**SQL**

Name: NARAYANA N GOURAV

|  |
| --- |
| Suppose a movie\_studio has several film crews. The crews might be designated by a given studio crew1, crew 2, and so on. However, other studios might use the same designations for crews, so the attribute crew\_number is not a key for crews. Movie\_studio holds the information like name, branch and several locations. Each crew holds information like sector and strength.  • List all movie studios which are not used a single crews.  • Retrieve the movie studio which uses highest strength crew.  • Write a before insert trigger to check maximum number of crews to any studio is limited to 10. |

|  |
| --- |
| create database movie\_studio;  use movie\_studio;  create table movie (name varchar(10), branch varchar(10), constraint pki1 primary key(name));  create table crew (crew\_no int, name varchar(10), strength int, sector varchar(20), primary key(name, crew\_no), foreign key(name) references movie(name));  create table locations(name varchar(10), location varchar(20), primary key(name, location));  show tables; |
|  |
| insert into movie values('Anu' , 'banglore');  insert into movie values('Vidhya' , 'pune');  insert into movie values('Tasrin' , 'goa');  insert into movie values('Sahana' , 'Mysore');  select \* from movie; |
|  |
| insert into crew values(1,'Sahana',14,'thrill');  insert into crew values(2,'Vidhya',11,'sijd');  insert into crew values(3,'Sahana',6,'sakjs');  insert into crew values(4,'Anu',2,'sawe');  insert into crew values(5,'Tasrin',12,'ssesd');  insert into crew values(5,'Anu',4,'ssesd');  insert into crew values(2,'Sahana',2,'thrill');  insert into crew values(6,'Tasrin',11,'ssessd');  insert into crew values(7,'Vidhya',11,'ssessd');  select \* from crew; |
|  |
| insert into locations values('Vidhya' , 'pune');  insert into locations values('Vidhya' , 'goa');  insert into locations values('Sahana' , 'blore');  select \* from locations; |
|  |
| select name from movie where name not in (select name from crew); |
|  |
| select name from crew c where c.strength in(select max(strength) from crew); |
|  |
| select name from crew where strength >=all(select strength from crew); |
|  |
| delimiter //  create trigger max\_crews\_trigger  before insert on crew  for each row  begin  declare crew\_count int;  select count(\*) into crew\_count from crew where crew\_no = new.crew\_no;  if crew\_count >= 10 then  signal sqlstate '45000'  set message\_text = 'maximum number of crews to this studio is limited to 10.';  end if;  end; //  delimiter ; |
|  |

|  |
| --- |
| The production company is organized into different studios. We store each studio’s  name branch and location; every studio must own at least one movie. We store each movie’s  title, sensor\_number and year of production. Star may act in any number of movies and we  store each actors name and address.  • List all the studios of the movie “Kantara”;  • List all the actors , acted in a movie ‘Kantara’  • Write a deletion trigger, does not allow to deleting current year movies. |

|  |
| --- |
| create database prod\_comp;  use prod\_comp;  create table studio (studio\_id int not null auto\_increment, name varchar(255) not null, branch varchar(255) not null, location varchar(255) not null, primary key (studio\_id));  create table movie (movie\_id int not null auto\_increment, title varchar(255) not null, censor\_number int not null, year\_of\_production int not null, studio\_id int not null, primary key (movie\_id), foreign key (studio\_id) references studio(studio\_id));  create table actor (actor\_id int not null auto\_increment, name varchar(255) not null, address varchar(255) not null, primary key (actor\_id));  create table movie\_cast (movie\_id int not null, actor\_id int not null, primary key (movie\_id, actor\_id), foreign key (movie\_id) references movie(movie\_id), foreign key (actor\_id) references actor(actor\_id)); |
|  |
| insert into studio (name, branch, location) values ('vane', 'bangalore', 'bangalore'),('major', 'mangalore', 'mangalore'),('vouge', 'mahalakshmi', 'bangalore'); |
|  |
| insert into movie (title, censor\_number, year\_of\_production, studio\_id) values ('kantara', 18, 2022, 1), ('the end', 16, 2021, 2), ('start again', 12, 2020, 1), ('rrr', 14, 2022, 2); |
|  |
| insert into actor (name, address) values ('rishab shetty', 'bangalore'), ('ravi verma', 'mangalore'), ('tillu', 'bangalore'); |
|  |
| insert into movie\_cast (movie\_id, actor\_id) values (1, 1), (1, 2), (2, 1), (3, 2), (4, 1), (4, 3); |
|  |
| select s.name from studio s inner join movie m on s.studio\_id = m.studio\_id where m.title = 'kantara'; |
|  |
| select a.name from actor a inner join movie\_cast mc on a.actor\_id = mc.actor\_id inner join movie m on mc.movie\_id = m.movie\_id where m.title = 'kantara'; |
|  |
| delimiter //  create trigger prevent\_delete\_current\_year\_movies  before delete on movie  for each row  begin  if old.year\_of\_production = year(curdate()) then  signal sqlstate '45000'  set message\_text = 'deletion not allowed for current year movies.';  end if;  end;//  delimiter ; |
|  |

|  |
| --- |
| The production company is organized into different studios. We store each studio’s name branch and location; a studio own any number of Cartoon-serials. We store each Cartoon-Serial’s title, sensor\_number and year of production. Star may do voices in any number of Cartoon-Serials and we store each actors name and address.  • Find total no of actors, do voiced in a Cartoon-Serials ‘Tom and Jerry’  • Retrieve name of studio, location and Cartoon-Serials title in which star “Richard Kind” is voiced.  • Write a deletion trigger, does not allow to deleting current year Cartoon-Serials. |

|  |
| --- |
| create database prod\_compy;  use prod\_compy;  create table studio (studio\_id int primary key, name varchar(50) not null, branch varchar(50) not null, location varchar(50) not null);  create table cartoon\_serial (serial\_id int primary key, studio\_id int not null, title varchar(50) not null, censor\_number varchar(10) not null, year\_of\_production int not null, foreign key (studio\_id) references studio(studio\_id));  create table actor (actor\_id int primary key,name varchar(50) not null,address varchar(100) not null);  create table voice\_actor (voice\_actor\_id int primary key,actor\_id int not null,serial\_id int not null,foreign key (actor\_id) references actor(actor\_id),foreign key (serial\_id) references cartoon\_serial(serial\_id)); |
|  |
| insert into studio(studio\_id, name, branch, location) values (1, 'warner bros.', 'hollywood', 'california'), (2, 'disney', 'burbank', 'california'), (3, 'nickelodeon', 'new york', 'new york'); |
|  |
| insert into cartoon\_serial(serial\_id, studio\_id, title, censor\_number, year\_of\_production) values (1, 1, 'tom and jerry', 'pg', 1940), (2, 1, 'looney tunes', 'pg', 1930), (3, 2, 'mickey mouse', 'g', 1928), (4, 3, 'spongebob squarepants', 'tv-y7', 1999); |
|  |
| insert into actor(actor\_id, name, address) values (1, 'tom hanks', 'los angeles'), (2, 'tim allen', 'santa monica'), (3, 'bill murray', 'new york'), (4, 'dan castellaneta', 'chicago'), (5, 'nancy cartwright', 'dayton'); |
|  |
| insert into voice\_actor(voice\_actor\_id, actor\_id, serial\_id) values (1, 1, 1), (2, 2, 1), (3, 3, 2), (4, 4, 2), (5, 5, 4), (6, 4, 4); |
|  |
| select count(\*) as total\_actors from actor as a join voice\_actor as vc on a.actor\_id = vc.actor\_id join cartoon\_serial as cs on vc.serial\_id = cs.serial\_id where cs.title = 'tom and jerry'; |
|  |
| select s.name as studio\_name, s.location as studio\_location, cs.title as cartoon\_title from studio s join cartoon\_serial cs on s.studio\_id = cs.studio\_id join voice\_actor va on cs.serial\_id = va.serial\_id join actor a on va.actor\_id = a.actor\_id where a.name = 'richard kind'; |
|  |
| delimiter //  create trigger prevent\_deletion\_current\_year  before delete on cartoon\_serial  for each row  begin  if old.year\_of\_production = year(curdate()) then  signal sqlstate '45000'  set message\_text = 'cannot delete cartoon-serials from the current year';  end if;  end;//  delimiter ; |
|  |

|  |
| --- |
| Car marketing company wants keep track of marketed cars and their owner. Each car must be associated with a single owner and owner may have any number of cars. We store car’s registration number, model & color and owner’s name, address & SSN. We also store date of purchase of each car. • Find a person who owns highest number of cars  • Retrieve persons and cars information purchased on the day 03-03-2023  • Write a insertion trigger to check date of purchase must be less than current date (must use system date) |

|  |
| --- |
| create database car\_mark;  use car\_mark;  create table owners (  id int primary key auto\_increment,  name varchar(50) not null,  address varchar(100) not null,  ssn varchar(20) not null  );  create table cars (  registration\_number varchar(20) primary key,  model varchar(50) not null,  color varchar(20) not null,  owner\_id int not null,  purchase\_date date not null,  foreign key (owner\_id) references owners(id)  ); |
|  |
| insert into owners (name, address, ssn)  values ('john doe', '123 main st, anytown, usa', '123-45-6789'),  ('jane smith', '456 oak st, othertown, usa', '987-65-4321'),  ('bob johnson', '789 elm st, thirdtown, usa', '555-55-5555'); |
|  |
| insert into cars (registration\_number, model, color, owner\_id, purchase\_date)  values ('abc123', 'honda civic', 'blue', 1, '2022-02-01'),  ('def456', 'toyota camry', 'red', 2, '2022-03-15'),  ('ghi789', 'ford mustang', 'black', 3, '2022-02-28'),  ('jkl012', 'chevrolet corvette', 'yellow', 1, '2023-01-01'),  ('mno345', 'tesla model s', 'white', 2, '2023-03-03'); |
|  |
| select o.name, count(\*) as car\_count  from cars c  join owners o on c.owner\_id = o.id  group by o.id  order by car\_count desc  limit 1; |
|  |
| select c.registration\_number, c.model, c.color, o.name, o.address, o.ssn  from cars c  join owners o on c.owner\_id = o.id  where c.purchase\_date = '2023-03-03'; |
|  |
| delimiter //  create trigger check\_purchase\_date  before insert on cars  for each row  begin  if new.purchase\_date > curdate() then  signal sqlstate '45000' set message\_text = 'purchase date must be less than or equal to current date';  end if;  end//  delimiter ; |
|  |

|  |
| --- |
| Puppy pet shop wants to keep track of dogs and their owners. The person can buy maximum three pet dogs. We store person’s name, SSN and address and dog’s name, date of purchase and sex. The owner of the pet dogs will be identified by SSN since the dog’s names are not distinct.  • List all pets owned by a person ‘Ramesh’.  • List all persons who are not owned a single pet  • Write a trigger to check the constraint that the person can buy maximum three pet dogs |

|  |
| --- |
| create database pet\_shop;  use pet\_shop;  create table person (  ssn varchar(9) primary key,  name varchar(50),  address varchar(100)  );  create table petdog (  name varchar(50),  purchase\_date date,  sex char(1),  owner\_ssn varchar(9),  foreign key (owner\_ssn) references person(ssn)  ); |
|  |
| insert into person (ssn, name, address)  values  ('111111111', 'alice', '123 main st'),  ('222222222', 'bob', '456 oak ave'),  ('333333333', 'charlie', '789 maple st'); |
|  |
| insert into petdog (name, purchase\_date, sex, owner\_ssn)  values  ('rufus', '2022-01-01', 'm', '111111111'),  ('fido', '2022-02-01', 'm', '111111111'),  ('lassie', '2022-03-01', 'f', '111111111'),  ('buddy', '2022-04-01', 'm', '222222222'),  ('daisy', '2022-05-01', 'f', '222222222'); |
|  |
| select name, purchase\_date, sex  from petdog  where owner\_ssn = (select ssn from person where name = 'ramesh'); |
|  |
| select name, ssn, address  from person  where ssn not in (select owner\_ssn from petdog); |
|  |
| delimiter //  create trigger max\_three\_pets  before insert on petdog  for each row  begin  declare pet\_count int;  select count(\*) into pet\_count from petdog where owner\_ssn = new.owner\_ssn;  if pet\_count >= 3 then  signal sqlstate '45000'  set message\_text = 'maximum of three pets allowed per person';  end if;  end//  delimiter; |
|  |

|  |
| --- |
| Education institute is managing the online course enrollment system. Students can enroll maximum of six courses of their choice and a maximum student to be enrolled to any course is 60. We store student details like name, USN, semester and several addresses, course details like unique title, unique id and credits.  • Find number of students enrolled for the course ‘ADBMS’.  • Retrieve student names that are enrolled for AI course but not enrolled for IOT.  • Write a trigger to establish the constraint that the students can enroll maximum of six courses of their choice. |

|  |
| --- |
| create database enroll\_sys;  use enroll\_sys;  create table students (  usn varchar(10) primary key,  name varchar(50) not null,  semester int not null,  address varchar(100) not null  );  create table courses (  course\_id int primary key,  course\_title varchar(50) not null,  credits int not null  );  create table student\_courses (  usn varchar(10) not null,  course\_id int not null,  foreign key (usn) references students (usn),  foreign key (course\_id) references courses (course\_id),  primary key (usn, course\_id)  ); |
|  |
| insert into students (usn, name, semester, address) values  ('1bm17cs001', 'alice', 5, '123 main st'),  ('1bm17cs002', 'bob', 5, '456 oak ave'),  ('1bm17cs003', 'charlie', 5, '789 maple blvd'); |
|  |
| insert into courses (course\_id, course\_title, credits) values  (1, 'adbms', 4),  (2, 'ai', 3),  (3, 'iot', 3); |
|  |
| insert into student\_courses (usn, course\_id) values  ('1bm17cs001', 1),  ('1bm17cs001', 2),  ('1bm17cs001', 3),  ('1bm17cs002', 1),  ('1bm17cs002', 2),  ('1bm17cs003', 2),  ('1bm17cs003', 3); |
|  |
| select count(\*) from student\_courses sc  join courses c on sc.course\_id = c.course\_id  where c.course\_title = 'adbms'; |
|  |
| select s.name  from students s  join student\_courses sc1 on s.usn = sc1.usn  join courses c1 on sc1.course\_id = c1.course\_id and c1.course\_title = 'ai'  left join student\_courses sc2 on s.usn = sc2.usn  left join courses c2 on sc2.course\_id = c2.course\_id and c2.course\_title = 'iot'  where sc2.usn is null; |
|  |
| delimiter //  create trigger max\_courses  before insert on student\_courses  for each row  begin  declare enrolled int;  select count(\*) into enrolled from student\_courses where usn = new.usn;  if enrolled >= 6 then  signal sqlstate '45000'  set message\_text = 'maximum number of courses exceeded';  end if;  end//  delimiter ; |
|  |

|  |
| --- |
| The commercial bank wants keep track of the customer’s account information. The each customer may have any number of accounts and account can be shared by any number of customers. The system will keep track of the date of last transaction. We store the following details.  a) Account: unique account-number, type and balance  b) Customer: unique customer-id, name and several addresses composed of street, city and state  • Add 3% interest to the customer who have less than 1000 balances and 6% interest to remaining customers.  • List joint accounts involving more than three customers  • Write a insertion trigger to allow only current date for date of last transaction field. |

|  |
| --- |
| create database bank;  use bank;  create table account (  accountnumber int primary key,  accounttype varchar(50),  balance decimal(10,2),  lasttransactiondate date  );  create table customer (  customerid int primary key,  name varchar(100),  street varchar(100),  city varchar(50),  state varchar(50)  );  create table accountcustomer (  accountnumber int,  customerid int,  foreign key (accountnumber) references account(accountnumber),  foreign key (customerid) references customer(customerid),  primary key (accountnumber, customerid)  ); |
|  |
| insert into account (accountnumber, accounttype, balance, lasttransactiondate) values  (1001, 'savings', 5000.00, '2022-03-01'),  (1002, 'checking', 2500.00, '2022-03-15'),  (1003, 'savings', 10000.00, '2022-02-28'); |
|  |
| insert into customer (customerid, name, street, city, state) values  (1, 'john doe', '123 main st', 'anytown', 'ca'),  (2, 'jane smith', '456 park ave', 'somewhere', 'ny'),  (3, 'bob johnson', '789 broadway', 'nowhere', 'tx'); |
|  |
| insert into accountcustomer (accountnumber, customerid) values  (1001, 1),  (1001, 2),  (1002, 2),  (1003, 1),  (1003, 3); |
|  |
| update account  set balance = case  when balance < 1000 then balance \* 1.03  else balance \* 1.06  end; |
|  |
| select a.accountnumber, count(ac.customerid) as numcustomers  from account a  join accountcustomer ac on a.accountnumber = ac.accountnumber  group by a.accountnumber  having count(ac.customerid) > 3; |
|  |
| delimiter //  create trigger setlasttransactiondate  before insert on account  for each row  begin  if new.lasttransactiondate is null then  set new.lasttransactiondate = current\_date();  end if;  end//  delimiter; |
|  |

|  |
| --- |
| The commercial bank wants keep track of the customer’s loan information. The customer can take any number of loans from the bank and loan will not be shared. The system will also keep track of the date of last transaction. We store the following details.  a) Customer: unique customer-id, name, Annual Income and several addresses composed of street, city and state  b) Loan: unique loan-number, type and amount  • Add 12% interest to the customer who have less than 50000 amounts and 14% interest to remaining customers.  • Retrieve the customers who have a single loan in a bank.  • Write an insertion trigger to loan, that does not allow if the loan amount is more than two times of customer annual income. |

|  |
| --- |
| create database com\_bank;  use com\_bank;  create table customer (  customer\_id int primary key,  name varchar(50),  annual\_income decimal(10,2),  street varchar(100),  city varchar(50),  state varchar(50)  );  create table loan (  loan\_number int primary key,  loan\_type varchar(50),  amount decimal(10,2),  customer\_id int,  last\_transaction\_date date,  foreign key (customer\_id) references customer(customer\_id)  ); |
|  |
| insert into customer (customer\_id, name, annual\_income, street, city, state) values  (1, 'john doe', 50000.00, '123 main st', 'anytown', 'ny'),  (2, 'jane smith', 75000.00, '789 elm st', 'anotherplace', 'fl'),  (3, 'bob johnson', 60000.00, '321 pine st', 'nowhere', 'tx'); |
|  |
| insert into loan (loan\_number, loan\_type, amount, customer\_id, last\_transaction\_date) values  (1001, 'car loan', 15000.00, 1, '2022-03-01'),  (1002, 'home loan', 250000.00, 2, '2022-04-01'),  (1003, 'personal loan', 10000.00, 3, '2022-05-01'),  (1004, 'business loan', 50000.00, 1, '2022-06-01'); |
|  |
| update loan  set amount = amount \* (case when (select annual\_income from customer where customer.customer\_id = loan.customer\_id) < 50000 then 1.12 else 1.14 end); |
|  |
| select customer.\*  from customer  join (select customer\_id, count(\*) as loan\_count from loan group by customer\_id having loan\_count = 1) as singleloancustomers  on customer.customer\_id = singleloancustomers.customer\_id; |
|  |
| delimiter //  create or replace trigger loancheck  before insert on loan  for each row  begin  declare annual\_income decimal(10,2);  select annual\_income into annual\_income from customer where customer\_id = new.customer\_id;  if (new.amount > annual\_income \* 2) then  signal sqlstate '45000' set message\_text = 'loan amount cannot be more than two times the customer''s annual income.';  end if;  end//  delimiter ; |
|  |

|  |
| --- |
| The Sapna Book shop wants keep track of orders of the book. The book is composed of unique id, title, year of publication, single author and single publisher. Each order will be uniquely identified by order-id and may have any number of books. We keep track of quantity of each book ordered. We store the following details for author and publisher.  AUTHOR: unique author-id, name, city, country  PUBLISHER: unique publisher-id, name, city, country.  • Find the author who has published highest number of books  • List the books published by specific publisher during the year 2022.  • Write before insertion trigger to book to check year of publication should allow current year only |

|  |
| --- |
| create database bookstore;  use bookstore;  create table books (  book\_id int primary key,  title varchar(255) not null,  year\_of\_publication int not null,  author\_id int not null,  publisher\_id int not null,  quantity int not null,  foreign key (author\_id) references authors(author\_id),  foreign key (publisher\_id) references publishers(publisher\_id)  );  create table authors (  author\_id int primary key,  name varchar(255) not null,  city varchar(255) not null,  country varchar(255) not null  );  create table publishers (  publisher\_id int primary key,  name varchar(255) not null,  city varchar(255) not null,  country varchar(255) not null  );  create table orders (  order\_id int primary key,  date date not null  ); |
|  |
| insert into authors (author\_id, name, city, country)  values  (1, 'j.k. rowling', 'edinburgh', 'scotland'),  (2, 'stephen king', 'bangor', 'usa'),  (3, 'margaret atwood', 'ottawa', 'canada'); |
|  |
| insert into publishers (publisher\_id, name, city, country)  values  (1, 'bloomsbury publishing', 'london', 'england'),  (2, 'scribner', 'new york city', 'usa'),  (3, 'mcclelland & stewart', 'toronto', 'canada'); |
|  |
| insert into books (book\_id, title, year\_of\_publication, author\_id, publisher\_id, quantity)  values  (1, 'harry potter and the philosopher\'s stone', 1997, 1, 1, 10),  (2, 'the stand', 1978, 2, 2, 5),  (3, 'the handmaid\'s tale', 1985, 3, 3, 3),  (4, 'harry potter and the chamber of secrets', 1998, 1, 1, 8),  (5, 'harry potter and the prisoner of azkaban', 1999, 1, 1, 6),  (6, 'the shining', 1977, 2, 2, 4),  (7, 'the testaments', 2019, 3, 3, 2); |
|  |
| insert into orders (order\_id, date)  values  (1, '2022-03-15'),  (2, '2022-06-20'),  (3, '2023-01-10'); |
|  |
| select a.name as author\_name, count(\*) as num\_books\_published  from books b  inner join authors a on b.author\_id = a.author\_id  group by a.author\_id  order by num\_books\_published desc  limit 1; |
|  |
| select \* from books  where publisher\_id = 1 and year\_of\_publication = 2022; |
|  |
| delimiter //  create trigger check\_year\_of\_publication  before insert on books  for each row  begin  if new.year\_of\_publication != year(curdate()) then  signal sqlstate '45000' set message\_text = 'year of publication must be current year';  end if;  end //  delimiter ; |
|  |