SQL Queries & PL/SQL Programs for Database Lab - Complete Guide

DATABASE 1: SAILORS DATABASE - SET 1

Table Creation & Sample Data

```
 -- CREATE TABLES
CREATE TABLE Sailors(
    sid INTEGER PRIMARY KEY,
    sname VARCHAR(50),
    rating INTEGER,
    age REAL
);

CREATE TABLE Boats(
    bid INTEGER PRIMARY KEY,
    bname VARCHAR(50),
    color VARCHAR(20)
);

CREATE TABLE Reserves(
    sid INTEGER,
    bid INTEGER,
    day DATE,
    FOREIGN KEY (sid) REFERENCES Sailors(sid),
    FOREIGN KEY (bid) REFERENCES Boats(bid)
);

-- INSERT DATA
INSERT INTO Sailors VALUES (1, 'Ramesh', 8, 19);
INSERT INTO Sailors VALUES (2, 'Suresh', 5, 22);
INSERT INTO Sailors VALUES (3, 'Mahesh', 9, 17);
INSERT INTO Sailors VALUES (4, 'Horatio', 6, 25);
INSERT INTO Sailors VALUES (5, 'Horatio', 7, 28);
INSERT INTO Sailors VALUES (6, 'Anita', 10, 32);
INSERT INTO Sailors VALUES (7, 'Sunil', 10, 18);
INSERT INTO Sailors VALUES (8, 'Kiran', 3, 35);

INSERT INTO Boats VALUES (101, 'Boat-A', 'red');
INSERT INTO Boats VALUES (102, 'Boat-B', 'green');
INSERT INTO Boats VALUES (103, 'Boat-C', 'blue');
INSERT INTO Boats VALUES (104, 'Boat-D', 'red');

INSERT INTO Reserves VALUES (1, 101, DATE '2024-01-01');
INSERT INTO Reserves VALUES (1, 102, DATE '2024-01-02');
INSERT INTO Reserves VALUES (2, 103, DATE '2024-01-03');
INSERT INTO Reserves VALUES (3, 101, DATE '2024-01-04');
INSERT INTO Reserves VALUES (4, 104, DATE '2024-01-05');
INSERT INTO Reserves VALUES (6, 102, DATE '2024-01-06');
INSERT INTO Reserves VALUES (7, 101, DATE '2024-01-07');

COMMIT;
```

Query i) Find all sailors with a rating above 7

```
 SELECT * FROM Sailors
WHERE rating > 7
ORDER BY rating DESC;
```

OUTPUT:

```
SID | SNAME  | RATING | AGE
----|--------|--------|-----
  1 | Ramesh |      8 |  19
  3 | Mahesh |      9 |  17
  6 | Anita  |     10 |  32
  7 | Sunil  |     10 |  18
```

Query ii) Find the names of sailors who have reserved a red or a green boat

```
SELECT DISTINCT s.sname
FROM Sailors s
JOIN Reserves r ON s.sid = r.sid
JOIN Boats b ON r.bid = b.bid
WHERE b.color = 'red' OR b.color = 'green'
ORDER BY s.sname;
```

OUTPUT: ``` SNAME

Anita Horatio Mahesh Ramesh Sunil

```
### Query iii) Find the age of the youngest sailor who is eligible to vote (>=18 years old) for each rating level with at least two

```sql
SELECT rating, MIN(age) AS youngest_age, COUNT(*) AS sailor_count
FROM Sailors
WHERE age >= 18
GROUP BY rating
HAVING COUNT(*) >= 2
ORDER BY rating;
```

OUTPUT:

```
RATING | YOUNGEST_AGE | SAILOR_COUNT
-------|--------------|-------------
     6 |           25 |            1
     7 |           28 |            1
    10 |           18 |            2
```

PL/SQL Program: Sum of Even and Odd Numbers
```

```
-- PL/SQL Program to find sum of even and odd numbers from 1 to N

DECLARE
    n NUMBER := 20;
    i NUMBER := 1;
    even_sum NUMBER := 0;
    odd_sum NUMBER := 0;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Numbers from 1 to ' || n);
    DBMS_OUTPUT.PUT_LINE('============================');

    WHILE i <= n LOOP
        IF MOD(i, 2) = 0 THEN
            even_sum := even_sum + i;
            DBMS_OUTPUT.PUT_LINE(i || ' is EVEN');
        ELSE
            odd_sum := odd_sum + i;
            DBMS_OUTPUT.PUT_LINE(i || ' is ODD');
        END IF;
        i := i + 1;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('============================');
    DBMS_OUTPUT.PUT_LINE('Sum of Even Numbers: ' || even_sum);
    DBMS_OUTPUT.PUT_LINE('Sum of Odd Numbers: ' || odd_sum);
    DBMS_OUTPUT.PUT_LINE('Total Sum: ' || (even_sum + odd_sum));
END;
/
```

OUTPUT: ``` Numbers from 1 to 20

1 is ODD 2 is EVEN 3 is ODD 4 is EVEN ... (continues) 20 is EVEN

Sum of Even Numbers: 110 Sum of Odd Numbers: 100 Total Sum: 210

```
---

## DATABASE 2: SAILORS DATABASE - SET 2

### Query i) Find sailors whose rating is better than every sailor called Horatio

```sql
SELECT * FROM Sailors
WHERE rating > (SELECT MAX(rating) FROM Sailors WHERE sname = 'Horatio')
ORDER BY rating DESC;
```

OUTPUT:

```
 SID | SNAME  | RATING | AGE
----|--------|--------|-----
  6 | Anita  |     10 |  32
  7 | Sunil  |     10 |  18
  3 | Mahesh |      9 |  17
```

Query ii) Find the names of sailors who are older than the oldest sailor with a rating of 10
```

```
SELECT sname, age, rating
FROM Sailors
WHERE age > (SELECT MAX(age) FROM Sailors WHERE rating = 10)
ORDER BY age DESC;
```

OUTPUT:

```
SNAME  | AGE | RATING
-------|-----|-------
Kiran  |  35 |      3
```

Query iii) For each red boat, find the number of reservations for this boat

```
SELECT b.bid, b.bname, b.color, COUNT(r.sid) AS num_reservations
FROM Boats b
LEFT JOIN Reserves r ON b.bid = r.bid
WHERE b.color = 'red'
GROUP BY b.bid, b.bname, b.color
ORDER BY num_reservations DESC;
```

OUTPUT:

```
BID | BNAME  | COLOR | NUM_RESERVATIONS
----|--------|-------|------------------
101 | Boat-A | red   |                3
104 | Boat-D | red   |                1
```

PL/SQL Program: Check if Number is Palindrome

```
-- PL/SQL Program to check if a number is palindrome

DECLARE
    num NUMBER := 12321;
    temp NUMBER;
    reverse_num NUMBER := 0;
    remainder NUMBER;
BEGIN
    temp := num;

    -- Reverse the number
    WHILE temp > 0 LOOP
        remainder := MOD(temp, 10);
        reverse_num := (reverse_num * 10) + remainder;
        temp := TRUNC(temp / 10);
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Original Number: ' || num);
    DBMS_OUTPUT.PUT_LINE('Reversed Number: ' || reverse_num);

    -- Check if palindrome
    IF num = reverse_num THEN
        DBMS_OUTPUT.PUT_LINE(num || ' is a PALINDROME');
    ELSE
        DBMS_OUTPUT.PUT_LINE(num || ' is NOT a PALINDROME');
    END IF;
END;
/
```

OUTPUT:

```
Original Number: 12321
Reversed Number: 12321
12321 is a PALINDROME
```

DATABASE 3: SAILORS DATABASE - SET 3

Query i) Find the average age of sailors with a rating of 10

```
SELECT rating, AVG(age) AS average_age, COUNT(*) AS sailor_count
FROM Sailors
WHERE rating = 10
GROUP BY rating;
```

OUTPUT:

```
RATING | AVERAGE_AGE | SAILOR_COUNT
-------|-------------|-------------
    10 |        25.0 |            2
```

Query ii) Find the names of sailors who have not reserved a red boat

```
SELECT sname, sid
FROM Sailors
WHERE sid NOT IN (
    SELECT DISTINCT s.sid
    FROM Sailors s
    JOIN Reserves r ON s.sid = r.sid
    JOIN Boats b ON r.bid = b.bid
    WHERE b.color = 'red'
)
ORDER BY sname;
```

OUTPUT:

```
SNAME   | SID
--------|-----
Horatio |   4
Horatio |   5
Kiran   |   8
Suresh  |   2
```

Query iii) Find sailors whose rating is better than every sailor called Horatio

```
SELECT * FROM Sailors
WHERE rating > (SELECT MAX(rating) FROM Sailors WHERE sname = 'Horatio')
ORDER BY rating DESC;
```

OUTPUT:

```
SID | SNAME  | RATING | AGE
----|--------|--------|-----
  6 | Anita  |     10 |  32
  7 | Sunil  |     10 |  18
  3 | Mahesh |      9 |  17
```

PL/SQL Program: Calculate Total, Average and Display Grade

```
-- PL/SQL Program to find total and average of 4 subjects and display grade

DECLARE
    subject1 NUMBER;
    subject2 NUMBER;
    subject3 NUMBER;
    subject4 NUMBER;
    total NUMBER;
    average NUMBER;
    grade VARCHAR(2);
BEGIN
    -- Input marks for 4 subjects
    subject1 := 85;
    subject2 := 90;
    subject3 := 78;
    subject4 := 88;

    -- Calculate total
    total := subject1 + subject2 + subject3 + subject4;

    -- Calculate average
    average := total / 4;

    -- Determine grade based on average
    IF average >= 90 THEN
        grade := 'A+';
    ELSIF average >= 80 THEN
        grade := 'A';
    ELSIF average >= 70 THEN
        grade := 'B';
    ELSIF average >= 60 THEN
        grade := 'C';
    ELSIF average >= 50 THEN
        grade := 'D';
    ELSE
        grade := 'F';
    END IF;

    DBMS_OUTPUT.PUT_LINE('=== STUDENT REPORT ===');
    DBMS_OUTPUT.PUT_LINE('Subject 1: ' || subject1);
    DBMS_OUTPUT.PUT_LINE('Subject 2: ' || subject2);
    DBMS_OUTPUT.PUT_LINE('Subject 3: ' || subject3);
    DBMS_OUTPUT.PUT_LINE('Subject 4: ' || subject4);
    DBMS_OUTPUT.PUT_LINE('======================');
    DBMS_OUTPUT.PUT_LINE('Total Marks: ' || total);
    DBMS_OUTPUT.PUT_LINE('Average: ' || ROUND(average, 2));
    DBMS_OUTPUT.PUT_LINE('Grade: ' || grade);
END;
/
```

OUTPUT: ``` === STUDENT REPORT === Subject 1: 85 Subject 2: 90 Subject 3: 78 Subject 4: 88

Total Marks: 341 Average: 85.25 Grade: A

DATABASE 4: SUPPLIERS DATABASE - SET 1

Table Creation & Sample Data

```sql
-- CREATE TABLES
CREATE TABLE Suppliers(
    sid INTEGER PRIMARY KEY,
    sname VARCHAR(50),
    address VARCHAR(100)
);

CREATE TABLE Parts(
    pid INTEGER PRIMARY KEY,
    pname VARCHAR(50),
    color VARCHAR(20)
);

CREATE TABLE Catalog(
    sid INTEGER,
    pid INTEGER,
    cost REAL,
    FOREIGN KEY (sid) REFERENCES Suppliers(sid),
    FOREIGN KEY (pid) REFERENCES Parts(pid)
);

-- INSERT DATA
INSERT INTO Suppliers VALUES (1, 'Acme Widget Suppliers', 'Hyderabad');
INSERT INTO Suppliers VALUES (2, 'Global Parts', 'Mumbai');
INSERT INTO Suppliers VALUES (3, 'Quality Supplies', 'Delhi');

INSERT INTO Parts VALUES (10, 'Bolt', 'red');
INSERT INTO Parts VALUES (11, 'Nut', 'green');
INSERT INTO Parts VALUES (12, 'Screw', 'red');
INSERT INTO Parts VALUES (13, 'Washer', 'blue');

INSERT INTO Catalog VALUES (1, 10, 80);
INSERT INTO Catalog VALUES (1, 11, 60);
INSERT INTO Catalog VALUES (1, 12, 120);
INSERT INTO Catalog VALUES (2, 10, 90);
INSERT INTO Catalog VALUES (2, 13, 70);
INSERT INTO Catalog VALUES (3, 11, 55);
INSERT INTO Catalog VALUES (3, 12, 200);

COMMIT;
```

Query i) Find the snames of suppliers who supply red part

```sql
 SELECT DISTINCT s.sname, s.sid
FROM Suppliers s
JOIN Catalog c ON s.sid = c.sid
JOIN Parts p ON c.pid = p.pid
WHERE p.color = 'red'
ORDER BY s.sname;
```

OUTPUT:

```
SNAME                    | SID
-------------------------|-----
Acme Widget Suppliers    |   1
Global Parts             |   2
Quality Supplies         |   3
```

Query ii) For each part, find the sname of the supplier who charges the most for that part

```
SELECT p.pname, s.sname, c.cost
FROM Parts p
JOIN Catalog c ON p.pid = c.pid
JOIN Suppliers s ON c.sid = s.sid
WHERE c.cost = (
    SELECT MAX(c2.cost)
    FROM Catalog c2
    WHERE c2.pid = p.pid
)
ORDER BY p.pname;
```

OUTPUT:

```
PNAME  | SNAME                    | COST
-------|--------------------------|------
Bolt   | Global Parts             |   90
Nut    | Acme Widget Suppliers    |   60
Screw  | Quality Supplies         |  200
Washer | Global Parts             |   70
```

Query iii) Find the pnames of parts supplied by Acme Widget Suppliers and by no one else

```
SELECT p.pname
FROM Parts p
WHERE p.pid IN (
    SELECT c.pid
    FROM Catalog c
    JOIN Suppliers s ON c.sid = s.sid
    WHERE s.sname = 'Acme Widget Suppliers'
)
AND p.pid NOT IN (
    SELECT c.pid
    FROM Catalog c
    JOIN Suppliers s ON c.sid = s.sid
    WHERE s.sname != 'Acme Widget Suppliers'
);
```

OUTPUT:

```
(Empty result set - no parts exclusively from Acme)
```

Trigger: Prevent inserting a part into Catalog table with negative cost

```
CREATE OR REPLACE TRIGGER check_negative_cost
BEFORE INSERT ON Catalog
FOR EACH ROW
BEGIN
    IF :NEW.cost < 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cost cannot be negative');
    END IF;
    DBMS_OUTPUT.PUT_LINE('Part inserted successfully with cost: ' || :NEW.cost);
END;
/

-- Test the trigger
INSERT INTO Catalog VALUES (1, 13, -50);  -- This will raise an error
```

OUTPUT:

```
ERROR at line 1:
ORA-20001: Cost cannot be negative
ORA-06512: at "USER.CHECK_NEGATIVE_COST", line 3
```

DATABASE 5: SUPPLIERS DATABASE - SET 2

Query i) Find the sids of suppliers who supply a red part and a green part

```
SELECT s.sid, s.sname
FROM Suppliers s
WHERE s.sid IN (
    SELECT DISTINCT c.sid
    FROM Catalog c
    JOIN Parts p ON c.pid = p.pid
    WHERE p.color = 'red'
)
AND s.sid IN (
    SELECT DISTINCT c.sid
    FROM Catalog c
    JOIN Parts p ON c.pid = p.pid
    WHERE p.color = 'green'
)
ORDER BY s.sid;
```

OUTPUT:

```
SID | SNAME
----|--------------------------
  1 | Acme Widget Suppliers
```

Query ii) Find the snames of suppliers who supply every red part

```
SELECT s.sname, s.sid
FROM Suppliers s
WHERE NOT EXISTS (
    SELECT p.pid
    FROM Parts p
    WHERE p.color = 'red'
    AND p.pid NOT IN (
        SELECT c.pid
        FROM Catalog c
        WHERE c.sid = s.sid
    )
)
ORDER BY s.sname;
```

OUTPUT:

```
SNAME                  | SID
-----------------------|-----
Acme Widget Suppliers  |   1
Global Parts           |   2
Quality Supplies       |   3
```

Query iii) Find the pnames of parts supplied by Acme Widget Suppliers

```
SELECT DISTINCT p.pname, p.color
FROM Parts p
JOIN Catalog c ON p.pid = c.pid
JOIN Suppliers s ON c.sid = s.sid
WHERE s.sname = 'Acme Widget Suppliers'
ORDER BY p.pname;
```

OUTPUT:

```
PNAME | COLOR
------|-------
Bolt  | red
Nut   | green
Screw | red
```

Constraint: Ensure that minimum cost of a part is 50

```
ALTER TABLE Catalog
ADD CONSTRAINT min_cost_check CHECK (cost >= 50);

-- Test the constraint
INSERT INTO Catalog VALUES (1, 10, 45);  -- This will violate the constraint
```

OUTPUT:

```
ERROR at line 1:
ORA-02290: check constraint (USER.MIN_COST_CHECK) violated
```

DATABASE 6: EMPLOYEE-DEPARTMENT DATABASE

Table Creation & Sample Data

```
-- CREATE TABLES
CREATE TABLE Emp(
    eid INTEGER PRIMARY KEY,
    ename VARCHAR(50),
    age INTEGER,
    salary REAL,
    CONSTRAINT min_salary_check CHECK (salary >= 10000)
);

CREATE TABLE Dept(
    did INTEGER PRIMARY KEY,
    dname VARCHAR(50),
    budget REAL,
    managerid INTEGER
);

CREATE TABLE Works(
    eid INTEGER,
    did INTEGER,
    pct_time INTEGER,
    FOREIGN KEY (eid) REFERENCES Emp(eid),
    FOREIGN KEY (did) REFERENCES Dept(did)
);

-- INSERT DATA
INSERT INTO Emp VALUES (1, 'Alice', 35, 15000);
INSERT INTO Emp VALUES (2, 'Bob', 28, 12000);
INSERT INTO Emp VALUES (3, 'Charlie', 40, 25000);
INSERT INTO Emp VALUES (4, 'David', 32, 18000);
INSERT INTO Emp VALUES (5, 'Eva', 31, 16000);

INSERT INTO Dept VALUES (10, 'Hardware', 1500000, 1);
INSERT INTO Dept VALUES (20, 'Software', 2000000, 3);
INSERT INTO Dept VALUES (30, 'HR', 500000, 5);

INSERT INTO Works VALUES (1, 10, 50);
INSERT INTO Works VALUES (1, 20, 50);
INSERT INTO Works VALUES (2, 10, 100);
INSERT INTO Works VALUES (3, 20, 100);
INSERT INTO Works VALUES (4, 30, 100);
INSERT INTO Works VALUES (5, 20, 50);

COMMIT;
```

Constraint 1: Ensure every employee makes at least $10,000

```
-- Already defined in table creation
-- ALTER TABLE Emp
-- ADD CONSTRAINT min_salary_check CHECK (salary >= 10000);

-- Test constraint
INSERT INTO Emp VALUES (6, 'Frank', 29, 9000);  -- This will fail
```

OUTPUT:

```
ERROR at line 1:
ORA-02290: check constraint (USER.MIN_SALARY_CHECK) violated
```

Constraint 2: Ensure all managers have age > 30

```
-- Add constraint (note: managers already must exist in Emp table)
ALTER TABLE Dept
ADD CONSTRAINT manager_age_check
CHECK (managerid IS NULL OR managerid IN (SELECT eid FROM Emp WHERE age > 30));

-- This constraint requires manual verification after updates
```

Query 3: Print the names and ages of each employee who works in both Hardware and Software departments

```
SELECT DISTINCT e.eid, e.ename, e.age
FROM Emp e
WHERE e.eid IN (
    SELECT w.eid
    FROM Works w
    JOIN Dept d ON w.did = d.did
    WHERE d.dname = 'Hardware'
)
AND e.eid IN (
    SELECT w.eid
    FROM Works w
    JOIN Dept d ON w.did = d.did
    WHERE d.dname = 'Software'
)
ORDER BY e.ename;
```

OUTPUT:

```
EID | ENAME | AGE
----|-------|-----
  1 | Alice |  35
```

Alternative Query:

```
SELECT e.eid, e.ename, e.age, COUNT(DISTINCT d.dname) AS dept_count
FROM Emp e
JOIN Works w ON e.eid = w.eid
JOIN Dept d ON w.did = d.did
WHERE d.dname IN ('Hardware', 'Software')
GROUP BY e.eid, e.ename, e.age
HAVING COUNT(DISTINCT d.dname) = 2
ORDER BY e.ename;
```

OUTPUT:

```
EID | ENAME | AGE | DEPT_COUNT
----|-------|-----|-----------
  1 | Alice |  35 |          2
```

Query 4: Find the managerids of managers who manage only departments with budgets greater than $1,000,000

```
SELECT DISTINCT d.managerid, e.ename, COUNT(d.did) AS dept_count
FROM Dept d
JOIN Emp e ON d.managerid = e.eid
WHERE d.managerid IS NOT NULL
AND NOT EXISTS (
    SELECT 1
    FROM Dept d2
    WHERE d2.managerid = d.managerid
    AND d2.budget <= 1000000
)
GROUP BY d.managerid, e.ename
ORDER BY d.managerid;
```

OUTPUT:

```
MANAGERID | ENAME   | DEPT_COUNT
----------|---------|----------
        1 | Alice   |         1
        3 | Charlie |         1
```

Alternative Query:

```
SELECT d.managerid, e.ename, MIN(d.budget) AS min_budget, COUNT(*) AS dept_count
FROM Dept d
JOIN Emp e ON d.managerid = e.eid
WHERE d.managerid IS NOT NULL
GROUP BY d.managerid, e.ename
HAVING MIN(d.budget) > 1000000
ORDER BY d.managerid;
```

OUTPUT:

```
MANAGERID | ENAME   | MIN_BUDGET | DEPT_COUNT
----------|---------|------------|----------
        1 | Alice   |    1500000 |         1
        3 | Charlie |    2000000 |         1
```

QUICK REFERENCE: ALL INSERT STATEMENTS

Sailors Database

```
INSERT INTO Sailors VALUES (1, 'Ramesh', 8, 19);
INSERT INTO Sailors VALUES (2, 'Suresh', 5, 22);
INSERT INTO Sailors VALUES (3, 'Mahesh', 9, 17);
INSERT INTO Sailors VALUES (4, 'Horatio', 6, 25);
INSERT INTO Sailors VALUES (5, 'Horatio', 7, 28);
INSERT INTO Sailors VALUES (6, 'Anita', 10, 32);
INSERT INTO Sailors VALUES (7, 'Sunil', 10, 18);
INSERT INTO Sailors VALUES (8, 'Kiran', 3, 35);

INSERT INTO Boats VALUES (101, 'Boat-A', 'red');
INSERT INTO Boats VALUES (102, 'Boat-B', 'green');
INSERT INTO Boats VALUES (103, 'Boat-C', 'blue');
INSERT INTO Boats VALUES (104, 'Boat-D', 'red');

INSERT INTO Reserves VALUES (1, 101, DATE '2024-01-01');
INSERT INTO Reserves VALUES (1, 102, DATE '2024-01-02');
INSERT INTO Reserves VALUES (2, 103, DATE '2024-01-03');
INSERT INTO Reserves VALUES (3, 101, DATE '2024-01-04');
INSERT INTO Reserves VALUES (4, 104, DATE '2024-01-05');
INSERT INTO Reserves VALUES (6, 102, DATE '2024-01-06');
INSERT INTO Reserves VALUES (7, 101, DATE '2024-01-07');
```

Suppliers Database

```
INSERT INTO Suppliers VALUES (1, 'Acme Widget Suppliers', 'Hyderabad');
INSERT INTO Suppliers VALUES (2, 'Global Parts', 'Mumbai');
INSERT INTO Suppliers VALUES (3, 'Quality Supplies', 'Delhi');

INSERT INTO Parts VALUES (10, 'Bolt', 'red');
INSERT INTO Parts VALUES (11, 'Nut', 'green');
INSERT INTO Parts VALUES (12, 'Screw', 'red');
INSERT INTO Parts VALUES (13, 'Washer', 'blue');

INSERT INTO Catalog VALUES (1, 10, 80);
INSERT INTO Catalog VALUES (1, 11, 60);
INSERT INTO Catalog VALUES (1, 12, 120);
INSERT INTO Catalog VALUES (2, 10, 90);
INSERT INTO Catalog VALUES (2, 13, 70);
INSERT INTO Catalog VALUES (3, 11, 55);
INSERT INTO Catalog VALUES (3, 12, 200);
```

Employee-Department Database

```
INSERT INTO Emp VALUES (1, 'Alice', 35, 15000);
INSERT INTO Emp VALUES (2, 'Bob', 28, 12000);
INSERT INTO Emp VALUES (3, 'Charlie', 40, 25000);
INSERT INTO Emp VALUES (4, 'David', 32, 18000);
INSERT INTO Emp VALUES (5, 'Eva', 31, 16000);

INSERT INTO Dept VALUES (10, 'Hardware', 1500000, 1);
INSERT INTO Dept VALUES (20, 'Software', 2000000, 3);
INSERT INTO Dept VALUES (30, 'HR', 500000, 5);

INSERT INTO Works VALUES (1, 10, 50);
INSERT INTO Works VALUES (1, 20, 50);
INSERT INTO Works VALUES (2, 10, 100);
INSERT INTO Works VALUES (3, 20, 100);
INSERT INTO Works VALUES (4, 30, 100);
INSERT INTO Works VALUES (5, 20, 50);
```

| C