized group records, whereas WHERE applies to individual records.
- Only the groups that meet the HAVING criteria will be returned.
- HAVING requires that a GROUP BY clause is present.
- **Syntax :**

```
SELECT column-names
FROM table-name
WHERE condition
GROUP BY column-names
HAVING condition
```

- **Example**

```
SELECT SUM(marks), city
FROM Student
GROUP BY city
HAVING city IN('Pune','Mumbai')
```

3.5 Schema Change Statements

Q.11 Describe DROP TABLE command of SQL with both the options CASCADE and RESTRICT. [SPPU : Nov.-18, Marks 5]

Ans. :

- The DROP command is used to remove the object (table, domains and constraints) from the database. There are two options for the DROP command - CASCADE and RESTRICT.
- To use the RESTRICT option, the user must first individually drop each element in the schema, then drop the schema itself. That means, the schema is dropped only if it has no elements in it, otherwise the DROP command can not be executed.

- Otherwise to remove completely some database schema CASCADE option is chosen. For example – to remove the Student_database

```
DROP SCHEMA Student_database CASCADE;
```

- If the table is to be deleted then the SQL command would be -

```
DROP TABLE Student CASCADE;
```

- The DROP TABLE command not only deletes all the records in the table if successful, but also removes the table definition from the catalog. If it is desired to delete only the records but to leave the table definition for future use, then the DELETE command. For example -
- The following SQL statement deletes all rows in the "Students" table, without deleting the table :

```
DELETE FROM Students;
```

3.6 Aggregate Functions

Q.12 What is aggregate function ? Explain with suitable examples.

Ans. : • An aggregate function allows you to perform a calculation on a set of values to return a single scalar value.

- SQL offers five built-in aggregate functions :
  1. Average : avg
  2. Minimum : min
  3. Maximum : max
  4. Total : sum
  5. Count :
- Syntax of all the Aggregate Functions

```
AVG( [ DISTINCT | ALL ] expression)
COUNT(*)
COUNT( [ DISTINCT | ALL ] expression )
MAX( [ DISTINCT | ALL ] expression)
MIN( [ DISTINCT | ALL ] expression)
```



```
SUM( [ DISTINCT | ALL ] expression)
```

- The avg function is used to compute average value. For example - To compute average marks of the students we can use

```
SQL Statement
SELECT AVG(marks)
FROM Students
```

- The **Count** function is used to count the total number of values in the specified field. It works on both numeric and non-numeric data type. COUNT (*) is a special implementation of the COUNT function that returns the count of all the rows in a specified table. COUNT (*) also considers Nulls and duplicates. For example Consider following table

Test

id	value
11	100
22	200
33	300
NULL	400

SQL Statement

```
SELECT COUNT(*)
FROM Test
Output
4
```

- The **min** function is used to get the minimum value from the specified column. For example - Consider the above created **Test** table

```
SQL Statement
SELECT Min(value)
FROM Test
Output
100
```

- The max function is used to get the maximum value from the specified column. For example - Consider the above created **Test** table

```
SQL Statement
SELECT Max(value)
FROM Test
Output
400
```

- The sum function is used to get total sum value from the specified column. For example - Consider the above created **Test** table

```
SQL Statement
SELECT sum(value)
FROM Test
Output
1000
```

Q.13 Consider, the following database,
Student(RollNo, Name, Address)
Subject(Sub_code, Sub_Name)
Marks (Roll_no, Sub_code, Marks)
Write following queries in SQL.
Find average marks of each student, along with the name of Student. [SPPU : Nov.-17, (End Sem), Marks 2]

Solution:

```
SELECT Name, AVG(Marks)
FROM Student,Marks
WHERE Student.Roll_No=Marks.Roll_No
```

3.7 Built-in Functions

Q.14 List and explain any five mathematical functions.

Ans. : Mathematical Functions

Function	Value Returned
ABS (m)	Absolute value of m

MOD (m, n)	Remainder of m divided by n
POWER (m, n)	m raised to the nth power
ROUND (m [, n])	m rounded to the nth decimal place
TRUNC (m [, n])	m truncated to the nth decimal place
SIN (n)	sine (n)
COS (n)	cosine (n)
TAN (n)	tan (n)
SQRT (n)	positive square root of n

Q.15 List and explain any five string functions.

Ans. : String Functions

Function	Value Returned
INITCAP (s)	First letter of each word is changed to uppercase and all other letters are in lower case.
LOWER (s)	All letters are changed to lowercase.
UPPER (s)	All letters are changed to uppercase.
CONCAT (s1, s2)	Concatenation of s1 and s2. Equivalent to s1 || s2.
LTRIM (s [, set])	Returns s with characters removed up to the first character not in set; defaults to space.
RTRIM (s [, set])	Returns s with final characters removed after the last character not in set; defaults to space.
REPLACE (s, search_s [, replace_s])	Returns s with every occurrence of search_s in s replaced by replace_s; default removes search_s.
LENGTH (s)	Returns the number of characters in s.

3.8 Set Operations

Q.16 Explain various set operations using SQL.

Ans. :

1) **Union :** To use this UNION clause, each SELECT statement must have

i) The same number of columns selected

ii) The same number of column expressions

iii) The same data type and

iv) Have them in the same order

This clause is used to **combine two tables** using UNION operator. It replaces the OR operator in the query. The union operator **eliminates** duplicate while the union all query will retain the duplicates.

Syntax :

The basic syntax of a UNION clause is as follows -

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
UNION
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Here, the given condition could be any given expression based on your requirement.

Example : Find the names of the students who have reserved the 'DBMS' book or 'OS' Book

The query can then be written by considering the Student, Reserve and Book table as


```
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
UNION
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

2) **Intersect :** The common entries between the two tables can be represented with the help of Intersect operator. It replaces the AND operator in the query.

Syntax :

The basic syntax of a INTERSECT clause is as follows -

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
INTERSECT
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Example : Find the students who have reserved both the 'DBMS' book and 'OS' Book

The query can then be written by considering the Student, Reserve and Book table as

```
SELECT S.sid, S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
INTERSECT
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

3) **Except :** The EXCEPT clause is used to represent the set-difference in the query. This query is used to represent the entries that are present in one table and not in other.

Syntax :

The basic syntax of a EXCEPT clause is as follows -

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
EXCEPT
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Example : Find the students who have reserved both the 'DBMS' book but not reserved 'OS' Book

The query can then be written by considering the Student, Reserve and Book table as

```
SELECT S.sid, S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
EXCEPT
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

3.9 Nested Queries

Q.17 What is co-related query ? Explain it with suitable example.

Ans. : In co-related nested queries, the output of inner query depends on the row which is being currently executed in outer query. For example To find names of employee who are department id 103

- If we want to find out sname of Student who live in city with cid as 101, it can be done with the help of co-related nested query as :

```
SELECT sname
FROM Student S
WHERE EXISTS
( SELECT *
```

SELECT E.ename
FROM Employee E
Where EXISTS(SELECT*FROM DepartmentD
WHERE D.id=103 AND D.id=E.id)


```
FROM Student_City SC
WHERE S.sid=SC.sid and SC.cid=101
)
```

- Here For each row of **Student S**, it will find the rows from **Student_City** where S.sid = SC.sid and SC.cid=101.
- If for a **sid** from **Student S**, atleast a row exists in **Student_City** SC with **cid**=101, then inner query will return true and corresponding **sid** will be returned as output.

3.10 Join and its Types

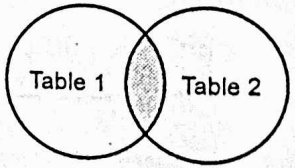
Q.18 Explain join operations with example.

☞ [SPPU : Nov.-19, End Sem, Marks 6, Nov.-17, End Sem, Marks 5]

Ans. :

1) Inner Join :

- The most important and frequently used of the joins is the INNER JOIN. They are also known as an EQUIJOIN.
- The INNER JOIN creates a new result table by combining column values of two tables (Table1 and Table2) based upon the join-predicate.
- The query compares each row of table1 with each row of Table2 to find all pairs of rows which satisfy the join-predicate.
- When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row. It can be represented as :



- **Syntax :** The basic syntax of the INNER JOIN is as follows.

```
SELECT Table1.column1, Table2.column2...
```

```
FROM Table1
INNER JOIN Table2
ON Table1.common_field = Table2.common_field;
```

- **Example :** For above given two tables namely Student and City, we can apply inner join. It will return the record that are matching in both tables using the common column cid.

The query will be

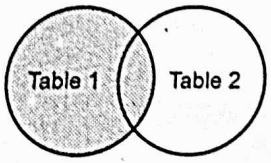
```
SELECT *
FROM Student Inner Join City on Student.cid=City.cid
```

The result will be

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai

2) Left Join(Outer Join) :

- The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in the right table; the join will still return a row in the result, but with NULL in each column from the right table.
- This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.
- It can be represented as -



- **Syntax :** The basic syntax of a LEFT JOIN is as follows.

```
SELECT
SELECT Table1.column1, Table2.column2...
```



```
FROM Table1
LEFT JOIN Table2
ON Table1.common_field = Table2.common_field;
```

- **Example :** For above given two tables namely Student and City, we can apply Left join. It will Return all records from the left table, and the matched records from the right table using the common column cid. The query will be

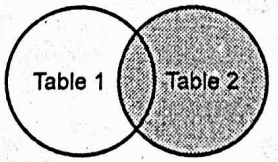
```
SELECT *
FROM Student Left Join City on Student.cid=City.cid
```

The result will be

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL

3) Right Join(Outer Join) :

- The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table.
- This means that if the ON clause matches 0 (zero) records in the left table; the join will still return a row in the result, but with NULL in each column from the left table.
- This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.
- It can be represented as follows :



- **Syntax :** The basic syntax of a RIGHT JOIN is as follow -

```
SELECT Table1.column1, Table2.column2...
```

```
FROM Table1
RIGHT JOIN Table2
ON Table1.common_field = Table2.common_field;
```

- **Example :** For above given two tables namely Student and City, we can apply Right join. It will return all records from the right table, and the matched records from the left table using the common column cid. The query will be

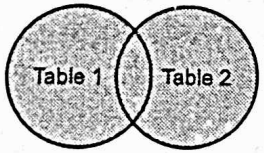
```
SELECT *
FROM Student Right Join City on Student.cid=City.cid
```

The result will be -

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
NULL	NULL	NULL	103	Chennai

4) Full Join (Outer Join) :

- The SQL FULL JOIN combines the results of both left and right outer joins.
- The joined table will contain all records from both the tables and fill in NULLs for missing matches on either side.
- It can be represented as,



- **Syntax :** The basic syntax of a FULL JOIN is as follows :

```
SELECT Table1.column1, Table2.column2...
FROM Table1 FULL JOIN Table2 ON Table1.common_field
= Table2.common_field;
```

The result will be -

- **Example :** For above given two tables namely **Student** and **City**, we can apply full join. It will return returns rows when there is a match in one of the tables using the common column cid. The query will be -

```
SELECT *
FROM Student Full Join City on Student.cid=City.cid
```

The result will be -

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL
NULL	NULL	NULL	103	Chennai

3.11 Exist, Any, All

Q.19 Explain Exists operator in SQL with suitable example.

☞ [SPPU : Nov.-19, Marks 5]

Ans. :

- The **EXISTS** operator is used to test for the existence of any record in a subquery. The **EXISTS** operator returns true if the subquery returns one or more records.
- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

- Example - Consider two tables namely **Student** and **Student_Score** table.
- If we want to get all students that have received a 10 grade in the subject 'Maths'.

- The SQL query using EXISTS can be written as follows -

```
SELECT
  RollNo, FName, LName
FROM
  Student
WHERE EXISTS (
  SELECT Student_Score.RollNo
  FROM
    Student_Score
  WHERE
    Student_Score.RollNo = Student.RollNo AND
    Student_Score.grade = 10 AND
    Student_Score.Subject = 'Maths'
)
```

Q.20 Explain Any and All Operators.

Ans. :

- The Any operator returns a Boolean value as a result. It returns true if any of the subquery meets the condition. When we use ANY operator then that means the condition will be true if the operation is true for any value in the range.
- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
 (SELECT column_name
   FROM table_name
   WHERE condition);
```

- Query : Find the name of the students who have scored the grade greater than 9.5 in any test.

We will write the query as

```
SELECT FName,LName
FROM Student
WHERE RollNo = ANY
 (SELECT RollNo
 FROM Student_Score
```



```
WHERE Grade > 9.5);
```

- The ALL operator returns a boolean value as result. It returns True if all of the subquery values meet the condition. When we use ALL operator then that means the condition will be true if the operation is true for all values in the range.
- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
 (SELECT column_name
   FROM table_name
   WHERE condition);
```

- **For example -** We will again consider the above specified **Student** and **Student_Score** tables. **Query :** Find the name of the students who have appeared only for three tests
- The SQL statement will be as follows -

```
SELECT FName,LName
FROM Student
WHERE RollNo = ALL
 (SELECT RollNo
 FROM Student_Score
 WHERE count(Test_no) = 3);
```

3.12 View and its Types

Q.21 Explain the concept of view along with its operations.

Ans. :

- 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 View

We can create a view using CREATE VIEW statement. The syntax is

```
CREATE VIEW name_of_view AS
SELECT column1,column2,...
FROM table_name1,table_name2,...
WHERE condition;
```

Example

i) **Creating a view using single table :** Consider table Empolyee and create a view EmployeeDetails whose Salary is <10000

EmpID	EName	Salary
101	Archana	20000
102	Madhura	5000
103	Poonam	8000
104	Sharda	15000
105	Monika	7000

```
CREATE VIEW EmployeeDetails(EmpID, EName) AS
SELECT E.EmpID, E.EName
OM Employee E
ERE E.Salary > 10000
```

The Output will be -

EmpID	EName
102	Madhura
103	Poonam

105	Monika

View Update

- The SQL UPDATE VIEW command can be used to modify the data of a view.
- All views are not updatable. So, UPDATE command is not applicable to all views.
- An updatable view is one which allows performing a UPDATE command on itself without affecting any other table.
- **Syntax for updating view**

```
UPDATE < view_name >
SET<column1>=<value1>,<column2>=<value2>,....
WHERE <condition>;
```

- **Example**

```
UPDATE VIEW Employee_dept_Details
SET Salary=1000
WHERE EName='Monika';
This view can be viewed by using following query
SELECT * FROM Employee_dept_Details;
```

3.13 Transaction Control Commands

Q.22 What are three types of TCL commands ? Illustrate.

Ans. : The COMMIT, ROLLBACK, and SAVEPOINT are collectively considered as Transaction Commands

1) **COMMIT :** The COMMIT command is used to save permanently any transaction to database. When we perform, Read or Write operations to the database then those changes can be undone by rollback operations. To make these changes permanent, we should make use of **commit**

2) **ROLLBACK :** The ROLLBACK command is used to undo transactions that have not already saved to database. For example

used to get back to previous permanent status of the table. similar to undo (ctrl+z)

- Consider the database table as

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig. 3.23.1 Student table

- Following command will delete the record from the database, but if we immediately performs ROLLBACK, then this deletion is undone.

For instance -

```
DELETE FROM Student
WHERE RollNo =2;
ROLLBACK;
```

Then the resultant table will be

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

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 SAVEPOINT can be created as

```
SAVEPOINT savepoint_name;
```

- Then we can ROLLBACK to SAVEPOIT as

```
ROLLBACK TO savepoint_name;
```

used along with the ROLLBACK command
It is used to mark a trasaction in table

- For example - Consider Student table as follows -

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig. 3.23.2 Student table

- Consider Following commands

```
SQL> SAVEPOINT S1
SQL>DELETE FROM Student
Where RollNo=2;
SQL> SAVEPOINT S2
SQL>DELETE FROM Student
Where RollNo=3;
SQL> SAVEPOINT S3
SQL>DELETE FROM Student
Where RollNo =4
SQL> SAVEPOINT S4
SQL>DELETE FROM Student
Where RollNo=5
SQL> ROLLBACK TO S3;
```

- Then the resultant table will be

RollNo	Name
1	AAA
2	BBB
3	CCC

- Thus the effect of deleting the record having RollNo 2, and RollNo 3 is undone.

Q.23 Schema definition for supplier-and-parts database.
Supplier = (supplier_number, supplier_name, status, city)
Parts = (part_number, part_name, colour, weight, city)
Shipments = (supplier_number, part_number, quantity)
Write SQL query for following requirements (any 2).
i) Find shipment information (supplier_number, supplier_name, part_number, part_name, quantity) for those having quantity less than 150.
ii) List supplier_number, suppler_name, part_number, part_name for those suppliers who made shipment of parts whose quantity is larger than the average quantity.
iii) Find aggregate quantity of part_number 'A692' of colour 'GREEN' for which shipment made by supplier_number who reside in 'MUMBAI'.
Ans. : ☞ [SPPU : May-18, End Sem, Marks 5]

```
i)  SELECT
    supplier_number,supplier_name,part_number,part_name,quantity
    FROM Supplier, Parts, Shipments
    WHERE Supplier.supplier_number = Shipments.supplier_number AND
    Parts.part_number=Shipments.part_number
    AND Shipments.quantity<150
ii) SELECT supplier_number, suppler_name, part_number, part_name
    FROM Supplier, Parts
    WHERE Supplier.supplier_number = Shipments.supplier_number AND
    Part.part_number= Shipments.part_number AND
    Shipments.quantity>(SELECT AVG(quantity) FROM Shipments)
iii) SELECT COUNT(part_number)
    FROM Parts, Supplier, Shipments
    WHERE Supplier.supplier_number = Shipments.supplier_number AND
    Part.part_number= Shipments.part_number AND
    Parts.part_number = 'A692' AND Parts.colour='GREEN' AND
    Supplier.city='MUMBAI'
```

Q.24 Consider the following database schema :
Loan (loan_no, Branch_name, Amount)
Borrower (Cust_name, Loan no).
Write SQL queries for following requirements (any two) :
i) Find all customers who have a loan from bank. Find their names, loan nos. and Loan amount.
ii) Find names of customers in alphabetical order who have a loan at Pune branch.

iii) Find all loan nos. for loan made at Pune branch with loan amount greater than 20000. ☞ [SPPU : Oct.-18, End Sem, Marks 5]
Solution :

```
(i)  SELECT Cust_name Loan_no, Amount
     FROM Loan, Borrower
     WHERE Loan.Loan_no = Borrower.Loan_no
(ii) SELECT Cust_name
     FROM Borrower, Loan
     WHERE Loan.Branch_name = 'Pune'
     ORDER BY Cust_name ASC
(iii) SELECT Loan_no
     FROM Loan, Borrower
     WHERE Loan.Loan_no = Borrower.Loan_no AND Loan.Branch_name
= 'Pune' AND
     Loan.Amount>20000
```

Q.25 Consider Employee database with following schema :
Employee(Emp_Id, First_Name, Last_Name, Salary, Joining_Date, Department)
Bonus(Emp_Ref_Id, Bonus_Amount, Bonus_Date)
Designation(Emp_Ref_Id, Emp_Designation, Affected_From)
Write queries in SQL for following requirements (any 2) :
i) To fetch the departments that have less than five people in it.
ii) To print the name of employees having the highest salary in each department.
iii) Write an SQL query to print details of the employee who are also Managers. ☞ [SPPU : Dec.-18, End Sem, Marks 5]
Solution :

```
(i)  SELECT Department, COUNT(Emp_Id) as 'Number of People'
     FROM Employee GROUP BY Department
     HAVING COUNT(Emp_Id) <5
(ii) SELECT E.Department,t.First_Name,t.Salary
     FROM
     (SELECT MAX(Salary) AS TotalSalary, Department FROM
     Employee GROUP BY Department)
     AS temp
     INNER JOIN ON temp.Department=E.Department
     AND temp.TotalSalary=E.Salary;
(iii) SELECT DISTINCT E.FIRST_NAME, D.Emp_Designation
     FROM Employee E, Designation D
```

```
WHERE E.Emp_Id = D.Emp_Ref_Id
AND D.Emp_Designation IN ('Manager');
```

Q.26 Consider following schema :
account (acct-no, branch - name, balance)
Depositor (cust - name, acct - no)
borrower (cust-name, loan-no)
loan (loan - no, branch - name, amount)
Write following queries using SQL (any 2)
i) Find names of all customers who have a loan at the redwood branch.
ii) Find all customers who are having an account and loan or both.
iii) Find average account balance at each branch.

☞ [SPPU : May-19, End Sem, Marks 5]

Ans. :

```
(i) SELECT DISTINCT cust-name
    FROM borrower, loan
    WHERE borrower.loan-no = loan.loan-no AND loan.branch-name
    = 'redwood'
(ii) SELECT cust-name FROM Depositor
    UNION
    SELECT cust-name FROM borrower
(iii) SELECT branch-name, AVG(balance)
    FROM account
    GROUP BY branch-name
```

Q.27 Emp(E_number, E_name, Dept_no)
Dept(Dept_no, Dept_name).
Consider the schema given above. Consider above tables are created without considering the Dept_no as primary key in Dept table and foreign key in Emp table. Assuming tables are already created write SQL queries for following requirements.
i) Create primary key in dept table considering above situations.
ii) Create foreign key considering EMP as child table and dept as master table also consider the above situation.
iii) Add column salary with appropriate data type in EMP table.

☞ [SPPU : Oct.-19, In Sem, Marks 5]

Solution :

```
i) ALTER TABLE Dept
   ADD PRIMARY KEY(Dept_no)
ii) ALTER TABLE Emp
   ADD CONSTRAINT FK_deptno
```



```
   FOREIGN KEY (Dept_no) REFERENCES Dept(Dept_no);
iii) ALTER TABLE Emp
   ADD Salary int;
```

3.14 PL/SQL Concepts

Q.28 What is PL/SQL ?
Ans. :

- PL/SQL stands for Procedural Language extensions to the Structured Query Language (SQL).
- PL/SQL is a combination of SQL along with the procedural features of programming languages.
- Oracle uses a PL/SQL engine to processes the PL/SQL statements.
- PL/SQL includes procedural language elements like conditions and loops. It allows declaration of constants and variables, procedures and functions, types and variable of those types and triggers.

Q.29 Write the PL/SQL block of code to calculate the factorial value of a number. ☞ [SPPU : May-19, Dec.-19, End Sem, Marks 5]

Solution :

```
SET SERVEROUTPUT ON;
DECLARE
-- The factorial f is initialized to 1
 f NUMBER := 1;
--user inputs the number here in variable n
 n NUMBER :=&1;
 num NUMBER := n;
BEGIN
 WHILE n>0 LOOP
  f:=n*f;
  n:=n-1;
 END LOOP;
 DBMS_OUTPUT.put_line('Factorial of ' || num || ' is ' || f);
END;
/
```

```
SET Serveroutpu on;
Declare
nnumber;
fact number := 1;
number;
begin
n := &n;
for i in 1 .. n
loop
fact := fact*i;
end loop;
Dbms_output.put_line('Factorial
'||fact
end;
/
```

Output

```
Factorial of 7 is 5040
```

Q.30 Write PI/SQL block of code for following requirement :
Student_fees (PRN, S_name, class, fees_paid).
Accept the PRN of student from user, check the fees paid by student, if fees paid is less than 30,000 then display the message on screen not paid full fees and display the total fees due. If fees_paid is greater than or equal to 30,000 then display message no fees due. ☞ [SPPU : Oct.-19, In Sem, Marks 5]

Ans. :

PL/SQL Program

```
SET SERVEROUTPUT ON;
DECLARE
 temp_PRN NUMBER(5);
 temp_fees NUMBER(10);
BEGIN
 temp_PRN := &temp_PRN;
 SELECT fees_paid INTO temp_fees FROM STUDENT_FEES WHERE PRN
= temp_PRN;
 DBMS_OUTPUT.put_line('PRN = ' || temp_prn);
 DBMS_OUTPUT.put_line('Fees(Rs.) = '|| temp_fees);
 IF(temp_fees<30000) THEN
  DBMS_OUTPUT.put_line('The Feees not Paid Fully');
 ELSE
  DBMS_OUTPUT.put_line('No Fees Due');
 END IF;
END;
/
```

Output(Run1)

```
PRN = 101
Fees(Rs.) = 30000
No Fees Due
```


3.15 Cursors

Q.31 Write short note on - Cursor.
Ans. :

- When an SQL statement is processed, Oracle creates a memory area known as **context area.** A cursor is a **pointer to this context area.**
- It contains all information needed for processing the statement.
- In PL/SQL, the context area is controlled by cursor.
- A cursor contains information on a SELECT statement and the rows of data accessed by it.
- The cursor is used to fetch and process the rows returned by SQL statement one at a time.

Q.32 Explain Explicit cursor.
Ans. :

- Explicit cursors are used when you are executing a SELECT statement query that will return more than one row.
- Cursors can process one record at a given point of time even though it stores more than one record.
- An explicit cursor is defined in the declaration section of the PL/SQL Block.
- Explicit cursors are used if you need to have better control over the Context Area via Cursor.

Syntax for Declaration of Explicit Cursor

```
CURSOR cusrsor_name
IS
SELECT statement
```

For example

```
CURSOR MyCursor
IS
SELECT * FROM Student;
```

3.16 Stored Procedures

Q.33 Write a PL/SQL code for addition of two numbers using procedure.

Ans. : Following example show the procedure for addition of two numbers. The result is stored in third variable which is an output variable.

Step 1 : We will create a procedure and store it in an sql file as follows

```
CREATE OR REPLACE PROCEDURE AddProc(x IN NUMPER, y IN
NUMBER, z OUT NUMBER)
IS
BEGIN
 z:=x+y;
END;
/
```

Two Input and One output parameter

Step 2 : The call to the procedure can be made from another file. The code for calling the procedure for execution is as follows -

```
SET SERVEROUTPUT ON;
DECLARE
a NUMBER;
b NUMBER;
c NUMBER;

BEGIN
 a := 10;
 b := 20;
 AddProc(a,b,c);
 dbms_output.put_line('Addition of two numbers:'||c);
END;
```

3.17 Stored Function

Q.34 How to write functions in PL/SQL ? Give its syntax and examples.

Ans. :

- Stored function is a named block or subprogram in PL/SQL.

- In PL/SQL, a function takes one or more parameter and returns one value.
- The syntax for a function in PL/SQL is as follows :

```
CREATE [Or REPLACE] Function Function_Name
[(Parameter,..)]
Return Datatype
[IS | AS]
[Declaration Section]
BEGIN
[Executable Section]
END Function_Name;
```

Q.35 Write PL/SQL block of code which accepts the roll.no. from user, the attendance of roll no entered by user will be checked in stud_att (Roll_no, Att) table. Attendance of Roll no entered is displayed on screen. ☞ [SPPU : Aug.-17, End Sem, Marks 5]

Ans. : PL/SQL Program

Step 1 : Create a function in a sql file as

```
example.sql
CREATE OR REPLACE FUNCTION Display(roll NUMBER)
 RETURN NUMBER IS
  temp_att NUMBER(5);
BEGIN
  SELECT att INTO temp_att FROM TESTUDENTS WHERE roll_no = roll;
  RETUFN temp_att;
END;
/
```

Step 2 : Following is a driver program that calls above created function

```
driver.sql
SET SERVEROUTPUT ON;
DECLARE
 temp_rollno NUMBER(3);
 temp_att NUMBER(5);
 total_days NUMBER(5) := 200;
BEGIN
  temp_rollno := &temp_rollno;
```

```
  temp_att := Display(temp_rollno);
  DBMS_OUTPUT.put_line('Roll No = ' || temp_rollno||' Attendance = '||
temp_att);
END;
/
```

Output

```
Roll No = 4 Attendance = 100
```

3.18 Database Triggers

Q.36 What is trigger ?
Ans. :

- Trigger is something that is invoked automatically when some event occurs.
- PL/SQL triggers are **block structures or pre-defined programs,** which may be in-built or even explicitly developed by the programmers for a particular task.
- Trigger is **stored into database** and invoked repeatedly, when specific condition match.
- Triggers are stored programs, which are automatically executed or fired when some event occurs.
- Triggers are associated with response-based events such as a,
  - Database Definition Language (DDL) statement such as CREATE, DROP or ALTER .
  - Database Manipulation Language (DML) statement such as UPDATE, INSERT or DELETE.
  - Any other database operation such as a Startup, Shutdown, Logging in and Logging Out.

Syntax

```
CREATE OR REPLACE TRIGGER Trigger_Name
BEFORE or AFTER or INSTEAD OF
INSERT or UPDATE or DELETE
```



```
of Column_Name
ON Table_Name
[REFERENCING OLD AS O New AS N]
FOR EACH ROW
WHEN (Condition)
DECLARE
Declaration Section
BEGIN
Execution Section
END;
```

Where

- CREATE [OR REPLACE] TRIGGER trigger_name: It creates or replaces an existing trigger with some trigger_name.
- {BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} : This specifies the DML operation.
- [OF col_name] : This specifies the column name that would be updated.
- [ON table_name] : This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] : This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.

- WHEN (condition) : This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

Q.37 Write a PL/SQL trigger for following requirement :
Event : Deletion of row from stud (roll_no, name, class) table.
Action : after deletion of values from stud table, values should be inserted into cancel_admission (roll_no, name) table
Note : for every row to be deleted, action should be performed.

☞ [SPPU : Dec.-17, End Sem, Marks 5]

Ans. :

Step 1 : Create table **stud_table** using following SQL statement

```
CREATE TABLE stud_table(
  roll_no NUMBER(5),
  sname VARCHAR2(30),
  sclass VARCHAR2(20)
);
```

Step 2 : Insert data into table using following commands

```
INSERT INTO stud_table VALUES(111,'AAA','TECOMP');
INSERT INTO stud_table VALUES(222,'BBB','SEMECH');
INSERT INTO stud_table VALUES(333,'CCC','BECIVIL');
INSERT INTO stud_table VALUES(444,'DDD','TEETC');
```

Step 3 : Create table **cancel_admission** using following SQL statement

```
CREATE TABLE cancel_admission(
  roll_no NUMBER(5),
  sname VARCHAR2(30)
);
```

Step 4 : Following is a PL/SQL program that deletes data from **stud_table** and insert the deleted data into **cancel_admission** table

```
TriggerDemo1.sql
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER Stud_AD
AFTER DELETE  ON stud_table
FOR EACH ROW
ENABLE
```



```
BEGIN
 INSERT INTO cancel_admission
 (roll_no,
  sname)
 VALUES
 ( :old.roll_no,
  :old.sname);
END;
/

DELETE FROM stud_table WHERE roll_no=222
```

Output

On compiling above trigger we get the output as follows -

```
TRIGGER Stud_AD Compiled.
1 rows deleted
```

The execution of this trigger can be verified by observing the data from student_table and cancel_admission table.

Stud_table Table

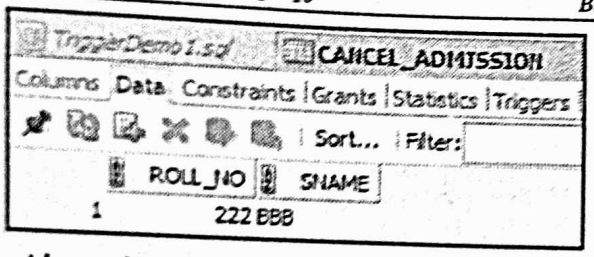
TriggerDemo1.sql | STUD_TABLE

Columns | Data | Constraints | Grants | Statistics | Tr

Sort... | Filter:

	ROLL_NO	SNAME	SCLASS
1	111	AAA	TECOMP
2	333	CCC	BECIVIL
3	444	DDD	TEETC

Cancel_admission Table



Q.38 Consider the schema : student_fees_detail (name, total_fees_deposited, till_date)
Answer the following :
i) Write a SQL query to display the total fees deposited by students whose minimum 3 character name starts with aj.
ii) Write Database Trigger to preserve the old values of student fees details before updating in table.

[SPPU : Dec.-18,19, End Sem, Marks 5]

Ans. : i) SQL Query is as follows -

```
SELECT name,total_fees_deposited
FROM student_fees_detail
WHERE LEN(name)>2 AND name LIKE 'aj%'
```

ii)

Step 1 : First of all we will create the table **stud_fees_detail** using following SQL statement.

```
CREATE TABLE stud_fees_detail
 (
  name VARCHAR2(30),
  fees_deposited NUMBER(5),
  till_date DATE
);
```

Step 2 : Insert values the values into this table using following SQL statement -

```
INSERT INTO stud_fees_detail VALUES('AAA',2000,'27-JUNE-19');
INSERT INTO stud_fees_detail VALUES('BBB',5000,'16-May-20');
INSERT INTO stud_fees_detail VALUES('CCC',1000,'01-August-19');
INSERT INTO stud_fees_detail VALUES('DDD',3000,'25-March-20');
INSERT INTO stud_fees_detail VALUES('EEE',2500,'12-November-20');
```


Step 3 : Simply create a table **backup_fees** using following SQL statement -

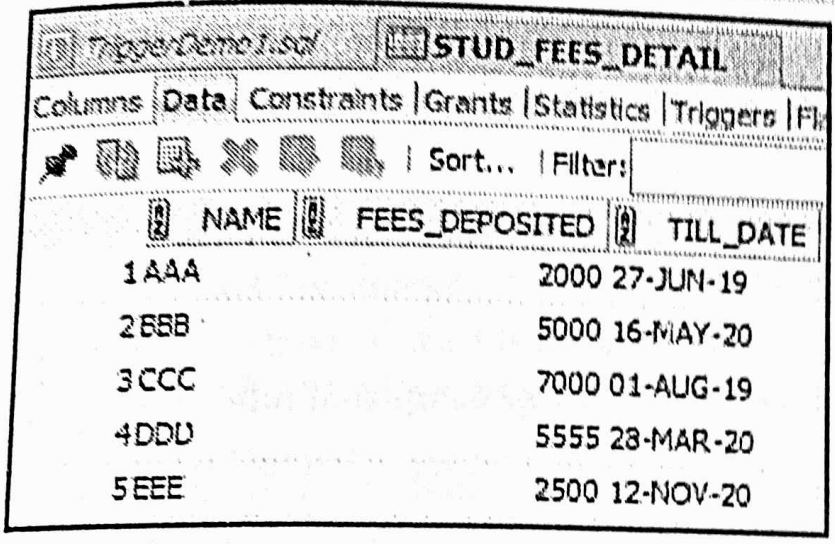
```
CREATE TABLE backup_fees
 (
  name VARCHAR2(30),
  fees_deposited NUMBER(5),
  till_date DATE
);
```

Step 4 : The PL/SQL script for the updating values is as shown below

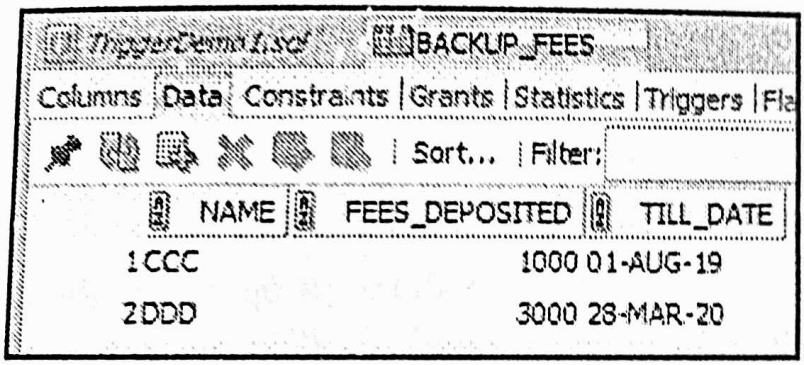
```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER Stud_fees_AU
AFTER UPDATE  ON stud_fees_detail
FOR EACH ROW
ENABLE

BEGIN
 INSERT INTO backup_fees
 (name,
 fees_deposited,
 till_date
 )
 VALUES
 (:old.name,
  :old.fees_deposited,
  :old.till_date
  );
END;
/
UPDATE stud_fees_detail
SET fees_deposited=7000
WHERE name='CCC'

UPDATE stud_fees_detail
SET fees_deposited=5555
WHERE name='DDD'
```

Output

Stud_fees_detail Table(This table is updated as trigger executes)



Backup_fees Table



END ...