DBMS

LAB ASSIGNMENT

1) True or False: if a query language is relationally complete, we are able to express every desired query in that language. Explain your answer.

2)**Assignment:**

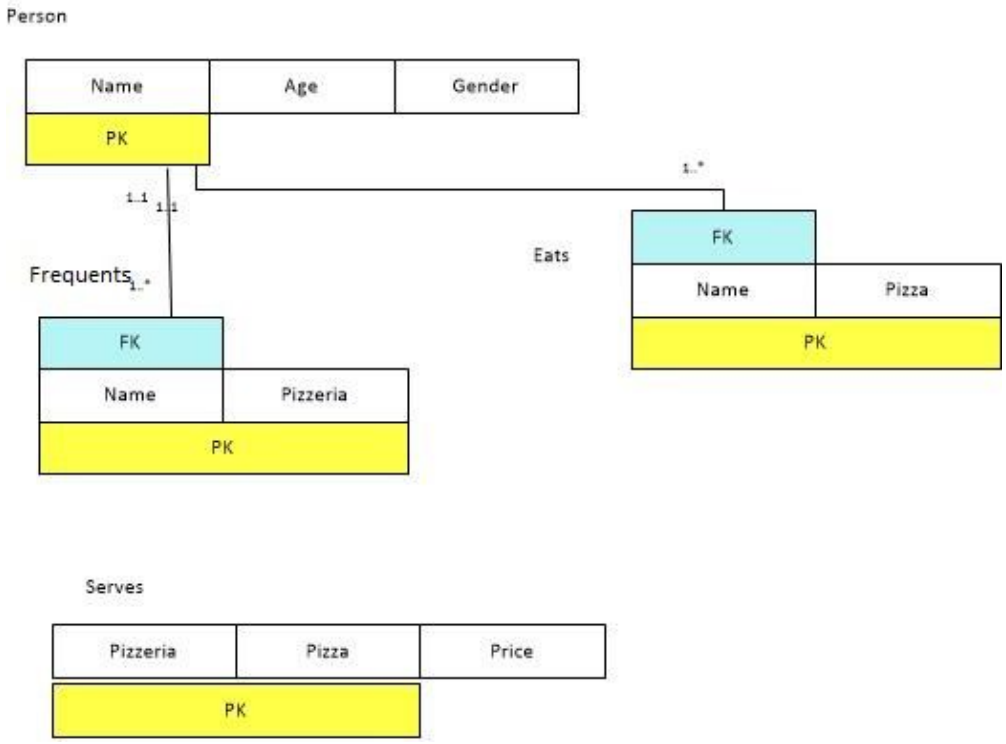
Create the pizza database by running the pizza.sql file.

Steps: 1) create database with name ‘pizza’ -> CREATE database pizza;

2) mysql -u root -p pizza < your sql file path

The file contains the table definitions and sample data.

Table	Attributes	Key
Person	name, age, gender	name
Frequents	name, pizzeria	(name, pizzeria)
Eats	name, pizza	(name, pizza)
Serves	pizzeria, pizza, price	(pizzeria, pizza)



For each of the following problems, write the Relational Algebra statement to solve the problem.
After you've done that, translate it into SQL, then run the SQL code to check your work.

Problems:

1. List all the customers

2. List the name of all the pizzeria

3. List the name and age of all the people who eat sausage pizza

4. Find the names and gender of all people who eat mushroom pizza.

5.

a. Find all pizzerias frequented by at least one person under the age of 18

b. Find the names of all females who eat either mushroom or pepperoni pizza (or both)

c. Find the names of all females who eat both mushroom and pepperoni pizza

d. Find all pizzerias that serve at least one pizza that Amy eats for less than $10.00

e. Find all pizzerias that are frequented by only females or only males

f. For each person, find all pizzas the person eats that are not served by any pizzeria the person frequents. Return all such person (name) / pizza pairs

g. Find the names of all people who frequent only pizzerias serving at least one pizza they eat

h. Find the names of all people who frequent every pizzeria serving at least one pizza they eat

i. Find the pizzeria serving the cheapest pepperoni pizza. In the case of ties, return all of the cheapest-pepperoni pizzerias.

3)Consider a schema with two relations, *R*(A, B) and *S*(B, C), where all values are integers. Make no assumptions about keys. Consider the following three relational algebra expressions:

a. $\pi_{A,C}(R \bowtie \sigma_{B=1} S)$

b. $\pi_A(\sigma_{B=1} R) \times \pi_C(\sigma_{B=1} S)$

c. $\pi_{A,C}(\pi_A R \times \sigma_{B=1} S)$

Two of the three expressions are equivalent (i.e., produce the same answer on all databases), while one of them can produce a different answer. Which query can produce a different answer? Give the simplest database instance you can think of where a different answer is produced

4) Assume you have two relations R and S, where R contains N tuples and S contains M tuples, and N > M (i.e., R has more rows). For each expression below, give the minimum and maximum number of tuples possible for the resulting relation.

(a) $R \cup S$

(b) $R \cap S$

(c) $R - S$

(d) R/S

(e) $\sigma_{(x=10)}R$

(f) $R \times S$

5) Consider a relation *Temp*(regionID, name, high, low) that records historical high and low temperatures for various regions. Regions have names, but they are identified by regionID, which is a key. Consider the following query, which uses the linear notation introduced at the end of the relational algebra videos.

$$T1(rID, h) = \pi_{regionID,high} Temp$$

$$T2(rID, h) = \pi_{regionID,low} Temp$$

$$T3(regionID) = \pi_{rID}(T1 \bowtie_{h<high} Temp)$$

$$T4(regionID) = \pi_{rID}(T2 \bowtie_{l>low} Temp)$$

$$T5(regionID) = \pi_{regionID} Temp - T3$$

$$T6(regionID) = \pi_{regionID} Temp - T4$$

$$Result(n) = \pi_{name}(Temp \bowtie (T5 \cup T6))$$

State in English what is computed as the final Result. The answer can be articulated in a single phrase.

6) 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)

TO-DO:

a) Create the above tables by properly specifying the primary keys and foreign keys.

b) Enter at least five tuples for each relation.

c) 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.

d) List the orderno for orders that were shipped from all the warehouses that the company has in a specific city.

e) 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.

LAB SOLUTIONS

1) Relational completeness means that a query language can express all the queries that can be expressed in relational algebra. It does not mean that the language can express any desired query

2) SELECT name from frequents;
3) SELECT name,age from Eats where pizza=”sausage:;
4) SELECT name,gender from Person where name in (SELECT name from Eats where pizza=”mushroom”)

   (OR)

   SELECT name,gender from Person natural join Eats where pizza="mushroom";
5)

a. $\pi_{pizzeria}(\sigma_{age<18}(Person) \bowtie Frequents)$

b. $\pi_{name}\left(\sigma_{gender='\text{female}' \wedge (pizza='\text{mushroom}' \vee pizza='\text{pepperoni}')}(Person \bowtie Eats)\right)$

c. $\pi_{name}(\sigma_{gender='\text{female}' \wedge pizza='\text{mushroom}'}(Person \bowtie Eats)) \cap \pi_{name}(\sigma_{gender='\text{female}' \wedge pizza='\text{pepperoni}'}(Person \bowtie Eats))$

d. $\pi_{pizzeria}\left(\sigma_{name='\text{Amy}'}(Eats) \bowtie \sigma_{price<10}(Serves)\right)$

e. $\left(\begin{array}{l}\pi_{pizzeria}(\sigma_{gender='\text{female}'}(Person) \bowtie Frequents) \; - \\ \pi_{pizzeria}(\sigma_{gender='\text{male}'}(Person) \bowtie Frequents)\end{array}\right) \cup \left(\begin{array}{l}\pi_{pizzeria}(\sigma_{gender='\text{male}'}(Person) \bowtie Frequents) \; - \\ \pi_{pizzeria}(\sigma_{gender='\text{female}'}(Person) \bowtie Frequents)\end{array}\right)$

f. $Eats - \pi_{name,pizza}(Frequents \bowtie Serves)$

g. $\pi_{name}(Person) - \pi_{name}(Frequents - \pi_{name,pizzeria}(Eats \bowtie Serves))$

h. $\pi_{name}(Person) - \pi_{name}(\pi_{name,pizzeria}(Eats \bowtie Serves) - Frequents)$

i. $\pi_{pizzeria}(\sigma_{pizza='\text{pepperoni}'}, Serves) \; - \; \pi_{pizzeria}\left(\sigma_{price>price2}\left(\begin{array}{l}\pi_{pizzeria,price}(\sigma_{pizza='\text{pepperoni}'}, Serves) \\ \times \\ \rho_{pizzeria2,price2}\left(\pi_{pizzeria,price}(\sigma_{pizza='\text{pepperoni}'}, Serves)\right)\end{array}\right)\right)$

Query results for SQL data:

a Straw Hat, New York Pizza, Pizza Hut

b Amy, Fay

c. Amy

d Little Caesars, Straw Hat, New York Pizza

e Little Caesars, Chicago Pizza, New York Pizza

f. Amy: mushroom, Dan: mushroom, Gus: mushroom

g Amy, Ben, Dan, Eli, Fay, Gus, Hil

h Fay

i. Straw Hat, New York Pizza

3) Query (c) is different. Let R = {(3, 4)} and S = {(1, 2)}. Then query (a) and (b) produce an empty result while (c) produces {(3, 2)}.

4)

a) Max : N+M (union we add all the tuples from both relations) when R and S have no common tuple.
   Min : the minimum is N (the greatest of the two sizes, m and n). When all the tuples of R also exist in S.

b) Max : M both relation contains same keys i.e all M tuples are in relation R
   Min : 0 because the two relations may have no common **tuples**

c) Max: N when no tuple is common
   Min : N-M when all the tuples in S are present in R also

d) Max:N when M=0
   Min : 0 when none of the M tuples of S matches in relation R in the same order as in the relation S

e) Max: N when x=10 for all tuples in R
   Min : 0 when x!=10 for all tuples in R

f) Max: M*N
   Min: 0 When M=0

5) Names of regions with the highest high temperature and/or lowest low temperature

6)

a)

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));

b)

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');

c)

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;

d)

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);

e)

SQL> delete from item where itemno=3;