SQL-QUESTIONS-SET1

Sample dataset-1

ID	NAME	COUNTRYCODE	DISTRICT	POPULATION
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

The CITY table is described as follows:

CITY

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

```
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	[redacted]

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:

ID	city	state	LAT_N	LONG_W
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

The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

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

	city varchar	state varchar
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

city
varchar

3 Sandy Hook

4 Tipton

.

..

QUESTION 17;

Q17.

Table: Product

Column Name	Type
product_id	int
product_name	varchar
unit_price	int

product_id is the primary key of this table.
Each row of this table indicates the name and the price of each product.

Table: Sales

Column Name	Type
seller_id	int
product_id	int
buyer_id	int
sale_date	date
quantity	int
price	int

This table has no primary key, it can have repeated rows.
product_id is a foreign key to the Product table.
Each row of this table contains some information about one sale.

Input data:

Input:
Product table:

product_id	product_name	unit_price
1	S8	1000
2	G4	800
3	iPhone	1400

Sales table:

seller_id	product_id	buyer_id	sale_date	quantity	price
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

```
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	product_name
int	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.

Column Name	Type
article_id	int
author_id	int
viewer_id	int
view_date	date

There is no primary key for this table, it may have duplicate rows.
Each row of this table indicates that some viewer viewed an article (written by some author) on some date.
Note that equal author_id and viewer_id indicate the same person.

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.
The query result format is in the following example.

Input:
Views table:

article_id	author_id	viewer_id	view_date
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

```
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
2
4
7

QUESTION 19;

Table: Delivery

Column Name	Type
delivery_id	int
customer_id	int
order_date	date
customer_pref_delivery_date	date

delivery_id is the primary key of this table.

The table holds information about food delivery to customers that make orders at some date and specify a preferred delivery date (on the same order date or after it).

If the customer's preferred delivery date is the same as the order date, then the order is called immediately; otherwise, it is called scheduled.

Input data:

Input:
Delivery table:

delivery_id	customer_id	order_date	customer_pref_delivery_date
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

```
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;

Q20.

Table: Ads

Column Name	Type
ad_id	int
user_id	int
action	enum

(ad_id, user_id) is the primary key for this table.
Each row of this table contains the ID of an Ad, the ID of a user, and the action taken by this user regarding this Ad.
The action column is an ENUM type of ('Clicked', 'Viewed', 'Ignored').

A company is running Ads and wants to calculate the performance of each Ad.
Performance of the Ad is measured using Click-Through Rate (CTR) where:

$$CTR = \begin{cases} 0, & \text{if Ad total clicks + Ad total views = 0} \\ \frac{\text{Ad total clicks}}{\text{Ad total clicks + Ad total views}} \times 100, & \text{otherwise} \end{cases}$$

Write an SQL query to find the ctr of each Ad. Round ctr to two decimal points.
Return the result table ordered by ctr in descending order and by ad_id in ascending order in case of a tie.
The query result format is in the following example.

Input:
Ads table:

ad_id	user_id	action
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

```sql
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;

Q21.

Table: Employee

Column Name	Type
employee_id	int
team_id	int

employee_id is the primary key for this table.
Each row of this table contains the ID of each employee and their respective team.

Write an SQL query to find the team size of each of the employees.
Return result table in any order.
The query result format is in the following example.

Input:
Employee Table:

employee_id	team_id
1	8
2	8
3	8
4	7
5	9
6	9

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;

Q22.

Table: Countries

Column Name	Type
country_id	int
country_name	varchar

country_id is the primary key for this table.
Each row of this table contains the ID and the name of one country.

Table: Weather

Column Name	Type
country_id	int
weather_state	int
day	date

(country_id, day) is the primary key for this table.
Each row of this table indicates the weather state in a country for one day.

The query result format is in the following example:

Input:
Countries table:

country_id	country_name
2	USA
3	Australia
7	Peru
5	China
8	Morocco
9	Spain

Weather table:

country_id	weather_state	day
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

```
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,'S
pain');
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;

Q23.

Table: Prices

Column Name	Type
product_id	int
start_date	date
end_date	date
price	int

(product_id, start_date, end_date) is the primary key for this table.
Each row of this table indicates the price of the product_id in the period from start_date to end_date.
For each product_id there will be no two overlapping periods. That means there will be no two intersecting periods for the same product_id.

Table: UnitsSold

Column Name	Type
product_id	int
purchase_date	date
units	int

There is no primary key for this table, it may contain duplicates.
Each row of this table indicates the date, units, and product_id of each product sold.

Insert data;

Input:
Prices table:

product_id	start_date	end_date	price
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

UnitsSold table:

product_id	purchase_date	units
1	2019-02-25	100
1	2019-03-01	15
2	2019-02-10	200
2	2019-03-22	30

```
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;

Q24.

Table: Activity

Column Name	Type
player_id	int
device_id	int
event_date	date
games_played	int

(player_id, event_date) is the primary key of this table.
This table shows the activity of players of some games.
Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on someday using some device.

Input:
Activity table:

player_id	device_id	event_date	games_played
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 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

player_id int	first_login date
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;

Q26.

Table: Products

Column Name	Type
product_id	int
product_name	varchar
product_category	varchar

product_id is the primary key for this table.
This table contains data about the company's products.

Table: Orders

Column Name	Type
product_id	int
order_date	date
unit	int

There is no primary key for this table. It may have duplicate rows.
product_id is a foreign key to the Products table.
unit is the number of products ordered in order_date.

Input:
Products table:

product_id	product_name	product_category
1	Leetcode Solutions	Book
2	Jewels of Stringology	Book
3	HP	Laptop
4	Lenovo	Laptop
5	Leetcode Kit	T-shirt

Orders table:

product_id	order_date	unit
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:

```
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	unit
varchar	newdecimal
1 Leetcode Solutions	130
2 Leetcode Kit	100

QUESTION 27;

Q27.

Table: Users

Column Name	Type
user_id	int
name	varchar
mail	varchar

user_id is the primary key for this table.
This table contains information of the users signed up in a website. Some emails are invalid.

Input:
Users table:

user_id	name	mail
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

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;

Q28.

Table: Customers

Column Name	Type
customer_id	int
name	varchar
country	varchar

customer_id is the primary key for this table.
This table contains information about the customers in the company.

Table: Product

Column Name	Type
customer_id	int
name	varchar
country	varchar

product_id is the primary key for this table.
This table contains information on the products in the company.
price is the product cost.

Table: Orders

Column Name	Type
order_id	int
customer_id	int
product_id	int
order_date	date
quantity	int

order_id is the primary key for this table.
This table contains information on customer orders.
customer_id is the id of the customer who bought "quantity" products with id "product_id".
Order_date is the date in format ('YYYY-MM-DD') when the order was shipped.

Input:
Customers table:

customer_id	name	country
1	Winston	USA
2	Jonathan	Peru
3	Moustafa	Egypt

Product table:

product_id	description	price
10	LC Phone	300
20	LC T-Shirt	10
30	LC Book	45
40	LC Keychain	2

Orders table:

order_id	customer_id	product_id	order_date	quantity
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

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	name
int	varchar
1	Winston

QUESTION 29;

Q29.

Table: TVProgram

Column Name	Type
program_date	date
content_id	int
channel	varchar

(program_date, content_id) is the primary key for this table.
This table contains information about the programs on the TV.
content_id is the id of the program in some channel on the TV.

Table: Content

Column Name	Type
content_id	varchar
title	varchar
Kids_content	enum
content_type	varchar

content_id is the primary key for this table.
Kids_content is an enum that takes one of the values ('Y', 'N') where:

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

Input:
TVProgram table:

program_date	content_id	channel
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

Content table:

content_id	title	Kids_content	content_type
1	Leetcode Movie	N	Movies
2	Alg. for Kids	Y	Series
3	Database Sols	N	Series
4	Aladdin	Y	Movies
5	Cinderella	Y	Movies

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;

Table: NPV

Column Name	Type
id	int
year	int
npv	int

(id, year) is the primary key of this table.
The table has information about the id and the year of each inventory and the corresponding net present value.

Table: Queries

Column Name	Type
id	int
year	int

(id, year) is the primary key of this table.
The table has information about the id and the year of each inventory query.

Input:
NPV table:

id	year	npv
1	2018	100
7	2020	30
13	2019	40
1	2019	113
2	2008	121
3	2009	12
11	2020	99
7	2019	0

Queries table;

id	year
1	2019
2	2008
3	2009
7	2018
7	2019
7	2020
13	2019

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:

```sql
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;

Q32.

Table: Employees

Column Name	Type
id	int
name	varchar

id is the primary key for this table.
Each row of this table contains the id and the name of an employee in a company.

Table: EmployeeUNI

Column Name	Type
id	int
unique_id	int

(id, unique_id) is the primary key for this table.
Each row of this table contains the id and the corresponding unique id of an employee in the company.

Input:
Employees table:

id	name
1	Alice
7	Bob
11	Meir
90	Winston
3	Jonathan

EmployeeUNI table:

id	unique_id
3	1
11	2
90	3

```
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:

```
```

```sql
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;

Q33.

Table: Users

Column Name	Type
id	int
name	varchar

id is the primary key for this table.
name is the name of the user.

Table: Rides

Column Name	Type
id	int
user_id	int
distance	int

id is the primary key for this table.
user_id is the id of the user who travelled the distance "distance".

Input:
Users table:

id	name
1	Alice
2	Bob
3	Alex
4	Donald
7	Lee
13	Jonathan
19	Elvis

Rides table:

id	user_id	distance
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

```
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');
```

```sql
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:

```sql
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	distance
varchar	newdecimal
Lee	450
Elvis	450
Bob	317
Jonathan	312
Alex	222
Alice	120

name distance
varchar newdecimal

Donald 0

QUESTION 34;

Table: Products

Column Name	Type
product_id	int
product_name	varchar
product_category	varchar

product_id is the primary key for this table.
This table contains data about the company's products.

Table: Orders

Column Name	Type
product_id	int
order_date	date
unit	int

There is no primary key for this table. It may have duplicate rows.
product_id is a foreign key to the Products table.
unit is the number of products ordered in order_date.

Input:
Products table:

product_id	product_name	product_category
1	Leetcode Solutions	Book
2	Jewels of Stringology	Book
3	HP	Laptop
4	Lenovo	Laptop
5	Leetcode Kit	T-shirt

```
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	unit
varchar	newdecimal
Leetcode Solutions	130
Leetcode Kit	100

QUESTION 35;

Table: Movies

Column Name	Type
movie_id	int
title	varchar

movie_id is the primary key for this table.
The title is the name of the movie.

Table: Users

Column Name	Type
user_id	int
name	varchar

user_id is the primary key for this table.

Table: MovieRating

Column Name	Type
movie_id	int
user_id	int
rating	int
created_at	date

(movie_id, user_id) is the primary key for this table.
This table contains the rating of a movie by a user in their review.
created_at is the user's review date

Movies table:

movie_id	title
1	Avengers
2	Frozen 2
3	Joker

Users table:

user_id	name
1	Daniel
2	Monica
3	Maria
4	James

MovieRating table:

movie_id	user_id	rating	created_at
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

```
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;

Input:

Departments table:

id	name
1	Electrical Engineering
7	Computer Engineering
13	Business Administration

Students table:

id	name	department_id
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

```
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,'J
asmine',77),
(3,'Steve',74),(6,'Luis',1),(8,'Jonathan',7),(7,'Daiana',33),(11,'Made
lynn',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	name
int	varchar
2	John
3	Steve
4	Jasmine
7	Daiana

QUESTION 39;

Q39.

Table: Calls

Column Name	Type
from_id	int
to_id	int
duration	int

This table does not have a primary key, it may contain duplicates.
This table contains the duration of a phone call between from_id and to_id.
from_id != to_id

Input:
Calls table:

from_id	to_id	duration
1	2	59
2	1	11
1	3	20
3	4	100
3	4	200
3	4	200
4	3	499

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;

Q41.

Table: Warehouse

Column Name	Type
name	varchar
product_id	int
units	int

(name, product_id) is the primary key for this table.
Each row of this table contains the information of the products in each warehouse.

Table: Products

Column Name	Type
product_id	int
product_name	varchar
Width	int
Length	int
Height	int

product_id is the primary key for this table.
Each row of this table contains information about the product dimensions (Width, Length, and Height) in feets of each product.

Input:
Warehouse table:

name	product_id	units
LCHouse1	1	1
LCHouse1	2	10
LCHouse1	3	5
LCHouse2	1	2
LCHouse2	2	2
LCHouse3	4	1

Products table:

product_id	product_name	Width	Length	Height
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

```sql
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	total_volume
varchar	newdecimal
LCHouse1	12250
LCHouse2	20250
LCHouse3	800

QUESTION 42;

Table: Sales

Column Name	Type
sale_date	date
fruit	enum
sold_num	int

(sale_date, fruit) is the primary key for this table.
This table contains the sales of "apples" and "oranges" sold each day.

Input:
Sales table:

sale_date	fruit	sold_num
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

```
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	diff
date	bigint
2020-05-01	2
2020-05-02	0
2020-05-03	20
2020-05-04	-1

QUESTION 43;

Q24.

Table: Activity

Column Name	Type
player_id	int
device_id	int
event_date	date
games_played	int

(player_id, event_date) is the primary key of this table.
This table shows the activity of players of some games.
Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on someday using some device.

Input:
Activity table:

player_id	device_id	event_date	games_played
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

```
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;

Table: Employee

Column Name	Type
id	int
name	varchar
department	varchar
managerId	int

id is the primary key column for this table.
Each row of this table indicates the name of an employee, their department, and the id of their manager.
If managerId is null, then the employee does not have a manager.
No employee will be the manager of themself.

Input:
Employee table:

id	name	department	managerId
101	John	A	None
102	Dan	A	101
103	James	A	101
104	Amy	A	101
105	Anne	A	101
106	Ron	B	101

```
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;

Table: Student

Column Name	Type
student_id	int
student_name	varchar
gender	varchar
dept_id	int

student_id is the primary key column for this table.
dept_id is a foreign key to dept_id in the Department tables.
Each row of this table indicates the name of a student, their gender, and the id of their department.

Table: Department

Column Name	Type
dept_id	int
dept_name	varchar

dept_id is the primary key column for this table.
Each row of this table contains the id and the name of a department.

Input:
Student table:

student_id	student_name	gender	dept_id
1	Jack	M	1
2	Jane	F	1
3	Mark	M	2

Department table:

dept_id	dept_name
1	Engineering
2	Science
3	Law

```
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;

Q46.

Table: Customer

Column Name	Type
customer_id	int
product_key	int

There is no primary key for this table. It may contain duplicates.
product_key is a foreign key to the Product table.

Table: Product

Column Name	Type
product_key	int

product_key is the primary key column for this table.

Input:
Customer table:

customer_id	product_key
1	5
2	6
3	5
3	6
1	6

Product table:

product_key
5
6

```
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;

Q47.

Table: Project

Column Name	Type
project_id	int
employee_id	int

(project_id, employee_id) is the primary key of this table.
employee_id is a foreign key to the Employee table.
Each row of this table indicates that the employee with employee_id is working on the project with project_id.

Table: Employee

Column Name	Type
employee_id	int
name	varchar
experience_years	int

employee_id is the primary key of this table.
Each row of this table contains information about one employee.

Input:
Project table:

project_id	employee_id
1	1
1	2
1	3
2	1
2	4

Employee table:

employee_id	name	experience_years
1	Khaled	3
2	Ali	2
3	John	3
4	Doe	2

```
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;

Q49.

Table: Enrollments

Column Name	Type
student_id	int
course_id	int
grade	int

(student_id, course_id) is the primary key of this table.

Input:
Enrollments table:

student_id	course_id	grade
2	2	95
2	3	95
1	1	90
1	2	99
3	1	80
3	2	75
3	3	82

```
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;

Table: Teams

Column Name	Type
team_id	int
team_name	varchar

team_id is the primary key of this table.
Each row of this table represents a single football team.

Table: Matches

Column Name	Type
match_id	int
host_team	int
guest_team	int
host_goals	int
guest_goals	int

match_id is the primary key of this table.
Each row is a record of a finished match between two different teams.
Teams host_team and guest_team are represented by their IDs in the Teams table (team_id), and they scored host_goals and guest_goals goals, respectively.

Input:
Players table:

player_id	group_id
15	1
25	1
30	1
45	1
10	2
35	2
50	2
20	3
40	3

Matches table:

match_id	first_player	second_player	first_score	second_score

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

```
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

```sql
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

```sql
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
