Database Applications Laboratory

1.

Consider the insurance database given below. The primary keys are made bold and the data types are specified.
PERSON(**driver_id**:string , name:string , address:string)
CAR(**regno**:string , model:string , year:int)
ACCIDENT(**report_number**:int , accd_date:date , location:string)
OWNS(**driver_id**:string , **regno**:string)
PARTICIPATED(**driver_id**:string , **regno**:string , **report_number**:int , damage_amount:int)
1)Create the above tables by properly specifying the primary keys and foreign keys.
2)Enter at least five tuples for each relation.
3)Demonstrate how you
a.Update the damage amount for the car with specific regno in the accident with report number 12 to 25000.
b.Add a new accident to the database.
4)Find the total number of people who owned cars that were involved in accidents in the year 2008.
5)Find the number of accidents in which cars belonging to a specific model were involved.

2.

Consider the following relations for a order processing database application in a company.
CUSTOMER(**custno**:int , cname:string , city:string)
ORDER(**orderno**:int , odate:date , custno:int , ord_amt:int)
ORDER_ITEM(**orderno**:int , **itemno**:int , quantity:int)
ITEM(**itemno**:int , unitprice:int)
SHIPMENT(**orderno**:int , **warehouseno**:int , ship_date:date)
WAREHOUSE(**warehouseno**:int , city:string)
1)Create the above tables by properly specifying the primary keys and foreign keys.
2)Enter at least five tuples for each relation.
3)Produce a listing: custname , No_of_orders , Avg_order_amount , where the middle column is the total number of orders by the customer and the last column is the average order amount for that customer.
4)List the orderno for orders that were shipped from ***all*** the warehouses that the company has in a specific city.
5)Demonstrate the deletion of an item from the ITEM table and demonstrate a method of handling the rows in the ORDER_ITEM table that contains this particular item.

3.

Consider the following database of student enrollment in courses and books adopted for that course.
STUDENT(**regno**:string , name:string , major:string , bdate:date)
COURSE(**courseno**:int , cname:string , dept:string)
ENROLL(**regno**:string , **courseno**:int , **sem**:int , marks:int)
BOOK_ADOPTION(**courseno**:int , **sem**:int , book_isbn:int)
TEXT(**book_isbn**:int , book_title:string , publisher:string , author:string)
1)Create the above tables by properly specifying the primary keys and foreign keys.
2)Enter atleast five tuples for each relation.
3)Demonstrate how you add a new text book to the database and make this book to be adopted by some department.
4)Produce a list of text books (includes courseno , book_isbn , book_title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.
5)List any department that has ***all*** its books published by a specific publisher.

4.

The following are maintained by abook dealer.
AUTHOR(**author_id**:int , name:string , city:string , country:string)
PUBLISHER(**publisher_id**:int , name:string , city:string , country:string)
CATALOG(**book_id**:int , title:string , author_id:int , publisher_id:int , category_id:int , year:int , price:int)

CATEGORY(**category_id**:int , description:string)
ORDER_DETAILS(**order_no**:int , **book_id**:int , quantity:int)
1)Create the above tables by properly specifying the primary keys and foreign keys.
2)Enter at least five tuples for each relation.
3)Give the details of the authors who have 2 or more books in the catalog and the price of the books is greater than the average price of the books in the catalog and the year of publication is after 2000.
4)Find the author of the book that has maximum sales.
5)Demonstrate how you increase the price of books published by a specific publisher by 10%.

5.
Consider the following database for a banking enterprise.
BRANCH(**branch_name**:string , branch_city:string , assets:real)
ACCOUNT(**accno**:int , branch_name:string , balance:real)
DEPOSITOR(**customer_name**:string , **accno**:int)
CUSTOMER(**customer_name**:string , customer_street:string , customer_city:string)
LOAN(**loan_number**:int , branch_name:string , amount:real)
BORROWER(**customer_name**:string , **loan_number**:int)
1)Create the above tables by properly specifying the primary keys and foreign keys.
2)Enter at least five tuples for each relation.
3)Find ***all*** the customers who have at least two accounts at the ***main*** branch.
4)Find all the customers who have an account at ***all*** the branches located in a specific city.
5)Demonstrate how you delete all account tuples at every branch located in a specific city.

1.Insurance database.

1.

SQL> create table person(driver_id varchar(10),name varchar(10),address varchar(10),primary key(driver_id));

SQL> create table car(regno varchar(10),model varchar(10),year int,primary key(regno));

SQL> create table accident(report_number int,accd_date date,location varchar(10),primary key(report_number));

SQL> create table owns(driver_id varchar(10),regno varchar(10),primary key(driver_id,regno),foreign key(driver_id) references person(driver_id),foreign key(regno) references car(regno));

SQL> create table participated(driver_id varchar(10),regno varchar(10),report_number int,damage_amount int,primary key(driver_id,regno,report_number),foreign key(driver_id) references person(driver_id),foreign key(regno) references car(regno),foreign key(report_number) references accident(report_number));

2.

SQL> insert into person values('&driver_id','&name','&address');

SQL> insert into car values('&regno','&model',&year);

SQL> insert into accident values(&report_number,'&accd_date','&location');

SQL> insert into owns values('&driver_id','&regno');

SQL> insert into participated values('&driver_id','&regno',&report_number,&damage_amount);

3a.

SQL> update participated set damage_amount=25000 where report_number=12 and regno='5';

3b.

```
SQL> insert into accident values(&report_number,'&accd_date','&location');

SQL> insert into participated values('&driver_id','&regno',&report_number,&damage_amount);
```

4.

```
SQL> select count(distinct o.driver_id) as People from owns o,participated p,accident a where
a.accd_date like
 '%08' and o.regno=p.regno and p.report_number=a.report_number;
```

5.

```
SQL> select count(*) as Totalcars from car c,participated p where c.regno=p.regno and
c.model='Alto';
```

2.Order processing database.

1.

```
SQL> create table customer(custno int,cname varchar(10),city varchar(10),primary key(custno));

SQL> create table order1(orderno int,odate date,custno int,ord_amt int,primary
key(orderno),foreign key(custno) references customer(custno));

SQL> create table item(itemno int,untiprice int,primary key(itemno));

SQL> create table order_item(orderno int,itemno int,quantity int,primary
key(orderno,itemno),foreign key(orderno) references order1(orderno),foreign key(itemno)
references item(itemno) on delete cascade);

SQL> create table warehouse(warehouseno int,city varchar(10),primary key(warehouseno));

SQL> create table shipment(orderno int,warehouseno int,ship_date date,primary
key(orderno,warehouseno),foreign key(orderno) references order1(orderno),foreign
key(warehouseno) references warehouse(warehouseno));
```

2.

```
SQL> insert into customer values(&custno,'&cname','&city');

SQL> insert into order1 values(&orderno,'&odate',&custno,&ord_amt);

SQL> insert into item values(&itemno,&unitprice);

SQL> insert into order_item values(&orderno,&itemno,&quantity);

SQL> insert into warehouse values(&warehouseno,'&city');

SQL> insert into shipment values(&orderno,&warehouseno,'&ship_date');
```

3.

```
SQL> select c.custno,count(*) as No_of_orders,avg(o.ord_amt) as Avg_order_amount from
customer c,order1 o where o.custno=c.custno group by c.custno;
```

4.

SQL> select distinct s.orderno from shipment s where not exists((select warehouseno from warehouse where city='Bangalore') minus (select w.warehouseno from shipment w where w.orderno=s.orderno))and exists (select warehouseno from warehouse where city='Bangalore');

OR

SQL> select s.orderno from shipment s,warehouse w where s.warehouseno=w.warehouseno and w.city='Bangalore' group by orderno having count(*)=(select count(*) from warehouse where city='Bangalore') and not(count(*)=0);

5.

SQL> delete from item where itemno=3;

3.Student enroll database.

1.

SQL> create table student(regno varchar(10),name varchar(10),major varchar(10),bdate date,primary key(regno));

SQL> create table course(courseno int,cname varchar(10),dept varchar(10),primary key(courseno));

SQL> create table enroll(regno varchar(10),courseno int,sem number(1),marks number(3),primary key(regno,courseno,sem),foreign key(regno) references student(regno),foreign key(courseno) references course(courseno));

SQL> create table text(book_isbn int,book_title varchar(10),publisher varchar(10),author varchar(10),primary key(book_isbn));

SQL> create table book_adoption(courseno int,sem int,book_isbn int,primary key(courseno,sem),foreign key(courseno) references course(courseno),foreign key(book_isbn) references text(book_isbn));

2.

SQL> insert into student values('&regno','&name','&major','&bdate');

SQL> insert into course values(&courseno,'&cname','&dept');

SQL> insert into enroll values('&regno',&courseno,&sem,&marks);

SQL> insert into text values(&book_isbn,'&book_title','&publisher','&author');

SQL> insert into book_adoption values(&courseno,&sem,&book_isbn);

3.

SQL> insert into text values(&book_isbn,'&book_title','&publisher','&author');

SQL> insert into book_adoption values(&courseno,&sem,&book_isbn);

4.

SQL> select b.courseno,t.book_isbn,t.book_title from book_adoption b,text t,course c where c.courseno=b.courseno and c.dept='CS' and b.book_isbn=t.book_isbn and c.courseno in (select courseno from (select courseno from book_adoption group by courseno,book_isbn) group by courseno having count(*)>=2) order by t.book_title;

5.

SQL> select distinct c.dept from course c where not exists ((select b.book_isbn from book_adoption b,course c1 where c1.courseno=b.courseno and c1.dept=c.dept) minus (select book_isbn from text where publisher='BPB')) and exists (select b.book_isbn from book_adoption b,course c1 where c1.courseno=b.courseno and c1.dept=c.dept);

4.Book dealer.

1.

SQL> create table author(author_id int,name varchar(10),city varchar(10),country varchar(10),primary key(author_id));

SQL> create table publisher(publisher_id int,name varchar(10),city varchar(10),country varchar(10),primary key(publisher_id));

SQL> create table category(category_id int,description varchar(10),primary key(category_id));

SQL> create table catalog(book_id int,title varchar(10),author_id int,publisher_id int,category_id int,year int,price int,primary key(book_id),foreign key(author_id) references author(author_id),foreign key(publisher_id) references publisher(publisher_id),foreign key(category_id) references category(category_id));

SQL> create table order_details(order_no int,book_id int,quantity int,primary key(order_no,book_id),foreign key(book_id) references catalog(book_id));

2.

SQL> insert into author values(&author_id,'&name','&city','&country');

SQL> insert into publisher values(&publisher_id,'&name','&city','&country');

SQL> insert into category values(&category_id,'&description');

SQL> insert into catalog values(&book_id,'&title',&author_id,&publisher_id,&category_id,&year,&price);

SQL> insert into order_details values(&order_no,&book_id,&quantity);

3.

SQL> select * from author where author_id in (select author_id from catalog where year>2000 and price>(select avg(price) from catalog) group by author_id having count(*)>=2);

4.

SQL> select a.author_id,a.name,a.city,a.country from author a,catalog c where c.author_id=a.author_id and c.book_id=(select book_id from order_details group by book_id having sum(quantity)=(select max(quantity) from (select sum(quantity) as quantity from order_details group by book_id)));

5.

SQL> update catalog set price=1.1*price where publisher_id=(select publisher_id from publisher where name='BPB');

5.Banking enterprise.

SQL> 1.

SQL> create table branch(branch_name varchar(10),branch_city varchar(10),assets int,primary key(branch_name));

SQL> create table account(accno int,branch_name varchar(10),balance int,primary key(accno));

```
SQL> create table customer(customer_name varchar(10),customer_street
varchar(10),customer_city varchar(10),primary key(customer_name));

SQL> create table depositor(customer_name varchar(10),accno int,primary
key(customer_name,accno),foreign key(customer_name) references
customer(customer_name),foreign key(accno) references account(accno) on delete cascade);

SQL> create table loan(loan_number int,branch_name varchar(10),amount int,primary
key(loan_number),foreign key(branch_name) references branch(branch_name));

SQL> create table borrower(customer_name varchar(10),loan_number int,primary
key(customer_name,loan_number),foreign key(customer_name) references
customer(customer_name),foreign key(loan_number) references loan(loan_number));
```

2.

```
SQL> insert into branch values('&branch_name','&branch_city',&assets);

SQL> insert into account values(&accno,'&branch_name',&balance);

SQL> insert into customer values('&customer_name','&customer_street','&customer_city');

SQL> insert into depositor values('&customer_name',&accno);

SQL> insert into loan values(&loan_number,'&branch_name',&amount);

SQL> insert into borrower values('&customer_name',&loan_number);
```

3.

```
SQL> select d.customer_name from depositor d,account a where a.accno=d.accno and
a.branch_name='KRMarket' group by d.customer_name having count(*)>=2;
```

4.

```
SQL> select c.customer_name from customer c where exists(select branch_name from branch
where branch_city='Bangalore') and not exists ((select branch_name from branch where
branch_city='Bangalore') minus (select b.branch_name from branch b,account a,depositor d
where d.customer_name=c.customer_name and d.accno=a.accno and
a.branch_name=b.branch_name));
```

5.

```
SQL> delete from account where accno in (select a.accno from account a,branch b where
a.branch_name=b.branch_name and b.branch_city='Bangalore');
```