I. Insurance Database

Consider the Insurance database given below.

PERSON (driver – id #: String, name: string, address: string)

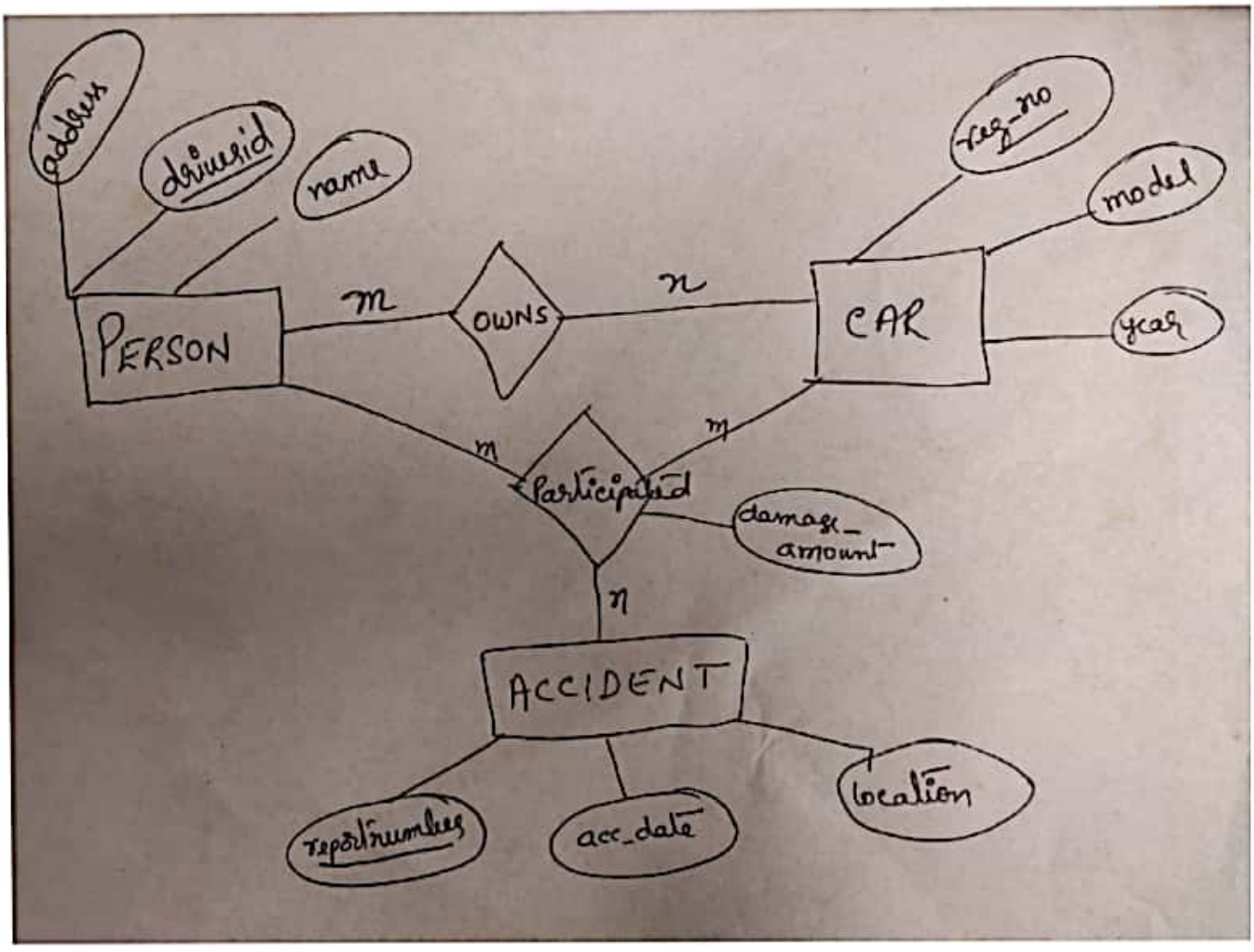
CAR (regno: string, model: string, year: int)

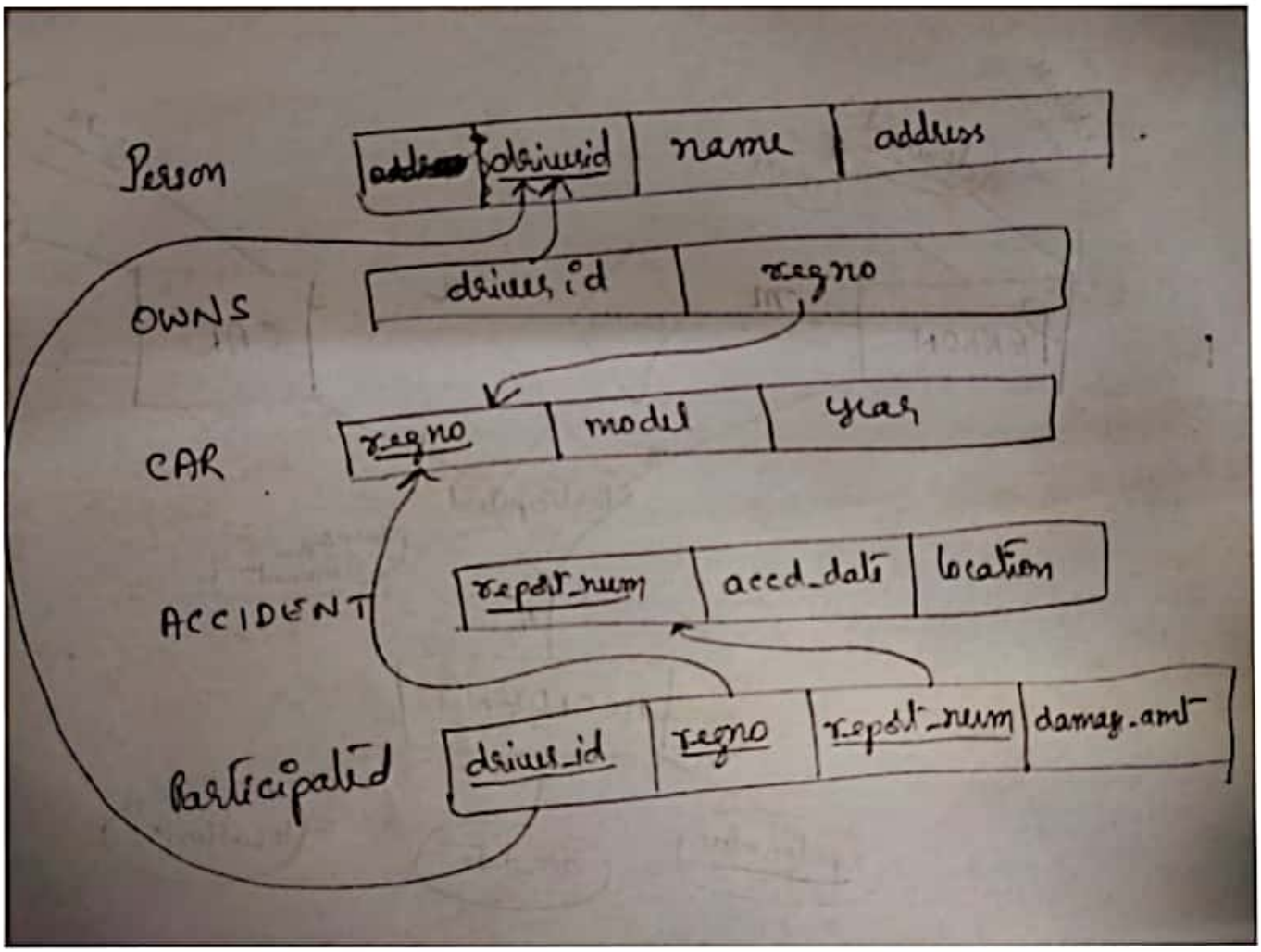
ACCIDENT (report-number: int, accd-date: date, location: string)

OWNS (driver-id #: string, regno: string)

PARTICIPATED (driver-id: string, Regno: string, report-number: int, damage amount: int)

1. Find the total number of people who owned cars that were involved in accidents in 1989.
2. Find the number of accidents in which the cars belonging to "John Smith" were involved.
3. Update the damage amount for the car with reg number "KA-12-1" in the accident with report number "1" to $3000.






```
CREATE TABLE PERSON (
                    driverid varchar(15),
                    pname VARCHAR(15) NOT NULL,
                    address varchar(30),
                    PRIMARY KEY (driverid)
                    )

CREATE TABLE CAR (
                    regno varchar(15),
                    model varchar(20)NOT NULL,
                    regyear INTEGER,
                    PRIMARY KEY(regno)
                )

CREATE TABLE OWNS(
                    driverid varchar(15),
                    regno varchar(15),
                    PRIMARY KEY(driverid, regno),
                    FOREIGN KEY(driverid) REFERENCES PERSON(driverid)ON
DELETE CASCADE ON UPDATE CASCADE,
                    FOREIGN KEY (regno) REFERENCES CAR(regno) ON DELETE
CASCADE ON UPDATE CASCADE     )
```

```
CREATE TABLE ACCIDENT (
                        reportno INTEGER,
                        accdate DATE,
                        location VARCHAR(20),
                        PRIMARY KEY(reportno)
                        )


CREATE TABLE PARTCIPATED(
                        driverid varchar(15) ,
                        regno varchar(15),
                        reportno  INTEGER,
                        damageamt INTEGER,
                        primary key(driverid, regno, reportno),
                        foreign key(driverid) references PERSON(driverid)on delete cascade
on update cascade,
                        foreign key(regno) references CAR(regno)on delete cascade on
update cascade,
                        foreign key(reportno) references ACCIDENT(reportno)  on delete
cascade on update cascade,
                        foreign key(driverid,regno) references OWNS(driverid,regno)
                       )

insert into  PERSON values (‘driver1','John Smith' , 'SP Road, Bangalore-12')
insert into  PERSON values (‘driver2','Ramesh Babu' , 'KP Nagar, Udupi -13')
insert into  PERSON values ('driver3','Raju SK' , 'KS Circle, Mangalore-12')
insert into  PERSON values ('driver4','Ramesh
Babu' , 'AS Road, Bangalore-14')
insert into  PERSON values ('driver5','Alica wallace' , 'SS Road, Karkala-16')


insert into  CAR values ('KA-12-1','FORD' ,1980)
insert into  CAR values ('KA-13-1','SWIFT' ,1990)
insert into  CAR values ('MH-11-1','INDIGO' ,1998)
insert into  CAR values ('AP-10-1','SWIFT' ,1988)

insert into  CAR values ('TN-11-1','FORD' ,2001)
insert into  CAR values ('TN-11-2','TOYATA' ,2001)
insert into  CAR values ('MH-14-1','SWIFT' ,2001)
insert into  CAR values ('KL-15-1','TOYATA' ,2001)
insert into  CAR values ('KL-4-1','INDIGO' ,2001)
insert into  CAR values ('AP-10-2','SANTRO' ,2001)

insert into  CAR values ('TN-12-1','scross' ,2018)

insert into  CAR values ('AP-05-1','baleno' ,2015)



insert into  OWNS values ('driver1','KA-13-1')
```

```
insert into  OWNS values ('driver1','KA-12-1')
insert into  OWNS values ('driver1','MH-11-1')
insert into  OWNS values ('driver2','AP-10-1')
insert into  OWNS values ('driver2','TN-11-1')

insert into  OWNS values ('driver3','TN-12-1')
insert into  OWNS values ('driver3','KL-15-1')
insert into  OWNS values ('driver4','AP-05-1')
insert into  OWNS values ('driver4','KL-4-1')
insert into  OWNS values ('driver5','MH-14-1')


insert into  ACCIDENT values (1,'1998-07-22' ,'Nitte')
insert into  ACCIDENT values (2,'1998-07-22','Karkala')
insert into  ACCIDENT values (12,'1998-07-22' ,'Mangalore')
insert into  ACCIDENT values (3,'1998-07-23','Mangalore')

insert into  ACCIDENT values (4,'1990-09-09','Bhatkal')
insert into  ACCIDENT values (5,'2001-02-22' ,'Udupi')
insert into  ACCIDENT values (6,'1990-09-09','Udupi')
insert into  ACCIDENT values (15,'1981-07-22' ,'Udupi')
insert into  ACCIDENT values (7,'1981-09-09','Karkala')
insert into  ACCIDENT values (8,'1990-09-09','Bhatkal')
insert into  ACCIDENT values (9,'2001-02-22' ,'Udupi')
insert into  ACCIDENT values (10,'1998-02-02','Udupi')
insert into  ACCIDENT values (11,'1998-01-02','Bhatkal')
insert into  ACCIDENT values (13,'1998-07-22','Udupi')
insert into  ACCIDENT values (14,'1998-07-22','Karkala')


insert into  PARTCIPATED values ('driver1','KA-12-1',1,20000)
insert into  PARTCIPATED values ('driver1','KA-13-1',2,10000)
insert into  PARTCIPATED values ('driver1','KA-12-1',3,60000)
insert into  PARTCIPATED values ('driver1','KA-12-1',4,60000)
insert into  PARTCIPATED values ('driver1','KA-12-1',5,60000)
insert into  PARTCIPATED values ('driver1','KA-12-1',15,40000)
insert into  PARTCIPATED values ('driver1','KA-13-1',6,10000)
insert into  PARTCIPATED values ('driver1','MH-11-1',12,20000)
insert into  PARTCIPATED values ('driver2','AP-10-1',7,30000)
insert into  PARTCIPATED values ('driver2','TN-11-1',8,40000)
insert into  PARTCIPATED values ('driver2','AP-10-1',13,20000)
insert into  PARTCIPATED values ('driver2','TN-11-1',14,10000)
insert into  PARTCIPATED values ('driver3','TN-12-1',9,40000)
insert into  PARTCIPATED values ('driver3','KL-15-1',10,50000)
insert into  PARTCIPATED values ('driver3','TN-12-1',11,20000)
```

Query 1: Find the total number of people who owned cars that were involved in accidents in 1989.

Answer:

SELECT COUNT(DISTINCT P.driverid) AS NO_OF_PEOPLE FROM ACCIDENT A, PARTICIPATED P WHERE A. reportno = P. reportno AND YEAR(A. accdate)=1989

Query 2. Find the number of accidents in which the cars belonging to "John Smith" were involved.

Answer:

SELECT COUNT(P. reportno) AS NO_OF_ACCIDENTS FROM PARTICIPATED P, PERSON PN WHERE P.driverid=PN.driverid AND PN.pname='John Smith'

Query 3. Update the damage amount for the car with reg number "KA-12-1" in the accident with report number "1" to $3000.

Answer:

UPDATE PARTICIPATED SET damageamt=30000 WHERE reportno=1 AND regno='KA-12-1'

II. Order Database

Consider the following relations for an order processing database application in a company:

CUSTOMER (cust #: int, cname: string, city: string)

ORDER (order #: int, odate: date, cust #: int, ord-Amt: int)

ORDER – ITEM (order #: int, item #: int, aqty: int)

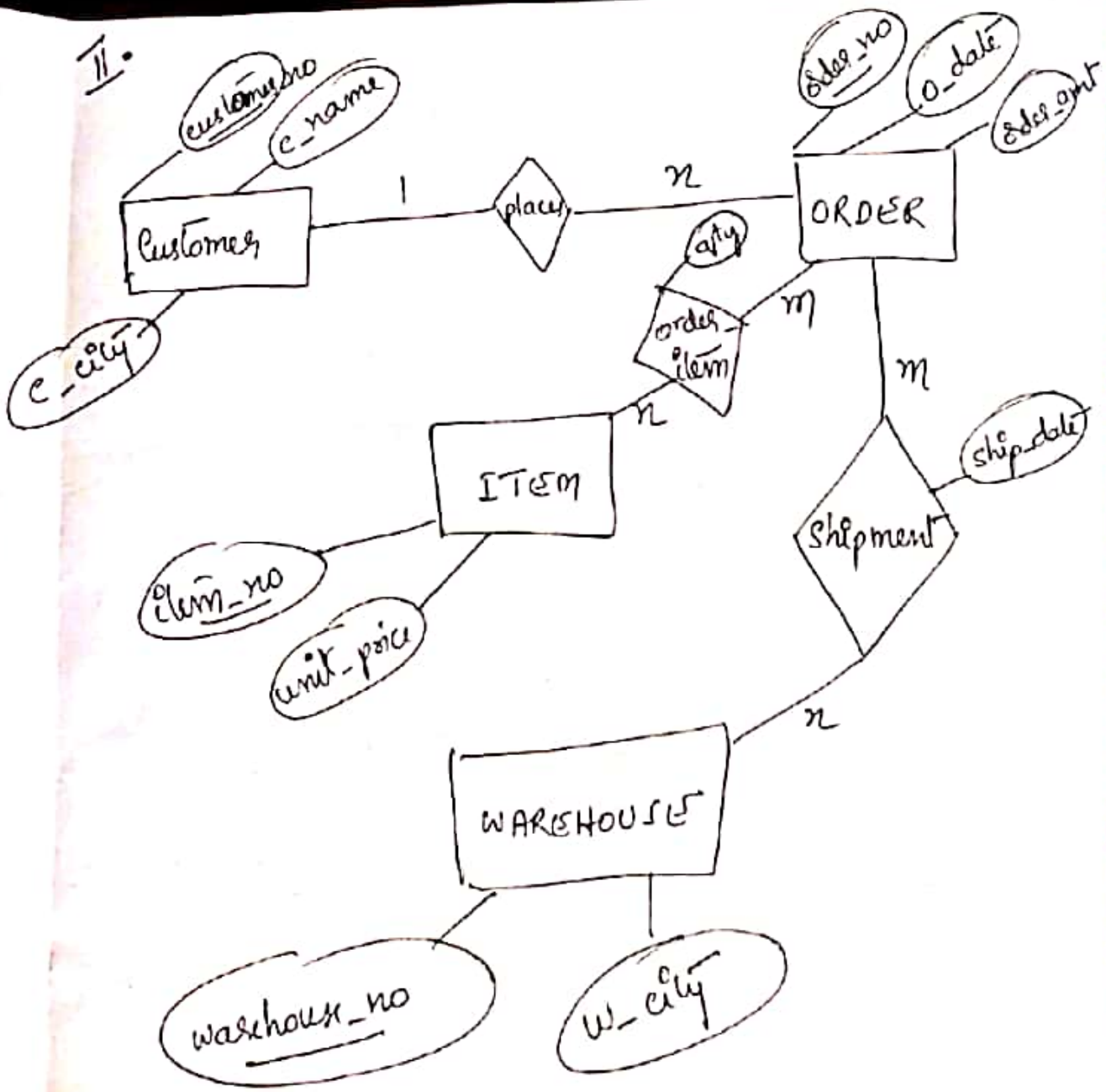
ITEM (item #: int, unit price: int)

SHIPMENT (order #: int, warehouse#: int, ship-date: date)

WAREHOUSE (warehouse #: int, city: string)

1. Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle column is the total numbers of orders by the customer and the last column is the average order amount for that customer.
2. For each item that has more than two orders, list the item, number of orders that are shipped from atleast two warehouses and total quantity of items shipped.
3. List the customers who have ordered for every item that the company produces.

11.



Relational schema:-

Customer: | customer_no | c_name | c_city |

ORDER: | order_no | o_date | customer_no | order_amt |

ITEM: | item_no | unit_price |

ORDER_ITEM: | order_no | item_no | qty |

WAREHOUSE: | warehouse_no | w_city |

SHIPMENT: | order_no | warehouse_no | ship_date |

```
CREATE TABLE CUSTOMER (
                        custid int,
                        custname varchar (20) NOT NULL,
                        custcity varchar (30),
                        primary key (custid)
                        )

insert into CUSTOMER values (111,'John Smith', 'Karkala')
insert into CUSTOMER values (112,'Ramesh N', 'Nitte')
insert into CUSTOMER values (113,'Franklin', 'Karkala')
insert into CUSTOMER values (114,'Alica', 'mangalore')
insert into CUSTOMER values (115,'Raju', 'Udupi')


CREATE TABLE ORDER (
                        orderid int,
                        odate datetime,
                        custid int,
                        ordamt int,
                        primary key (orderid) ,
                        foreign key(custid) references CUSTOMER (custid) on delete cascade
on update cascade
                        )

insert into ORDER values (201,'2001-08-03', 111, 0)
insert into ORDER values (202,'2002-08-03', 111, 0)
insert into ORDER values (203,'2001-08-04', 112, 0)
insert into ORDER values (204,'2004-02-01', 113, 0)
insert into ORDER values (205,'2001-04-02', 114, 0)
insert into ORDER values (206,'2005-02-01', 115, 0)
insert into ORDER values (207,'2008-04-01', 115, 0)
insert into ORDER values (209,'2008-02-01', 114, 0)
insert into ORDER values (208,'2008-12-01', 111, 0)
insert into ORDER values (200,'2008-11-01', 111, 0)
insert into ORDER values (210,'2008-10-01', 111, 0)


CREATE TABLE ITEM (
                        itemid  int,
                        price int,
                        primary key (itemid)
                )

insert into ITEM values (301,2000)
insert into ITEM values (302,2000)
insert into ITEM values (303,1000)
insert into ITEM values (304,5000)
insert into ITEM values (305,4000)
```

```
CREATE TABLE ORDER_ITEM (
                        orderid int,
                        itemid int,
                        qty int,
                        primary key (orderid, itemid),
                        foreign key(orderid) references ORDER(orderid) on delete cascade on
update cascade,
                        foreign key(itemid) references ITEM(itemid) on delete cascade on
update cascade
                        )

insert into ORDER_ITEM values (201,301,2)
insert into ORDER_ITEM values (201,302,4)
insert into ORDER_ITEM values (201,303,4)
insert into ORDER_ITEM values (201,304,4)
insert into ORDER_ITEM values (201,305,3)

insert into ORDER_ITEM values (202,303,2)
insert into ORDER_ITEM values (202,305,4)
insert into ORDER_ITEM values (203,302,1)
insert into ORDER_ITEM values (204,305,2)
insert into ORDER_ITEM values (205,301,3)
insert into ORDER_ITEM values (206,301,5)
insert into ORDER_ITEM values (206,302,3)


update ORDER set ordamt =
(select sum(OI.qty * I.price) from ORDER_ITEM OI, ITEM I
                        where OI.itemid = I.itemid and OI.orderid = 201)
where orderid = 201

update ORDER set ordamt =
(select sum(OI.qty * I.price) from ORDER_ITEM O, ITEM I
                        where OI.itemid = I.itemid and OI.orderid = 202)
where orderid = 202

update ORDER set ordamt =
(select sum(OI.qty * I.price) from ORDER_ITEM O, ITEM I
                        where OI.itemid = I.itemid and OI.orderid = 203)
where orderid = 203

update ORDER set ordamt =
(select sum(OI.qty * I.price) from ORDER_ITEM O, ITEM I
                        where OI.itemid = I.itemid and OI.orderid = 204)
where orderid = 204

update ORDER set ordamt =
(select sum(OI.qty * I.price) from ORDER_ITEM O, ITEM I
                        where OI.itemid = I.itemid and OI.orderid = 205)
```

```
where orderid = 205

update ORDER set ordamt  =
(select sum(OI.qty * I.price) from ORDER_ITEM O, ITEM I
                    where OI.itemid = I.itemid and OI.orderid = 206)
where orderid = 206



CREATE TABLE WAREHOUSE (
                        warehouseid int,
                        w_city varchar(20) not null,
                        primary key (warehouseid)
                    )

insert into WAREHOUSE values (1,'MAGALORE')
insert into WAREHOUSE values (2,'MAGALORE')
insert into WAREHOUSE values (3,'MAGALORE')
insert into WAREHOUSE values (4,'UDUPI')
insert into WAREHOUSE values (5,'UDUPI')
insert into WAREHOUSE values (6,'KARKALA')


CREATE TABLE SHIPMENT (
                        orderid int,
                        warehouseid int,
                        shipdate date,
                        primary key (orderid, warehouseid) ,
                        foreign key(orderid) references ORDER(orderid) on delete cascade on
update cascade,
                        foreign key(warehouseid) references WAREHOUSE(warehouseid) on
delete cascade on update cascade
                    )


insert into SHIPMENT values (201,1,'2001-04-02')
insert into SHIPMENT values (201,2,'2001-04-04')
insert into SHIPMENT values (202,1,'2001-05-02')

insert into SHIPMENT values (202,2,'2002-05-12')
insert into SHIPMENT values (202,3,'2003-06-01')
insert into SHIPMENT values (202,4,'2003-06-01')
insert into SHIPMENT values (203,1,'2004-02-01')
insert into SHIPMENT values (203,2,'2004-02-01')
insert into SHIPMENT values (203,3,'2004-02-01')
insert into SHIPMENT values (204,4,'2004-06-02')
insert into SHIPMENT values (204,2,'2004-06-02')
```

1. Produce a listing: CUSTNAME, nooforders, AVG_ORDER_AMT, where the middle column is the total numbers of orders by the customer and the last column is the average order amount for that customer.

```
select C.custname, count(O.orderid) as NO_OF_ORDR, avg(O.ordamt) as AVG_ORD_AMT from CUSTOMER C, ORDER O
where C.custid = O.custid group by C. custname
```

2. For each item that has more than two orders, list the item, number of orders that are shipped from atleast two warehouses and total quantity of items shipped.

```
select itemid,count(*),sum(qty) from ORDER_ITEM where orderid in
  (select orderid from SHIPMENT group by orderid having count(*) >=2) group by itemid having count(*) > =2
```

3. List the customers who have ordered for every item that the company produces.

```
select C.custname from CUSTOMER C where NOT EXISTS
(
        (select itemid from ITEM)
        EXCEPT
        (select distinct(OI.itemid) from ORDER O,ORDER_ITEM OI where O.orderid = OI.orderid and O.custid = C.custid )
)
```

III. STUDENT COURSE ENROLLMENT

Consider the following database of student enrollment in courses & books adopted for each course:

STUDENT (regno: string, name: string, major: string, bdate: date)

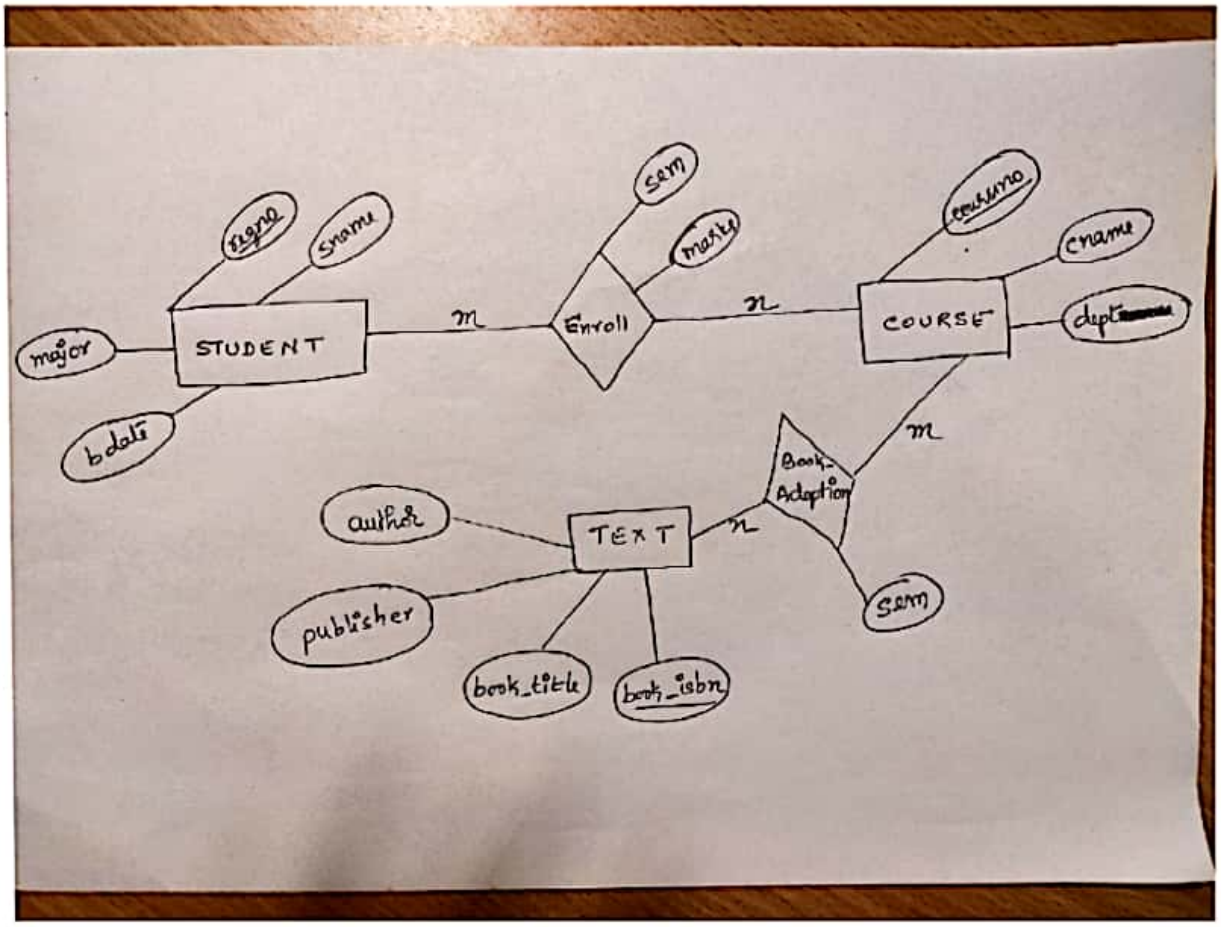
COURSE (course #: int, cname: string, dept: string)

ENROLL (regno: string, course#: int, sem: int marks: int)

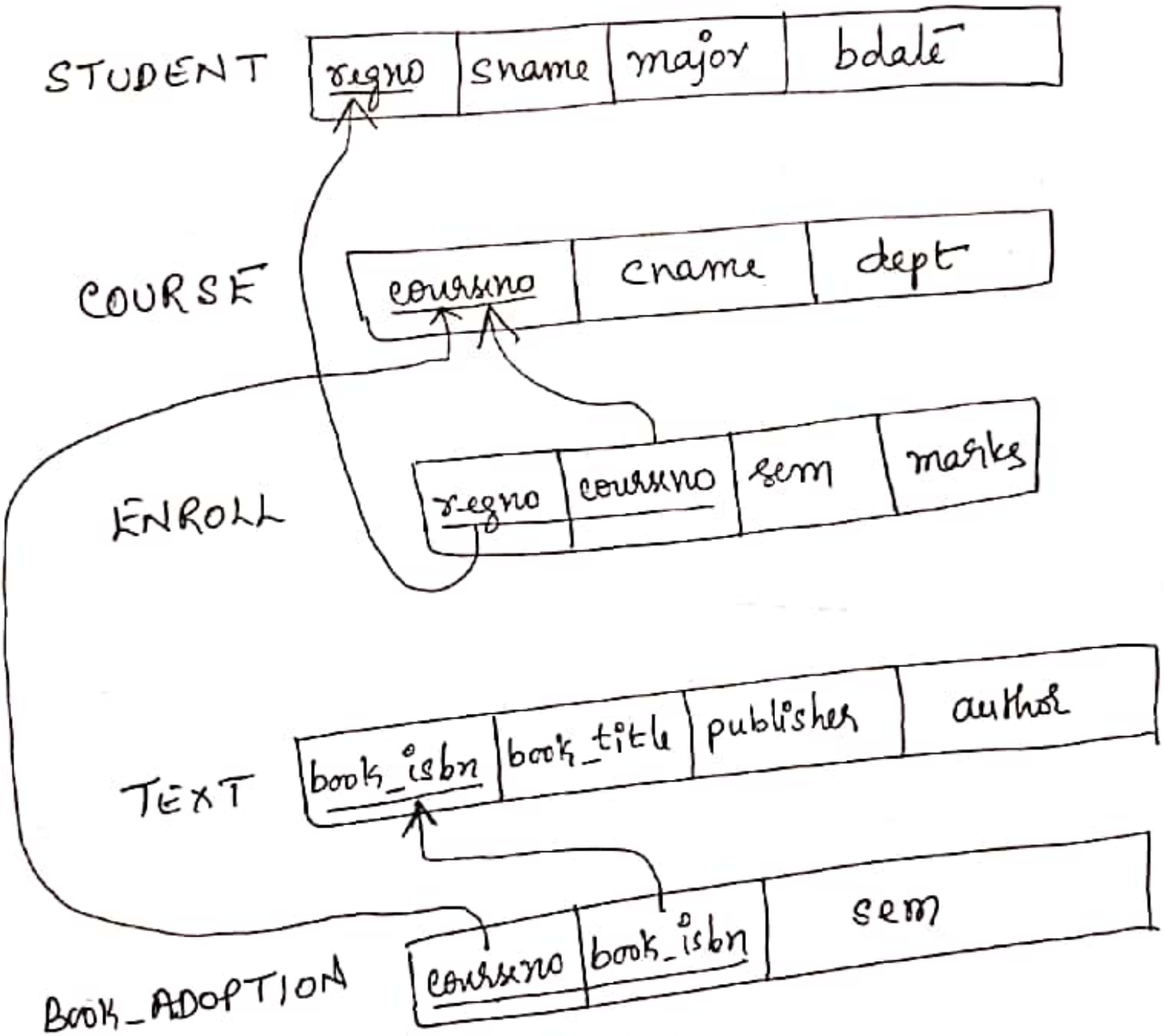
BOOK _ ADOPTION (course#: int, sem: int, book-ISBN: int)

TEXT (book-ISBN: int, book-title: string, publisher: string, author: string)

1. Produce a list of text books (include Course #, Book-ISBN,Book-title) in the alphabetical order for courses offered by th 'CS' department that use more than two books.
2. List any department that has all its adopted books published by a specific publisher
3. List the bookISBNs and book titles of the department that has maximum number of students



schema.



```
create table STUDENT (
                        regno varchar(10),
                        sname varchar(15),
                        major char (20),
                        bdate date,
                        primary key(regno)
                    )

insert into STUDENT values ('111','ravi','academic','1989-11-09')
insert into STUDENT values ('112','sudha','academic','1979-07-04')
insert into STUDENT values ('113','kumar','academic','1979-01-06')
insert into STUDENT values ('114','raju','academic','1999-10-02')
insert into STUDENT values ('115','hemanth','academic','1988-11-04')

create table COURSE (
                        courseno int,
                        cname varchar(25),
                        dept  varchar (20),
                        primary key(course)
                    )


insert into COURSE values (1,'DBMS','CS')
insert into COURSE values (2,'COMPILER','CS')
insert into COURSE values (3,'JAVA','CS')
insert into COURSE values (4,'SIG PROCESSING','ENC')
insert into COURSE values (5,'DIGTAL CIRCUITS','ENC')
insert into COURSE values (6,'MACHINE DESIGN','MECH')
insert into COURSE values (7,'THEMODYNAICS','MECH')
insert into COURSE values (8,'AUTOCAD','MECH')
```

```
insert into TEXTBOOK  values (204,'JAVA complete Reference', 'McGraw', 'BALAGURU')
insert into TEXTBOOK  values (205,'Singals and Fundumentals', 'McGraw', 'NITHIN')
insert into TEXTBOOK  values (206,'Machine Theory','McGraw', 'Ragavan')
insert into TEXTBOOK  values (208,'Circuit design','McGraw', 'Rajkamal')
insert into TEXTBOOK  values (207,'Thermodynamics','McGraw', 'Alfred')
insert into TEXTBOOK  values (209,'Electronic Circuits', 'McGraw', 'Alfred')
insert into TEXTBOOK  values (210,'Circuits Theory', 'McGraw', 'Alfred')


create table BOOK_ADOPTION (
                        courseno int,
                        sem int,
                        bookISBN int,
                        primary key (course, sem, bookISBN),
                        foreign key(courseno) references COURSE (courseno) on delete cascade on
update cascade,
                        foreign key(bookISBN) references TEXTBOOK (bookISBN) on delete cascade
on update cascade,
                )

insert into BOOK_ADOPTION values (1,5,201)
insert into BOOK_ADOPTION values (1,7,202)
insert into BOOK_ADOPTION values (2,5,203)
insert into BOOK_ADOPTION values (2,6,203)
insert into BOOK_ADOPTION values (3,7,204)
insert into BOOK_ADOPTION values (4,3,205)
insert into BOOK_ADOPTION values (4,5,209)
insert into BOOK_ADOPTION values (5,5,205)
insert into BOOK_ADOPTION values (5,6,208)
insert into BOOK_ADOPTION values (5,2,210)
insert into BOOK_ADOPTION values (6,7,206)
insert into BOOK_ADOPTION values (7,3,207)
```

insert into BOOK_ADOPTION values (7,3,206)

insert into BOOK_ADOPTION values (8,3,207)

1. Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.

```
SELECT DISTINCT(T.bookISBN), T.booktitle, C.courseno, C.cname FROM
Textbook T, Course C, Book_ADOPTION B

WHERE B.bookISBN=T.bookISBN AND C.courseno=B.courseno AND
C.dept='CSE' AND C.courseno IN (

SELECT courseno FROM Book_ADOPTION GROUP BY courseno HAVING
COUNT(DISTINCT(bookISBN))>2

)

ORDER BY T.booktitle
```

2. List any department that has all its adopted books published by a specific publisher

```
select distinct(C.dept) from COURSE C  where

NOT EXISTS

(

(select bookISBN from BOOK_ADOPTION where courseno in

            (select courseno from COURSE where dept = C.dept) )

      EXCEPT

       (select bookISBN from TEXTBOOK where publisher='McGraw')

)
```

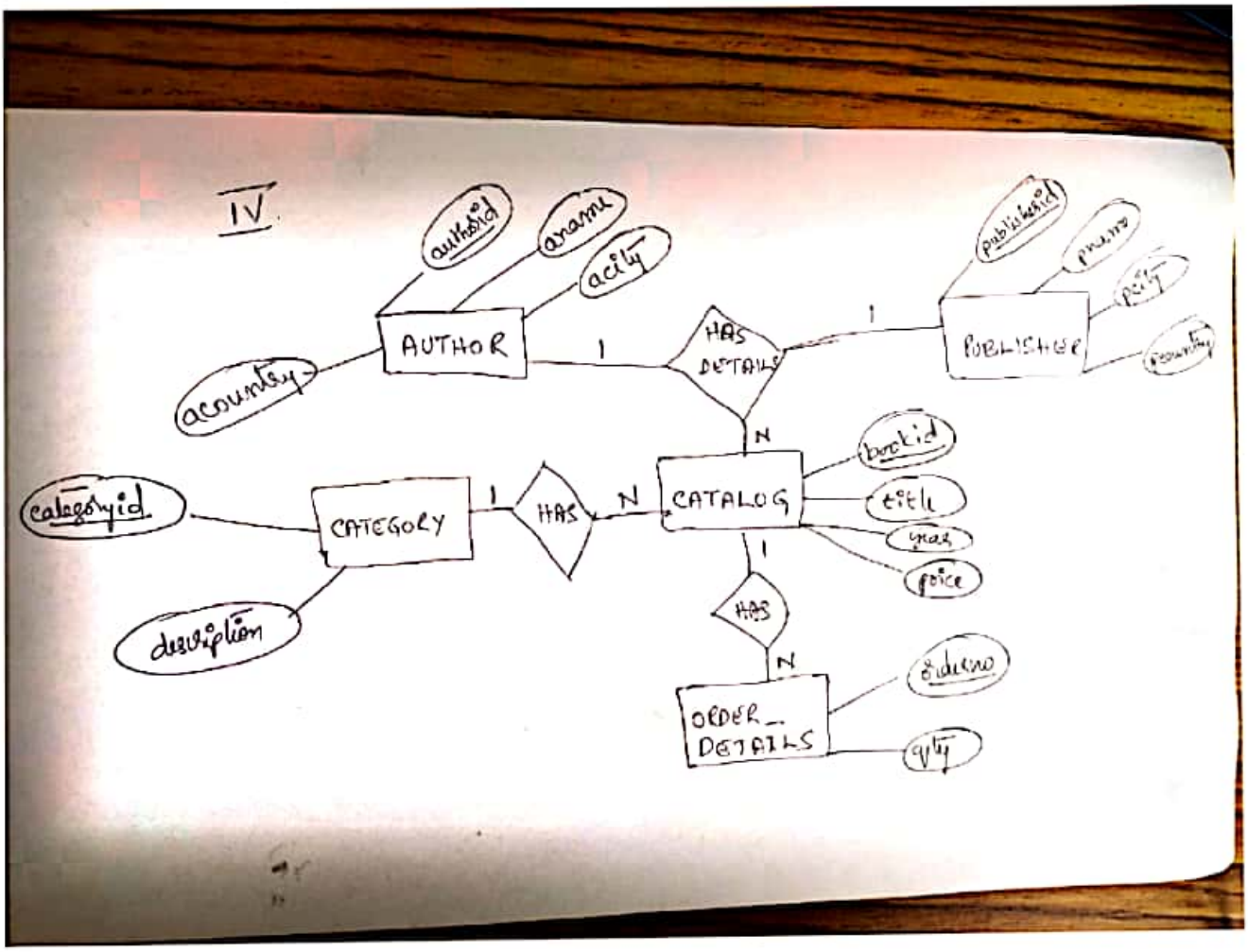
3. List the bookISBNs and book titles of the department that has maximum number of students

```
select  distinct T.bookISBN, T.title  from TEXTBOOK T, BOOK_ADOPTION BA, COURSE C where
T.bookISBN = BA.bookISBN and C.courseno = BA.courseno  and  C.dept in (select C.dept from
COURSE C, ENROLL E where C.courseno = E.courseno group by C.dept having count(*) >= all (select
count(*) from COURSE C, ENROLL E where C.courseno = E.courseno group by C.dept))
```

IV.

The following tables are maintained by a book 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. Find the author of the book which has maximum sales.
2. Increase the price of the books published by a specific publisher by 10%
3. Find the number of orders for the book that has minimum sales.



IV.

Relational schema:

AUTHOR:

authorid	aname	acity	acountry

PUBLISHER:

publisherid	pname	pcity	pcountry

CATEGORY:

categoryid	description

CATALOG:

bookid	title	authorid	publisherid	kategoryid	year	pric

ORDER_DETAILS:

order_no	bookid	qty

```
create table AUTHOR
        (
                authorid int primary key,
                aname  varchar(20),
                acity    varchar(20),
                acountry varchar(20),
                PRIMARY KEY (authorid)
        )


insert into AUTHOR values (110,'Elmasri','Houston','Canada')
insert into AUTHOR values (111,'sebesta','mangalore','India')
insert into AUTHOR values (112,'Elmasri','Houston','Canada')
insert into AUTHOR values (113,'Bharath K','Bangalore','India')
insert into AUTHOR values (114,'Willy Z','California','USA')
insert into AUTHOR values (115,'Salma','Dakha','Bangladesh')


create table PUBLISHER
        (
                publisherid int,
                pname varchar (20),
                pcity varchar (20),
                pcountry varchar (20),
                PRIMARY KEY (publisherid)
        )

insert into PUBLISHER values(201,'McGRAW','mangalore','India')
insert into PUBLISHER values(202,'Pearson','Bangalore','India')
insert into PUBLISHER values(203,'GKP','Bangalore','India')
insert into PUBLISHER values(204,'MediTech','Delhi','India')
insert into PUBLISHER values(205,'Sun','Ahmadbad','India')


create table CATEGORY
        (
        categoryid int primary key,
        description varchar (30),
        )

insert into CATEGORY values(1,'All children Books')
insert into CATEGORY values(2,'Cooking Books')
insert into CATEGORY values(3,'Popular Novels')
insert into CATEGORY values(4,'Small Story Books')
insert into CATEGORY values(5,'Medical Books')
```

```
create table CATALOGUE
		(
			bookid int primary key,
			title varchar (20),
			publisherid int,
			authorid int,
			categoryid int,
			cyear int,
			price int,
			foreign key(publisherid) references PUBLISHER (publisherid) on
delete cascade on update cascade,
			foreign key(authorid) references AUTHOR (authorid) on delete
cascade on update cascade,
			foreign key(categoryid) references CATEGORY (categoryid) on delete
cascade on update cascade
		)

insert into CATALOGUE values(301,'Panchatantra',201,111,1,2000,300)
insert into CATALOGUE values(302,'Vegetables',202,111,2,2000,400)
insert into CATALOGUE values(303,'Yogasana',203,112,5,2002,600)
insert into CATALOGUE values (304,'Stories of Village',204,113,4,2005,100)
insert into CATALOGUE values(305,'Triangle',205,114,3,2008,1000)
insert into CATALOGUE values (306,'Naughtiest Girl',201,110,3,2007,1500)
insert into CATALOGUE values(307,'Cookery',205,115,2,2006,100)


create table ORDER_DET
		(
			ordno int ,
			bookid int,
			qty int,
			primary key (ordno,bookid),
			foreign key(bookid) references CATALOGUE (bookid) on delete
cascade on update cascade,
		)


insert into ORDER_DET values(1,301,10)
insert into ORDER_DET values(1,302,6)
insert into ORDER_DET values(1,307,23)

insert into ORDER_DET values(2,301,15)
insert into ORDER_DET values(2,304,11)

insert into ORDER_DET values(3,304,15)

insert into ORDER_DET values(4,301,3)
insert into ORDER_DET values(4,305,8)

insert into ORDER_DET values(5,303,20)
```

insert into ORDER_DET values(5,306,6)
insert into ORDER_DET values(5,305,7)

1. Find the author of the book which has maximum sales.

select A.authorid, A.aname, A.acity, sum(O. qty) as QTY_SUM from Author A, CATALOGUE C, Order_Det O where A.authorid = C.authorid and C.bookid = O.bookid group by A.authorid, A.aname, A.acity, C.bookid having sum(O.qty) >= all (select sum(qty) from Order_Det group by bookid)

2. Increase the price of the books published by a specific publisher by 10%

update CATALOGUE set price = price * 1.1 where publisherid in (select publisherid from publisher where pname ='Pearson')

3. Find the number of orders for the book that has minimum sales.

```
SELECT COUNT(ordno) AS 'No. of orders',bookid
FROM ORDER_DET
GROUP BY bookid
HAVING SUM(qty) <=ALL (SELECT SUM(qty)FROM ORDER_DET GROUP
BY bookid
)
```

V. 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)

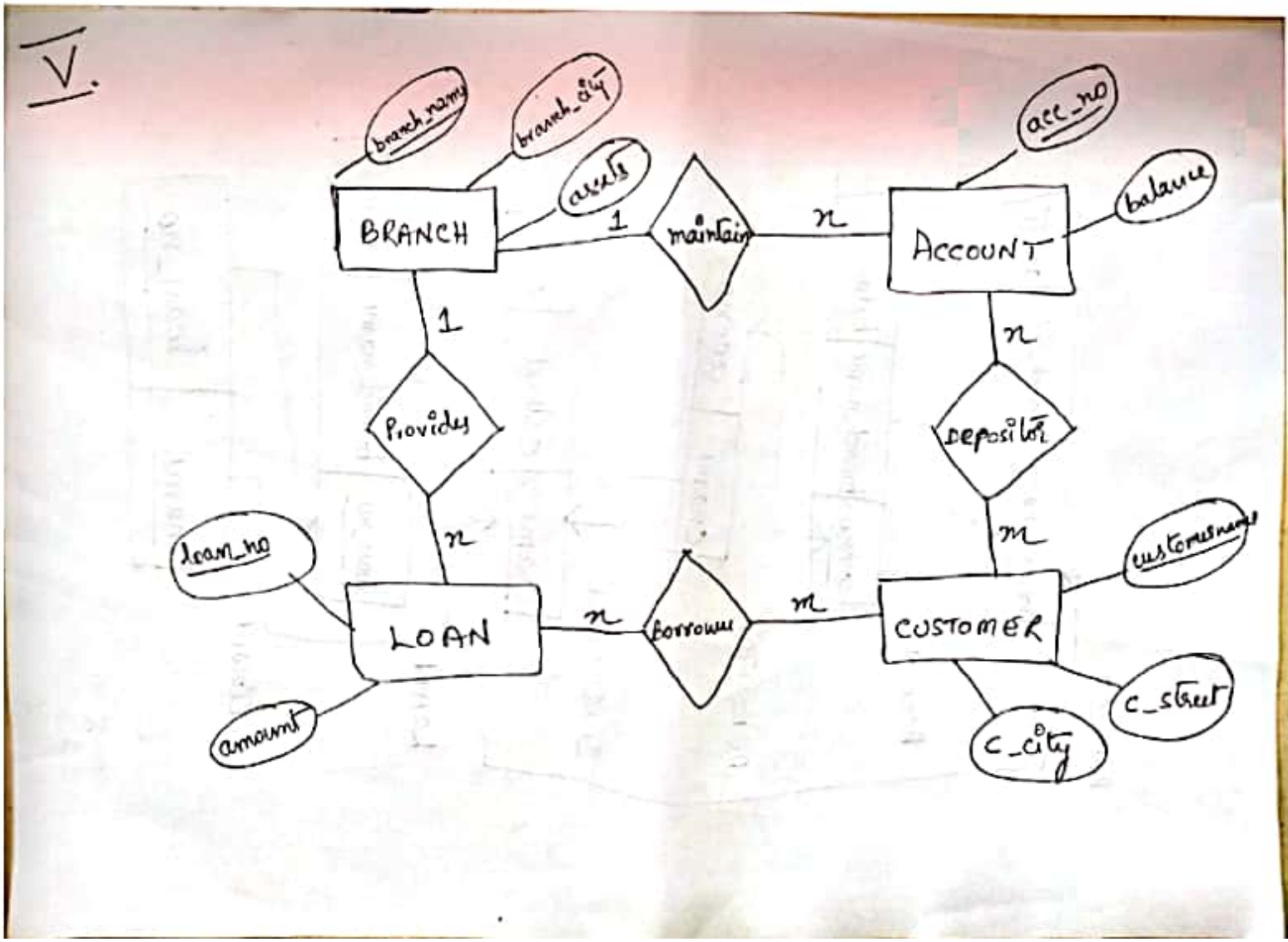
DEPOSITOR (customer-name: string, accno: int)

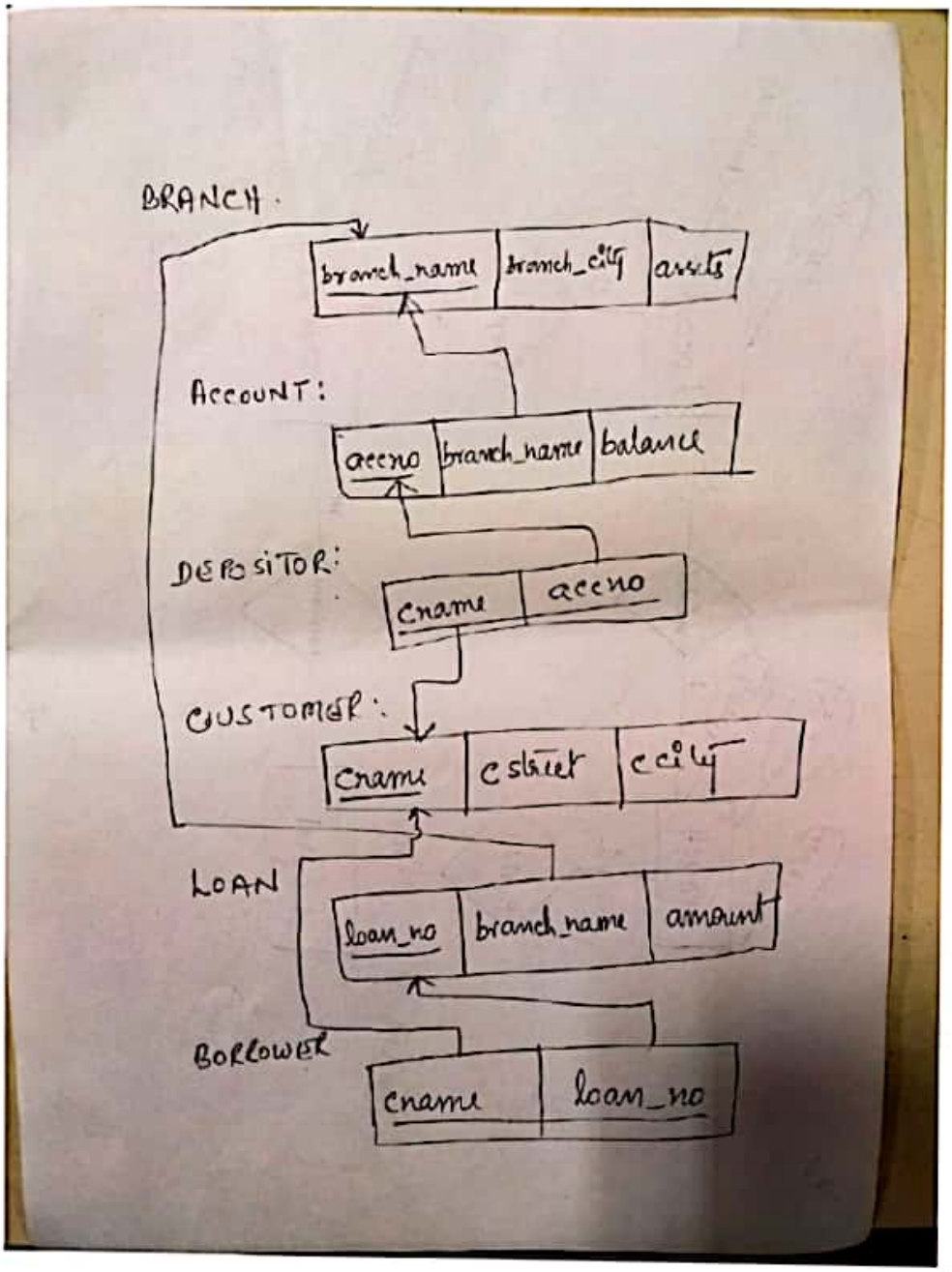
CUSTOMER (customer-name: string, customer-street: string, customer-city: string)

LOAN (loan-number: int, branch-name: string, amount: real)

BORROWER (customer-name: string, loan-number: int)

1. Find all the customers who have at least 2 accounts at all the branches located in a specific city.
2. Find all the customers who have accounts in at least 1 branch located in all the cities
3. Find all the customers who have accounts in at least 2 branches located in a specific city.






```
create table BRANCH(
                bname varchar(15)primary key,
                bcity varchar(15),
                assets real
              )

insert into BRANCH values('synd_nitte','karkala',200000)
insert into BRANCH values('Corp_nitte','karkala',300000)
insert into BRANCH values('PNB_nitte','karkala',100000)
insert into BRANCH values('Corp_mang','Mangalore',300000)
insert into BRANCH values('PNB_mang','Mangalore',500000)
insert into BRANCH values('state_udupi','Udupi',500000)
insert into BRANCH values('synd_udupi','Udupi',500000)
```

```
create table ACCOUNT(
                accno int,
                bname varchar(15),
                balance real,
                primary key(accno),
                foreign key(bname) references BRANCH(bname) on
delete cascade on update cascade
               )


insert into ACCOUNT values(12345,'synd_nitte',6000)
insert into ACCOUNT values(12340,'synd_nitte',6000)
insert into ACCOUNT values(21345,'synd_nitte',10000)
insert into ACCOUNT values(14341,'Corp_nitte',15000)
insert into ACCOUNT values(14345,'Corp_nitte',15000)
insert into ACCOUNT values(12455,'Corp_nitte',17000)
insert into ACCOUNT values(13345,'PNB_nitte',11000)
insert into ACCOUNT values(13346,'PNB_nitte',11000)
insert into ACCOUNT values(13347,'PNB_nitte',11000)
insert into ACCOUNT values(13340,'PNB_nitte',11000)
insert into ACCOUNT values(15345,'synd_udupi',11000)
insert into ACCOUNT values(12453,'PNB_mang',17000)
insert into ACCOUNT values(21346,'PNB_mang',10000)
insert into ACCOUNT values(12450,'PNB_mang',17000)
insert into ACCOUNT values(12452,'PNB_mang',17000)
insert into ACCOUNT values(13245,'state_udupi',5000)
insert into ACCOUNT values(13241,'state_udupi',5000)
insert into ACCOUNT values(12375,'state_udupi',12000)
insert into ACCOUNT values(12377,'state_udupi',12000)
insert into ACCOUNT values(12378,'state_udupi',12000)
insert into ACCOUNT values(15342,'state_udupi',19000)
insert into ACCOUNT values(12451,'state_udupi',17000)
```

```
create table CUSTOMER(
                cname varchar(20)primary key,
                cstreet varchar(25),
                ccity varchar(20)
               )


insert into CUSTOMER values('Rakesh','3rd main','karkala')
insert into CUSTOMER values('Ramesh','4th main','karkala')
insert into CUSTOMER values('Rajesh','4th block','mangalore')
insert into CUSTOMER values('Kareem','456 nagar','mangalore')
insert into CUSTOMER values('John smith','452 street','Udupi')
```

```
create table DEPOSITOR(
                cname varchar(20),
                accno int,
                primary key(cname,accno),
                foreign key(cname) references CUSTOMER(cname) on
delete cascade on update cascade,
                foreign key(accno) references ACCOUNT(accno) on
delete cascade on update cascade,
                unique(accno)
                 )

insert into DEPOSITOR values('Rakesh',12340)
insert into DEPOSITOR values('Rakesh',13345)
insert into DEPOSITOR values('Rakesh',14345)
insert into DEPOSITOR values('Rakesh',13346)
insert into DEPOSITOR values('Rakesh',15342)
insert into DEPOSITOR values('Rakesh',14341)

insert into DEPOSITOR values('Ramesh',12345)
insert into DEPOSITOR values('Ramesh',12375)
insert into DEPOSITOR values('Ramesh',12377)
insert into DEPOSITOR values('Ramesh',12378)
insert into DEPOSITOR values('Ramesh',12450)
insert into DEPOSITOR values('Ramesh',13340)
insert into DEPOSITOR values('Ramesh',12451)
insert into DEPOSITOR values('Ramesh',12452)
insert into DEPOSITOR values('Ramesh',12455)

insert into DEPOSITOR values('Kareem',21346)
insert into DEPOSITOR values('Kareem',13245)

insert into DEPOSITOR values('Rajesh',15345)
insert into DEPOSITOR values('Rajesh',13241)

insert into DEPOSITOR values('John smith',21345)
insert into DEPOSITOR values('John smith',12453)
insert into DEPOSITOR values('John smith',13347)
```

```
create table LOAN(
                loanno int,
                bname varchar(15),
                amount real,
                primary key(loanno),
                foreign key(bname) references BRANCH(bname) on
delete cascade on update cascade)


insert into LOAN values(1,'Corp_mang',12000)
insert into LOAN values(2,'Corp_mang',11000)
```

```
insert into LOAN values(3,'Corp_mang',10000)

insert into LOAN values(4,'Corp_nitte',16000)
insert into LOAN values(5,'Corp_nitte',13000)

insert into LOAN values(6,'PNB_mang',12000)
insert into LOAN values(11,'Corp_mang',10000)

insert into LOAN values(7,'state_udupi',20000)
insert into LOAN values(8,'state_udupi',23000)
insert into LOAN values(12,'synd_nitte',10000)
insert into LOAN values(9,'synd_nitte',32000)
insert into LOAN values(10,'PNB_nitte',12000)
insert into LOAN values(13,'state_udupi',12000)
insert into LOAN values(14,'synd_udupi',12000)

create table BORROWER(
                cname varchar(20),
                loanno int
                primary key(cname,loanno),
                foreign key(cname) references CUSTOMER(cname) on
delete cascade on update cascade,
                foreign key(loanno) references LOAN(loanno) on
delete cascade on update cascade,
                unique(loanno)
                )

insert into BORROWER values('John smith',1)
insert into BORROWER values('John smith',2)
insert into BORROWER values('John smith',3)
insert into BORROWER values('John smith',13)
insert into BORROWER values('John smith',14)
insert into BORROWER values('Kareem',4)
insert into BORROWER values('Kareem',5)
insert into BORROWER values('Rajesh',6)
insert into BORROWER values('Rajesh',11)
insert into BORROWER values('Rajesh',12)
insert into BORROWER values('Rajesh',7)
insert into BORROWER values('Rajesh',8)
insert into BORROWER values('Rakesh',9)
insert into BORROWER values('Ramesh',10)
```

1. Find all the customers who have at least 2 accounts at all the branches located in a specific city.

```
select C.cname from CUSTOMER  C where NOT EXISTS
(
        (select B.bname from  BRANCH B where B.bcity  = 'karkala')
        EXCEPT
        (select A.bname from ACCOUNT A, DEPOSITOR D, BRANCH B1 where D.accno =
        A.accno  and A.bname = B1.bname  and D.cname = C.cname  group by A.bname having
        count(*) >= 2)
)
```

2. Find all the customers who have accounts in at least 1 branch located in all the cities.

```
select C.cname from CUSTOMER C where NOT EXISTS
(
        (select distinct(B.bcity) from  BRANCH B)
        EXCEPT
        (select B1.bcity from BRANCH B1, ACCOUNT A ,DEPOSITOR D where A.bname =
        B1.bname and D.accno = A.accno and D.cname = C.cname   group by B1.bcity having
        count(*) >=1)
)
```

3. Find all the customers who have accounts in at least 2 branches located in a specific city.

```
select C.cname from CUSTOMER  C where  EXISIS
(
        select  count(distinct B.bname) from BRANCH B, ACCOUNT A, DEPOSITOR D where
        A.bname = B.bname and D.accno = A.accno and B.bcity  = 'karkala' and D.cname =
        C.cname   group by B.bcity having count(*) >=2
)
```

OR

```
select C.cname from CUSTOMER  C, BRANCH B, ACCOUNT A, DEPOSITOR D where
A.bname = B.bname and D.accno = A.accno and B.bcity  = 'karkala' and D.cname = C.cname
group by B.bcity, C.cname having count(distinct B.bname) >=2
```