1-- SUPPLIER table

```
CREATE TABLE SUPPLIER ( Sno VARCHAR(6) PRIMARY KEY CHECK (Sno LIKE 'S%' AND
TRY_CAST(SUBSTRING(Sno, 2, LEN(Sno)) AS INT) BETWEEN 0 AND 9999), Sname VARCHAR(50) NOT
NULL, Address VARCHAR(100) NOT NULL, City VARCHAR(50) NOT NULL CHECK (City IN ('London', 'Paris',
'Rome', 'New York', 'Amsterdam')) );
```

-- PARTS table

```
CREATE TABLE PARTS ( Pno VARCHAR(6) PRIMARY KEY, Pname VARCHAR(50) NOT NULL, Color
VARCHAR(20) NOT NULL, Weight DECIMAL(5,2) NOT NULL, Price DECIMAL(10,2) NOT NULL );
```

-- PROJECT table

```
CREATE TABLE PROJECT ( Jno VARCHAR(6) PRIMARY KEY, Jname VARCHAR(50) NOT NULL UNIQUE, City
VARCHAR(50) NOT NULL CHECK (City IN ('London', 'Paris', 'Rome', 'New York', 'Amsterdam')) );
```

-- SPJ table

```
CREATE TABLE SPJ ( Sno VARCHAR(6) NOT NULL FOREIGN KEY REFERENCES SUPPLIER(Sno), Pno
VARCHAR(6) NOT NULL FOREIGN KEY REFERENCES PARTS(Pno), Jno VARCHAR(6) NOT NULL FOREIGN
KEY REFERENCES PROJECT(Jno), Qty INT NOT NULL, PRIMARY KEY (Sno, Pno, Jno) );

INSERT INTO SUPPLIER VALUES ('S101', 'Alpha Ltd', '12 King St', 'London');

INSERT INTO PARTS VALUES ('P1', 'Bolt', 'Red', 0.50, 1.00);

INSERT INTO PROJECT VALUES ('J1', 'Apollo', 'London');

INSERT INTO SPJ VALUES ('S101', 'P1', 'J1', 100);
```

Queries---

1 Projects that have 3 or more parts

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

2) Full details of projects in London

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

2----product

```
CREATE TABLE PRODUCT ( Maker VARCHAR(50) NOT NULL, Modelno INT PRIMARY KEY, Type
VARCHAR(20) NOT NULL CHECK (Type IN ('PC', 'Laptop', 'Printer')) );
```

CREATE TABLE PC (Modelno INT PRIMARY KEY, Speed INT NOT NULL, RAM INT NOT NULL, HD INT NOT NULL, CD VARCHAR(10) NOT NULL, Price INT 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 INT NOT NULL CHECK (Price >= 30000), FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno));

CREATE TABLE PRINTER (Modelno INT PRIMARY KEY, Color CHAR(1) NOT NULL CHECK (Color IN ('T', 'F')), Type VARCHAR(20) NOT NULL CHECK (Type IN ('laser', 'ink jet', 'dot-matrix', 'dry')), Price INT NOT NULL, FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno));

INSERT INTO PRODUCT VALUES ('IBM', 1001, 'PC'), ('IBM', 1002, 'Laptop'), ('Compaq', 1003, 'PC');

INSERT INTO PC VALUES (1001, 200, 8, 500, '52X', 40000);

INSERT INTO LAPTOP VALUES (1002, 250, 8, 512, 45000);

INSERT INTO PRINTER VALUES (1004, 'T', 'laser', 25000);

Queries--

1PC models with speed >= 150 MHz

SELECT * FROM PC WHERE Speed >= 150;

2) Manufacturers that make Laptops but not PCs

SELECT DISTINCT p.Maker FROM PRODUCT p WHERE p.Type = 'Laptop' AND p.Maker NOT IN (SELECT DISTINCT Maker FROM PRODUCT WHERE Type = 'PC');

3 Different types of printers produced by Epson

SELECT DISTINCT p.Type FROM PRODUCT pr JOIN PRINTER p ON pr.Modelno = p.Modelno WHERE pr.Maker = 'Epson';

4 Hard disk sizes that occur in two or more PCs

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

5 Find the manufacturers of color printers.

SELECT DISTINCT p.Maker FROM PRODUCT p JOIN PRINTER pr ON p.Modelno = pr.Modelno WHERE pr.Color = 'T';

6 Find the laptops whose speed is slower than that of any PC.

SELECT * FROM LAPTOP WHERE Speed < ALL (SELECT Speed FROM PC);

7 SQL Assertion: No black & white printer should have price greater than any color printer. -- Check for invalid printers

SELECT * FROM PRINTER bw WHERE Color = 'F' AND Price > ANY (SELECT Price FROM PRINTER WHERE Color = 'T');

3-- DOCTOR Table

CREATE TABLE DOCTOR (Did INT PRIMARY KEY, Dname VARCHAR(50) NOT NULL, Daddress VARCHAR(100) NOT NULL, qualification VARCHAR(50) NOT NULL);

-- PATIENTMASTER Table CREATE TABLE

PATIENTMASTER (Pcode INT PRIMARY KEY, Pname VARCHAR(50) NOT NULL, Padd VARCHAR(100) NOT NULL, age INT NOT NULL, gender CHAR(1) NOT NULL CHECK (gender IN ('M', 'F')), bloodgroup VARCHAR(3) NOT NULL, Pid INT NOT NULL);

-- ADMITTEDPATIENT Table

CREATE TABLE ADMITTEDPATIENT (P_code INT NOT NULL FOREIGN KEY REFERENCES PATIENTMASTER(Pcode), EntryDate DATE NOT NULL, DischargeDate DATE NOT NULL, wardno INT NOT NULL CHECK (wardno < 6), disease VARCHAR(100) NOT NULL, Did INT NOT NULL FOREIGN KEY REFERENCES DOCTOR(Did));

INSERT INTO DOCTOR VALUES(1, 'Dr. A Sharma', 'Delhi', 'MD');

INSERT INTO PATIENTMASTER VALUES (101, 'Ravi Verma', 'Delhi', 30, 'M', 'A', 1);

INSERT INTO ADMITTEDPATIENT VALUES (101, '2012-03-01', '2012-03-10', 3, 'Fever', 1);

queries

1 Doctors treating patients in ward no. 3

SELECT DISTINCT D.* FROM DOCTOR D JOIN ADMITTEDPATIENT A ON D.Did = A.Did WHERE A.wardno = 3;

2 Patients discharged between 03/03/12 and 25/03/12

SELECT P.* FROM PATIENTMASTER P JOIN ADMITTEDPATIENT A ON P.Pcode = A.P_code WHERE A.DischargeDate BETWEEN '2012-03-03' AND '2012-03-25';

3 Name of disease with maximum patients

SELECT TOP 1 disease, COUNT(*) AS PatientCount FROM ADMITTEDPATIENT GROUP BY disease ORDER BY PatientCount DESC;

4 Patients treated by M.B.B.S. doctors:

```
SELECT P.* FROM PATIENTMASTER P JOIN DOCTOR D ON P.aid = D.Did WHERE D.qualification = 'M.B.B.S.';
```

5) Patient suffering from blood cancer, age < 50, blood group A:

```
SELECT P.* FROM PATIENTMASTER P JOIN ADMITTEDPATIENT A ON P.Pcode = A.Pcode WHERE A.disease = 'Blood Cancer' AND P.age < 50 AND P.bloodgroup = 'A';
```

6Patients treated by M.S. Doctors

```
SELECT P.* FROM PATIENTMASTER P JOIN DOCTOR D ON P.Did = D.Did WHERE D.qualification = 'M.S.';
```

7 Doctor Treating Maximum Patients

```
SELECT TOP 1 D.Dname, COUNT(*) AS PatientCount FROM PATIENTMASTER P JOIN DOCTOR D ON P.Did = D.Did GROUP BY D.Dname ORDER BY PatientCount DESC;
```

8) Findthe details of patient who are discharge d within the period 03/03/12 to 25/03/12

```
select p.pcode,p.pname,p.age,p.gender,a.disch_date from PATIENTMASTER p,ADMITTEDPATIENT a

where p.pcode=a.pcode and disch_date between '3-mar-2008'and'25-mar-2008';
```

4—-- ACCOUNT Table

```
CREATE TABLE ACCOUNT ( accno INT PRIMARY KEY CHECK (accno < 100), -- Less than 3 digits opendate DATE NOT NULL, acctype CHAR(1) NOT NULL CHECK (acctype IN ('P', 'J')), balance DECIMAL(18, 2) NOT NULL CHECK (balance >= 0) );
```

-- CUSTOMER Table

```
CREATE TABLE CUSTOMER ( custid INT PRIMARY KEY, name VARCHAR(100) NOT NULL, address VARCHAR(255) NOT NULL, accno INT NOT NULL, FOREIGN KEY (accno) REFERENCES ACCOUNT(accno) );
```

-- TRANSACTION Table

```
CREATE TABLE TRANSACTION ( transid INT PRIMARY KEY, transdate DATE NOT NULL, accno INT NOT NULL, transtype CHAR(1) NOT NULL CHECK (transtype IN ('C', 'D')), amount DECIMAL(18, 2) NOT NULL CHECK (amount > 0), FOREIGN KEY (accno) REFERENCES ACCOUNT(accno) );
```

```
INSERT INTO ACCOUNT VALUES (1, '2006-03-25', 'P', 150000);
```

```
INSERT INTO CUSTOMER VALUES (101, 'Alice', 'New York', 1);
```

INSERT INTO TRANSACTION VALUES (1, '2024-04-01', 1, 'C', 10000);

queries

1 Customers with minimum balance = 1 lakh

SELECT C.* FROM CUSTOMER C JOIN ACCOUNT A ON C.accno = A.accno WHERE A.balance >= 100000;

2 Amount credited between 25-03-2012 and 28-03-2012

SELECT * FROM TRANSACTION WHERE transtype = 'C' AND transdate BETWEEN '2012-03-25' AND '2012-03-28';

3 Customers with Personal Account and balance < 2 Lakhs

SELECT C.* FROM CUSTOMER C JOIN ACCOUNT A ON C.accno = A.accno WHERE A.acctype = 'P' AND A.balance < 200000;

4 Customers with Joint Account

SELECT C.* FROM CUSTOMER C JOIN ACCOUNT A ON C.accno = A.accno WHERE A.acctype = 'J';

5 Details of transactions for account number 101 and customer names

SELECT T.*, C.name FROM TRANSACTION T JOIN CUSTOMER C ON T.accno = C.accno WHERE T.accno = 101;

6) Amount credited between 15-3-2012 to 18-3-2012

SELECT * FROM TRANSACTION WHERE transtype = 'C' AND [trans date] BETWEEN '2012-03-15' AND '2012-03 18';

7Find the details of customers who have opened accounts within the period 25-03-2012 to 28-03-2012.

SELECT c.cust_id, c.name, c.address, a.accno, a.open_date, a.acctype, a.balance FROM CUSTOMER c JOIN ACCOUNT a ON c.accno = a.accno WHERE a.open_date BETWEEN '2012-03-25' AND '2012 03-28';

8 Find the details of customers who have joint accounts & balance is less than 2 lakhs. Since there is no explicit field to define whether an account is a joint account or not, we assume that the 'M' (joint) type represents joint accounts.

SELECT c.cust_id, c.name, c.address, a.accno, a.open_date, a.acctype, a.balance FROM CUSTOMER c JOIN ACCOUNT a ON c.accno = a.accno WHERE a.acctype = 'M' AND a.balance < 2000

9 Customers with accounts opened between 25-3-2006 and 28-3-2006

SELECT C.* FROM CUSTOMER C JOIN ACCOUNT A ON C.accno = A.accno WHERE A.open_date BETWEEN '2006-03-25' AND '2006-03 28';

10 Joint account customers with balance < 200000

SELECT C.* FROM CUSTOMER C JOIN ACCOUNT A ON C.accno = A.accno WHERE A.acctype = 'J' AND A.balance < 200000;

--5---bookmaster

CREATE TABLE BOOKMASTER (bid INT PRIMARY KEY, title VARCHAR(100) NOT NULL, author VARCHAR(100) NOT NULL, price DECIMAL(10, 2) NOT NULL);

CREATE TABLE STUDENTMASTER (enrollno INT PRIMARY KEY, sname VARCHAR(100) NOT NULL, class VARCHAR(50) NOT NULL, dept VARCHAR(50) NOT NULL);

CREATE TABLE ACCESSIONTABLE (bid INT, accession_no INT PRIMARY KEY, 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, enrollno INT, issuedate DATE NOT NULL, cluedate DATE NOT NULL, ret_date DATE, bid INT, FOREIGN KEY (accession_no) REFERENCES ACCESSIONTABLE(accession_no), FOREIGN KEY (enrollno) REFERENCES STUDENTMASTER(enrollno), FOREIGN KEY (bid) REFERENCES BOOKMASTER(bid));

INSERT INTO BOOKMASTER (bid, title, author, price) VALUES (1, 'Book A', 'Author A', 200.00);

INSERT INTO STUDENTMASTER (enrollno, sname, class, dept) VALUES (1, 'Alice', '1st Year', 'Computer');

INSERT INTO ACCESSIONTABLE (bid, accession_no, avail) VALUES (1, 101, 'T');

INSERT INTO ISSUETABLE (issueid, accession_no, enrollno, issuedate, cluedate, ret_date, bid) VALUES (1, 101, 1, '2025-04-01', '2025-04-08', '2025-04-10', 1),

queries

1 Find the name of books which are issued the maximum times.

SELECT b.title, COUNT(i.issueid) AS issue_count FROM BOOKMASTER b JOIN ISSUETABLE i ON b.bid = i.bid GROUP BY b.title ORDER BY issue_count DESC LIMIT 1;

2 Find the detailed information of books that are issued by computer department students.

SELECT b.*, s.sname, s.class, s.dept, i.issuedate, i.cluedate, i.ret_date FROM BOOKMASTER b JOIN ISSUETABLE i ON b.bid = i.bid JOIN STUDENTMASTER s ON i.enrollno = s.enrollno WHERE s.dept = 'Computer';

3)Data Report: Display all books available in the library.

SELECT b.title, b.author, a.avail FROM BOOKMASTER b JOIN ACCESSIONTABLE a ON b.bid = a.bid WHERE a.avail = 'T';

4 Students who have issued books between two given dates

DECLARE @startDate DATE = '2024-04-01'; DECLARE @endDate DATE = '2024-04-10'; SELECT s.* FROM STUDENTMASTER s JOIN ISSUETABLE i ON s.stud_enrollno = i.stud_enrollno WHERE i.issuedate BETWEEN @startDate AND @endDate;

5 Create a view for accession info for bid = 100

CREATE VIEW Book100Accessions AS SELECT a.accession_no, a.avail, i.bid FROM ACCESSIONTABLE a JOIN ISSUETABLE i ON a.accession_no = i.accession_no WHERE i.bid = 100;

6 Information of books issued by MCA students

SELECT b.* FROM BOOKMASTER b JOIN ISSUETABLE i ON b.bid = i.bid JOIN STUDENTMASTER s ON s.stud_enrollno = i.stud_enrollno WHERE s.dept = 'MCA';

7 Number of books issued by each student

SELECT S.sname, S.stud_enroll_no, COUNT(I.issueid) AS books_issued FROM STUDENTMASTER S LEFT JOIN ISSUETABLE I ON S.stud_enroll_no = I.stud_enroll_no GROUP BY S.sname, S.stud_enroll_no;

8 Count of available books by "Henry Korth"

SELECT COUNT(*) AS AvailableBooksByHenryKorth FROM BOOKMASTER B JOIN ACCESSIONTABLE A ON B.bid = A.bid WHERE B.author = 'Henry Korth' AND A.avail = 'T';

9 Class-wise issue report of books

SELECT SM.class, COUNT(I.issueid) AS total_issues FROM STUDENTMASTER SM JOIN ISSUETABLE I ON SM.stud_enroll_no = I.stud_enroll_no GROUP BY SM.class;

6---Employee

```
CREATE TABLE EMPLOYEE (
    fname       VARCHAR(50) NOT NULL,
    mname       VARCHAR(50) NOT NULL,
    ssn         CHAR(9)     NOT NULL PRIMARY KEY,  -- Assuming fixed 9-digit SSN
    sex         CHAR(1)     NOT NULL CHECK (sex IN ('M', 'F')),
    salary      DECIMAL(10, 2) NOT NULL,
```

```
   joindate    DATE       NOT NULL,
   superssn    CHAR(9),  -- Can be NULL if top-level employee (e.g., CEO), but per your rule, assume NOT
NULL
   dno        INT        NOT NULL CHECK (dno < 10000),
   FOREIGN KEY (superssn) REFERENCES EMPLOYEE(ssn);

CREATE TABLE DEPT (
   dname    VARCHAR(50) NOT NULL,
   dnum     INT        NOT NULL PRIMARY KEY CHECK (dnum < 10000),
   mgrssn   CHAR(9)     NOT NULL,
   dlocation VARCHAR(100) NOT NULL,
   FOREIGN KEY (mgrssn) REFERENCES EMPLOYEE(ssn)
);

ALTER TABLE DEPT
ADD CONSTRAINT fk_mgrssn FOREIGN KEY (mgrssn) REFERENCES EMPLOYEE(ssn);

  CREATE TABLE PROJECT (
  pname     VARCHAR(50) NOT NULL,
   pno      INT        NOT NULL PRIMARY KEY,
   plocation VARCHAR(100) NOT NULL,
   dnumber   INT        NOT NULL CHECK (dnumber < 10000),
   FOREIGN KEY (dnumber) REFERENCES DEPT(dnum)
);
CREATE TABLE WORKS_ON (
   ssn    CHAR(9) NOT NULL,
```

```
    pno    INT     NOT NULL,
    hours  DECIMAL(4,2) NOT NULL CHECK (hours >= 0),
    PRIMARY KEY (ssn, pno),
    FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn),
    FOREIGN KEY (pno) REFERENCES PROJECT(pno)
);
 INSERT INTO DEPT (dname, dnum, mgrssn, dlocation) VALUES ('HR', 1, '123456789', 'New York');
INSERT INTO EMPLOYEE VALUES ('John', 'Doe', '123456789', 'M', 80000, '2020-01-10', '123456789', 101);
INSERT INTO PROJECT VALUES ('HR System', 1, 'Jalgaon', 101);
INSERT INTO WORKS_ON VALUES ('123456789', 1, 20);
```

Query

a) For every project located in 'jalgaon". List the pno,the controlling detptno and dept manager last name.

```
SELECT P.pno, P.dnumber AS deptno,
  E.lname AS manager_last_name
FROM  PROJECT P
JOIN   DEPT D ON P.dnumber = D.dnum
JOIN
    EMPLOYEE E ON D.mgrssn = E.ssn
WHERE
    LOWER(P.plocation) = 'jalgaon';
```

b) For each project on which more than two employee work, Find the pno, pname & no. of employees who work on the project.

```
SELECT  P.pno,  P.pname,
 COUNT(W.ssn) AS num_of_employees
```

```
FROM  PROJECT P

JOIN  WORK_ON W ON P.pno = W.pno

GROUP BY  P.pno, P.pname

HAVING   COUNT(W.ssn) > 2;
```

c) Express the following constraint as SQL assertions - "salary of employee must not be greater than the salary of the manager of the dept".

```
  CREATE ASSERTION salary_check

CHECK ( NOT EXISTS ( SELECT *

  FROM EMPLOYEE E

    JOIN DEPT D ON E.dno = D.dnum

   JOIN EMPLOYEE M ON D.mgrssn = M.ssn

WHERE E.salary > M.salary

    )

);
```