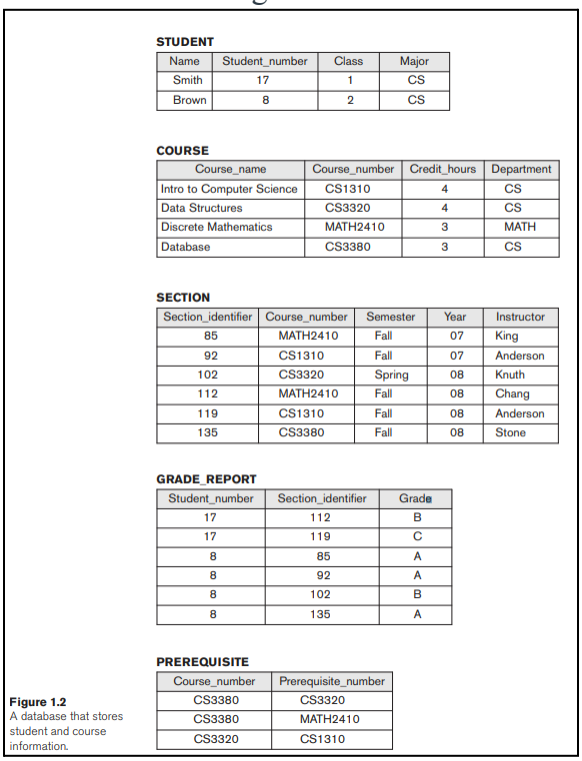
Figure 1.2



STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

1a) Retrieve the names of all senior students majoring in ‘cs’ (computer science).

```
SELECT   Name
FROM     STUDENT
WHERE    Class=4
   AND   Major=’CS’;
```

1b) Retrieve the names of all courses taught by Professor King in 2007 and 2008.

```
SELECT   Course_name
FROM     (COURSE NATURAL JOIN SECTION s)
WHERE    (Year = 7 OR Year = 8)
   AND   s.Instructor = ’King’;
```

1c) For each section taught by Professor King, retrieve the course number, semester, year, and number of students who took the section.

```
SELECT          Course_number, Semester, Year, COUNT(Student_number)
FROM            (SECTION NATURAL JOIN GRADE_REPORT)
WHERE           Instructor=’King’
GROUP BY        Section_identiefier;
```

1d) Retrieve the name and transcript of each senior student (Class = 4) majoring in CS. A transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.

```
SELECT   Name, Course_name, Course_number, Credit_hours, Semester, Year, Grade
FROM     (((STUDENT NATURAL JOIN GRADE_REPORT) NATURAL JOIN
         SECTION) NATURAL JOIN COURSE)
WHERE    Class = 4
   AND   Major = ‘CS’;
```

1e) Retrieve the names and major departments of all straight-A students (students who have a grade of A in all their courses).

```
SELECT   Name, Major
FROM     STUDENT AS S
WHERE    'A' = ALL (SELECT  Grade
                    FROM     GRADE_REPORT AS GR
                    WHERE    S.Student_number = GR.Student_number);
```

1f) Retrieve the names and major departments of all students who do not have a grade of A in any of their courses.

```
SELECT   Name, Major
FROM     STUDENT AS S
WHERE    'A' <> ANY (SELECT        Grade
                     FROM          GRADE_REPORT AS GR
                     WHERE         S.Student_number = GR.Student_number);
```

Figure 5.5

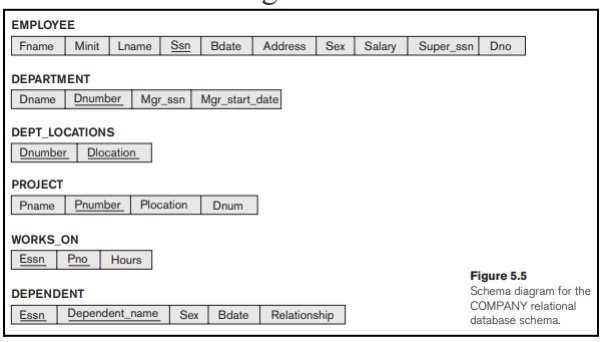


Figure 5.5
Schema diagram for the COMPANY relational database schema.

2a) For each department whose average employee salary is more than $30,000, retrieve the department name and the number of employees working for that department.

```
SELECT      Dname, COUNT(E.*)
FROM        (DEPARTMENT D JOIN EMPLOYEE E ON Dno=Dnumber)
GROUP BY    D.Dnumber;
HAVING      AVG(E.Salary)>30000;
```

2b) Suppose that we want the number of male employees in each department making more than $30,000, rather than all employees. Can we specify this query in SQL? Why or why not?

```
SELECT      Dname, COUNT(E.*)
FROM        (DEPARTMENT D JOIN EMPLOYEE E ON Dno=Dnumber))
WHERE       E.Salary>30000 AND E.Sex='M'
GROUP BY    Dno;
```

Yes we can do this, there is nothing preventing us from specifing only the male employees making more than 30000 in the where clause.

2c) Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees

```
SELECT   N.Fname
FROM     EMPLOYEE N
WHERE    Dno = (SELECT      D.Dno
                FROM        EMPLOYEE D
                WHERE       Salary = (SELECT    MAX(E.Salary)
                                      FROM      EMPLOYEE E));
```

2d) Retrieve the names of all employees whose supervisor's supervisor has '888665555' for Ssn.

```
SELECT   N.Fname
FROM     EMPLOYEE N
WHERE    N.Super_ssn = (SELECT S.Ssn
                        FROM EMPLOYEE S
                        WHERE s.Super_ssn = 888665555);
```

2e) Retrieve the names of employees who make at least $10,000 more than the employee who is paid the least in the company.

```
SELECT   N.Fname
FROM     EMPLOYEE N
WHERE    N.Salary >= 10000 + (SELECT      MIN(M.Salary)
                              FROM        EMPLOYEE M);
```

2f) Find the average salary for employees in each department.

```
SELECT      Dname, AVG(E.Salary)
FROM        (DEPARTMENT D JOIN EMPLOYEE E ON Dno=Dnumber))
GROUP BY    Dno;
```

Figure 6.6

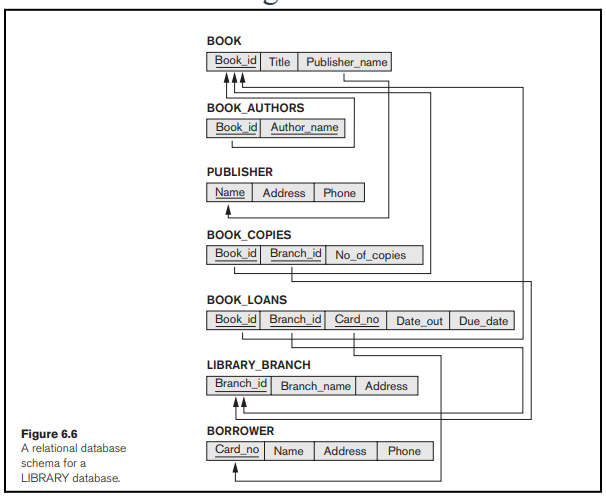


Figure 6.6
A relational database schema for a LIBRARY database.

3a) Retrieve the most popular books in the library

```
SELECT      B.*
FROM        (BOOK B NATURAL JOIN BOOK_LOANS)
GROUP BY    Book_id
HAVING      COUNT(*) = (SELECT      MAX(counts.c)
                        FROM        (SELECT     COUNT(*) as c
                                     FROM       BOOK_LOANS
                                     GROUP BY   Book_id)counts)
```

3b) List branch addresses that house the book titled ‘Don Quixote’

```
SELECT  Address
FROM    ((LIBRARY_BRANCH NATURAL JOIN BOOK_COPIES) NATURAL JOIN
        BOOK)
WHERE Title = ’Don Quixote’
```

3c) Find all borrowers who have checked books authored by: 'JK Rowling'

```
SELECT      B.*
FROM        ((BORROWER B NATURAL JOIN BOOK_LOANS) NATURAL JOIN
            BOOK_AUTHRS)
WHERE       Author_name = ‘JK Rowling’
GROUP BY    Card_no;
```

3d) Retrieve the number of books checked out by a particular borrower: 'Hughie Prim'

```
SELECT      COUNT(*)
FROM        (BORROWER NATURAL JOIN BOOK_LOANS)
WHERE       Name = ‘Hughie Prim’
GROUP BY    Card_no;
```

3e) Retrieve the total number of checkouts for each borrower.

```
SELECT      COUNT(*)
FROM        (BORROWER NATURAL JOIN BOOK_LOANS)
GROUP BY    Card_no;
```

3f) Find the book that had the minimum number of checkouts.

```
SELECT  B.*
FROM    (BOOK B NATURAL JOIN BOOK_COPIES BC)
WHERE   BC.No_of_copies = (SELECT MIN(No_of_copies)
```

FROM BOOK_COPIES);

DEPT_SUMMARY definition

```
CREATE VIEW     DEPT_SUMMARY (D, C, Total_s, Average_s)
AS SELECT       Dno, COUNT (*), SUM (Salary), AVG (Salary)
FROM            EMPLOYEE
GROUP BY        Dno;
```

Figure 5.6

Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

4) State which of the following queries and updates would be allowed on the view. If a query or update would be allowed, show what the corresponding query or update on the base relations would look like, and give its result when applied to the database in Figure 5.6.

a. SELECT *
FROM DEPT_SUMMARY;

```
sqlite> SELECT  *
   ...> FROM    DEPT_SUMMARY;
1|1|55000|55000.0
4|3|93000|31000.0
5|4|133000|33250.0
```

b. SELECT D, C
FROM DEPT_SUMMARY
WHERE TOTAL_S>100000;

```
sqlite> SELECT  D, C
   ...> FROM    DEPT_SUMMARY
   ...> WHERE   TOTAL_S>100000;
5|4
```

c. SELECT D, AVERAGE_S
FROM DEPT_SUMMARY
WHERE C > (SELECT C FROM DEPT_SUMMARY WHERE D = 4);

```
sqlite> SELECT  D, AVERAGE_S
   ...> FROM    DEPT_SUMMARY
   ...> WHERE   C>(SELECT C FROM DEPT_SUMMARY WHERE D = 4)
   ...> ;
5|33250.0
```

d. UPDATE DEPT_SUMMARY
SET D = 3
WHERE D = 4;

```
sqlite> UPDATE  DEPT_SUMMARY
   ...> SET     D=3
   ...> WHERE   D=4;
Parse error: cannot modify DEPT_SUMMARY because it is a view
```

e. DELETE FROM DEPT_SUMMARY
WHERE C > 4;

```
sqlite> DELETE  FROM DEPT_SUMMARY
   ...> WHERE   C>4;
Parse error: cannot modify DEPT_SUMMARY because it is a view
```