**5. Apply Integrity Constraints**

CREATE TABLE Employees (

EmpID INT PRIMARY KEY, -- Primary Key

Name VARCHAR(50) NOT NULL, -- Not Null

Age INT CHECK (Age >= 18 AND Age <= 65), -- Check Constraint

Salary DECIMAL(10, 2)

);

**6. DML Commands**

-- Insert data

INSERT INTO Employees (EmpID, Name, Age, Salary)

VALUES (1, 'John Doe', 30, 50000.00);

-- Update data

UPDATE Employees

SET Salary = 55000.00

WHERE EmpID = 1;

-- Delete data

DELETE FROM Employees

WHERE EmpID = 1;

**7. Queries with Operators**

**(i) Range Searching - BETWEEN**

SELECT \* FROM Employees

WHERE Age BETWEEN 25 AND 40;

**(ii) Pattern Matching - LIKE**

SELECT \* FROM Employees

WHERE Name LIKE 'J%'; -- Names starting with J

**8. Queries with Functions**

**(i) String Functions**

SELECT UPPER(Name) AS UpperCaseName, LENGTH(Name) AS NameLength

FROM Employees;

**(ii) Arithmetic Functions**

SELECT Salary \* 0.1 AS Bonus, Salary + (Salary \* 0.1) AS TotalPay

FROM Employees;

**9. Queries with Date, Time & Aggregate Functions**

**(i) Date & Time Functions**

SELECT CURRENT\_DATE AS Today, YEAR(CURRENT\_DATE) AS Year;

**(ii) Aggregate Functions**

SELECT AVG(Salary) AS AverageSalary, COUNT(\*) AS TotalEmployees

FROM Employees;

**10. SELECT with WHERE and HAVING Clauses**

**(i) WHERE Clause**

SELECT \* FROM Employees

WHERE Age > 30;

**(ii) HAVING Clause**

SELECT COUNT(\*), AVG(Salary)

FROM Employees

GROUP BY Age

HAVING AVG(Salary) > 40000;

**11. GROUP BY and ORDER BY Clauses**

**(i) GROUP BY**

SELECT Age, COUNT(\*) AS EmployeeCount

FROM Employees

GROUP BY Age;

**(ii) ORDER BY**

SELECT \* FROM Employees

ORDER BY Salary DESC;

**12. Joins**

**(i) Inner Join**

SELECT E.EmpID, E.Name, D.DeptName

FROM Employees E

INNER JOIN Departments D

ON E.EmpID = D.ManagerID;

**(ii) Outer Join**

SELECT E.EmpID, E.Name, D.DeptName

FROM Employees E

LEFT OUTER JOIN Departments D

ON E.EmpID = D.ManagerID;

**13. Views**

**(i) Create Views**

CREATE VIEW HighEarners AS

SELECT Name, Salary

FROM Employees

WHERE Salary > 50000;

**(ii) Insert, Modify, Delete via View**

INSERT INTO HighEarners (Name, Salary)

VALUES ('Alice', 60000.00);

UPDATE HighEarners

SET Salary = 65000.00

WHERE Name = 'Alice';

DELETE FROM HighEarners

WHERE Name = 'Alice';

**(iii) Delete View**

DROP VIEW HighEarners;

**14. Sequences**

CREATE SEQUENCE EmpSeq

START WITH 1

INCREMENT BY 1;

-- Using the sequence

INSERT INTO Employees (EmpID, Name, Age, Salary)

VALUES (EmpSeq.NEXTVAL, 'Mark', 28, 40000.00);

**15. Synonyms**

CREATE SYNONYM EmpSynonym FOR Employees;

-- Using the synonym

SELECT \* FROM EmpSynonym;

**16-19. PL/ Loops**

**IF THEN ELSE**

DECLARE

bonus NUMBER;

BEGIN

IF Salary > 50000 THEN

bonus := Salary \* 0.10;

ELSE

bonus := Salary \* 0.05;

END IF;

END;

**FOR Loop**

BEGIN

FOR i IN 1..10 LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || i);

END LOOP;

END;

**Reverse FOR Loop**

BEGIN

FOR i IN REVERSE 1..10 LOOP

DBMS\_OUTPUT.PUT\_LINE('Reverse Employee ID: ' || i);

END LOOP;

END;

**WHILE Loop**

DECLARE

i NUMBER := 1;

BEGIN

WHILE i <= 10 LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || i);

i := i + 1;

END LOOP;

END;

**20-21. Cursors**

**Implicit Cursor**

BEGIN

FOR rec IN (SELECT \* FROM Employees) LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || rec.Name);

END LOOP;

END;

**Explicit Cursor**

DECLARE

CURSOR EmpCursor IS SELECT Name FROM Employees;

EmpRec Employees%ROWTYPE;

BEGIN

OPEN EmpCursor;

FETCH EmpCursor INTO EmpRec;

WHILE EmpCursor%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || EmpRec.Name);

FETCH EmpCursor INTO EmpRec;

END LOOP;

CLOSE EmpCursor;

END;

**22-23. Exception Handling**

**Pre-defined Exception**

BEGIN

INSERT INTO Employees (EmpID, Name) VALUES (1, NULL);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred.');

END;

**User-defined Exception**

DECLARE

invalid\_salary EXCEPTION;

BEGIN

IF Salary < 0 THEN

RAISE invalid\_salary;

END IF;

EXCEPTION

WHEN invalid\_salary THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid Salary!');

END;

**24. DCL Commands**

**(i) Create Users**

CREATE USER new\_user IDENTIFIED BY password;

**(ii) Grant Privileges**

GRANT SELECT, INSERT ON Employees TO new\_user;