```
-------------------------------------------------
-- 3. DDL Commands with ALTER operations
-------------------------------------------------
CREATE TABLE student1 (
   id NUMBER(30),
   name VARCHAR2(15),
   salary NUMBER(20)
);

INSERT INTO student1 VALUES (10, 'Krishna', 25000);
INSERT INTO student1 VALUES (11, 'Ravi', 30000);
INSERT INTO student1 VALUES (12, 'Sneha', 28000);

ALTER TABLE student1 ADD age NUMBER(3);
ALTER TABLE student1 MODIFY name VARCHAR2(25);
ALTER TABLE student1 RENAME COLUMN salary TO fees;
ALTER TABLE student1 DROP COLUMN age;
ALTER TABLE student1 RENAME TO student_details;

SELECT * FROM student_details;

-------------------------------------------------
-- 4. Integrity Constraints
-------------------------------------------------
CREATE TABLE student2 (
   id NUMBER(5) PRIMARY KEY,
   name VARCHAR2(20) NOT NULL,
   salary NUMBER(10),
   age NUMBER(3) CHECK (age >= 18)
);

DESC student2;

INSERT INTO student2 VALUES (1, 'Krishna', 25000, 20);
INSERT INTO student2 VALUES (2, 'Raj', 30000, 22);

SELECT * FROM student2;

-------------------------------------------------
-- 5. DML Commands
-------------------------------------------------
CREATE TABLE student3 (
   id NUMBER(5) PRIMARY KEY,
   name VARCHAR2(20),
   salary NUMBER(10)
);

INSERT INTO student3 VALUES (1, 'Krishna', 25000);
INSERT INTO student3 VALUES (2, 'Ravi', 30000);
INSERT INTO student3 VALUES (3, 'Sneha', 28000);

UPDATE student3 SET salary = 32000 WHERE id = 2;
DELETE FROM student3 WHERE id = 3;

SELECT * FROM student3;
```

```
-------------------------------------------------
-- 6. TCL Commands
-------------------------------------------------
CREATE TABLE student_tcl (
   sid NUMBER(5) PRIMARY KEY,
   sname VARCHAR2(20),
   marks NUMBER(5)
);

INSERT INTO student_tcl VALUES (1, 'Rahul', 80);
INSERT INTO student_tcl VALUES (2, 'Priya', 90);

SAVEPOINT sp1;

UPDATE student_tcl SET marks = 85 WHERE sid = 1;

ROLLBACK TO sp1;

COMMIT;

SELECT * FROM student_tcl;

-------------------------------------------------
-- 7. Operators
-------------------------------------------------
CREATE TABLE student_operators (
   id NUMBER(5) PRIMARY KEY,
   name VARCHAR2(20),
   salary NUMBER(10)
);

INSERT INTO student_operators VALUES (1, 'Amit', 20000);
INSERT INTO student_operators VALUES (2, 'Rohit', 28000);
INSERT INTO student_operators VALUES (3, 'Neha', 35000);

SELECT * FROM student_operators WHERE salary BETWEEN 20000 AND 30000;
SELECT * FROM student_operators WHERE name LIKE 'N%';

-------------------------------------------------
-- 8. Functions
-------------------------------------------------
CREATE TABLE student_functions (
   id NUMBER(5) PRIMARY KEY,
   name VARCHAR2(20),
   salary NUMBER(10)
);

INSERT INTO student_functions VALUES (1, 'Kiran', 27000);
INSERT INTO student_functions VALUES (2, 'Sneha', 30000);

SELECT UPPER(name), LOWER(name), LENGTH(name) FROM student_functions;
SELECT id, salary, salary + 2000 AS Bonus, salary * 0.10 AS Hike FROM student_functions;
```

```
-------------------------------------------------
-- 9. Date & Aggregate Functions
-------------------------------------------------
CREATE TABLE student_functions2 (
   id NUMBER(5) PRIMARY KEY,
   name VARCHAR2(20),
   salary NUMBER(10)
);

INSERT INTO student_functions2 VALUES (1, 'Varun', 24000);
INSERT INTO student_functions2 VALUES (2, 'Riya', 35000);
INSERT INTO student_functions2 VALUES (3, 'Ankit', 28000);

SELECT SYSDATE AS Today, ADD_MONTHS(SYSDATE, 3) AS After_3_Months FROM dual;
SELECT COUNT(*), MAX(salary), MIN(salary), AVG(salary), SUM(salary) FROM student_functions2;

-------------------------------------------------
-- 10. SELECT Clauses
-------------------------------------------------
CREATE TABLE student_clauses (
   id NUMBER(5) PRIMARY KEY,
   name VARCHAR2(20),
   marks NUMBER(5)
);

INSERT INTO student_clauses VALUES (1, 'Rohan', 75);
INSERT INTO student_clauses VALUES (2, 'Mira', 88);
INSERT INTO student_clauses VALUES (3, 'Tina', 67);

SELECT * FROM student_clauses WHERE marks > 70;
SELECT name, AVG(marks) FROM student_clauses GROUP BY name HAVING AVG(marks) > 70;

-------------------------------------------------
-- 11. GROUP BY and ORDER BY
-------------------------------------------------
CREATE TABLE student_grouporder (
   id NUMBER(5) PRIMARY KEY,
   name VARCHAR2(20),
   city VARCHAR2(20)
);

INSERT INTO student_grouporder VALUES (1, 'Amit', 'Pune');
INSERT INTO student_grouporder VALUES (2, 'Neha', 'Mumbai');
INSERT INTO student_grouporder VALUES (3, 'Amit', 'Pune');

SELECT name, COUNT(*) FROM student_grouporder GROUP BY name;
SELECT * FROM student_grouporder ORDER BY city ASC;
```

```
-------------------------------------------------
-- 12. JOINS
-------------------------------------------------
CREATE TABLE student_join (
   sid NUMBER(5) PRIMARY KEY,
   sname VARCHAR2(20),
   dept_id NUMBER(5)
);

CREATE TABLE department_join (
   dept_id NUMBER(5) PRIMARY KEY,
   dept_name VARCHAR2(20)
);

INSERT INTO student_join VALUES (1, 'Kiran', 101);
INSERT INTO student_join VALUES (2, 'Ravi', 102);
INSERT INTO student_join VALUES (3, 'Neha', 103);

INSERT INTO department_join VALUES (101, 'IT');
INSERT INTO department_join VALUES (102, 'HR');
INSERT INTO department_join VALUES (104, 'Finance');

SELECT s.sid, s.sname, d.dept_name
FROM student_join s
INNER JOIN department_join d
ON s.dept_id = d.dept_id;

SELECT s.sid, s.sname, d.dept_name
FROM student_join s
LEFT OUTER JOIN department_join d
ON s.dept_id = d.dept_id;

-------------------------------------------------
-- 13. VIEWS
-------------------------------------------------
CREATE TABLE student_viewtable (
   id NUMBER(5) PRIMARY KEY,
   name VARCHAR2(20),
   salary NUMBER(10)
);

INSERT INTO student_viewtable VALUES (1, 'Raj', 25000);
INSERT INTO student_viewtable VALUES (2, 'Simran', 30000);

CREATE VIEW student_view AS
SELECT id, name FROM student_viewtable;

SELECT * FROM student_view;

INSERT INTO student_viewtable VALUES (3, 'Ravi', 28000);
UPDATE student_viewtable SET name = 'Rajan' WHERE id = 1;
DELETE FROM student_viewtable WHERE id = 2;

DROP VIEW student_view;
```

```
-----------------------------------------------
-- 14. SEQUENCE
-----------------------------------------------
CREATE SEQUENCE seq14_example
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 20;

CREATE TABLE student_sequence (
   id NUMBER(5) PRIMARY KEY,
   name VARCHAR2(20)
);

INSERT INTO student_sequence VALUES (seq14_example.NEXTVAL, 'Kunal');
INSERT INTO student_sequence VALUES (seq14_example.NEXTVAL, 'Pooja');
INSERT INTO student_sequence VALUES (seq14_example.NEXTVAL, 'Vikas');

SELECT * FROM student_sequence;
```

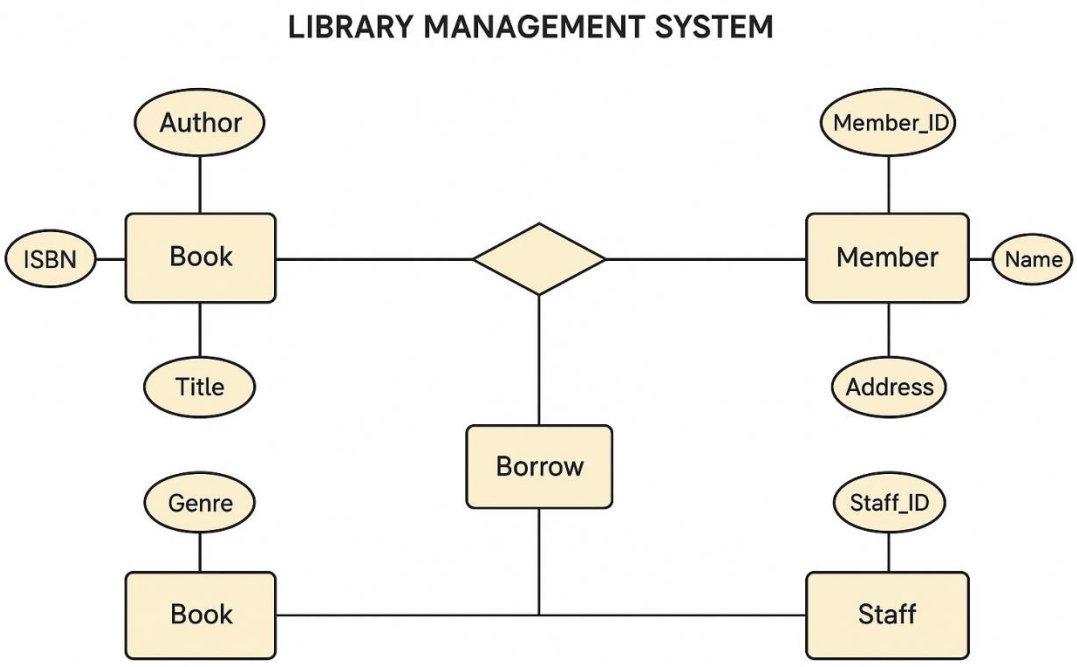
```
-------------------------------------------------
-- 15. IF THEN ELSE (Bind Variable)
-------------------------------------------------
DECLARE
   marks NUMBER := :marks;
   grade VARCHAR2(20);
BEGIN
   IF marks >= 75 THEN
      grade := 'Distinction';
   ELSIF marks >= 60 THEN
      grade := 'First Class';
   ELSE
      grade := 'Fail';
   END IF;
   DBMS_OUTPUT.PUT_LINE('Marks: ' || marks || ' -> Grade: ' || grade);
END;
/
-- Example: :marks = 80
-- Output: Marks: 80 -> Grade: Distinction

-------------------------------------------------
-- 16. FOR LOOP (Bind Variable)
-------------------------------------------------
DECLARE
   n NUMBER := :n;
   i NUMBER;
BEGIN
   FOR i IN 1..n LOOP
      DBMS_OUTPUT.PUT_LINE('Number: ' || i);
   END LOOP;
END;
/
-- Example: :n = 5

-------------------------------------------------
-- 17. REVERSE FOR LOOP (Bind Variable)
-------------------------------------------------
DECLARE
   n NUMBER := :n;
   i NUMBER;
BEGIN
   FOR i IN REVERSE 1..n LOOP
      DBMS_OUTPUT.PUT_LINE('Count: ' || i);
   END LOOP;
END;
/
-- Example: :n = 5
```

```
-------------------------------------------------
-- 18. WHILE LOOP (Bind Variable)
-------------------------------------------------
DECLARE
   limit NUMBER := :limit;
   num NUMBER := 1;
BEGIN
   WHILE num <= limit LOOP
      DBMS_OUTPUT.PUT_LINE('Value: ' || num);
      num := num + 1;
   END LOOP;
END;
/
-- Example: :limit = 5

-------------------------------------------------
-- 19. IMPLICIT CURSOR (Bind Variable)
-------------------------------------------------
CREATE TABLE emp_implicit (
   id NUMBER,
   name VARCHAR2(20),
   salary NUMBER
);

INSERT INTO emp_implicit VALUES (1, 'Krishna', 25000);
INSERT INTO emp_implicit VALUES (2, 'Ravi', 30000);
COMMIT;

DECLARE
   v_limit NUMBER := :salary_limit;
   v_count NUMBER;
BEGIN
   UPDATE emp_implicit SET salary = salary + 2000 WHERE salary < v_limit;
   v_count := SQL%ROWCOUNT;
   DBMS_OUTPUT.PUT_LINE('Rows Updated: ' || v_count);
END;
/
-- Example: :salary_limit = 28000
```

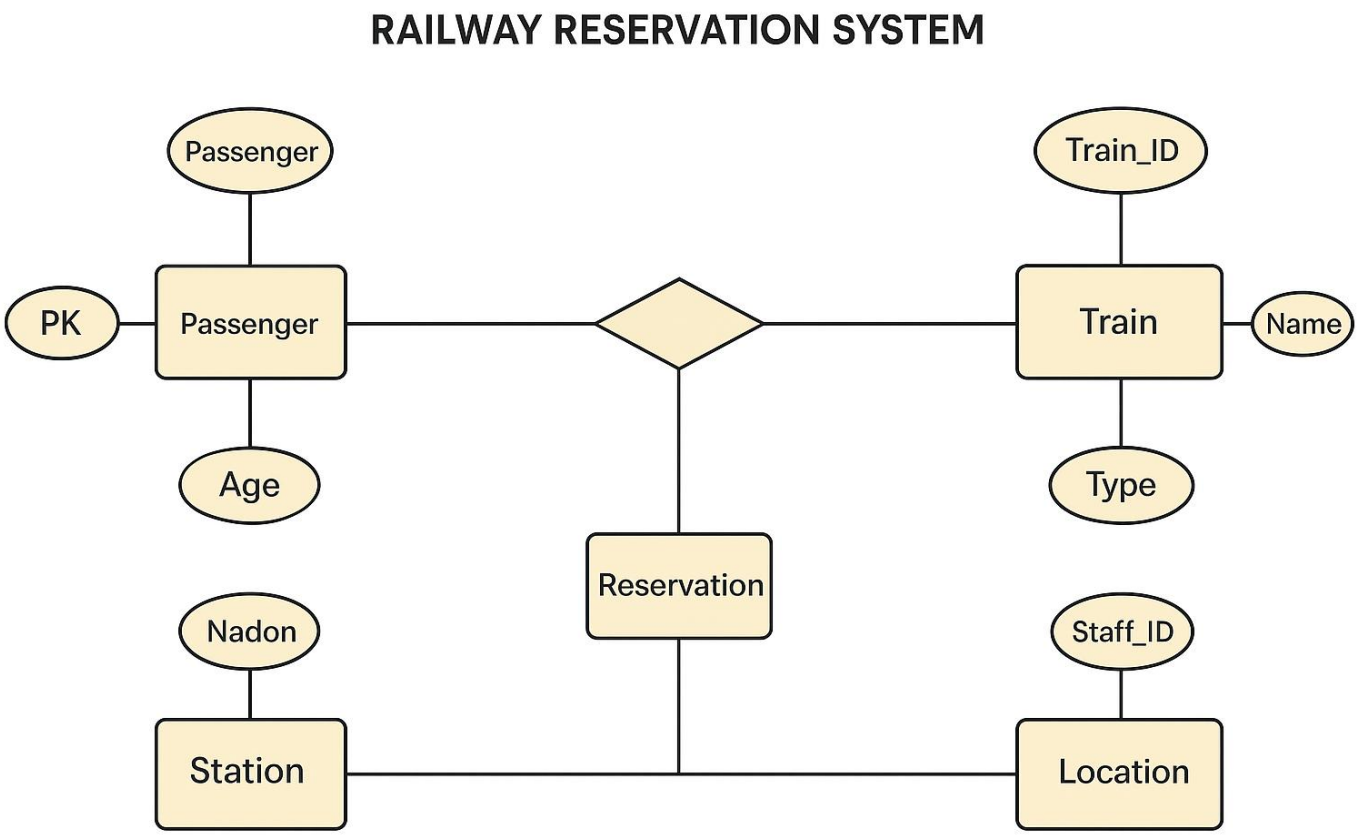
```
-------------------------------------------------
-- 20. EXPLICIT CURSOR (Bind Variable)
-------------------------------------------------
CREATE TABLE emp_explicit (
   id NUMBER,
   name VARCHAR2(20),
   salary NUMBER
);

INSERT INTO emp_explicit VALUES (1, 'Amit', 27000);
INSERT INTO emp_explicit VALUES (2, 'Sneha', 32000);
COMMIT;

DECLARE
   v_min_salary NUMBER := :min_salary;
   CURSOR c1 IS
      SELECT name, salary
      FROM emp_explicit
      WHERE salary >= v_min_salary;
   v_name emp_explicit.name%TYPE;
   v_salary emp_explicit.salary%TYPE;
BEGIN
   OPEN c1;
   LOOP
      FETCH c1 INTO v_name, v_salary;
      EXIT WHEN c1%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Name: ' || v_name || ', Salary: ' || v_salary);
   END LOOP;
   CLOSE c1;
END;
/
-- Example: :min_salary = 28000

-------------------------------------------------
-- 21. PROCEDURE (Bind Variable)
-------------------------------------------------
CREATE OR REPLACE PROCEDURE display_square(num IN NUMBER) IS
   result NUMBER;
BEGIN
   result := num * num;
   DBMS_OUTPUT.PUT_LINE('Square of ' || num || ' is ' || result);
END;
/

DECLARE
   n NUMBER := :n;
BEGIN
   display_square(n);
END;
/
-- Example: :n = 6
```

```
-------------------------------------------------
-- 22. PREDEFINED EXCEPTION (Bind Variables)
-------------------------------------------------
DECLARE
   num1 NUMBER := :num1;
   num2 NUMBER := :num2;
   result NUMBER;
BEGIN
   result := num1 / num2;
   DBMS_OUTPUT.PUT_LINE('Result: ' || result);
EXCEPTION
   WHEN ZERO_DIVIDE THEN
      DBMS_OUTPUT.PUT_LINE('Error: Cannot divide by zero.');
END;
/
-- Example: :num1 = 10, :num2 = 0

-------------------------------------------------
-- 23. USER DEFINED EXCEPTION (Bind Variable)
-------------------------------------------------
DECLARE
   age NUMBER := :age;
   under_age EXCEPTION;
BEGIN
   IF age < 18 THEN
      RAISE under_age;
   ELSE
      DBMS_OUTPUT.PUT_LINE('Valid age for registration.');
   END IF;
EXCEPTION
   WHEN under_age THEN
      DBMS_OUTPUT.PUT_LINE('Error: Age must be 18 or above.');
END;
/
-- Example: :age = 15
```

1.



2.


RAILWAY RESERVATION SYSTEM
Passenger
PK
Passenger
Age
Train_ID
Train
Name
Type
Reservation
Nadon
Station
Staff_ID
Location