MCA-432 DBMS Practical Slips - SQL Answers (Slips 1 to 3)

```
-- Slip 1 Table Creation

CREATE TABLE SUPPLIER (

    Sno VARCHAR(5) CHECK (Sno LIKE 'S%' AND LENGTH(Sno) <= 5),

    Sname VARCHAR(50) NOT NULL,

    Address VARCHAR(100) NOT NULL,

    City VARCHAR(50) CHECK (City IN ('London', 'Paris', 'Rome', 'New
York', 'Amsterdam')),

    PRIMARY KEY (Sno)

);

CREATE TABLE PARTS (

    Pno INT PRIMARY KEY,

    Pname VARCHAR(50) NOT NULL,

    Color VARCHAR(20) NOT NULL,

    Weight DECIMAL(5,2) NOT NULL,

    Price DECIMAL(10,2) NOT NULL

);

CREATE TABLE PROJECT (

    Jno INT PRIMARY KEY,

    Jname VARCHAR(50) UNIQUE NOT NULL,

    City VARCHAR(50) CHECK (City IN ('London', 'Paris', 'Rome', 'New
York', 'Amsterdam')) NOT NULL

);

CREATE TABLE SPJ (

    Sno VARCHAR(5),

    Pno INT,

    Jno INT,

    Qty INT NOT NULL,

    FOREIGN KEY (Sno) REFERENCES SUPPLIER(Sno),

    FOREIGN KEY (Pno) REFERENCES PARTS(Pno),

    FOREIGN KEY (Jno) REFERENCES PROJECT(Jno)

);
```

```
-- Slip 1 Queries

-- a)

SELECT Jno FROM SPJ GROUP BY Jno HAVING COUNT(DISTINCT Pno) >= 3;



-- b)

CREATE OR REPLACE TRIGGER trg_no_duplicate_jname

BEFORE UPDATE ON PROJECT

FOR EACH ROW

BEGIN

    IF EXISTS (SELECT 1 FROM PROJECT WHERE Jname = :NEW.Jname AND Jno
!= :OLD.Jno) THEN

        RAISE_APPLICATION_ERROR(-20001, 'Duplicate project name not
allowed');

    END IF;

END;



-- c)

SELECT * FROM PROJECT WHERE City = 'Paris';
```

-- Slip 2 Table Creation

```
CREATE TABLE PRODUCT (

    Maker VARCHAR(50) NOT NULL,

    Modelno INT PRIMARY KEY,

    Type VARCHAR(10) CHECK (Type IN ('PC', 'Laptop', 'Printer')) NOT
NULL

);


CREATE TABLE PC (

    Modelno INT PRIMARY KEY,

    Speed INT NOT NULL,

    RAM INT NOT NULL,

    HD INT NOT NULL,

    CD VARCHAR(20) NOT NULL,

    Price DECIMAL(10,2) NOT NULL,

    FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);


CREATE TABLE LAPTOP (

    Modelno INT PRIMARY KEY,

    Speed INT NOT NULL,

    RAM INT NOT NULL,

    HD INT NOT NULL,

    Price DECIMAL(10,2) NOT NULL,

    FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);


CREATE TABLE PRINTER (

    Modelno INT PRIMARY KEY,

    Color CHAR(1) CHECK (Color IN ('T', 'F')) NOT NULL,

    Type VARCHAR(20) CHECK (Type IN ('laser', 'ink-jet', 'dot-matrix',
'dry')) NOT NULL,

    Price DECIMAL(10,2) NOT NULL,

    FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)
);


-- Slip 2 Queries

-- a)

SELECT * FROM PC WHERE Speed >= 160;


-- b)

SELECT DISTINCT p.Maker FROM PRODUCT p

WHERE p.Type = 'Laptop'

AND p.Maker NOT IN (

    SELECT Maker FROM PRODUCT WHERE Type = 'PC'

);


-- c)

CREATE OR REPLACE PROCEDURE MostExpensiveLaptop AS

    v_maker PRODUCT.Maker%TYPE;

BEGIN

    SELECT Maker INTO v_maker

    FROM PRODUCT p

    JOIN LAPTOP l ON p.Modelno = l.Modelno

    WHERE l.Price = (SELECT MAX(Price) FROM LAPTOP);

    DBMS_OUTPUT.PUT_LINE('Maker: ' || v_maker);

END;


-- Slip 3 Queries (same schema as Slip 2)

-- a)

SELECT DISTINCT pr.Type FROM PRINTER pr

JOIN PRODUCT p ON pr.Modelno = p.Modelno

WHERE p.Maker = 'Epson';


-- b)

SELECT HD FROM PC GROUP BY HD HAVING COUNT(*) >= 2;


-- c)
```

```
CREATE OR REPLACE TRIGGER trg_laptop_min_speed

BEFORE INSERT OR UPDATE ON LAPTOP

FOR EACH ROW

BEGIN

    IF :NEW.Speed < 150 THEN

        RAISE_APPLICATION_ERROR(-20002, 'Minimum speed should be
150MHz');

    END IF;

END;
```

-- Slip 4 Table Creation

```
CREATE TABLE DOCTOR (

    Did INT PRIMARY KEY,

    Dname VARCHAR(50) NOT NULL,

    Daddress VARCHAR(100) NOT NULL,

    Qualification VARCHAR(20) NOT NULL

);


CREATE TABLE PATIENTMASTER (

    Pcode INT PRIMARY KEY,

    Pname VARCHAR(50) NOT NULL,

    Padd VARCHAR(100) NOT NULL,

    Age INT NOT NULL,

    Gender CHAR(1) CHECK (Gender IN ('M', 'F')) NOT NULL,

    BloodGroup VARCHAR(5) NOT NULL,

    Did INT NOT NULL,

    FOREIGN KEY (Did) REFERENCES DOCTOR(Did)

);


CREATE TABLE ADMITTEDPATIENT (

    P_code INT,

    Entry_date DATE NOT NULL,

    Discharge_date DATE,

    Ward_no INT CHECK (Ward_no < 5) NOT NULL,

    Disease VARCHAR(50) NOT NULL,

    FOREIGN KEY (P_code) REFERENCES PATIENTMASTER(Pcode)

);
```

-- Slip 4 Queries

```
-- a)

SELECT DISTINCT d.* FROM DOCTOR d

JOIN PATIENTMASTER p ON d.Did = p.Did

JOIN ADMITTEDPATIENT a ON a.P_code = p.Pcode

WHERE a.Ward_no = 4;


-- b)

SELECT * FROM PATIENTMASTER p

JOIN ADMITTEDPATIENT a ON p.Pcode = a.P_code

WHERE a.Discharge_date BETWEEN TO_DATE('13-08-2022', 'DD-MM-YYYY') AND
TO_DATE('28-08-2022', 'DD-MM-YYYY');


-- c)

CREATE OR REPLACE PROCEDURE CalculateBill AS

BEGIN

    FOR rec IN (SELECT P_code, Entry_date, Discharge_date, Ward_no FROM
ADMITTEDPATIENT WHERE Discharge_date IS NOT NULL) LOOP

        DECLARE

            v_days INT;

            v_bill INT;

        BEGIN

            v_days := rec.Discharge_date - rec.Entry_date;

            v_bill := v_days * rec.Ward_no * 100;

            DBMS_OUTPUT.PUT_LINE('Patient Code: ' || rec.P_code || '
Bill: ' || v_bill);

        END;

    END LOOP;

END;
```

-- Slip 5 Queries

```
-- a)

SELECT d.Dname, p.Pname, a.Disease FROM DOCTOR d

JOIN PATIENTMASTER p ON d.Did = p.Did

JOIN ADMITTEDPATIENT a ON p.Pcode = a.P_code

WHERE a.Ward_no = 3;


-- b)

SELECT Disease FROM ADMITTEDPATIENT

GROUP BY Disease

ORDER BY COUNT(*) ASC FETCH FIRST 1 ROWS ONLY;


-- c)

CREATE OR REPLACE TRIGGER trg_ward_range

BEFORE INSERT OR UPDATE ON ADMITTEDPATIENT

FOR EACH ROW

BEGIN

    IF :NEW.Ward_no NOT BETWEEN 1 AND 5 THEN

        RAISE_APPLICATION_ERROR(-20003, 'Ward number must be between 1
and 5');

    END IF;

END;
```

-- Slip 6 Queries

```
-- a)

SELECT * FROM PATIENTMASTER p

JOIN DOCTOR d ON p.Did = d.Did

WHERE d.Qualification = 'M.S.';


-- b)

SELECT * FROM PATIENTMASTER p

JOIN ADMITTEDPATIENT a ON p.Pcode = a.P_code

WHERE a.Disease = 'blood cancer' AND p.Age < 40 AND p.BloodGroup = 'A';


-- c)

DECLARE

    CURSOR cur_last IS SELECT * FROM PATIENTMASTER ORDER BY Pcode DESC;

    v_row PATIENTMASTER%ROWTYPE;

BEGIN

    OPEN cur_last;

    FETCH cur_last INTO v_row;

    CLOSE cur_last;

    DBMS_OUTPUT.PUT_LINE('Last Patient: ' || v_row.Pname);

END;
```

-- Slip 7 Table Creation

```
CREATE TABLE ACCOUNT (

    accno INT PRIMARY KEY CHECK (accno < 1000),

    open_date DATE NOT NULL,

    acctype CHAR(1) CHECK (acctype IN ('P', 'J')) NOT NULL,

    balance DECIMAL(12,2) NOT NULL

);


CREATE TABLE TRANSACTION (

    trans_id INT PRIMARY KEY,

    trans_date DATE NOT NULL,

    accno INT NOT NULL,

    trans_type CHAR(1) CHECK (trans_type IN ('C', 'D')) NOT NULL,

    amount DECIMAL(10,2) NOT NULL,

    FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)

);


CREATE TABLE CUSTOMER (

    cust_id INT PRIMARY KEY,

    name VARCHAR(50) NOT NULL,

    address VARCHAR(100) NOT NULL,

    accno INT NOT NULL,

    FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)

);
```

```
-- Slip 7 Queries

-- a)

SELECT * FROM CUSTOMER WHERE balance >= 100000;



-- b)

SELECT * FROM TRANSACTION

WHERE trans_type = 'C'

AND trans_date BETWEEN TO_DATE('25-06-2022','DD-MM-YYYY') AND
TO_DATE('28-06-2022','DD-MM-YYYY');



-- c)

CREATE OR REPLACE TRIGGER trg_update_balance

AFTER INSERT ON TRANSACTION

FOR EACH ROW

BEGIN

    IF :NEW.trans_type = 'C' THEN

        UPDATE ACCOUNT SET balance = balance + :NEW.amount WHERE accno
= :NEW.accno;

    ELSE

        UPDATE ACCOUNT SET balance = balance - :NEW.amount WHERE accno
= :NEW.accno;

    END IF;

END;
```

-- Slip 8 Queries

```
-- a)

SELECT * FROM CUSTOMER c

JOIN ACCOUNT a ON c.accno = a.accno

WHERE a.acctype = 'P' AND a.balance < 300000;


-- b)

SELECT * FROM CUSTOMER c

JOIN ACCOUNT a ON c.accno = a.accno

WHERE a.acctype = 'J';


-- c)

CREATE OR REPLACE PROCEDURE InsertTransaction(

    p_trans_id INT, p_trans_date DATE, p_accno INT, p_trans_type CHAR,
p_amount DECIMAL

) AS

BEGIN

    INSERT INTO TRANSACTION VALUES (p_trans_id, p_trans_date, p_accno,
p_trans_type, p_amount);

    IF p_trans_type = 'C' THEN

        UPDATE ACCOUNT SET balance = balance + p_amount WHERE accno =
p_accno;

    ELSE

        UPDATE ACCOUNT SET balance = balance - p_amount WHERE accno =
p_accno;

    END IF;

END;
```

```
-- Slip 9 Queries

-- a)

SELECT t.*, c.name FROM TRANSACTION t

JOIN CUSTOMER c ON t.accno = c.accno

WHERE t.accno = 103;


-- b)

SELECT * FROM TRANSACTION

WHERE trans_type = 'C'

AND trans_date BETWEEN TO_DATE('15-03-2022','DD-MM-YYYY') AND
TO_DATE('18-03-2022','DD-MM-YYYY');


-- c)

CREATE OR REPLACE TRIGGER trg_balance_check

BEFORE INSERT ON TRANSACTION

FOR EACH ROW

DECLARE

    v_balance ACCOUNT.balance%TYPE;

BEGIN

    SELECT balance INTO v_balance FROM ACCOUNT WHERE accno =
:NEW.accno;

    IF :NEW.trans_type = 'D' AND v_balance <= 800 THEN

        RAISE_APPLICATION_ERROR(-20004, 'Account balance too low to
allow debit');

    END IF;

END;
```

-- Slip 10 Queries

```
-- a)

SELECT * FROM CUSTOMER

WHERE open_date BETWEEN TO_DATE('25-03-2022', 'DD-MM-YYYY') AND
TO_DATE('28-03-2022', 'DD-MM-YYYY');


-- b)

SELECT * FROM CUSTOMER c

JOIN ACCOUNT a ON c.accno = a.accno

WHERE a.acctype = 'J' AND a.balance < 300000;


-- c)

DECLARE

    CURSOR cur_last IS SELECT * FROM CUSTOMER ORDER BY cust_id DESC;

    v_row CUSTOMER%ROWTYPE;

BEGIN

    OPEN cur_last;

    FETCH cur_last INTO v_row;

    CLOSE cur_last;

    DBMS_OUTPUT.PUT_LINE('Last Customer: ' || v_row.name);

END;


-- Slip 11 Table Creation

CREATE TABLE BOOKMASTER (

    bid INT PRIMARY KEY,

    title VARCHAR(100) NOT NULL,

    author VARCHAR(50) NOT NULL,

    price DECIMAL(8,2) NOT NULL

);
```

```
CREATE TABLE STUDENTMASTER (

    stud_enrollno INT PRIMARY KEY,

    sname VARCHAR(50) NOT NULL,

    class VARCHAR(20) NOT NULL,

    dept VARCHAR(50) NOT NULL

);


CREATE TABLE ACCESSIONTABLE (

    accession_no INT PRIMARY KEY,

    bid INT NOT NULL,

    avail CHAR(1) CHECK (avail IN ('T', 'F')) NOT NULL,

    FOREIGN KEY (bid) REFERENCES BOOKMASTER(bid)

);


CREATE TABLE ISSUETABLE (

    issueid INT PRIMARY KEY,

    accession_no INT NOT NULL,

    stud_enrollno INT NOT NULL,

    issuedate DATE NOT NULL,

    duedate DATE NOT NULL,

    ret_date DATE,

    bid INT NOT NULL,

    FOREIGN KEY (accession_no) REFERENCES ACCESSIONTABLE(accession_no),

    FOREIGN KEY (stud_enrollno) REFERENCES
STUDENTMASTER(stud_enrollno),

    FOREIGN KEY (bid) REFERENCES BOOKMASTER(bid)

);
```

-- Slip 11 Queries

```
-- a)

SELECT b.title FROM BOOKMASTER b

JOIN ISSUETABLE i ON b.bid = i.bid

GROUP BY b.title

ORDER BY COUNT(*) DESC FETCH FIRST 1 ROW ONLY;


-- b)

SELECT b.*, i.*, s.*

FROM BOOKMASTER b

JOIN ISSUETABLE i ON b.bid = i.bid

JOIN STUDENTMASTER s ON i.stud_enrollno = s.stud_enrollno

WHERE s.dept = 'Computer';


-- c)

CREATE OR REPLACE PROCEDURE CalcFine AS

BEGIN

    FOR rec IN (

        SELECT issueid, ret_date, duedate FROM ISSUETABLE

        WHERE ret_date > duedate

    ) LOOP

        DECLARE

            days_late INT := rec.ret_date - rec.duedate;

            fine INT := days_late * 10;

        BEGIN

            DBMS_OUTPUT.PUT_LINE('Issue ID: ' || rec.issueid || ', 
Fine: ' || fine);

        END;

    END LOOP;

END;
```

-- Slip 12 Queries

```
-- a)

SELECT s.*

FROM STUDENTMASTER s

JOIN ISSUETABLE i ON s.stud_enrollno = i.stud_enrollno

WHERE i.issuedate BETWEEN TO_DATE('01-01-2022', 'DD-MM-YYYY') AND
TO_DATE('31-12-2022', 'DD-MM-YYYY');


-- b)

CREATE VIEW View_Book_Accession AS

SELECT * FROM ACCESSIONTABLE WHERE bid = 100;


-- c)

DECLARE

    CURSOR cur_first IS SELECT * FROM View_Book_Accession;

    v_row View_Book_Accession%ROWTYPE;

BEGIN

    OPEN cur_first;

    FETCH cur_first INTO v_row;

    CLOSE cur_first;

    DBMS_OUTPUT.PUT_LINE('First Book Accession No: ' ||
v_row.accession_no);

END;
```

-- Slip 13 Queries

```
-- a)

CREATE OR REPLACE PROCEDURE GetAvailableBooks AS

BEGIN

    FOR rec IN (

        SELECT * FROM BOOKMASTER b

        WHERE EXISTS (

            SELECT 1 FROM ACCESSIONTABLE a

            WHERE a.bid = b.bid AND a.avail = 'T'

        )

    ) LOOP

        DBMS_OUTPUT.PUT_LINE('Book: ' || rec.title);

    END LOOP;

END;


-- b)

SELECT stud_enrollno, COUNT(*) AS BooksIssued

FROM ISSUETABLE

GROUP BY stud_enrollno;


-- c)

SELECT COUNT(*) AS AvailableBooks

FROM BOOKMASTER b

JOIN ACCESSIONTABLE a ON b.bid = a.bid

WHERE a.avail = 'T' AND b.author = 'E.Navathe';
```

-- Slip 14 Queries

```
-- a)

SELECT * FROM CUSTOMER

WHERE open_date BETWEEN TO_DATE('25-03-2018', 'DD-MM-YYYY') AND
TO_DATE('28-03-2018', 'DD-MM-YYYY');



-- b)

SELECT * FROM CUSTOMER c

JOIN ACCOUNT a ON c.accno = a.accno

WHERE a.acctype = 'J' AND a.balance < 200000;



-- c)

DECLARE

    CURSOR cur_last_cust IS SELECT * FROM CUSTOMER ORDER BY cust_id
DESC;

    v_row CUSTOMER%ROWTYPE;

BEGIN

    OPEN cur_last_cust;

    FETCH cur_last_cust INTO v_row;

    CLOSE cur_last_cust;

    DBMS_OUTPUT.PUT_LINE('Last Customer: ' || v_row.name);

END;
```

```
-- Slip 15 Queries

-- a)

SELECT DISTINCT p.Maker

FROM PRODUCT p

JOIN PRINTER pr ON p.Modelno = pr.Modelno

WHERE pr.Color = 'T';


-- b)

SELECT * FROM LAPTOP l

WHERE l.Speed < (SELECT MIN(Speed) FROM PC);


-- c)

CREATE OR REPLACE TRIGGER trg_hd_check

BEFORE INSERT OR UPDATE ON PC

FOR EACH ROW

BEGIN

    IF :NEW.HD <= 20 THEN

        RAISE_APPLICATION_ERROR(-20005, 'PC Hard disk must be greater
than 20 GB');

    END IF;

END;


CREATE OR REPLACE TRIGGER trg_hd_check_laptop

BEFORE INSERT OR UPDATE ON LAPTOP

FOR EACH ROW

BEGIN

    IF :NEW.HD <= 20 THEN

        RAISE_APPLICATION_ERROR(-20006, 'Laptop Hard disk must be
greater than 20 GB');

    END IF;

END;
```

-- Slip 16 Queries

```
-- a)

SELECT DISTINCT pr.Type FROM PRINTER pr

JOIN PRODUCT p ON pr.Modelno = p.Modelno

WHERE p.Maker = 'Epson';


-- b)

SELECT HD FROM PC

GROUP BY HD

HAVING COUNT(*) >= 2;


-- c)

CREATE OR REPLACE TRIGGER trg_min_speed_laptop

BEFORE INSERT OR UPDATE ON LAPTOP

FOR EACH ROW

BEGIN

    IF :NEW.Speed < 250 THEN

        RAISE_APPLICATION_ERROR(-20007, 'Laptop speed must be at least
250 MHz');

    END IF;

END;
```

-- Slip 17 Queries

```
-- a)

SELECT DISTINCT pr.Type FROM PRINTER pr

JOIN PRODUCT p ON pr.Modelno = pr.Modelno

WHERE p.Maker = 'Epson';


-- b)

SELECT HD FROM PC

GROUP BY HD

HAVING COUNT(*) >= 2;


-- c)

DECLARE

    CURSOR cur_product IS SELECT * FROM PRODUCT ORDER BY Modelno;

    v_row PRODUCT%ROWTYPE;

BEGIN

    OPEN cur_product;

    FETCH cur_product INTO v_row;

    CLOSE cur_product;

    DBMS_OUTPUT.PUT_LINE('First Product: ' || v_row.Modelno);

END;
```

```
-- Slip 18 Queries

-- a)

SELECT * FROM ADMITTEDPATIENT

WHERE Entry_date BETWEEN TO_DATE('03-03-2022', 'DD-MM-YYYY') AND
TO_DATE('25-03-2022', 'DD-MM-YYYY');


-- b)

SELECT DISTINCT d.Dname FROM DOCTOR d

JOIN PATIENTMASTER p ON d.Did = p.Did

JOIN ADMITTEDPATIENT a ON a.P_code = p.Pcode

WHERE a.Disease = 'TB';


-- c)

CREATE OR REPLACE PROCEDURE CalcCurrentBill AS

BEGIN

    FOR rec IN (

        SELECT P_code, Entry_date, Ward_no FROM ADMITTEDPATIENT

        WHERE Discharge_date IS NULL

    ) LOOP

        DECLARE

            v_days INT := SYSDATE - rec.Entry_date;

            v_bill INT := v_days * 800;

        BEGIN

            DBMS_OUTPUT.PUT_LINE('Patient Code: ' || rec.P_code || '
Current Bill: ' || v_bill);

        END;

    END LOOP;

END;
```

-- Slip 19 Queries

```
-- a)

SELECT DISTINCT p.Maker

FROM PRODUCT p

JOIN PRINTER pr ON p.Modelno = pr.Modelno

WHERE pr.Color = 'T';


-- b)

SELECT * FROM LAPTOP

WHERE Speed < (SELECT MIN(Speed) FROM PC);


-- c)

CREATE OR REPLACE TRIGGER trg_hd_check_pc_laptop

BEFORE INSERT OR UPDATE ON PC

FOR EACH ROW

BEGIN

    IF :NEW.HD <= 20 THEN

        RAISE_APPLICATION_ERROR(-20008, 'PC Hard disk must be > 20
GB');

    END IF;

END;

CREATE OR REPLACE TRIGGER trg_hd_check_laptop_2

BEFORE INSERT OR UPDATE ON LAPTOP

FOR EACH ROW

BEGIN

    IF :NEW.HD <= 20 THEN

        RAISE_APPLICATION_ERROR(-20009, 'Laptop Hard disk must be > 20
GB');

    END IF;

END;
```

-- Slip 20 Queries

```
-- a)

SELECT DISTINCT pr.Type FROM PRINTER pr

JOIN PRODUCT p ON pr.Modelno = p.Modelno

WHERE p.Maker = 'Epson';


-- b)

SELECT HD FROM PC

GROUP BY HD

HAVING COUNT(*) >= 2;


-- c)

DECLARE

    CURSOR cur_product IS SELECT * FROM PRODUCT ORDER BY Modelno DESC;

    v_row PRODUCT%ROWTYPE;

BEGIN

    OPEN cur_product;

    FETCH cur_product INTO v_row;

    CLOSE cur_product;

    DBMS_OUTPUT.PUT_LINE('Last Product Model: ' || v_row.Modelno);

END;
```