Ans1. Keeping organizationalinformationinafile-processingsystemhasanumber of major disadvantages:

• Data redundancy and inconsistency. Since different programmers create the files and application programs over a long period, the various files are likelytohavedifferentstructuresandtheprogramsmaybewritteninseveral programminglanguages.Moreover,thesameinformationmaybeduplicated in several places (files). For example, if a student has a double major (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematicsdepartment.Thisredundancyleadstohigherstorageandaccess cost.Inaddition,itmayleadtodatainconsistency;thatis,thevariouscopies ofthesamedatamaynolongeragree.Forexample,achangedstudentaddress may be reflected in the Music department records but not elsewhere in the system.

• Difficulty in accessing data. Suppose thatone ofthe universityclerksneeds tofindoutthenamesofallstudentswholivewithinaparticularpostal-code area. The clerk asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of all students. The university clerk has now two choices: either obtain the list of all students and extract the needed information manually or ask a programmer to write the necessary applicationprogram.Bothalternativesareobviouslyunsatisfactory.Suppose that such a program is written, and that, several days later, the same clerk needs to trim that list to include only those studentswho have taken at least 60 credit hours. As expected, a program to generate such a list does not exist. Again, the clerk has the preceding two options, neither of which is satisfactory. The point here is that conventional file-processing environments do not allowneededdatatoberetrievedinaconvenientandefficientmanner.More responsive data-retrievalsystemsare requiredfor general use.

• Data isolation. Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

• Integrity problems. The data values stored in the database must satisfy certain types of consistency constraints. Suppose the university maintains an account for each department, and records the balance amount in each account.Supposealsothattheuniversityrequiresthattheaccountbalanceofa department may never fall below zero. Developersenforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change theprogramstoenforcethem.Theproblemiscompoundedwhenconstraints involve several data items from differentfiles.

Ans2.

- Comparision on character string: The SQL standard specifies that the equality operation on strings is case sensitive; as a result the expression “’comp. sci.’ = ’Comp. Sci.’” evaluates to false. Pattern matching can be performed on strings, using the operator like. We describe patterns by

using two special characters:• Percent (%): The % character matches any substring. • Underscore ( ): The character matches any character.select dept_name from department where building like '%Watson%';

- Having : Predicates in the having clause are applied after the formation of groups to specify some conditions on the query. example: to find name and avg salary of all departments whose avg salary is greater then 42000. select dept_name, avg(salary) from instructor group by dept_name having salary > 42000

- Set comparison: In sql we can compare two sets by using 'some' and 'all' constructs. The phrase"greater than at least one"is represented in SQL by > some.The construct > all corresponds to the phrase "greater than all." example: "Find the departments that have the highest average salary." - select dept name from instructor group by dept name having avg (salary) >=all (select avg (salary) from instructor group by dept name);

- Subqueries in from clause: SQL allows a subquery expression to be used in the from clause.The key concept applied here is that any select-from-where expression returns a relation as a result and, therefore, can be inserted into another select-from-where anywhere that a relation can appear. Example - "Find the average instructors' salaries of those departments where the average salary is greater than $42,000." select dept name, avg salary from (select dept name, avg (salary) as avg salary from instructor group by dept name) where avg salary > 42000;

Ans3.

1. select cname from customers,items,buys where customer.cid=buys.cid and items.iid=buys.iid and items.brand='Cotton World';

2. select count(iid),brand from items group_by brand having count(iid)>=10;

3. select distinct cname from customer as c where not exists ((select iid from items where brand='MTR') minus (select iid from buys as b where c.cid=b.cid))

4. with table1(brand,total_cost) as (select brand,sum(cost) from item group by brand) select brand from table1 where total_cost > all (select avg(total_cost) from table1);

Ans4.

1. with table1(aid,bcount) as (select aid,count(isbn) from writes group by aid) select aname from author a,table1 w where a.aid=w.aid and w.bcount>1;

2. with table1(isbn,auth_count) as (select isbn,count(aid) from writes group by isbn) select title from book b,table1 w where b.isbn=w.isbn and w.auth_count=1;

3. with table1(aid,b_count) as (select aid,count(isbn) from writes group by aid)select aname from author a,table1 w where a.aid=w.aid and w.b_count >= all(select max(b_count) from table1);

4. select pname from publisher p where not exists((select distinct category from book) minus (select distinct category from book b where b.pname=p.pname));

Ans5.

1. select e1.employee_name from employee as e1,employees as e2
2. To find name of employees whose salary is greater than at least one of the employee who works in 'TISCO'. select employee_name from works where salary>some(select salary from works where company_name='TISCO');
3. To find employee names who are not managers. select employee_name from employee E where not exists(select * from Manages M where M.manager_name=E.employee_name);
4. To find the name of employees who works in 'Microsoft' and has salary > 40000. with emp(name,salary) as (select employee_name,salary from works where company='Microsoft') select name from emp where salary>40000;