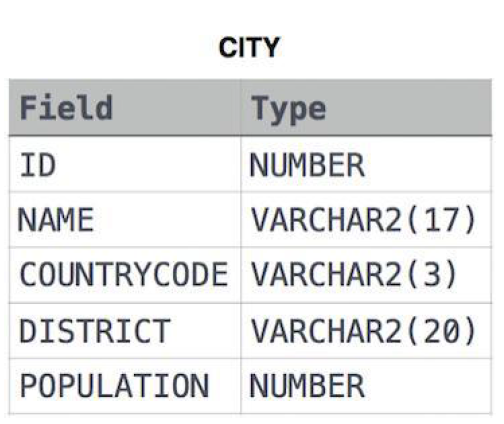
SQL-questions-set1



The CITY table is described as follows:



create table city(

    ID int,

    NAME varchar(17),

    COUNTRYCODE varchar(3),

    DISTRICT VARCHAR(20),

    POPULATION int

);

Insert the data:

insert into city VALUES(6,'Rotterdam','NLD','Zuid-Holland',593321),

(3878,'Scottsdale','USA','Arizona',202705),

(3965,'Corona','USA','California',124966),

(3973,'Concord','USA','California',121780),

(3977,'Cedar Rapids','USA','Iowa',120758),

(3982,'Coral Springs','USA','Florida',117549),

(4054,'Fairfield','USA','California',92256),

(4058,'Boulder','USA','Colorado',91238),

(4061,'Fall River','USA','Massachusetts',90555);

**Q1**. Query all columns for all American cities in the CITY table with populations larger than 100000.The CountryCode for America is USA.

***Solution:***

**select \*from city where POPULATION>=100000 and COUNTRYCODE='USA';**

| ID int | | NAME varchar | COUNTRYCODE varchar | | DISTRICT varchar | POPULATION int |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 3878 | Scottsdale | USA | Arizona | 202705 |
|  | 2 | 3965 | Corona | USA | California | 124966 |
|  | 3 | 3973 | Concord | USA | California | 121780 |
|  | 4 | 3977 | Cedar Rapids | USA | Iowa | 120758 |
|  | 5 | 3982 | Coral Springs | USA | Florida | 117549 |

**Q2**. Query the NAME field for all American cities in the CITY table with populations larger than 120000.The CountryCode for America is USA.

***Solution:***

**select NAME from city where population >=120000 and countrycode='USA';**

| NAME varchar | | |
| --- | --- | --- |
|  | 1 | Scottsdale |
|  | 2 | Corona |
|  | 3 | Concord |
|  | 4 | Cedar Rapids |

**Q3.** Query all columns (attributes) for every row in the CITY table.

***Solution:***

**select \*from city;**

| ID int | | NAME varchar | COUNTRYCODE varchar | | DISTRICT varchar | POPULATION int |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 6 | Rotterdam | NLD | Zuid-Holland | 593321 |
|  | 2 | 3878 | Scottsdale | USA | Arizona | 202705 |
|  | 3 | 3965 | Corona | USA | California | 124966 |
|  | 4 | 3973 | Concord | USA | California | 121780 |
|  | 5 | 3977 | Cedar Rapids | USA | Iowa | 120758 |
|  | 6 | 3982 | Coral Springs | USA | Florida | 117549 |
|  | 7 | 4054 | Fairfield | USA | California |  |

**Q4**. Query all columns for a city in CITY with the ID 1661.

***Solution:***

**select \*from city where ID=1661;**

| ID int | | NAME varchar | COUNTRYCODE varchar | DISTRICT varchar | | POPULATION int |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 1661 | Tokiyo | JPN | Zukan | 559321 |

**Q5**. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.

***Solution:***

**select \*from city where countrycode='JPN';**

| ID int | | NAME varchar | COUNTRYCODE varchar | DISTRICT varchar | | POPULATION int |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 1661 | Tokiyo | JPN | Zukan | 559321 |
|  | 2 | 1331 | Nagasaki | JPN | dhada | 679321 |

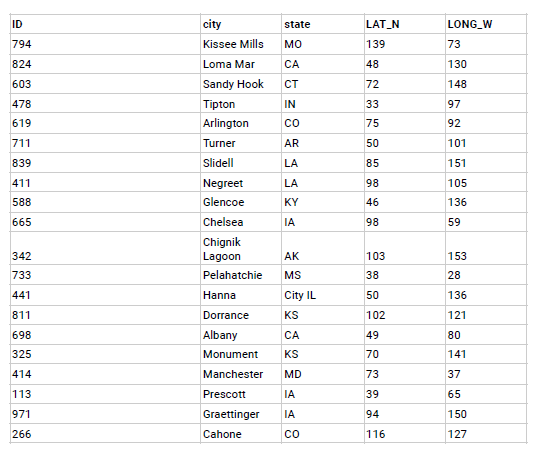
**Q6.** Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.

***Solution:***

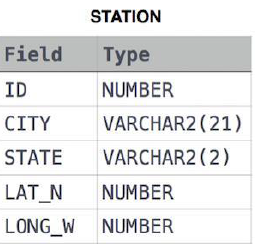
**select name from city where countrycode='JPN';**

| name varchar | | |
| --- | --- | --- |
|  | 1 | Tokiyo |
|  | 2 | Nagasaki |

**Sample Data set 2:**

****

The STATION table is described as follows:

****

where LAT\_N is the northern latitude and LONG\_W is the western longitude

create table station (

    id int,

    city varchar(21),

    state varchar(2),

    lat\_n int,

    long\_w int

);

**Insert the data;**

insert into station values(794,'Kissee Mills','MO',139,73),

(824,'Loma Mar','CA',48,130),

(603,'Sandy Hook','CT',72,148),

(478,'Tipton','IN',33,97),

(619,'Arlington','CO',75,92),

(711,'Turner','AR',50,101),

(839,'Slidell','LA',85,151),

(411,'Negreet','LA',98,105),

(588,'Glencoe','KY',46,136),

(665,'Chelsea','IA',98,59),

(342,'Chignik Lagoon','AK',103,153),

(733,'Pelahatchie','MS',38,28),

(441,'Hanna City','IL',50,136),

(811,'Dorrance','KS',102,121),

(698,'Albany','CA',49,80),

(325,'Monument','KS',70,141),

(414,'Manchester','MD',73,37),

(113,'Prescott','IA',39,65),

(971,'Graettinger','IA',94,150),

(266,'Cahone','CO',116,127);

**Q7.** Query a list of CITY and STATE from the STATION table.

***Solution:***

**select city,state from station;**

| city varchar | | | state varchar | |
| --- | --- | --- | --- | --- |
|  | 1 | Kissee Mills | | MO |
|  | 2 | Loma Mar | | CA |
|  | 3 | Sandy Hook | | CT |
|  | 4 | Tipton | | IN |

**.**

**.**

**.**

**Q8.** Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.

***Solution:***

**select id, city from station GROUP BY id, city;**

| id int | | | city varchar | |
| --- | --- | --- | --- | --- |
|  | 1 | 794 | | Kissee Mills |
|  | 2 | 824 | | Loma Mar |
|  | 3 | 603 | | Sandy Hook |
|  | 4 | 478 | | Tipton |
|  | 5 | 619 | | Arlington |

**.**

**.**

**.**

**Q9**. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

***Solution:***

**select (count(city)-count(DISTINCT(city)))as city\_diff from station;**

**Q10.** Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

***Solution:***

**Step 1:** Query the city and length

**select city,length(city)as len from station**

**Step 2:** If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

**with cte as(**

**select city,length(city)as len from station**

**),**

**select city,len from cte**

**where len=(select min(len) from cte )**

**order by city**

**limit 1;**

**Step 3:** Union the results

**with cte as(**

**select city,length(city)as len from station**

**),**

**result\_1 as(select city,len from cte**

**where len=(select min(len) from cte )**

**order by city**

**limit 1)**

**select city,len from cte where len=(select max(len) from cte)**

**UNION**

**select city,len from result\_1;**

| city varchar | | | len bigint | |
| --- | --- | --- | --- | --- |
|  | 1 | Chignik Lagoon | | 14 |
|  | 2 | Albany | | 6 |

**Q11**. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

***Solution:***

**select distinct(city) from station where substr(lower(city),1,1) in ('a','e','i','o','u');**

| city varchar | | |
| --- | --- | --- |
|  | 1 | Arlington |
|  | 2 | Albany |

**Q12.** Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

***Solution:***

**select distinct(city) from station**

**where substr(lower(city),-1,1) in ('a','e','i','o','u');**

| city varchar | | |
| --- | --- | --- |
|  | 1 | Glencoe |
|  | 2 | Chelsea |
|  | 3 | Pelahatchie |
|  | 4 | Dorrance |
|  | 5 | Cahone |

**Q13.** Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

***Solution:***

**select distinct(city) from station**

**where substr(lower(city),1,1) not in ('a','e','i','o','u');**

| city varchar | | |
| --- | --- | --- |
|  | 1 | Kissee Mills |
|  | 2 | Loma Mar |
|  | 3 | Sandy Hook |
|  | 4 | Tipton |
|  | 5 | Turner |

**.**

**.**

**.**

**Q14.** Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.

***Solution:***

**select distinct(city) from station**

**where substr(lower(city),-1,1) not in ('a','e','i','o','u');**

| city varchar | | |
| --- | --- | --- |
|  | 1 | Kissee Mills |
|  | 2 | Loma Mar |
|  | 3 | Sandy Hook |
|  | 4 | Tipton |
|  | 5 | Arlington |

**.**

**.**

**.**

**Q15.** Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

***Solution:***

**select distinct(city) from station**

**where substr(lower(city),1,1) not in('a','e','i','o','u') or**

**substr(lower(city),-1,1) not in ('a','e','i','o','u');**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | city varchar | | | | --- | --- | --- | |  | 1 | Kissee Mills | |  | 2 | Loma Mar | |  | 3 | Sandy Hook | |  | 4 | Tipton | |  | 5 | Arlington | | | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**.**

**.**

**.**

**Q16.** Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.

***Solution:***

**select distinct(city) from station**

**where substr(lower(city),1,1) not in('a','e','i','o','u') and**

**substr(lower(city),-1,1) not in ('a','e','i','o','u');**

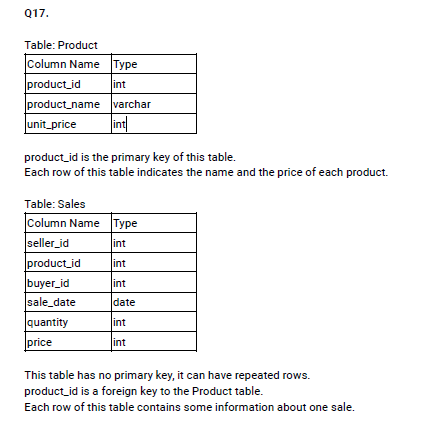
| city varchar | | |
| --- | --- | --- |
|  | 1 | Kissee Mills |
|  | 2 | Loma Mar |
|  | 3 | Sandy Hook |
|  | 4 | Tipton |

.

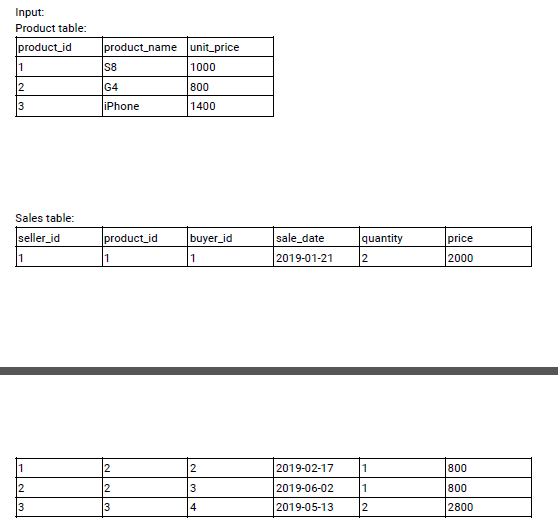
..

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 17;**

****

**Input data:**

****

create table Product(

    product\_id int,

    product\_name VARCHAR(20),

    unit\_price int

);

create table sales(

    seller\_id int,

    product\_id int,

    buyer\_id int,

    sale\_date date,

    quantity int,

    price int

);

insert into Product values(1,'S8',1000),

(2,'G4',800),

(3,'iPhone',1400);

insert into sales VALUES(1,1,1,'2019-01-21',2,2000),

(1,2,2,'2019-02-17',1,800),

(2,2,3,'2019-06-02',1,800),

(3,3,4,'2019-05-13',2,2800);

Write an SQL query that reports the products that were only sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive.

***Solution:***

**select product\_id,product\_name from Product**

**where product\_id not in (select product\_id from sales**

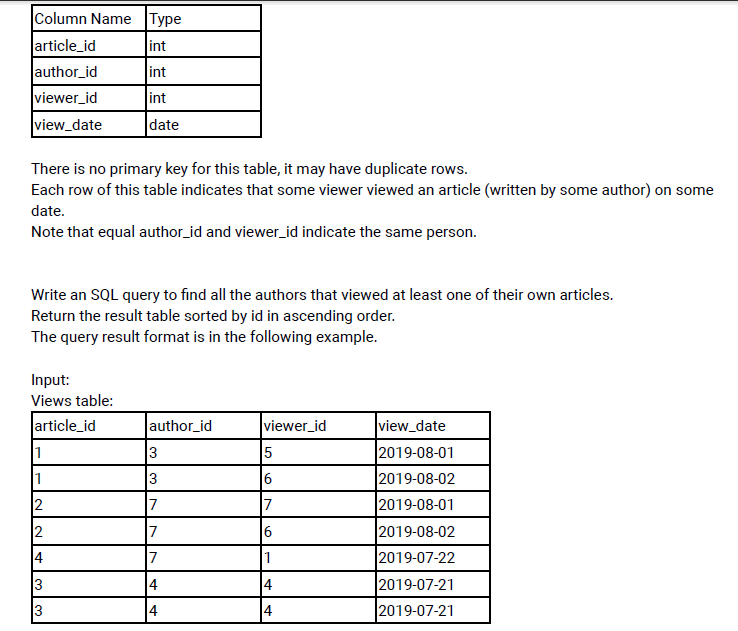
**where sale\_date not between '2019-01-01' and '2019-03-31' );**

| product\_id int | | | product\_name varchar | |
| --- | --- | --- | --- | --- |
|  | 1 | 1 | | S8 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 18;**

Write an SQL query to find all the authors that viewed at least one of their own articles. Return the result table sorted by id in ascending order.

****

create table views(

    article\_id int,

    author\_id int,

    viewer\_id int,

    view\_date date

);

insert into views values(1,3,5,'2019-08-01'),

(1,3,6,'2019-08-02'),

(2,7,7,'2019-08-01'),

(2,7,6,'2019-08-02'),

(4,7,1,'2019-07-22'),

(3,4,4,'2019-07-21'),

(3,4,4,'2019-07-21');

***Solution:***

**select author\_id from views**

**where author\_id=viewer\_id**

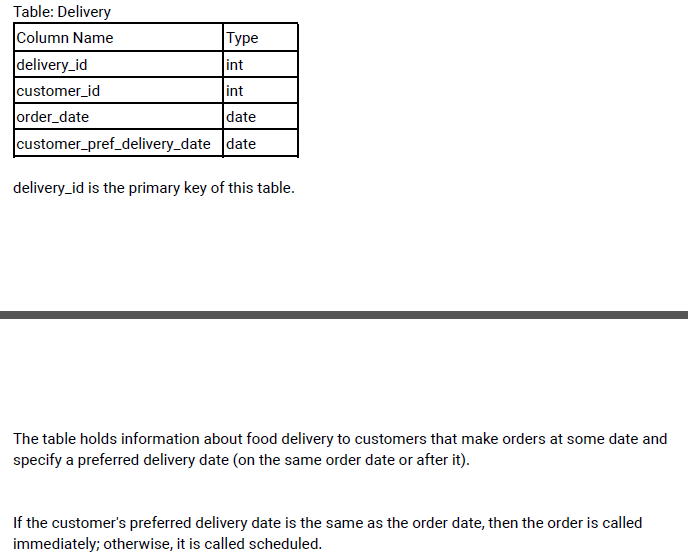
**group by author\_id**

**order by author\_id;**

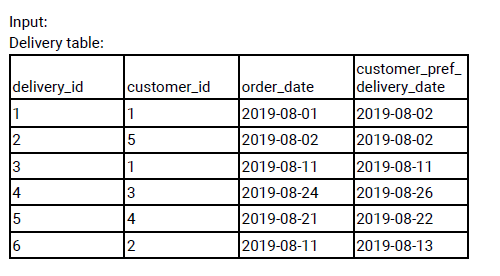
| author\_id int | | |
| --- | --- | --- |
|  | 1 | 4 |
|  | 2 | 7 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 19;**

****

**Input data:**

****

create table delivery(

    delivery\_id int,

    customer\_id int,

    order\_date date,

    customer\_pref\_delivery\_date date,

constraint pk PRIMARY KEY(delivery\_id)

);

insert into delivery values(1,1,'2019-08-01','2019-08-02'),

(2,5,'2019-08-02','2019-08-02'),

(3,1,'2019-08-11','2019-08-11'),

(4,3,'2019-08-24','2019-08-26'),

(5,4,'2019-08-21','2019-08-22'),

(6,2,'2019-08-11','2019-08-13');

Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.

***Solution:***

**Step 1:** find immediate delivery count and total count

**select count(\*)as immediate\_del,**

**(select count(\*) from delivery)as count**

**from delivery**

**where order\_date=customer\_pref\_delivery\_date;**

**Step 2:** Find percentage.

**with cte as(**

**select count(\*)as immediate\_del,(select count(\*) from delivery)as count**

**from delivery**

**where order\_date=customer\_pref\_delivery\_date**

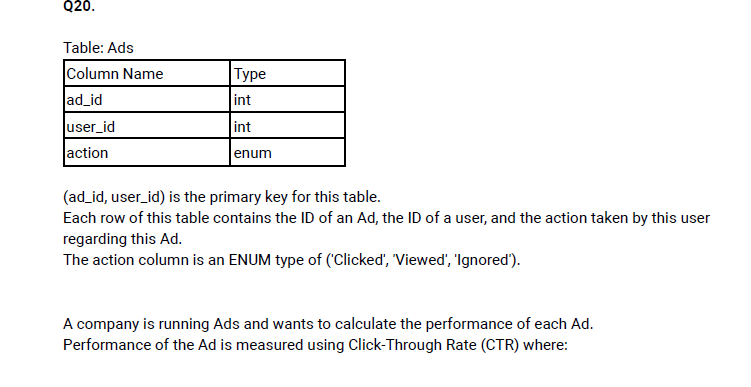
**)**

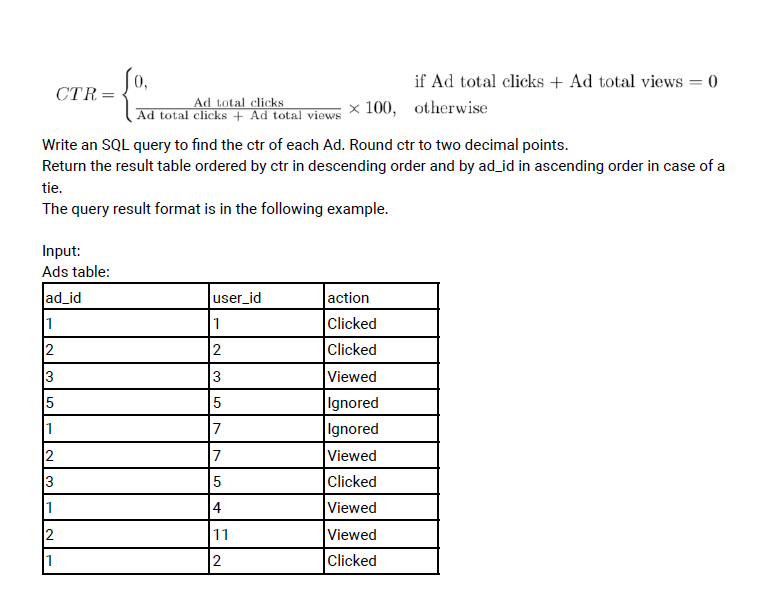
**select round(100.00\*immediate\_del/count,2) as immediate\_percentage from cte;**

| immediate\_percentage newdecimal | | |
| --- | --- | --- |
|  | 1 | 33.33 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 20;**

****

****

create table Ads(

    ad\_id int,

    user\_id int,

    action enum('Clicked', 'Viewed', 'Ignored'),

constraint pk PRIMARY KEY(ad\_id,user\_id)

);

insert into Ads VALUES(1,1,'Clicked'),

(2,2,'Clicked'),

(3,3,'Viewed'),

(5,5,'Ignored'),

(1,7,'Ignored'),

(2,7,'Viewed'),

(3,5,'Clicked'),

(1,4,'Viewed'),

(2,11,'Viewed'),

(1,2,'Clicked');

***Solution:***

**Step 1:** find the number of clicks and view for each ad\_id

**select ad\_id,**

**count(case when action='Clicked' then 1 end)as total\_click,**

**count(case when action='Viewed' then 1 end)as total\_view**

**from Ads**

**group by ad\_id;**

**Step 2:** Calculated the ctr

**with cte as(select ad\_id,**

**count(case when action='Clicked' then 1 end)as total\_click,**

**count(case when action='Viewed' then 1 end)as total\_view**

**from Ads**

**group by ad\_id)**

**select ad\_id,**

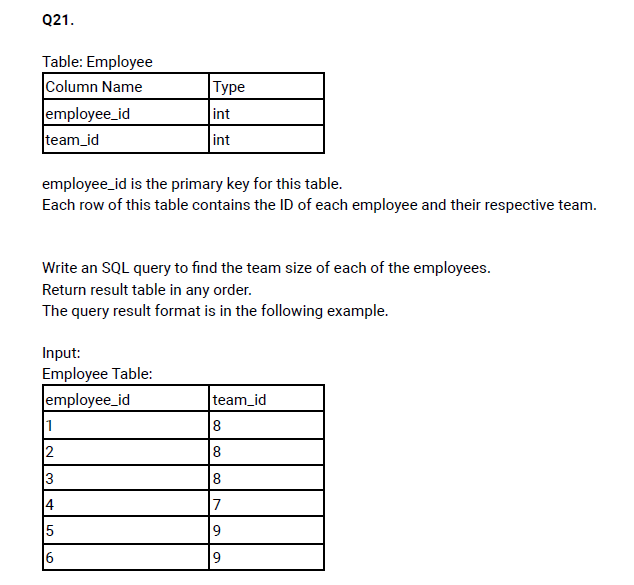
**round((total\_click/(total\_click+total\_view))\*100,2) as ctr**

**from cte;**

| ad\_id int | | | ctr newdecimal |
| --- | --- | --- | --- |
|  | 1 | 1 | 66.67 |
|  | 2 | 2 | 33.33 |
|  | 3 | 3 | 50.00 |
|  | 4 | 5 | 0 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 21;**

****

***Solution:***

create table employee(

    employee\_id int,

    team\_id int,

constraint emp\_pk PRIMARY KEY(employee\_id)

);

insert into employee values(1,8),(2,8),(3,8),(4,7),(5,9),(6,9);

**select employee\_id,**

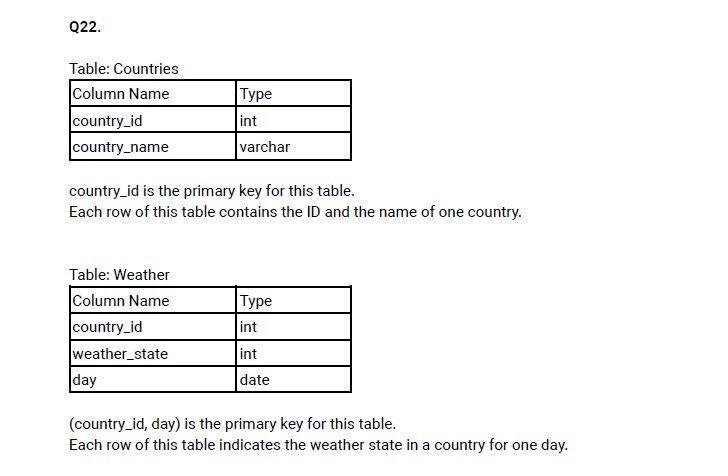
**count(team\_id) over(partition by team\_id) as team\_count from employee**

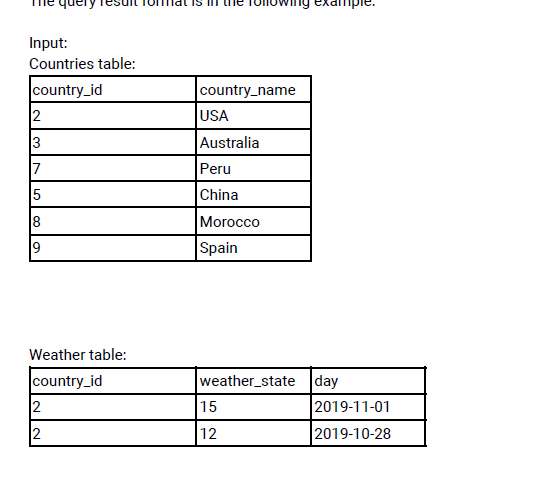
**order by employee\_id;**

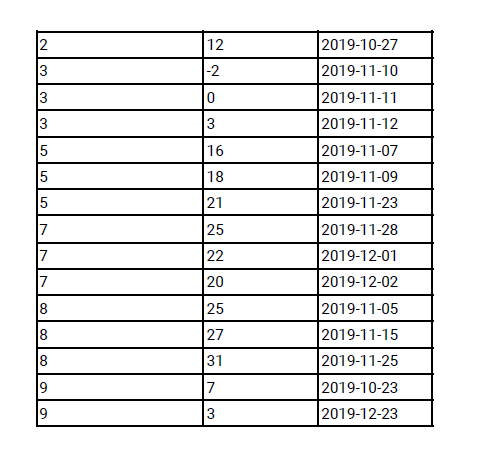
| employee\_id int | | | team\_count bigint | |
| --- | --- | --- | --- | --- |
|  | 1 |  | | 3 |
|  | 2 |  | | 3 |
|  | 3 |  | | 3 |
|  | 4 |  | | 1 |
|  | 5 |  | | 2 |
|  | 6 |  | | 2 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 22;**

****

****

****

create table countries(

    country\_id int,

    country\_name varchar(20),

    constraint pk PRIMARY KEY(country\_id)

);

insert into countries values(2,'USA'),(3,'Australia'),(7,'Peru'),(5,'China'),(8,'Morocco'),(9,'Spain');

create table Weather(

    country\_id int,

    weather\_state int,

    day date,

    constraint pk PRIMARY KEY(country\_id,day)

);

insert into Weather VALUES(2,15,'2019-11-01'),(2,12,'2019-10-28'),(2,12,'2019-10-27'),

(3,-2,'2019-11-10'),(3,0,'2019-11-11'),(3,3,'2019-11-12'),(5,16,'2019-11-07'),

(5,18,'2019-11-09'),(5,21,'2019-11-23'),(7,25,'2019-11-28'),(7,22,'2019-12-01'),

(7,20,'2019-12-02'),(8,25,'2019-11-05'),(8,27,'2019-11-15'),(8,31,'2019-11-25'),

(9,7,'2019-10-23'),(9,3,'2019-12-23');

Write an SQL query to find the type of weather in each country for November 2019.

The type of weather is:

* Cold if the average weather\_state is less than or equal 15,
* Hot if the average weather\_state is greater than or equal to 25, and
* Warm otherwise.

***Solution:***

**Step 1:** join the Countries and Weather table.

**Step 2:** find Novmber 2019 data and Average weather\_state

**select c.country\_name,**

**Round(avg(w.weather\_state),2)as weather\_state**

**from countries c left join Weather w**

**on c.country\_id=w.country\_id**

**where extract(month from day)=11**

**group by c.country\_name;**

**Step 3:** Find the weather\_type based on questions

with cte as(select c.country\_name,round(avg(w.weather\_state),2)as weather\_state from countries c left join Weather w

on c.country\_id=w.country\_id

where extract(month from day)=11

GROUP BY c.country\_name

)

select country\_name,

(case when weather\_state <=15 then 'Cold'

when weather\_state>=25 then 'Hot'

else 'Warm' end)as weather\_type

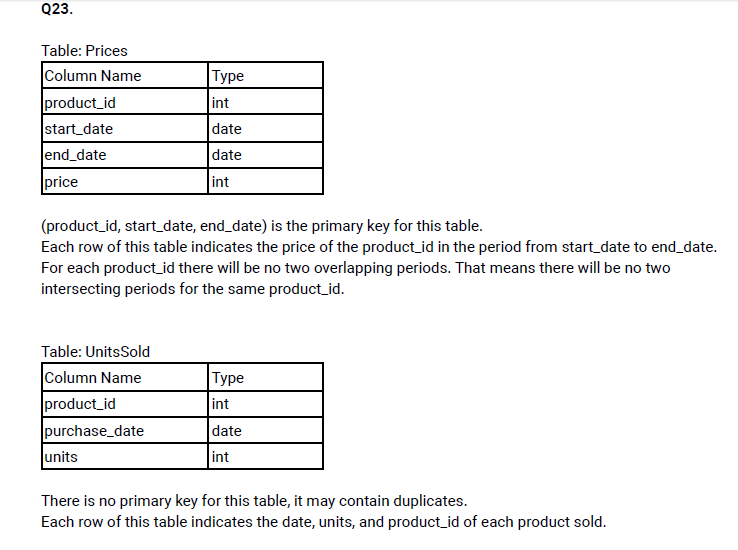
from cte

;

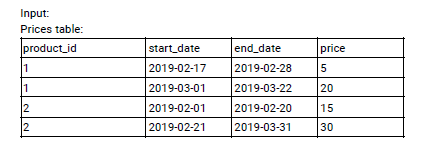
| country\_name varchar | | | weather\_type varchar | |
| --- | --- | --- | --- | --- |
|  | 1 | USA | | Cold |
|  | 2 | Australia | | Cold |
|  | 3 | China | | Warm |
|  | 4 | Peru | | Hot |
|  | 5 | Morocco | | Hot |

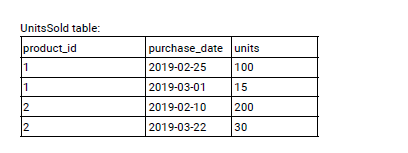
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 23 & 40;**

****

**Insert data;**

****

****

create table prices(

    product\_id int,

    start\_date date,

    end\_date date,

    price int,

    constraint pk PRIMARY KEY(product\_id,start\_date,end\_date)

);

insert into prices values(1,'2019-02-17','2019-02-28',5),

(1,'2019-03-01','2019-03-22',20),

(2,'2019-02-01','2019-02-20',15),

(2,'2019-02-21','2019-03-31',30);

create table unitssold(

    product\_id int,

    purchase\_date date,

    units int

);

insert into unitssold VALUES(1,'2019-02-25',100),

(1,'2019-03-01',15),

(2,'2019-02-10',200),

(2,'2019-03-22',30);

Write an SQL query to find the average selling price for each product. average\_price should be rounded to 2 decimal places.

***Solution:***

**Step 1:** Join the two tables and find purchase\_date between start\_date and end\_date

**select \*from prices p left join unitssold u**

**on p.product\_id=u.product\_id**

**where u.purchase\_date BETWEEN p.start\_date and p.end\_date;**

**step 2:** calculate average unit,

with cte as(select p.product\_id,p.price,u.units from  prices p left join unitssold u

on p.product\_id=u.product\_id

where u.purchase\_date BETWEEN p.start\_date and p.end\_date)

select product\_id,round(sum(price\*units)/sum(units),2)as avg\_units

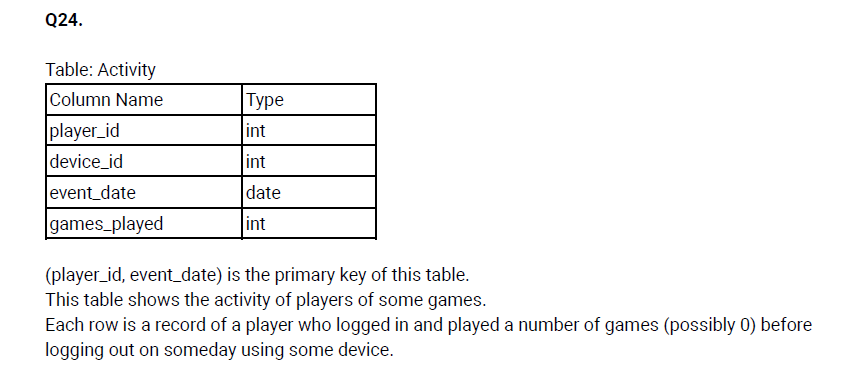
from cte

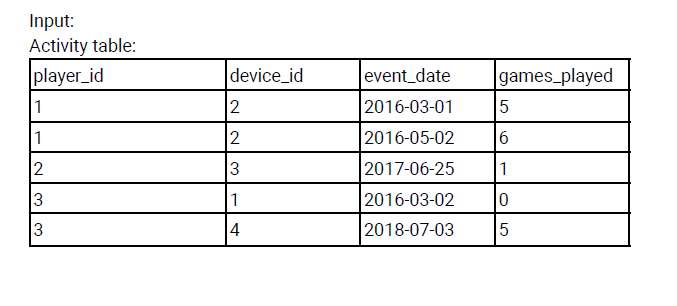
group by product\_id;

| product\_id int | | | avg\_units newdecimal |
| --- | --- | --- | --- |
|  | 1 |  | 6.96 |
|  | 2 |  | 16.96 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 24;**

****

****Write an SQL query to report the first login date for each player.

***Solution:***

create table Activity(

    player\_id int,

    device\_id int,

    event\_date date,

    games\_played int,

    constraint pk PRIMARY KEY(player\_id,event\_date)

);

insert into Activity values(1,2,'2016-03-01',5),

(1,2,'2016-05-02',6),

(2,3,'2017-06-25',1),

(3,1,'2016-03-02',0),

(3,4,'2018-07-03',5);

**with cte as(select \*,**

**row\_number() over(partition by player\_id order by event\_date)as row\_num**

**from Activity)**

**select player\_id,event\_date as first\_login from cte where row\_num=1;**

| player\_id int | | | first\_login date |
| --- | --- | --- | --- |
|  | 1 |  | 2016-03-01 |
|  | 2 |  | 2017-06-25 |
|  | 3 |  | 2016-03-02 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 25;**

**Table will be follow above question number 26.**

Write an SQL query to report the device that is first logged in for each player.

***Solution:***

**with cte as(select \*,**

**row\_number() over(partition by player\_id order by event\_date)as row\_num**

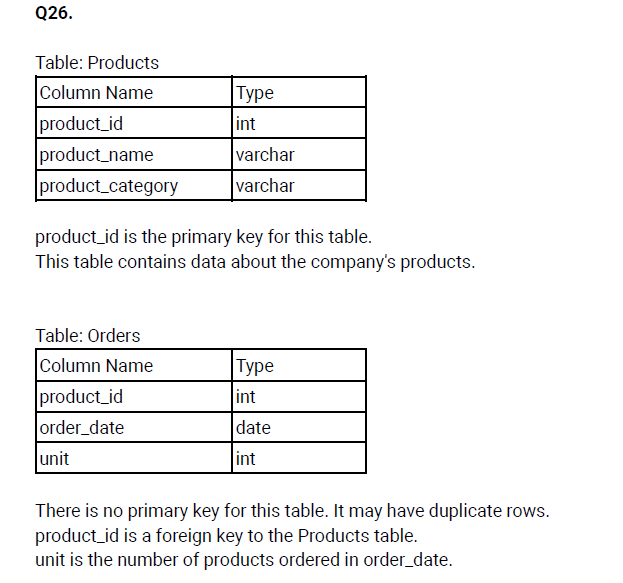
**from Activity)**

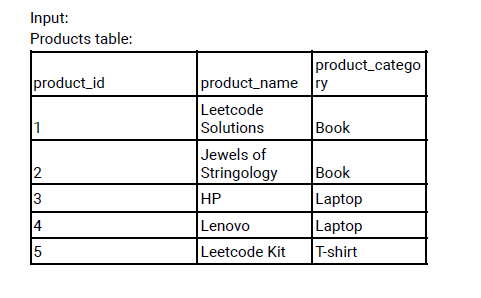
**select player\_id,device\_id from cte where row\_num=1;**

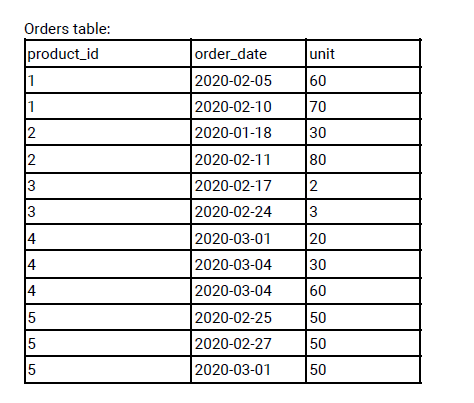
| player\_id int | | | device\_id int | |
| --- | --- | --- | --- | --- |
|  | 1 |  | | 2 |
|  | 2 |  | | 3 |
|  | 3 |  | | 1 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 26;**

****

****

****

Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount.

***Solution:***

create table products(

    product\_id int,

    product\_name varchar(40),

    product\_category varchar(20),

    constraint pk PRIMARY KEY(product\_id)

);

insert into products values(1,'Leetcode Solutions','Book'),

(2,'Jewels of Stringology','Book'),

(3,'HP','Laptop'),

(4,'Lenovo','Laptop'),

(5,'Leetcode Kit','T-shirt');

Create table orders(

    product\_id int,

    order\_date date,

    unit int

);

insert into orders values(1,'2020-02-05',60),(1,'2020-02-10',70),(2,'2020-01-18',30),

(2,'2020-02-11',80),(3,'2020-02-17',2),(3,'2020-02-24',3),(4,'2020-03-01',20),

(4,'2020-03-04',30),(4,'2020-03-04',60),(5,'2020-02-25',50),(5,'2020-02-27',50),

(5,'2020-03-01',50);

**select \*from (**

**select p.product\_name,sum(o.unit)as unit**

**from products p left join orders o**

**on p.product\_id=o.product\_id**

**where o.order\_date BETWEEN '2020-02-01' and '2020-02-29'**

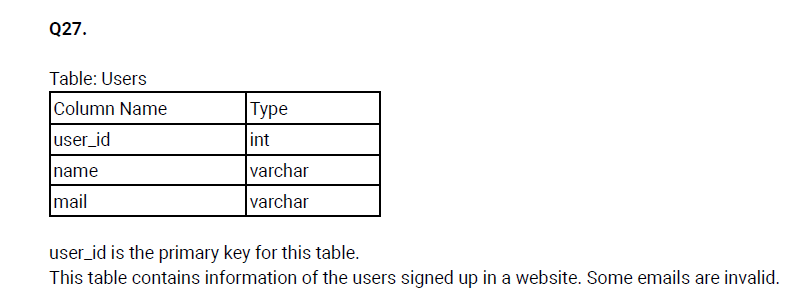
**group by p.product\_name)s**

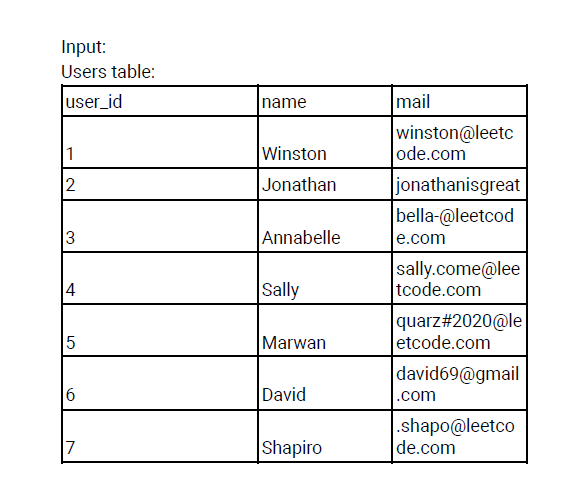
**where unit>=100;**

| product\_name varchar | | | unit newdecimal | |
| --- | --- | --- | --- | --- |
|  | 1 | Leetcode Solutions | | 130 |
|  | 2 | Leetcode Kit | | 100 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 27;**

****

****

Write an SQL query to find the users who have valid emails.

A valid e-mail has a prefix name and a domain where:

* The prefix name is a string that may contain letters (upper or lower case), digits, underscore '\_', period '.', and/or dash '-'. The prefix name must start with a letter.
* The domain is '@leetcode.com'.

***Solution:***

create table user(

    user\_id int,

    name varchar(40),

    mail varchar(40),

    constraint pk PRIMARY KEY(user\_id)

);

insert into user values(1,'Winston','winston@leetcode.com'),

(2,'Jonathan','jonathanisgreat'),

(3,'Annabelle','bella-@leetcode.com'),

(4,'Sally','sally.come@leetcode.com'),

(5,'Marwan','quarz#2020@leetcode.com'),

(6,'David','david69@gmail.com'),

(7,'Shapiro','.shapo@leetcode.com');

**Step 1:** Find domain name and name from given table;

**select user\_id,mail,**

**substr(mail,1,instr(mail,'@')-1)as name,**

**substr(mail,instr(mail,'@')+1)as domain\_name from user;**

**Step 2:** filter the domain name like ‘Leetcode.come’ and name start with letter**,**

**with cte as(select user\_id,mail,**

**substr(mail,1,instr(mail,'@')-1)as name,**

**substr(mail,instr(mail,'@')+1)as domain\_name from user)**

**select user\_id,mail from cte**

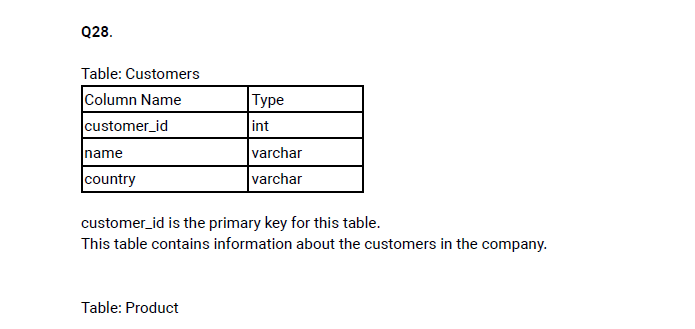
**where domain\_name='leetcode.com' and substr(lower(name),1,1) in('a','b','c','d','e','f','g','i','j','k','l','m','n',**

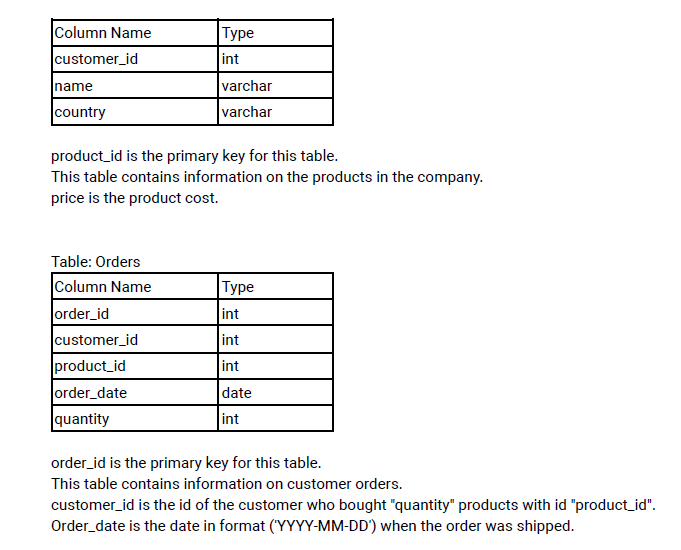
**'o','p','q','r','s','t','u','v','w','x','y','z');**

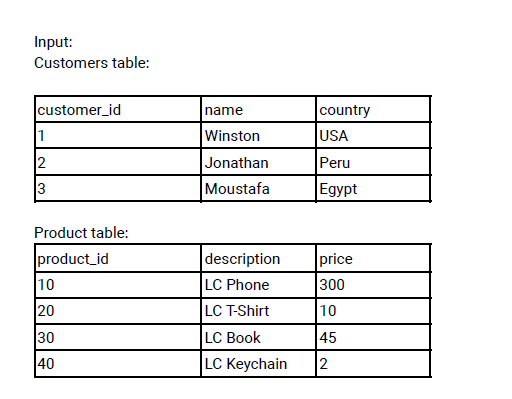
| user\_id int | | | mail varchar |
| --- | --- | --- | --- |
|  | 1 |  | winston@leetcode.com |
|  | 3 |  | bella-@leetcode.com |
|  | 4 |  | sally.come@leetcode.com |
|  | 5 |  | quarz#2020@leetcode.com |

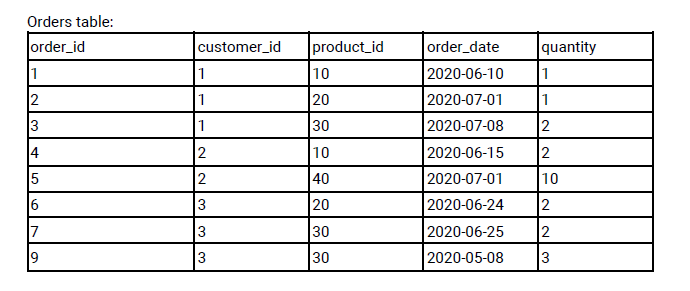
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 28;**

****

****

****

****

Write an SQL query to report the customer\_id and customer\_name of customers who have spent at least $100 in each month of June and July 2020.

***Solution:***

create table customers(

    customer\_id int,

    name varchar(20),

    country varchar(20),

    constraint pk PRIMARY KEY(customer\_id)

);

insert into customers values(1,'Winston','USA'),(2,'Jonathan','Peru'),(3,'Moustafa','Egypt');

create table product(

    product\_id int,

    description varchar(20),

    price int,

    constraint pk PRIMARY KEY(product\_id)

);

insert into product VALUES(10,'LC Phone',300),(20,'LC T-Shirt',10),

(30,'LC Book',45),(40,'LC Keychain',2);

create table ordered(

    order\_id int,

    customer\_id int,

    product\_id int,

    order\_date date,

    quantity int,

    constraint pk PRIMARY KEY(order\_id)

);

insert into ordered VALUES(1,1,10,'2020-06-10',1),(2,1,20,'2020-07-01',1),(3,1,30,'2020-07-08',2),

(4,2,10,'2020-06-15',2),(5,2,40,'2020-07-01',10),(6,3,20,'2020-06-24',2),(7,3,30,'2020-06-25',2),

(9,3,30,'2020-05-08',3);

**Step 1:** Join the three table first,

**select \*from ordered o left join customers c**

**on o.customer\_id=c.customer\_id**

**left join product p**

**on o.product\_id=p.product\_id;**

**step 2:** Find the month of june and july records

**select c.customer\_id,c.name,extract(month from o.order\_date)as month,(p.price\*o.quantity)as total\_price**

**from ordered o left join customers c**

**on o.customer\_id=c.customer\_id**

**left join product p**

**on o.product\_id=p.product\_id**

**where o.order\_date between '2020-06-01' and '2020-07-31';**

**Step 3:** find total price of each month per person,

**with cte as(**

**select c.customer\_id,c.name,extract(month from o.order\_date)as month,(p.price\*o.quantity)as total\_price**

**from ordered o left join customers c**

**on o.customer\_id=c.customer\_id**

**left join product p**

**on o.product\_id=p.product\_id**

**where o.order\_date between '2020-06-01' and '2020-07-31')**

**select customer\_id,name,sum(case when month=6 then total\_price end)as june,**

**sum(case when month=7 then total\_price end)as july**

**from cte**

**group by customer\_id,name;**

**Step 4:** Find to who have order above 100$ each month

**with cte as(select c.customer\_id,c.name,extract(month from o.order\_date)as**

**month,(p.price\*o.quantity)as total\_price**

**from ordered o left join customers c**

**on o.customer\_id=c.customer\_id**

**left join product p**

**on o.product\_id=p.product\_id**

**where o.order\_date between '2020-06-01' and '2020-07-31'),**

**result\_1 as (select customer\_id,name,sum(case when month=6 then total\_price end)as june,**

**sum(case when month=7 then total\_price end)as july**

**from cte**

**group by customer\_id,name)**

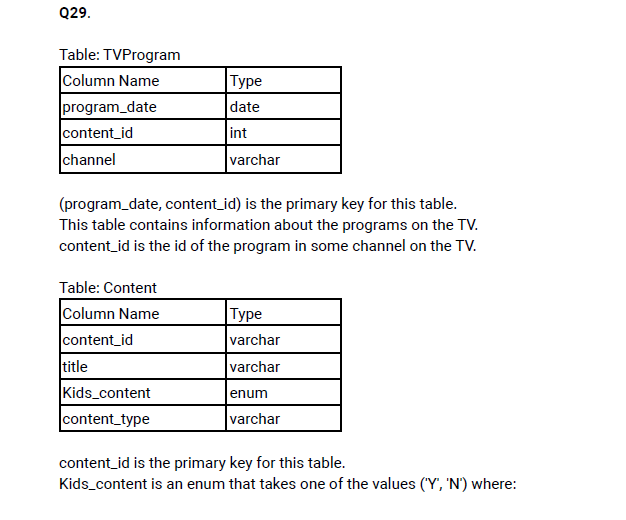
**select customer\_id,name from result\_1**

**where june>=100 and july>=100;**

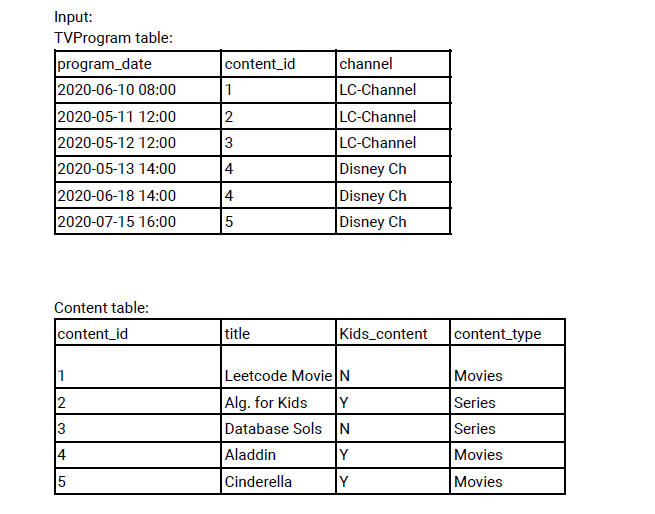
| customer\_id int | | | name varchar |
| --- | --- | --- | --- |
|  | 1 |  | Winston |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 29;**

****

'Y' means content for kids, otherwise 'N' is not content for kids. content\_type is the category of the content as movies, series, etc.

****

Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020.

***Solution:***

create table TVProgram(

    program\_date date,

    content\_id int,

    channel varchar(20),

    constraint pk PRIMARY KEY(program\_date,content\_id)

);

insert into TVProgram values('2020-06-10 08:00',1,'LC-Channel'),

('2020-05-11 12:00',2,'LC-Channel'),

('2020-05-12 12:00',3,'LC-Channel'),

('2020-05-13 14:00',4,'Disney Ch'),

('2020-06-18 14:00',4,'Disney Ch'),

('2020-07-15 16:00',5,'Disney Ch');

create table content (

    content\_id VARCHAR(40),

    title VARCHAR(30),

    kids\_content enum('Y','N'),

    content\_type VARCHAR(30),

    constraint pk PRIMARY KEY(content\_id)

);

insert into content values(1,'Leetcode Movie','N','Movies'),

(2,'Alg. for Kids','Y','Series'),

(3,'Database Sols','N','Series'),

(4,'Aladdin','Y','Movies'),

(5,'Cinderella','Y','Movies');

select distinct(c.title) from TVProgram t left join content c

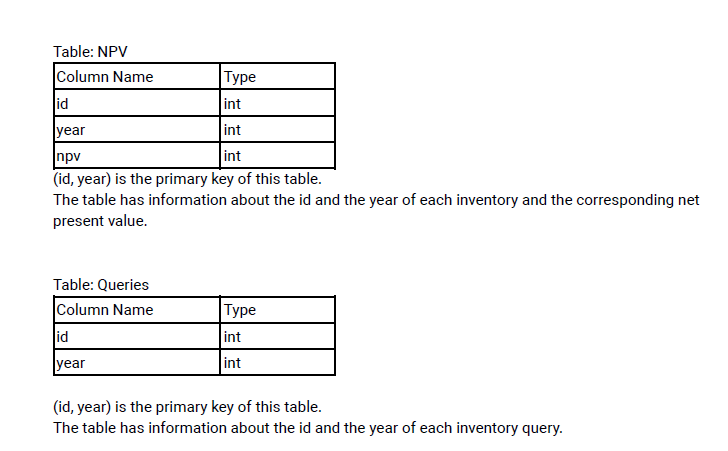
on t.content\_id=c.content\_id

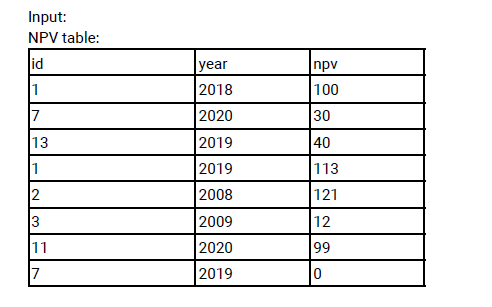
where c.kids\_content like 'Y' and c.content\_type like 'Movies' and extract(month from t.program\_date)=6;

| title varchar | | |
| --- | --- | --- |
|  |  | Aladdin |

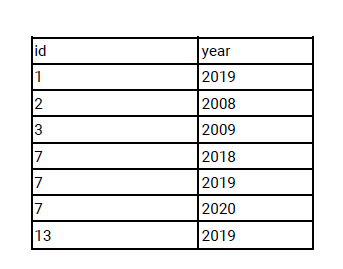
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 30 & 31;**

****

****

**Queries table;**

****

Write an SQL query to find the npv of each query of the Queries table.

create table NPV(

    id int,

    year int,

    npv int,

    constraint pk PRIMARY key(id,year)

);

insert into NPV values(1,2018,100),(7,2020,30),(13,2019,40),(1,2019,113),(2,2008,121),(3,2009,12),(11,2020,99),(7,2019,0);

create table Queries (

    id int,

    year int,

    constraint pk PRIMARY KEY(id,year)

) ;

insert into Queries values(1,2019),(2,2008),(3,2009),(7,2018),(7,2019),(7,2020),(13,2019);

***Solution:***

select q.\*,

(case when n.npv is null then 0 else n.npv end)as npv

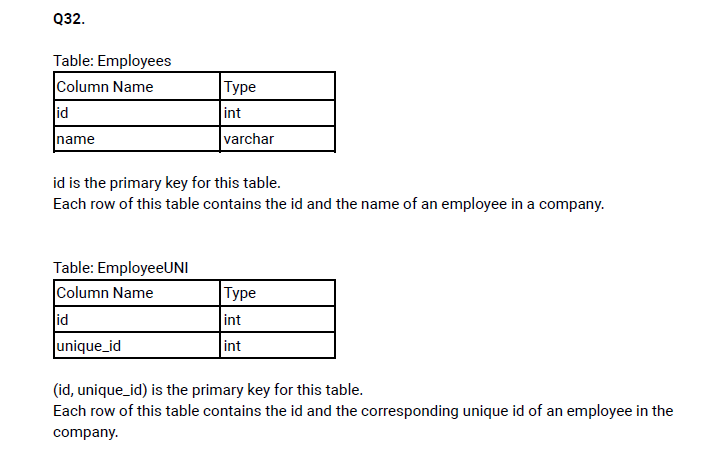
from  Queries q left join NPV n

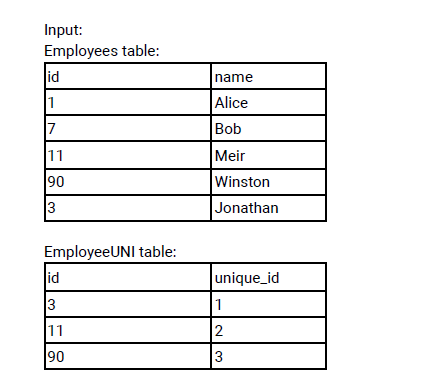
on n.id=q.id and n.year=q.year;

| id int | | year int | | npv bigint |
| --- | --- | --- | --- | --- |
|  | 1 |  | 2019 | 113 |
|  | 2 |  | 2008 | 121 |
|  | 3 |  | 2009 | 12 |
|  | 7 |  | 2018 | 0 |
|  | 7 |  | 2019 | 0 |
|  | 7 |  | 2020 | 30 |
|  | 13 |  | 2019 | 40 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 32 & 37;**

****

****

create table employee (

    id int,

    name VARCHAR(20),

    constraint pk PRIMARY KEY(id)

);

insert into employee values(1,'Alice'),(7,'Bob'),(11,'Meir'),(90,'Winston'),(3,'Jonathan');

create table employee\_uni(

    id int,

    unique\_id int,

    constraint pk PRIMARY KEY(id,unique\_id)

);

insert into employee\_uni values(3,1),(11,2),(90,3);

Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null.

***Solution:***

select u.unique\_id,e.name from employee e left join employee\_uni u

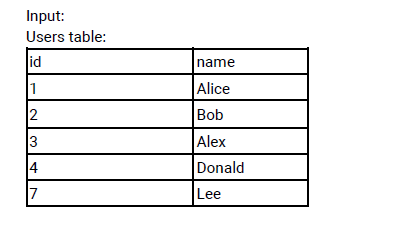
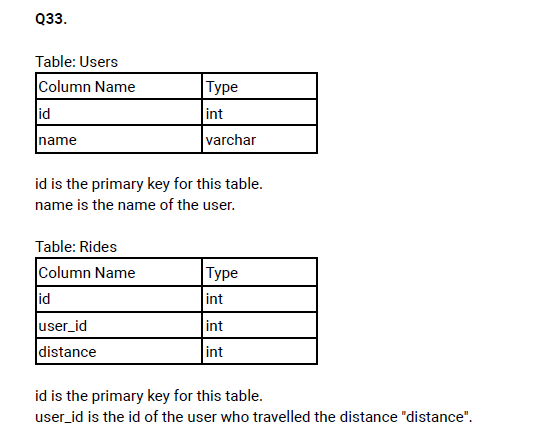
on e.id=u.id

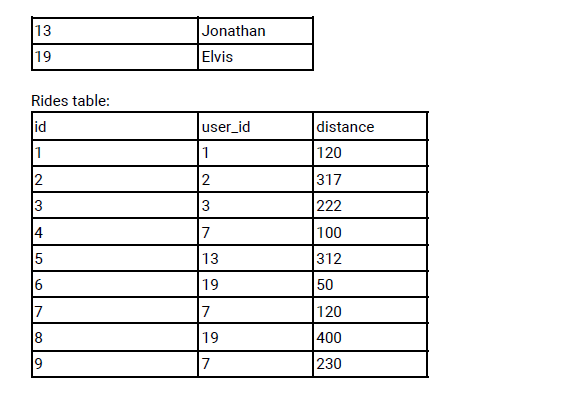
order by e.name;

| unique\_id int | | | name varchar |
| --- | --- | --- | --- |
|  |  | (NULL) | Alice |
|  |  | (NULL) | Bob |
|  |  | 1 | Jonathan |
|  |  | 2 | Meir |
|  |  | 3 | Winston |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 33 & 36;**

****

****

create table users(

    id int,

    name VARCHAR(20),

    constraint pk PRIMARY KEY(id)

);

insert into users values(1,'Alice'),(2,'Bob'),(3,'Alex'),(4,'Donald'),(7,'Lee'),(13,'Jonathan'),(19,'Elvis');

create table rides(

    id int,

    user\_id int ,

    distance int,

    constraint pk PRIMARY KEY(id)

);

insert into rides values(1,1,120),(2,2,317),(3,3,222),(4,7,100),(5,13,312),(6,19,50),(7,7,120),(8,19,400),(9,7,230);

Write an SQL query to report the distance travelled by each user. Return the result table ordered by travelled\_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order.

***Solution:***

select u.name,

sum(case when r.distance is null then 0 else r.distance end) as distance

from users u left join rides r

on u.id=r.user\_id

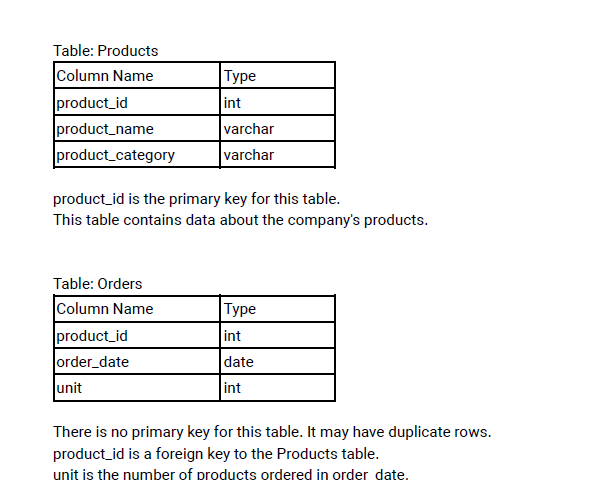
group by u.name

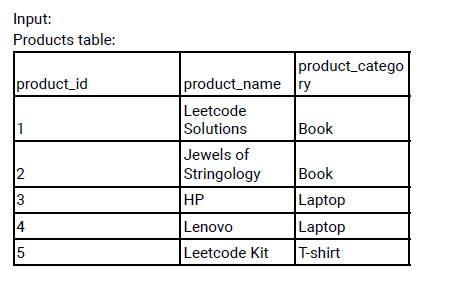
order by distance desc, u.name asc;

| name varchar | | | distance newdecimal | |
| --- | --- | --- | --- | --- |
|  |  | Lee | | 450 |
|  |  | Elvis | | 450 |
|  |  | Bob | | 317 |
|  |  | Jonathan | | 312 |
|  |  | Alex | | 222 |
|  |  | Alice | | 120 |
|  |  | Donald | | 0 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 34;**

****

****

create table products(

    product\_id int,

    product\_name varchar(40),

    product\_category varchar(20),

    constraint pk PRIMARY KEY(product\_id)

);

insert into products values(1,'Leetcode Solutions','Book'),

(2,'Jewels of Stringology','Book'),

(3,'HP','Laptop'),

(4,'Lenovo','Laptop'),

(5,'Leetcode Kit','T-shirt');

Create table orders(

    product\_id int,

    order\_date date,

    unit int

);

insert into orders values(1,'2020-02-05',60),(1,'2020-02-10',70),(2,'2020-01-18',30),

(2,'2020-02-11',80),(3,'2020-02-17',2),(3,'2020-02-24',3),(4,'2020-03-01',20),

(4,'2020-03-04',30),(4,'2020-03-04',60),(5,'2020-02-25',50),(5,'2020-02-27',50),

(5,'2020-03-01',50);

Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount.

***Solution:***

with cte as(select p.product\_name,sum(o.unit)as unit  from products p left join orders o

on p.product\_id=o.product\_id

where extract(month from o.order\_date)=2

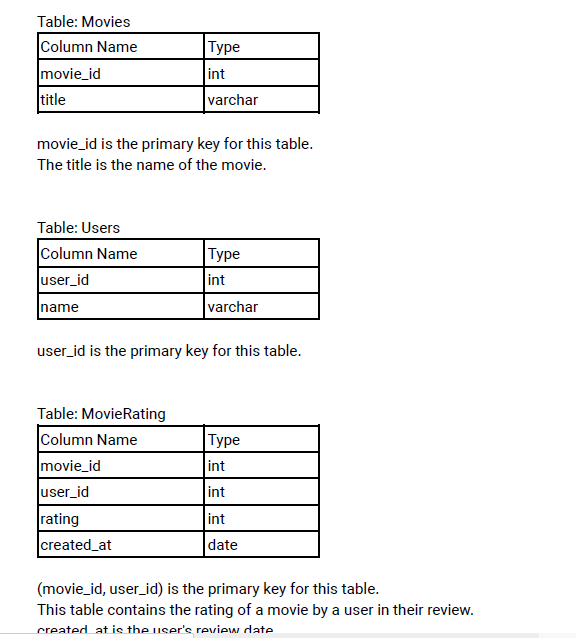
group by p.product\_name)

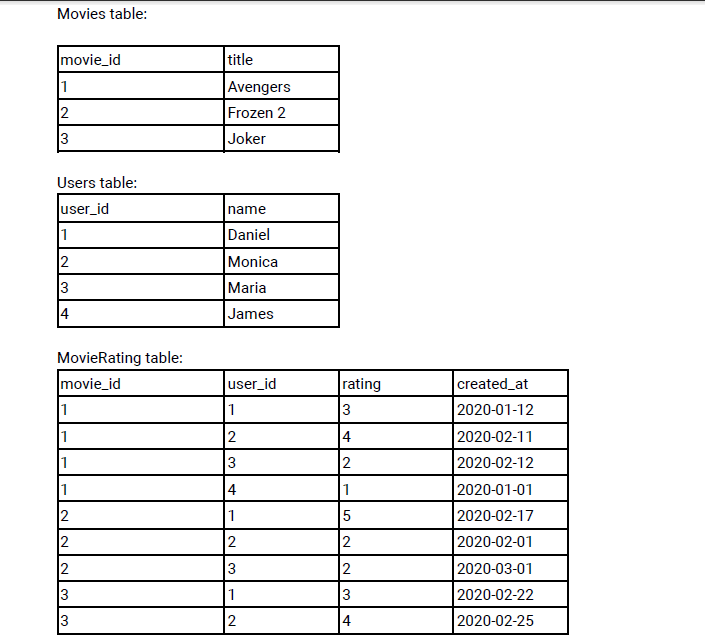
select \*from cte where unit>=100;

| product\_name varchar | | | unit newdecimal | |
| --- | --- | --- | --- | --- |
|  |  | Leetcode Solutions | | 130 |
|  |  | Leetcode Kit | | 100 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 35;**

****

****

create table Movies(

movie\_id int,

    title varchar(20),

    constraint pk PRIMARY KEY(movie\_id)

);

insert into Movies values(1,'Avengers'),(2,'Frozen2'),(3,'Joker');

create table users(

    user\_id int,

    name VARCHAR(20),

    constraint pk PRIMARY KEY(user\_id)

);

insert into users values(1,'Daniel'),(2,'Monica'),(3,'Maria'),(4,'James');

create table MovieRating(

    movie\_id int,

    user\_id int,

    rating int,

    created\_at date,

    constraint pk PRIMARY KEY (movie\_id,user\_id)

);

insert into MovieRating values(1,1,3,'2020-01-12'),

(1,2,4,'2020-02-11'),

(1,3,2,'2020-02-12'),(1,4,1,'2020-01-01'),(2,1,5,'2020-02-17'),

(2,2,2,'2020-02-01'),(2,3,2,'2020-03-01'),

(3,1,3,'2020-02-22'),(3,2,4,'2020-02-25');

Write an SQL query to:

* Find the name of the user who has rated the greatest number of movies. In case of a tie, return the lexicographically smaller user name.
* Find the movie name with the highest average rating in February 2020. In case of a tie, return the lexicographically smaller movie name.

Step 1: Query1

select result from(select u.name as result from MovieRating r left join Movies M

on r.movie\_id=M.movie\_id

left join users u on r.user\_id=u.user\_id

group by u.name

order by u.name limit 1)s1

step 2:Query2

select title as result from (select M.title,avg(r.rating)as rating from MovieRating r left join Movies M

on r.movie\_id=M.movie\_id

left join users u on r.user\_id=u.user\_id

where extract(month from r.created\_at)=2

group by M.title

order by rating desc,M.title limit 1 )s

;

Step 3: Union the query,

select result from(select u.name as result from MovieRating r left join Movies M

on r.movie\_id=M.movie\_id

left join users u on r.user\_id=u.user\_id

group by u.name

order by u.name limit 1)s1

union

select title as result from (select M.title,avg(r.rating)as rating from MovieRating r left join Movies M

on r.movie\_id=M.movie\_id

left join users u on r.user\_id=u.user\_id

where extract(month from r.created\_at)=2

group by M.title

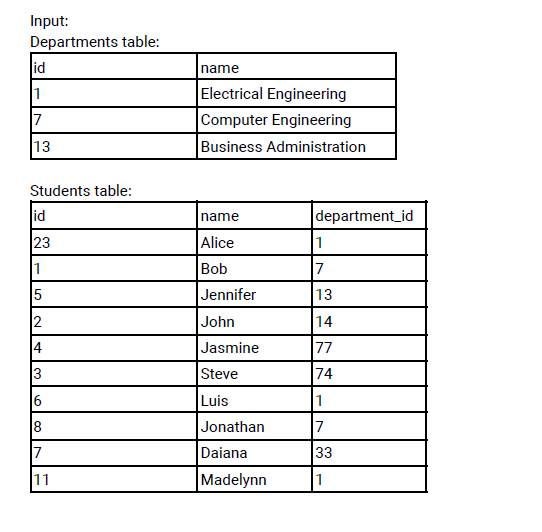
order by rating desc,M.title limit 1 )s

;

| result varchar | | |
| --- | --- | --- |
|  |  | Daniel |
|  |  | Frozen2 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 38;**



create table departments(

    id int,

    name VARCHAR(60),

    constraint pk PRIMARY KEY(id)

);

insert into departments values(1,'Electrical Engineering'),(7,'Computer Engineering'),(13,'Business Administration');

create table students(

    id int,

    name varchar(20),

    department\_id int,

    constraint pk PRIMARY KEY(id)

);

insert into students values(23,'Alice',1),(1,'Bob',7),(5,'Jennifer',13),(2,'John',14),(4,'Jasmine',77),

(3,'Steve',74),(6,'Luis',1),(8,'Jonathan',7),(7,'Daiana',33),(11,'Madelynn',1);

Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist.

***Solution:***

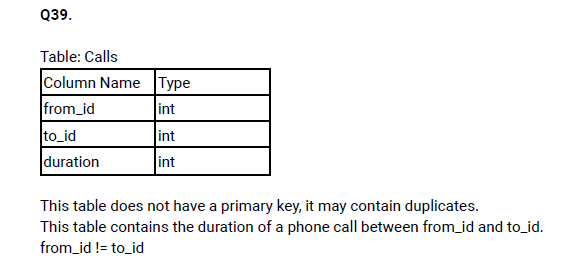
select id,name from  students

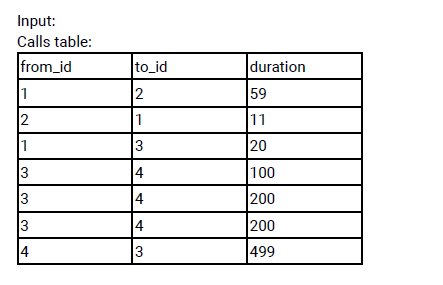
where department\_id not in (select id from departments);

| id int | | | name varchar |
| --- | --- | --- | --- |
|  | 2 |  | John |
|  | 3 |  | Steve |
|  | 4 |  | Jasmine |
|  | 7 |  | Daiana |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 39;**





Write an SQL query to report the number of calls and the total call duration between each pair of distinct persons (person1, person2) where person1 < person2.

***Solution:***

with cte as(select from\_id as person1 , to\_id as person2,duration

            from calls

            union ALL

            select to\_id as person1, from\_id as person2, duration

            from calls

),

result\_1 as(select person1,person2,duration from cte

            where person1<person2

)

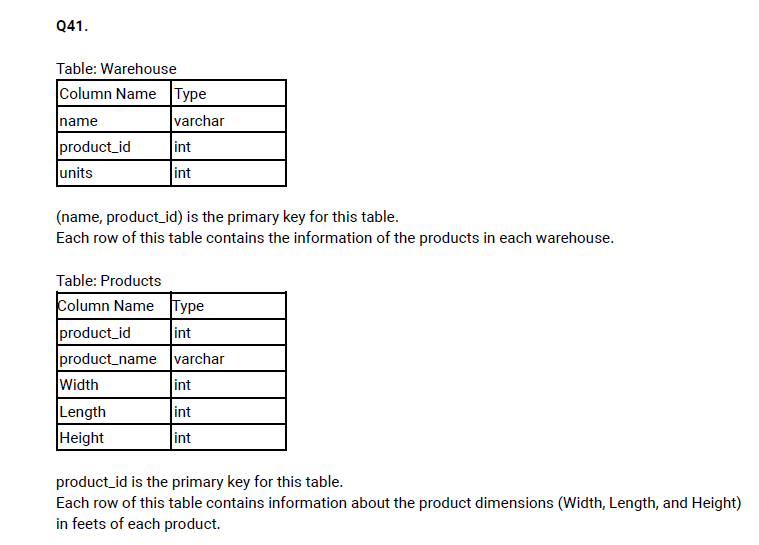
select person1,person2,count(\*)as call\_count,sum(duration)as total\_duration

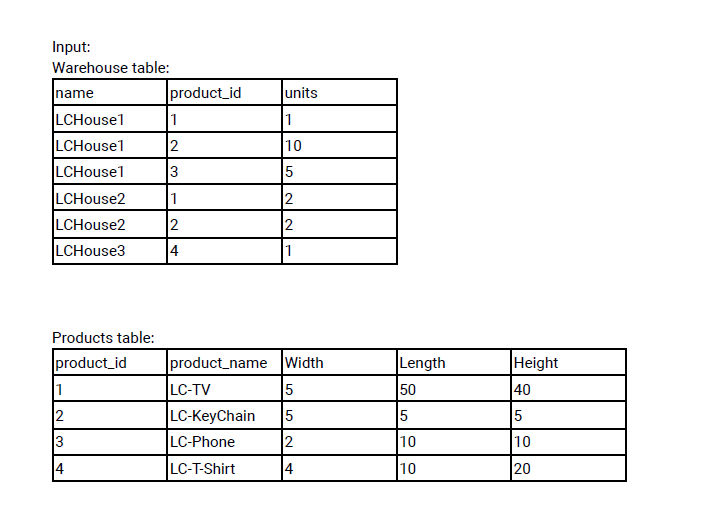
from result\_1

group by person1,person2;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 41;**





create table warehouse(

    name VARCHAR(20),

    product\_id int,

    units int,

    constraint pk PRIMARY KEY(name,product\_id)

);

insert into warehouse values('LCHouse1',1,1),('LCHouse1',2,10),('LCHouse1',3,5),

('LCHouse2',1,2),('LCHouse2',2,2),('LCHouse3',4,1);

create table products(

    product\_id int,

    product\_name varchar(20),

    Width int,

    Length int,

    Height int,

    constraint pk PRIMARY Key(product\_id)

);

insert into products values(1,'LC-TV',5,50,40),(2,'LC-KeyChain',5,5,5),

(3,'LC-Phone',2,10,10),(4,'LC-T-Shirt',4,10,20);

Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse.

***Solution:***

with cte as(select w.name,w.product\_id,(w.units\*p.Width\*p.Length\*p.Height)as volume

            from warehouse w left join products p

            on w.product\_id=p.product\_id

)

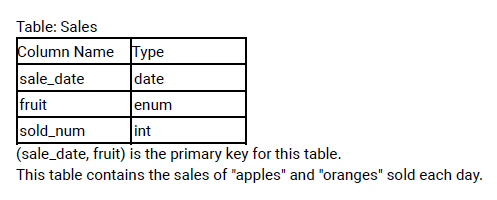
select name,sum(volume)as total\_volume from cte

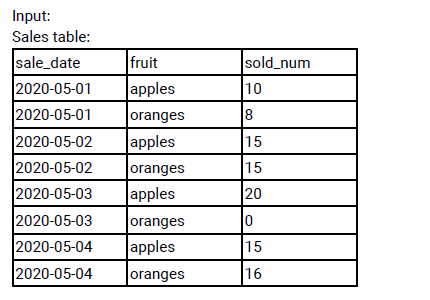
group by name;

| name varchar | | | total\_volume newdecimal | |
| --- | --- | --- | --- | --- |
|  |  | LCHouse1 | | 12250 |
|  |  | LCHouse2 | | 20250 |
|  |  | LCHouse3 | | 800 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 42;**





CREATE table sales(

    sale\_date date,

    fruit enum('apples','oranges'),

    sold\_num int,

    constraint pk PRIMARY Key(sale\_date,fruit)

);

insert into sales values('2020-05-01','apples',10),('2020-05-01','oranges',8),

('2020-05-02','apples',15),('2020-05-02','oranges',15),('2020-05-03','apples',20),

('2020-05-03','oranges',0),('2020-05-04','apples',15),('2020-05-04','oranges',16);

Write an SQL query to report the difference between the number of apples and oranges sold each day. Return the result table ordered by sale\_date.

***Solution:***

Step 1:

select s1.sale\_date,s1.sold\_num as apple\_sal,s2.sold\_num as orange\_sal,

row\_number() over(partition by s1.sale\_date)as row\_num

from sales s1 left join sales s2

on s1.sale\_date = s2.sale\_date and s1.fruit!=s2.fruit

Step 2: Find difference

with cte as(select s1.sale\_date,s1.sold\_num as apple\_sal,s2.sold\_num as orange\_sal,

            row\_number() over(partition by s1.sale\_date)as row\_num

            from sales s1 left join sales s2

            on s1.sale\_date = s2.sale\_date and s1.fruit!=s2.fruit

)

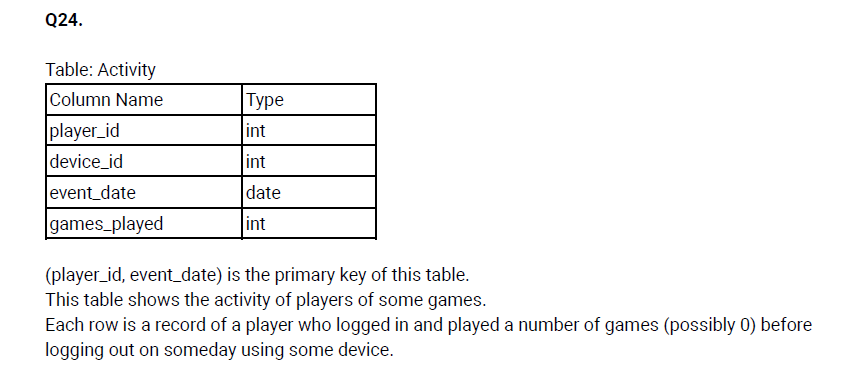
select sale\_date,(apple\_sal-orange\_sal)as diff

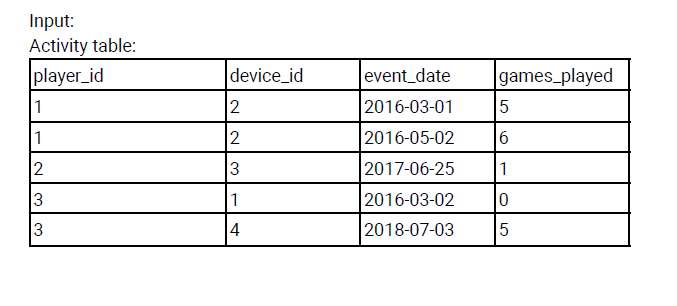
from cte where row\_num=1;

| sale\_date date | | | diff bigint | |
| --- | --- | --- | --- | --- |
|  |  | 2020-05-01 | | 2 |
|  |  | 2020-05-02 | | 0 |
|  |  | 2020-05-03 | | 20 |
|  |  | 2020-05-04 | | -1 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 43;**





create table Activity(

    player\_id int,

    device\_id int,

    event\_date date,

    games\_played int,

    constraint pk PRIMARY KEY(player\_id,event\_date)

);

insert into Activity values(1,2,'2016-03-01',5),

(1,2,'2016-05-02',6),

(2,3,'2017-06-25',1),

(3,1,'2016-03-02',0),

(3,4,'2018-07-03',5);

Write an SQL query to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that

number by the total number of players.

***Solution:***

**Step 1:** using LAG function find prev\_date

select \*,lag(event\_date) over(partition by player\_id order by event\_date)as prev\_date

from Activity

**Step 2:** using case statement if null then 0 in prev\_date

with cte as(select \*,lag(event\_date) over(partition by player\_id order by event\_date)as prev\_date

from Activity),

select player\_id,event\_date,(case when prev\_date is null then 0 else prev\_date end)as date  from cte

**Step 3:** find date diff from event\_date and prev\_date

with cte as(select \*,lag(event\_date) over(partition by player\_id order by event\_date)as prev\_date

from Activity),

result\_1 as(select player\_id,event\_date,(case when prev\_date is null then 0 else prev\_date end)as date  from cte),

select count(\*)as immediate\_login,

(select count(distinct player\_id)from Activity)

as total\_player

from (select \*,DATEDIFF(event\_date,date)as diff from result\_1 ) s

where diff in (1,2);

**Step 4:** calculate the fraction

with cte as(

    select \*,lag(event\_date) over(partition by player\_id order by event\_date)as prev\_date

     from Activity

),

result\_1 as(

    select player\_id,event\_date,(case when prev\_date is null then 0 else prev\_date end)as date

     from cte

),

result\_2 as(

select count(\*)as immediate\_login, (select count(distinct player\_id)from Activity)as total\_player

from (select \*,DATEDIFF(event\_date,date)as diff from result\_1 ) s

where diff in (1,2)

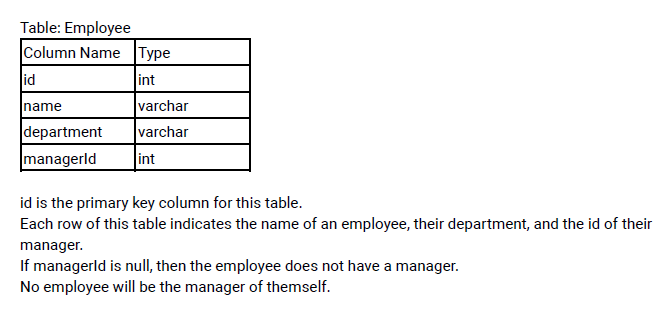
)

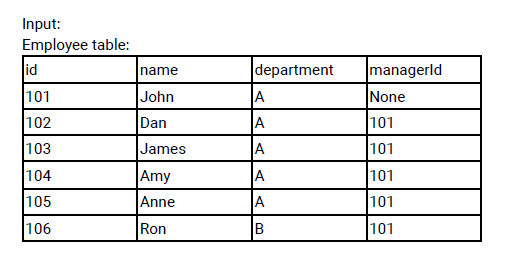
select round(immediate\_login/total\_player,2)as login\_count from result\_2;

| login\_count newdecimal | | |
| --- | --- | --- |
|  |  | 0.33 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 44;**





create table employee(

    id INT,

    name varchar(20),

    department varchar(20),

    managerId int,

    constraint pk PRIMARY KEY(id)

);

insert into employee values (101,'John','A',NULL),

(102,'Dan','A',101),(103,'James','A',101),

(104,'Amy','A',101),

(105,'Anne','A',101),

(106,'Ron','B',101);

Write an SQL query to report the managers with at least five direct reports.

***Solution:***

**Step 1:** count manager\_id

select count(managerid) as count,managerid from employee

group by managerid

**Step 2:** Find who are getting 5 direct report

with cte as(select count(managerid) as count,managerid from employee

group by managerid)

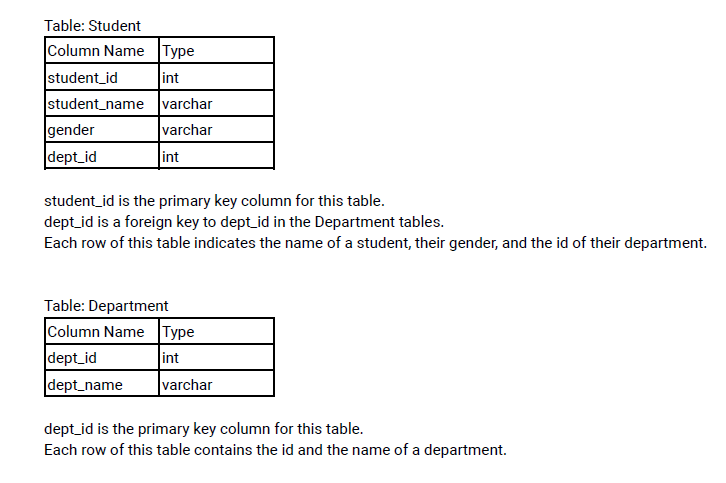
select name from employee

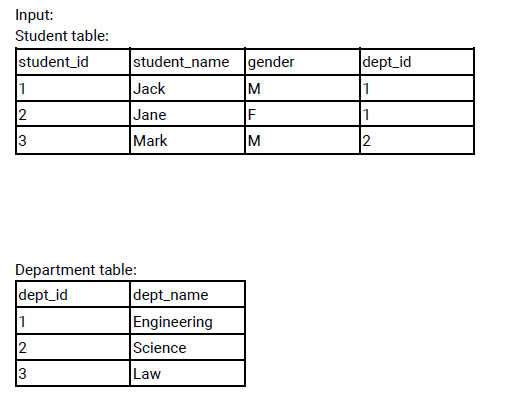
where id =  (select managerid from cte where count=5);

| name varchar | | |
| --- | --- | --- |
|  |  | John |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 45;**

****

****

create table student(

    student\_id int,

    student\_name varchar(20),

    gender varchar(10),

    dept\_id int,

    constraint pk PRIMARY KEY(student\_id,dept\_id)

);

insert into student VALUES(1,'Jack','M',1),(2,'Jane','F',1),(3,'Mark','M',2);

create table department(

    dept\_id int,

    dept\_name varchar(20),

    constraint pk PRIMARY KEY(dept\_id)

);

insert into department values(1,'Engineering'),(2,'Science'),(3,'Law');

Write an SQL query to report the respective department name and number of students majoring in each department for all departments in the Department table (even ones with no current students). Return the result table ordered by student\_number in descending order. In case of a tie, order them by

dept\_name alphabetically.

***Solution:***

**Step 1:** Find count student\_id based on dept\_name

select d.dept\_name,

           count(s.student\_id) over(partition by d.dept\_name )as stud\_num

           from department d left join student s

on s.dept\_id=d.dept\_id;

**Step 2:** group by the dept\_name, stud\_num and order by stud\_num

with cte as(select d.dept\_name,

           count(s.student\_id) over(partition by d.dept\_name )as stud\_num

           from department d left join student s

on s.dept\_id=d.dept\_id)

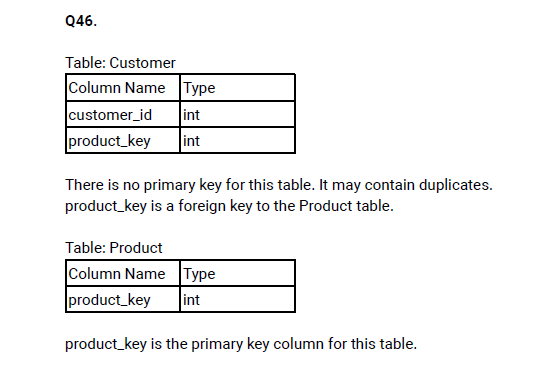
select dept\_name,stud\_num from  cte group by dept\_name,stud\_num

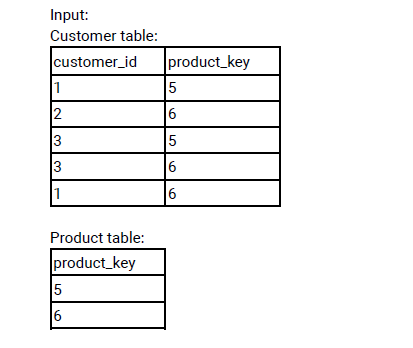
order by stud\_num desc;

| dept\_name varchar | | | stud\_num bigint | |
| --- | --- | --- | --- | --- |
|  |  | Engineering | | 2 |
|  |  | Science | | 1 |
|  |  | Law | | 0 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 46;**





create table product(

    product\_key int,

    constraint pk PRIMARY key(product\_key)

);

INSERT into product values(5),(6);

create table customer(

    customer\_id int,

    product\_key int,

    constraint fk FOREIGN KEY(product\_key) REFERENCES product(product\_key)

);

insert into customer VALUES(1,5),(2,6),(3,5),(3,6),(1,6);

Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table.

***Solution:***

**Step 1:** Merge product\_key based on customer\_id

select customer\_id,

group\_concat(distinct product\_key separator ',')as product\_key

from customer

group by customer\_id;

**Step 2:** Merge the product\_key

select group\_concat(product\_key separator ',')as product\_key

from  product

**Step 3:** find the product\_key are eqeual in customer table

with cte as(select customer\_id,

group\_concat(distinct product\_key separator ',')as product\_key

from customer

group by customer\_id

),

result\_1 as(select group\_concat(product\_key separator ',')as product\_key

from  product)

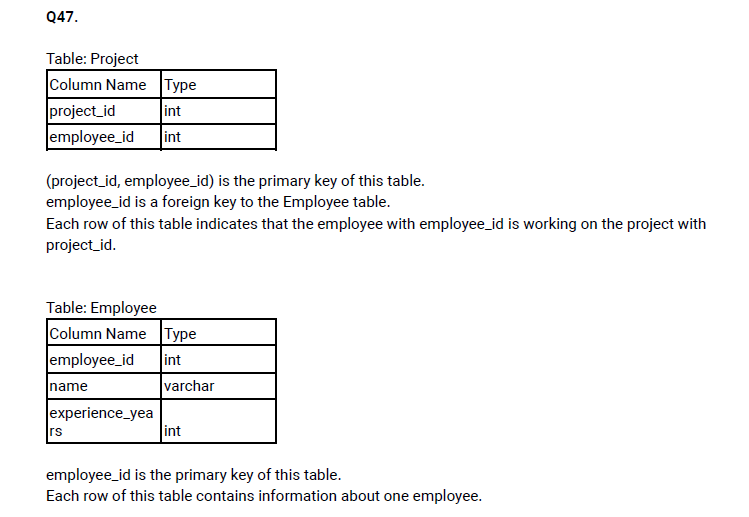
select customer\_id from cte

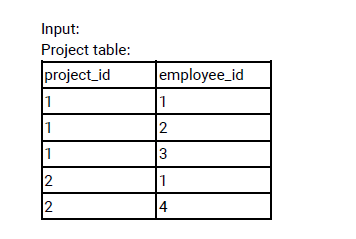
where product\_key = (select product\_key from result\_1);

| customer\_id int | | |
| --- | --- | --- |
|  |  | 1 |
|  |  | 3 |

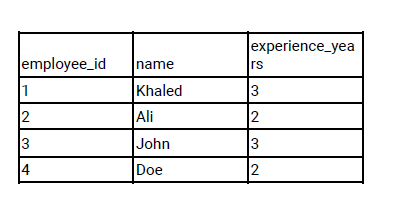
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 47;**





Employee table:



create table project(

    project\_id int,

    employee\_id int,

    constraint pk PRIMARY KEY(project\_id,employee\_id)

);

insert into project values(1 ,1),(1, 2),(1, 3),(2 ,1),(2, 4);

create table employee(

    employee\_id int,

    name varchar(20),

    experience\_years int,

    constraint pk PRIMARY KEY(employee\_id)

);

insert into employee values(1,'Khaled',3),(2,'Ali',2),(3,'John',3),(4,'Doe',2);

Write an SQL query that reports the most experienced employees in each project. In case of a tie, report all employees with the maximum number of experience years.

***Solution:***

**Step 1:** find max\_exp from those two table based on project\_id

select p.\*,e.experience\_years,

max(e.experience\_years) over(partition by p.project\_id)as max\_exp

from project p left join employee e

on p.employee\_id=e.employee\_id;

**Step 2:** filter the dataset where experience is equal to max\_exp

with cte as(select p.\*,e.experience\_years,

max(e.experience\_years) over(partition by p.project\_id)as max\_exp

from project p left join employee e

on p.employee\_id=e.employee\_id)

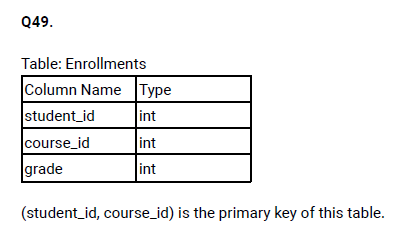
select project\_id,employee\_id from cte

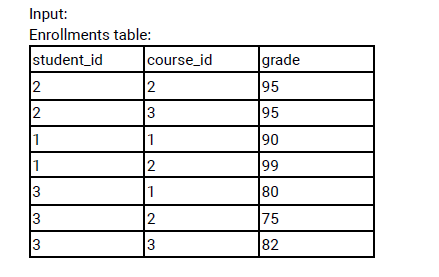
where experience\_years=max\_exp;

| project\_id int | | | employee\_id int | |
| --- | --- | --- | --- | --- |
|  |  | 1 | | 1 |
|  |  | 1 | | 3 |
|  |  | 2 | | 1 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 49;**





create table enrollments(

    student\_id int,

    course\_id int,

    grade int,

    constraint pk PRIMARY KEY(student\_id, course\_id)

);

insert into enrollments values(2 ,2, 95),(2, 3, 95),(1,1,90),(1,2,99),(3,1, 80),(3,2,75),(3,3,82);

Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest course\_id.

Return the result table ordered by student\_id in ascending order.

***Solution:***

**Step 1:** Find the max\_grade based on student\_id order by course\_id,grade desc

select student\_id,course\_id,grade,

rank() over(partition by student\_id order by course\_id,grade desc )as max\_grade

from enrollments;

**Step 2:** Filter the dataset where max\_grade=1

with cte as(select student\_id,course\_id,grade,

rank() over(partition by student\_id order by course\_id,grade desc )as max\_grade

from enrollments

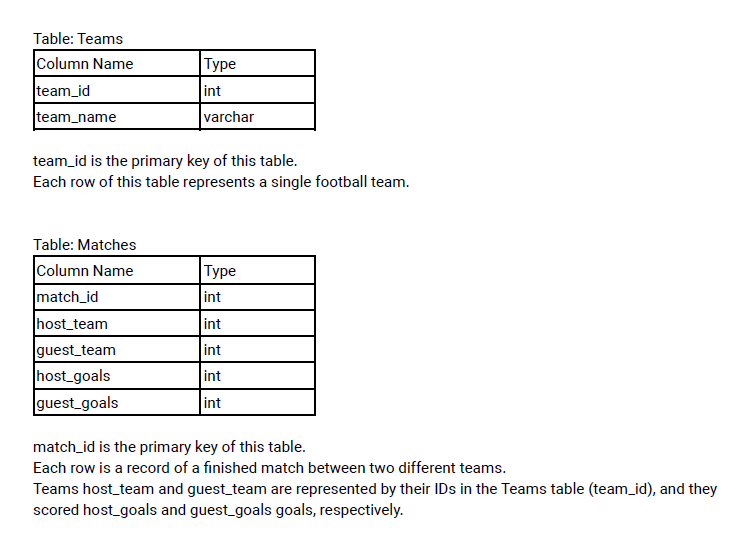
)

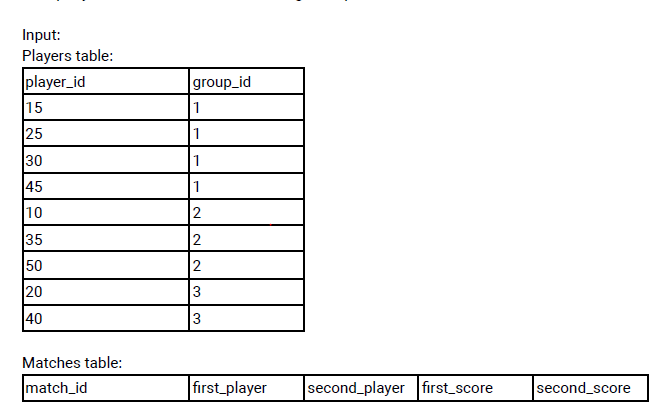
select student\_id,course\_id,grade from cte where max\_grade=1;

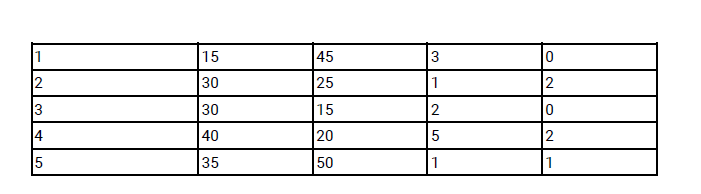
| student\_id int | | | course\_id int | | grade int |
| --- | --- | --- | --- | --- | --- |
|  | 1 |  | | 1 | 90 |
|  | 2 |  | | 2 | 95 |
|  | 3 |  | | 1 | 80 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 50;**







create table teams(

    player\_id int,

    group\_id VARCHAR(20),

    constraint pk PRIMARY KEY(player\_id)

);

insert into teams values(15 ,1),(25,1),(30,1),(45,1),

(10,2),(35,2),(50,2),(20,3),(40,3);

create table matches(

    match\_id int,

    first\_player int,

    second\_plyer int,

    first\_score int,

    second\_score int,

    constraint pk PRIMARY KEY(match\_id)

);

insert into matches values(1,15,45,3,0),(2,30,25,1,2),

(3,30,15,2,0),(4,40,20,5,2),(5,35,50,1,1);

The winner in each group is the player who scored the maximum total points within the group. In the case of a tie, the lowest player\_id wins.

Write an SQL query to find the winner in each group.

***Solution:***

**Step 1:** Join the those table and find rank of player\_id based on group\_id order by first\_player

select m.match\_id,m.first\_player,t.group\_id,

rank() over(partition by t.group\_id order by m.first\_player )as player\_id

 from matches m left join teams t

on m.first\_player=t.player\_id;

**Step 2:** filtter the data where player\_id min

with cte as (select m.match\_id,m.first\_player,t.group\_id,

rank() over(partition by t.group\_id order by m.first\_player )as player\_id

 from matches m left join teams t

on m.first\_player=t.player\_id)

SELECT group\_id,first\_player from cte where player\_id=1;

| group\_id varchar | | | first\_player int | |
| --- | --- | --- | --- | --- |
|  | 1 |  | | 15 |
|  | 2 |  | | 35 |
|  | 3 |  | | 40 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_