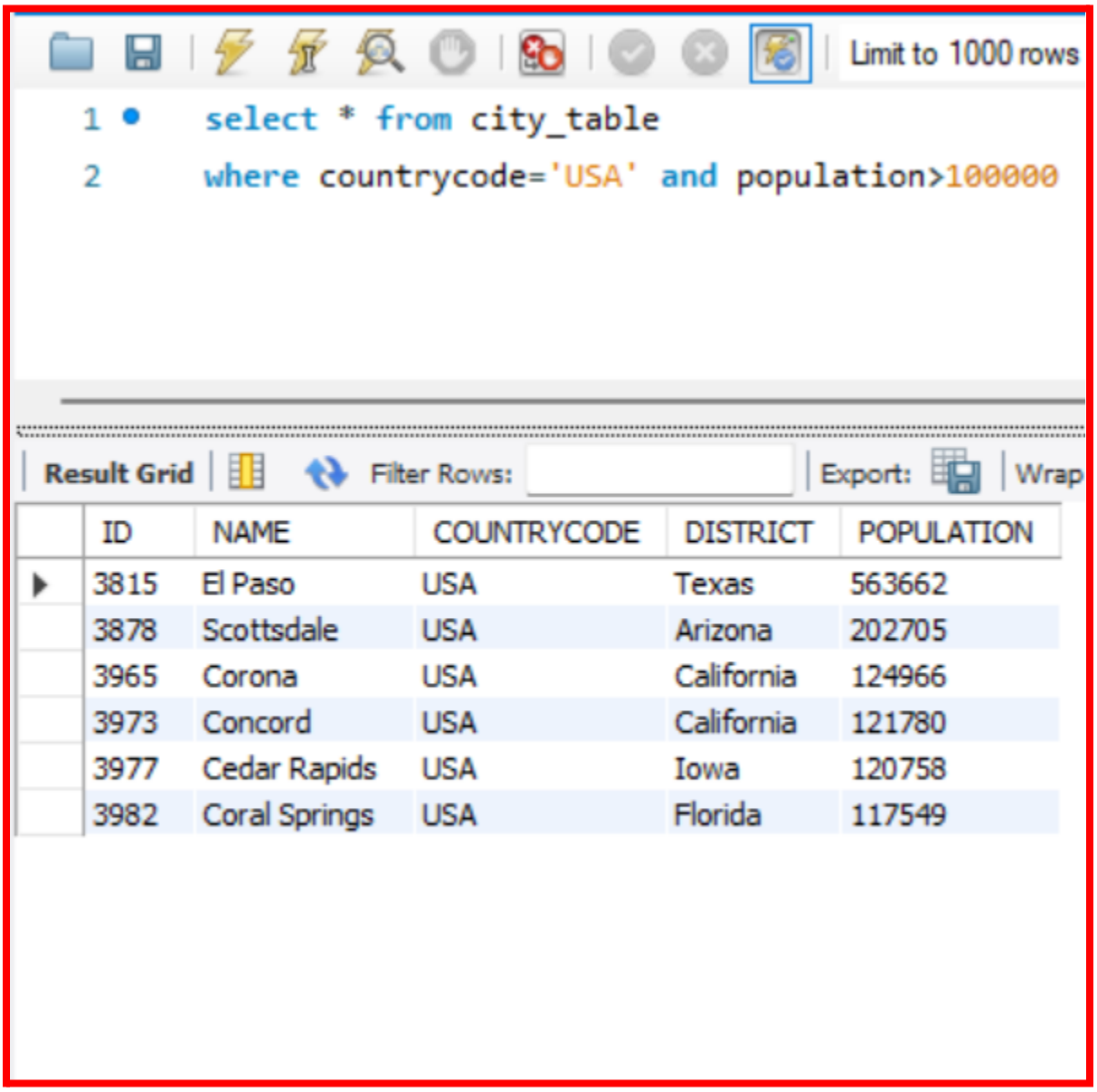
Q1. Query all columns for all American cities in the CITY table with populations larger than 100000.

The CountryCode for America is USA

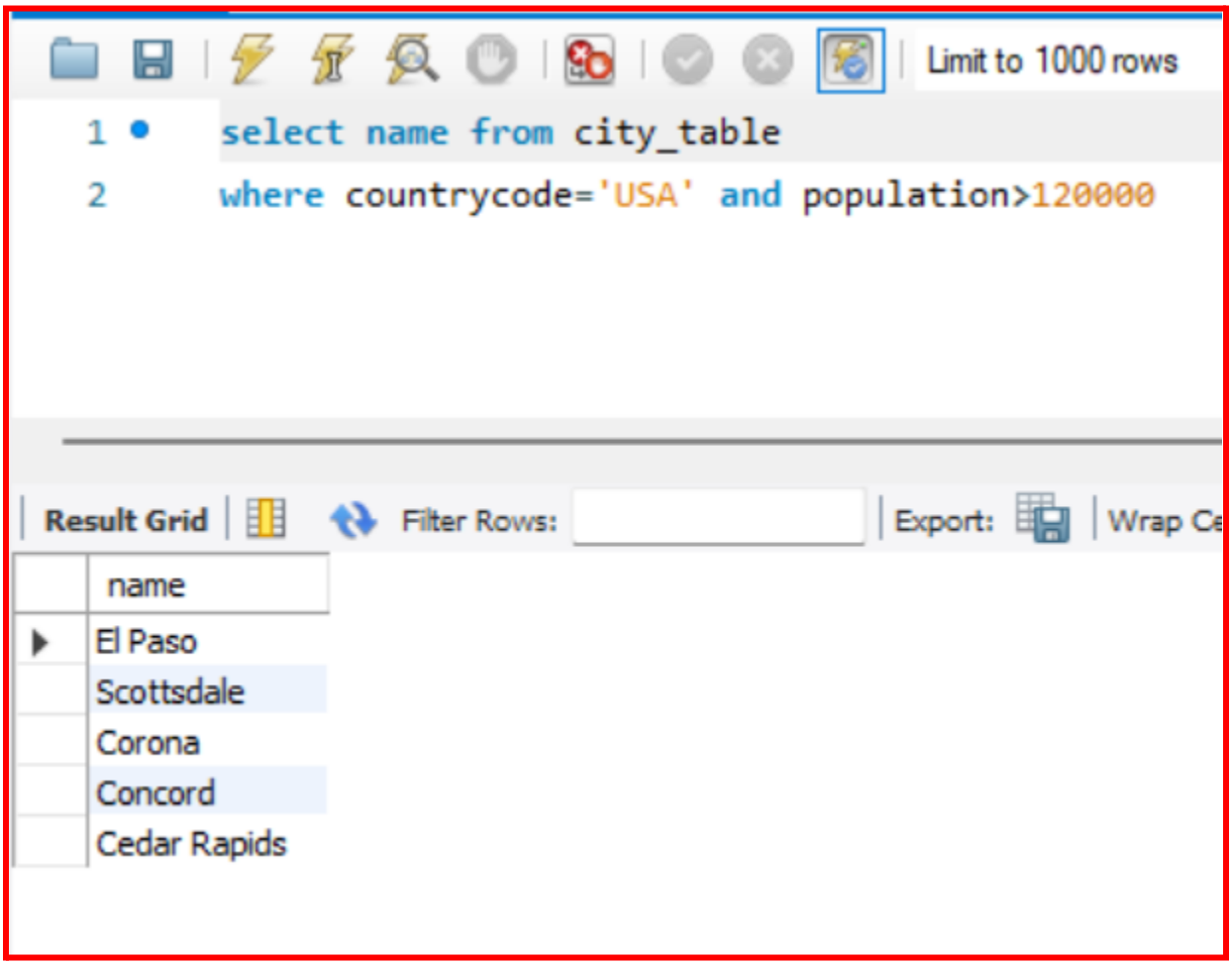
City-Dataset:

https://docs.google.com/spreadsheets/d/1dk9kRwcMxj5USuJqxtlTD05S-aOUD6fzNzVW41dcpgc/edit?usp=sharing

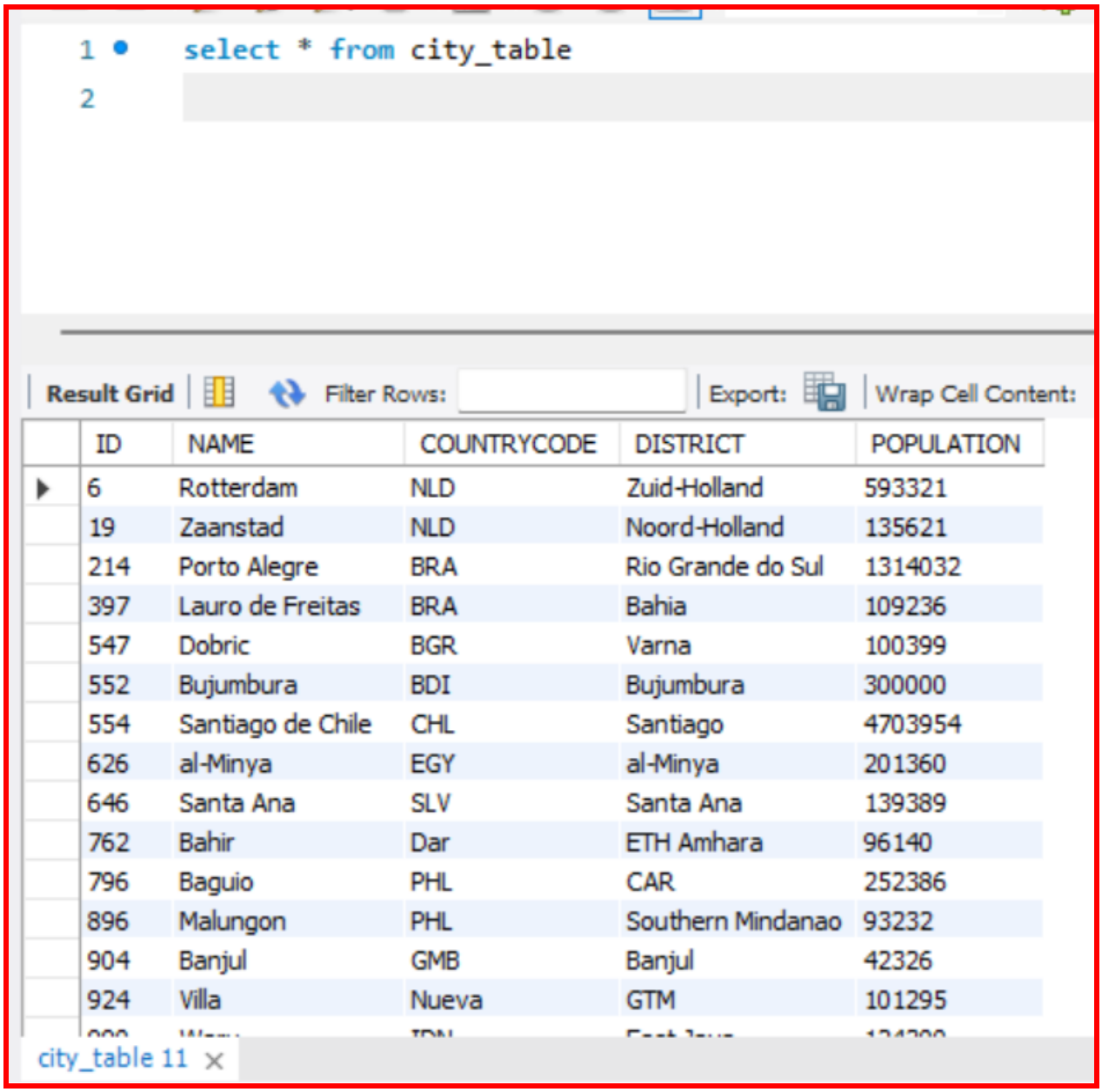


ID	NAME	COUNTRYCODE	DISTRICT	POPULATION
3815	El Paso	USA	Texas	563662
3878	Scottsdale	USA	Arizona	202705
3965	Corona	USA	California	124966
3973	Concord	USA	California	121780
3977	Cedar Rapids	USA	Iowa	120758
3982	Coral Springs	USA	Florida	117549

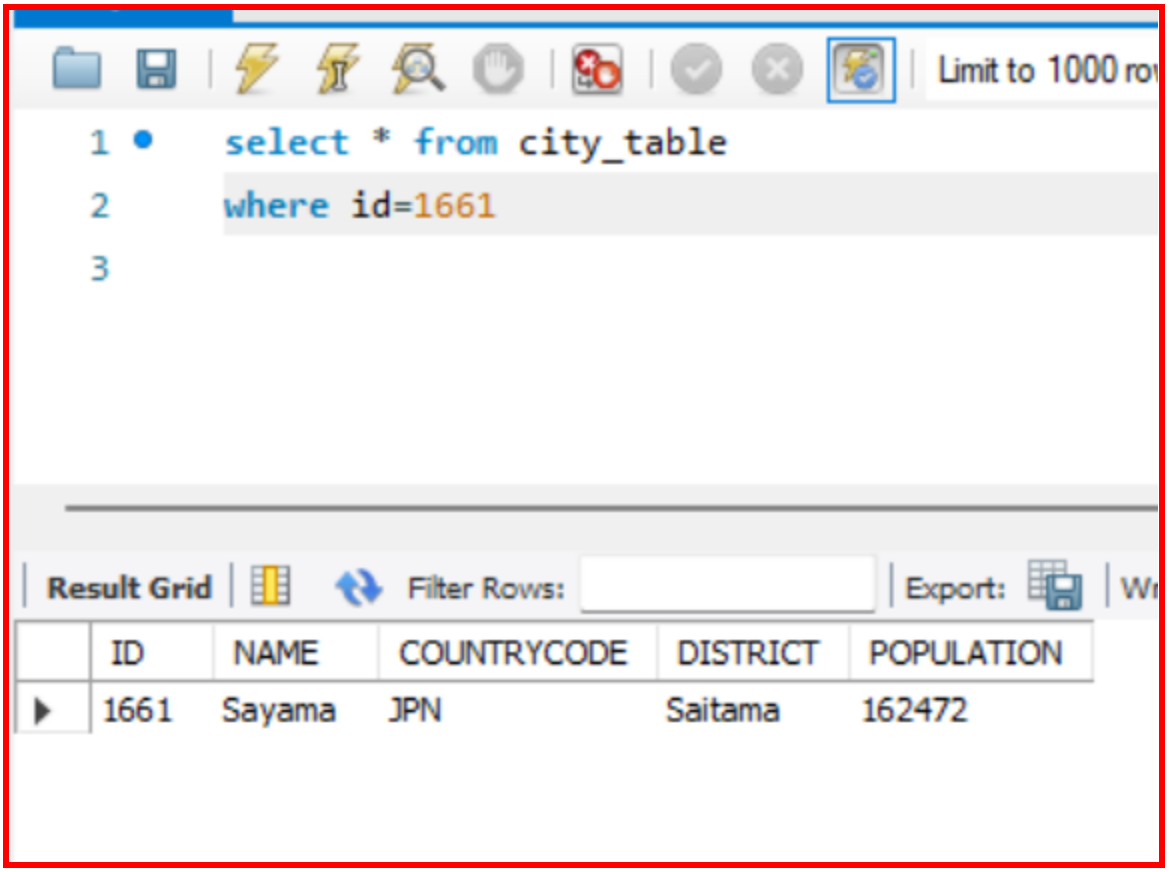
Q2. Query the NAME field for all American cities in the CITY table with populations larger than 120000. The CountryCode for America is USA.



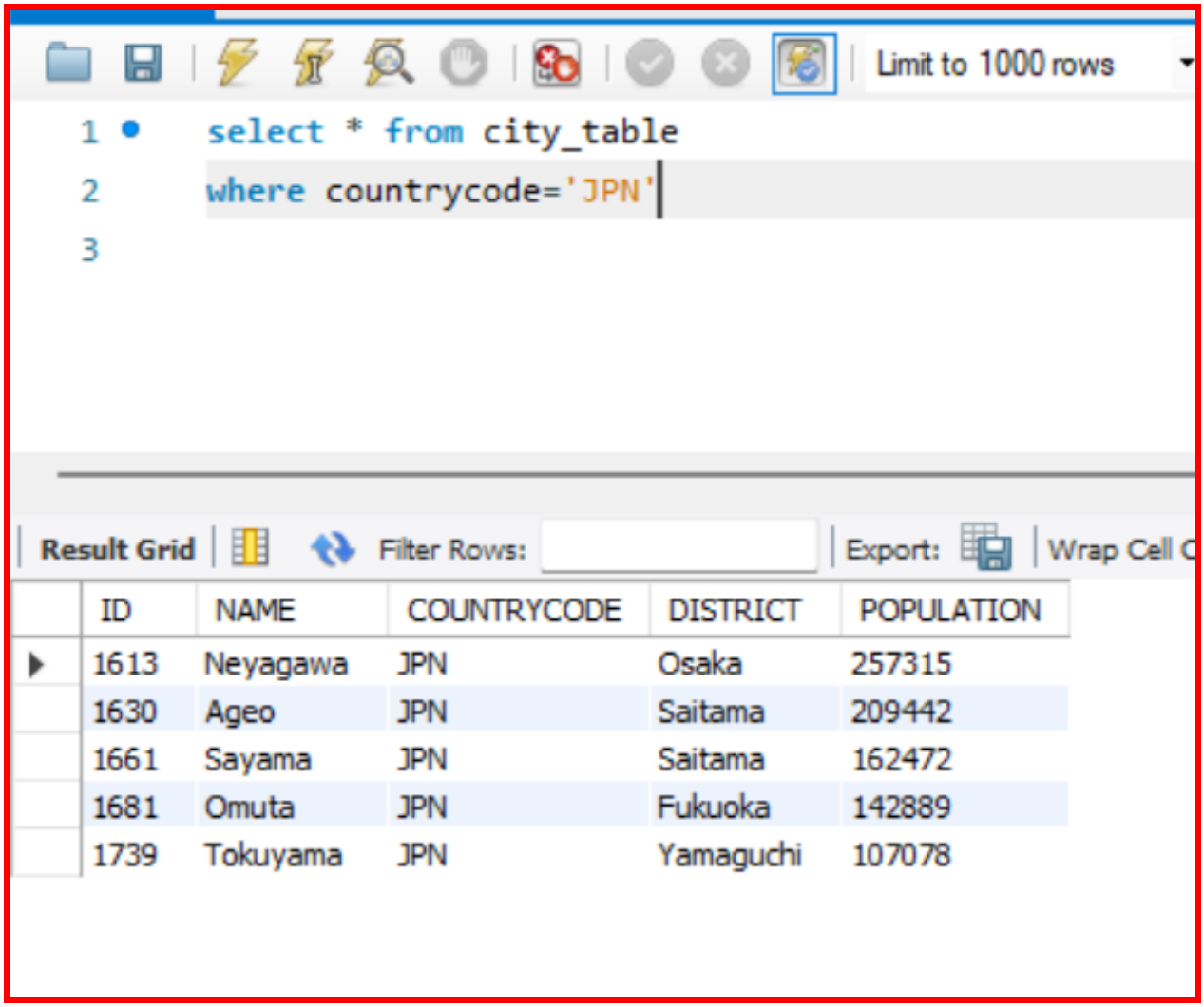
Q3. Query all columns (attributes) for every row in the CITY table.



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

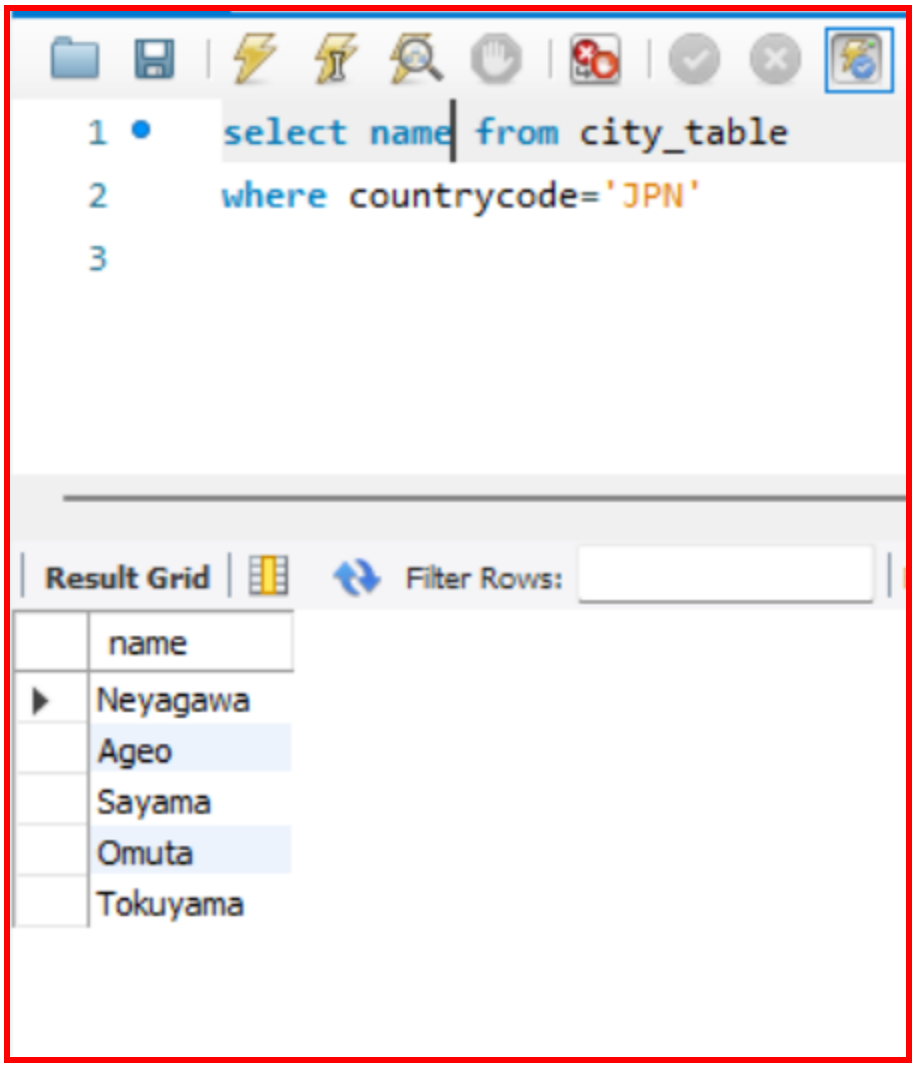


Q5. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.



ID	NAME	COUNTRYCODE	DISTRICT	POPULATION
1613	Neyagawa	JPN	Osaka	257315
1630	Ageo	JPN	Saitama	209442
1661	Sayama	JPN	Saitama	162472
1681	Omuta	JPN	Fukuoka	142889
1739	Tokuyama	JPN	Yamaguchi	107078

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



Station-table:

https://docs.google.com/spreadsheets/d/1sHPhE7walQD5mL7ppFNqybyoOJY3E51N0cWYzhp2UH4/edit?usp=sharing

Q7. Query a list of CITY and STATE from the STATION table. The STATION table is described as follows:

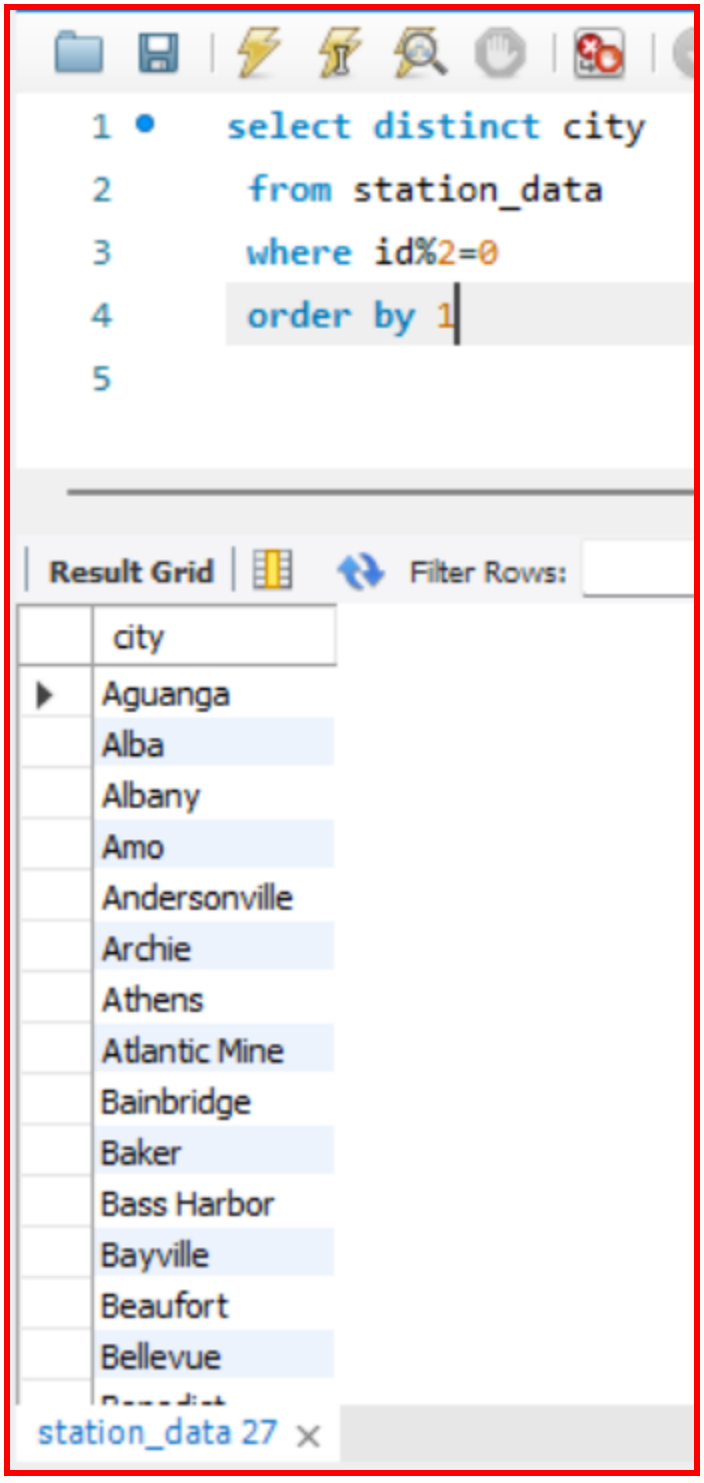
where LAT_N is the northern latitude and LONG_W is the western longitude.

```
1 •  select city,state from station_data
2
```

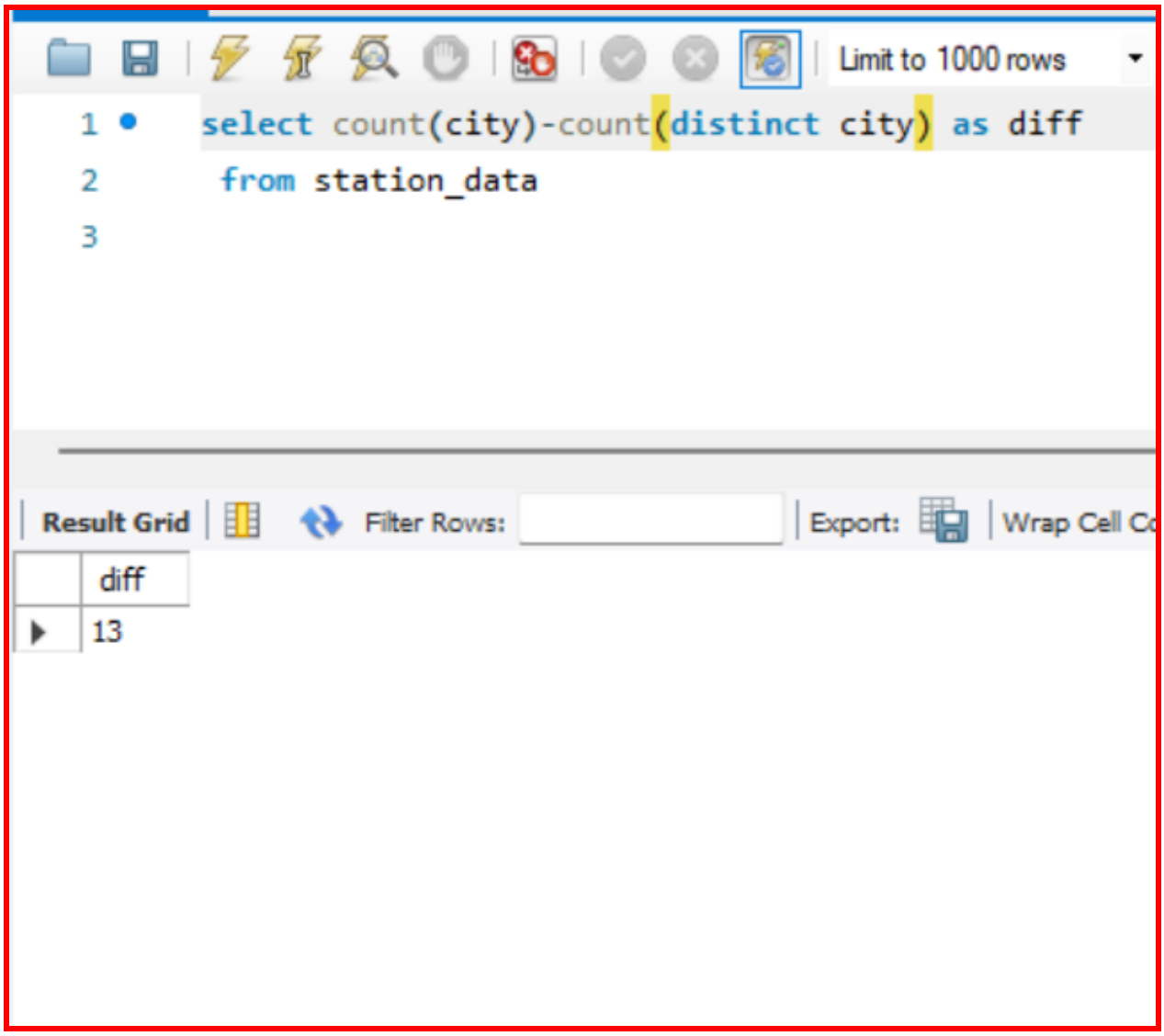
Result Grid | Filter Rows: | Expor

city	state
Kissee Mills	MO
Loma Mar	CA
Sandy Hook	CT
Tipton	IN
Arlington	CO
Turner	AR
Slidell	LA
Negreet	LA
Glencoe	KY
Chelsea	IA
Chignik Lag...	AK
Pelahatchie	MS
Hanna City	IL
Dorrance	KS

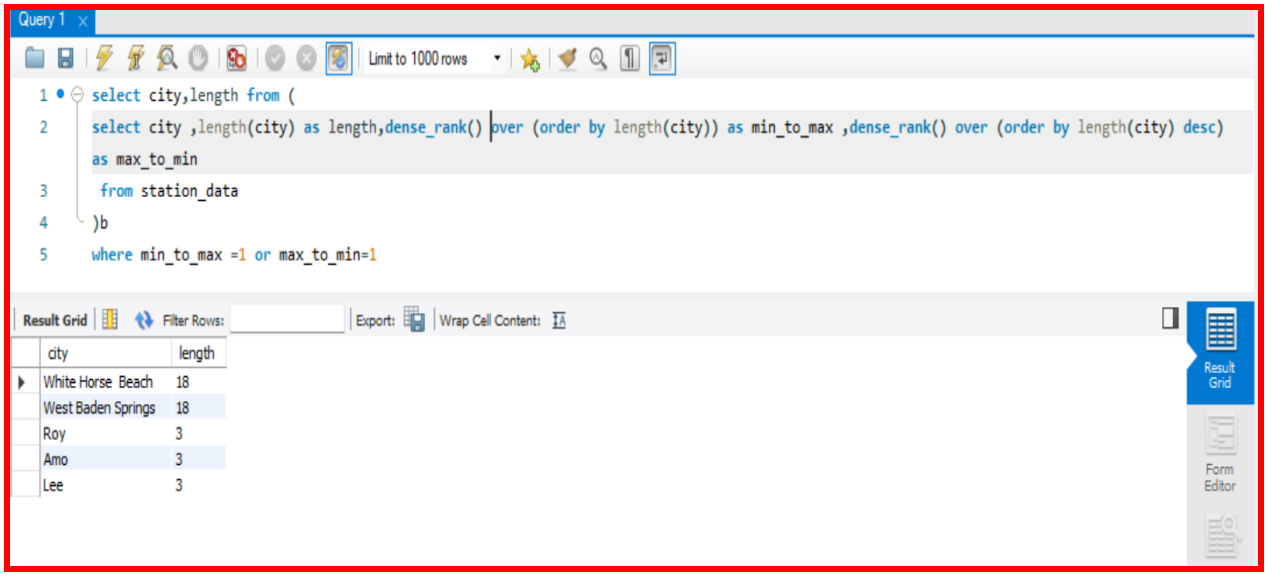
Q8. Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.



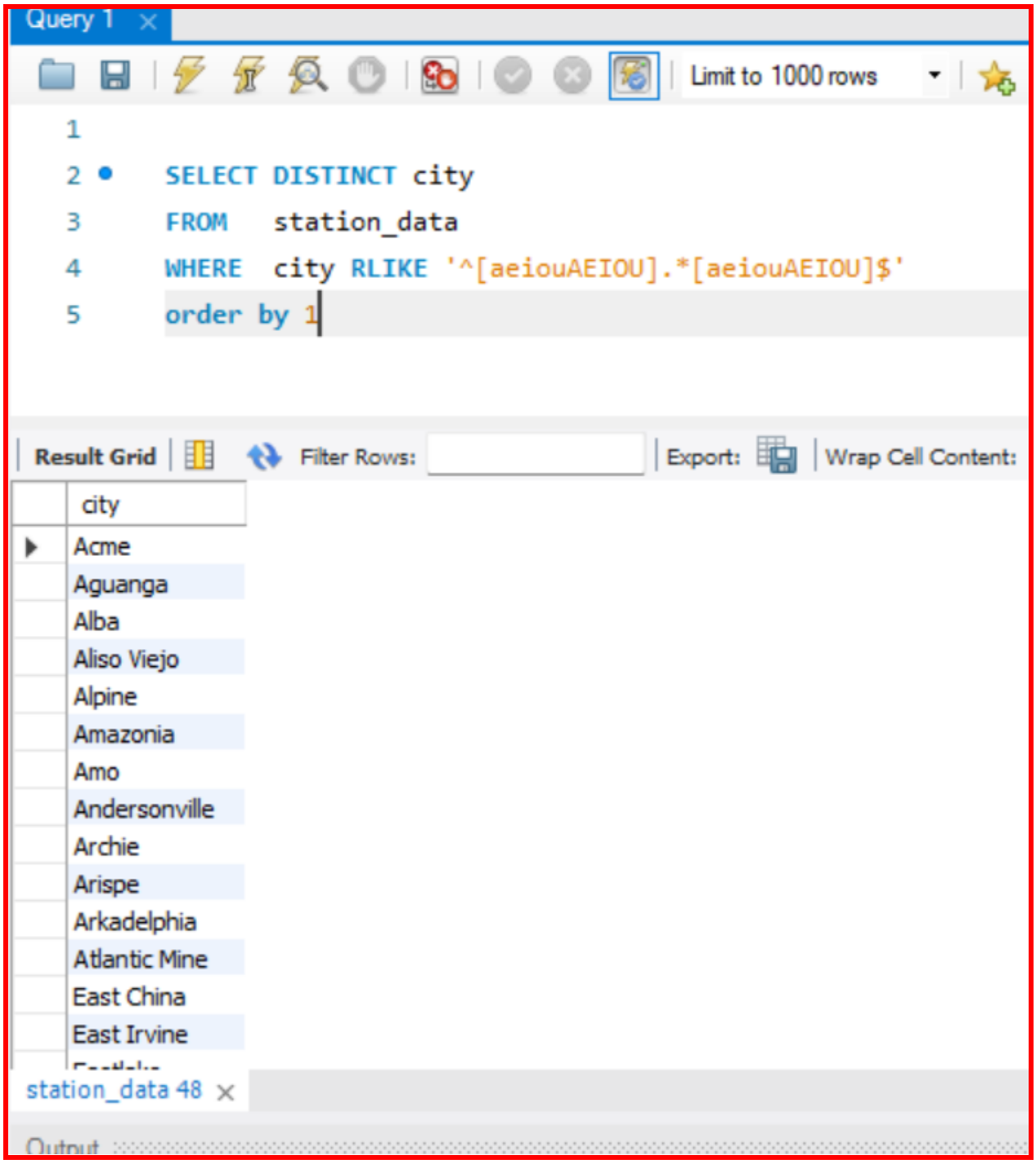
Q9. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.



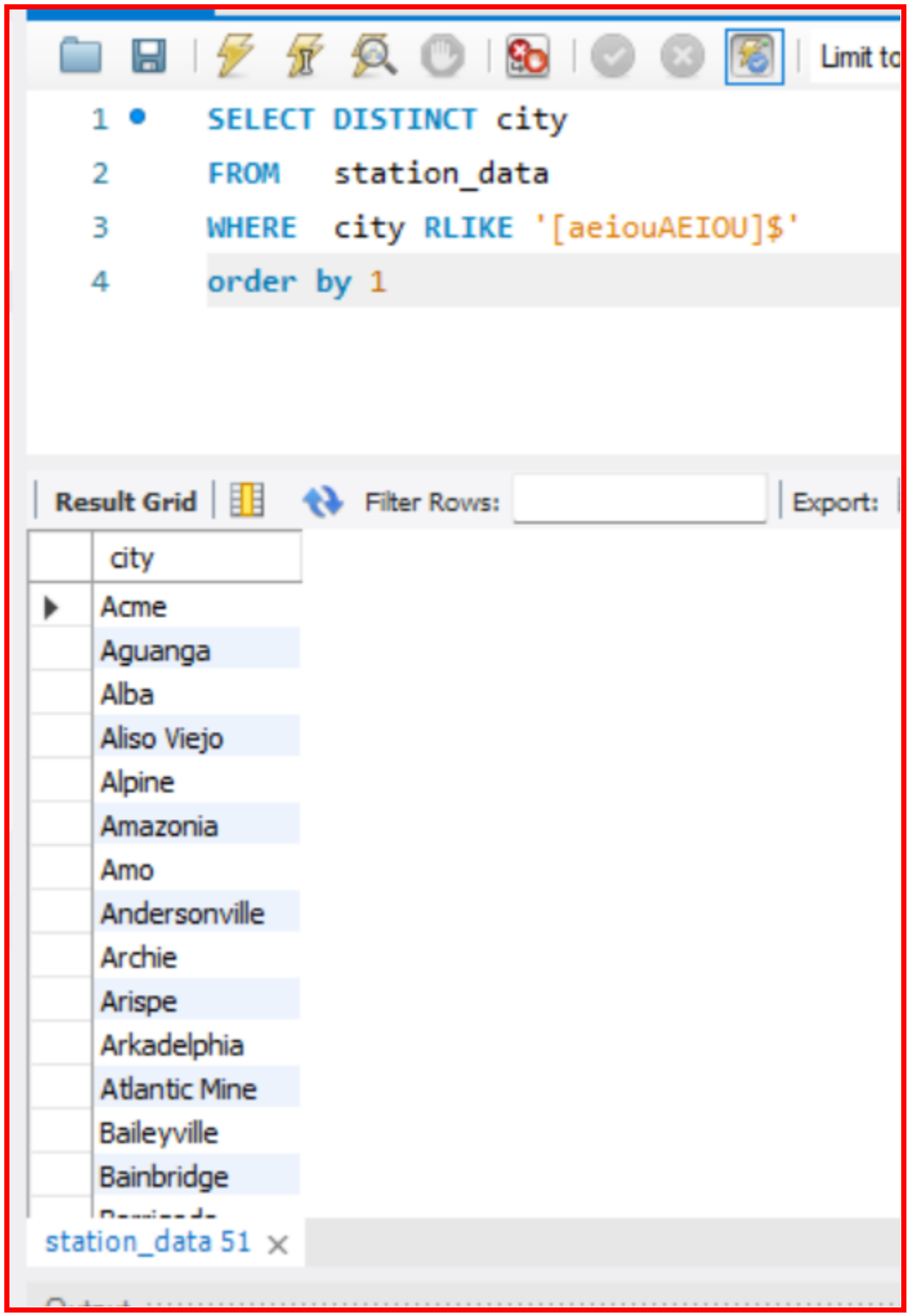
Q10. Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically



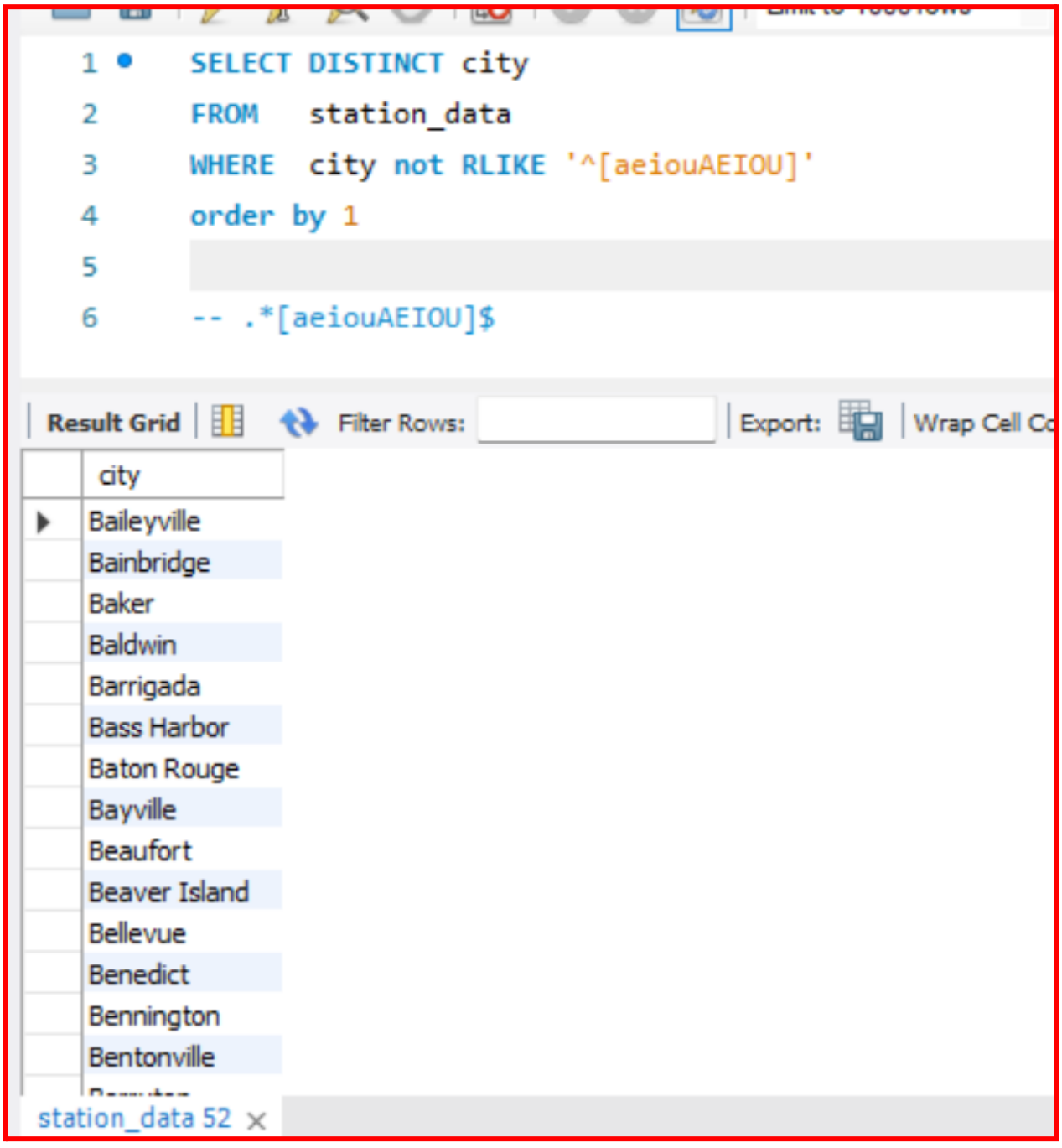
Q11. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.



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


Limit to
1 SELECT DISTINCT city
2 FROM station_data
3 WHERE city RLIKE '[aeiouAEIOU]$'
4 order by 1
Result Grid
Filter Rows:
Export:
city
Acme
Aguanga
Alba
Aliso Viejo
Alpine
Amazonia
Amo
Andersonville
Archie
Arispe
Arkadelphia
Atlantic Mine
Baileyville
Bainbridge
station_data 51

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



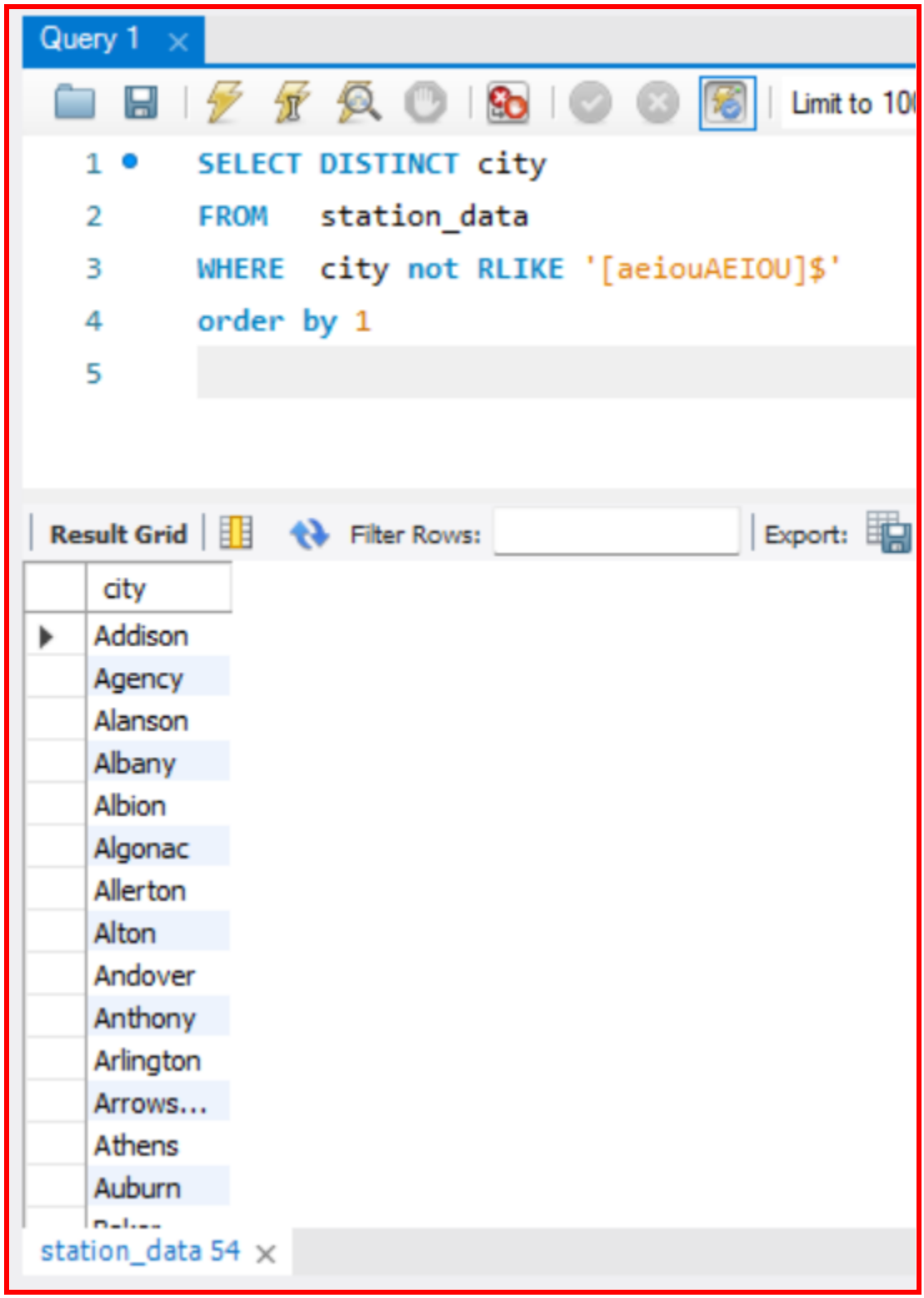
```
1  SELECT DISTINCT city
2  FROM   station_data
3  WHERE  city not RLIKE '^[aeiouAEIOU]'
4  order by 1
5
6  -- .*[aeiouAEIOU]$
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

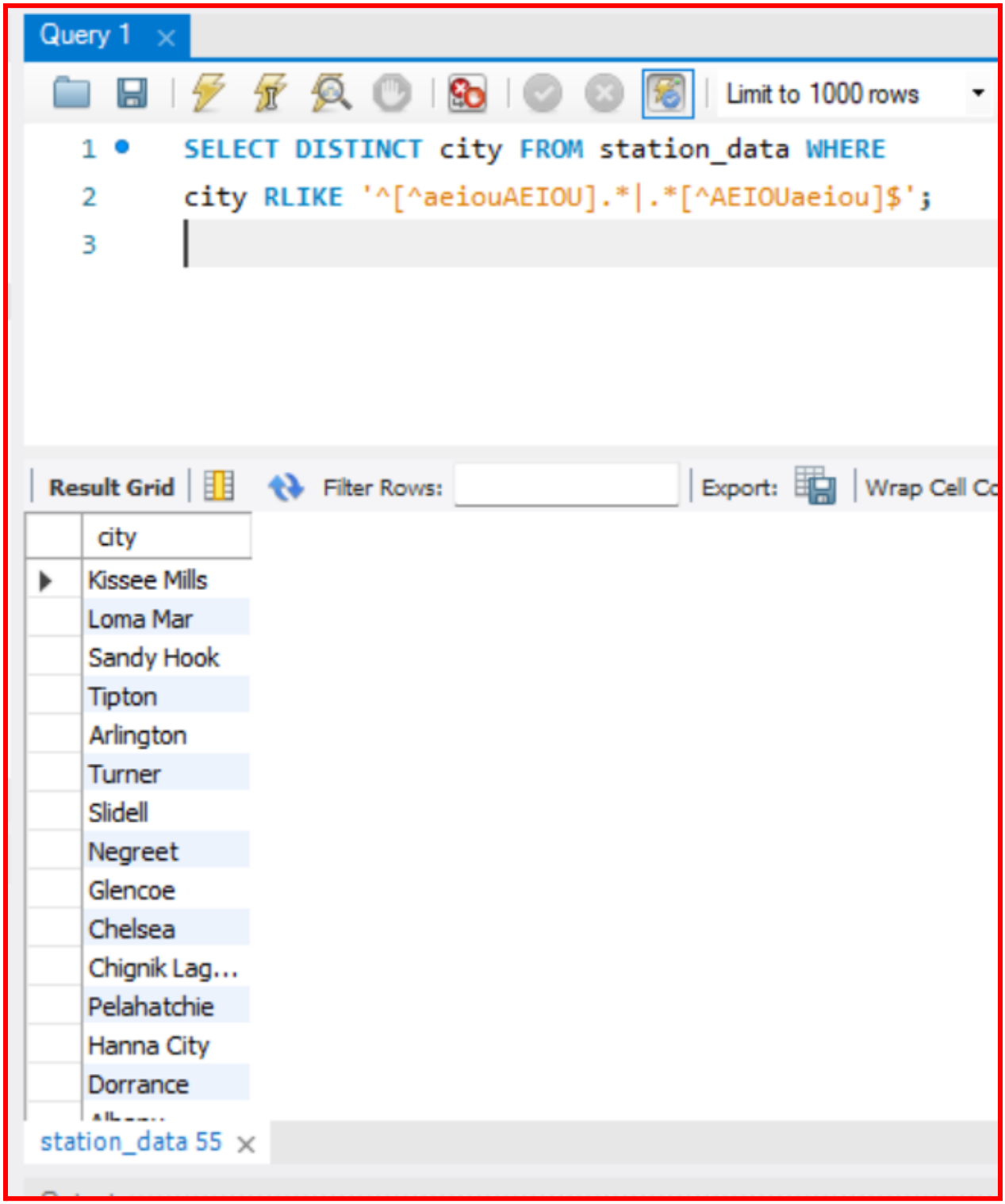
city
Baileyville
Bainbridge
Baker
Baldwin
Barrigada
Bass Harbor
Baton Rouge
Bayville
Beaufort
Beaver Island
Bellevue
Benedict
Bennington
Bentonville

station_data 52

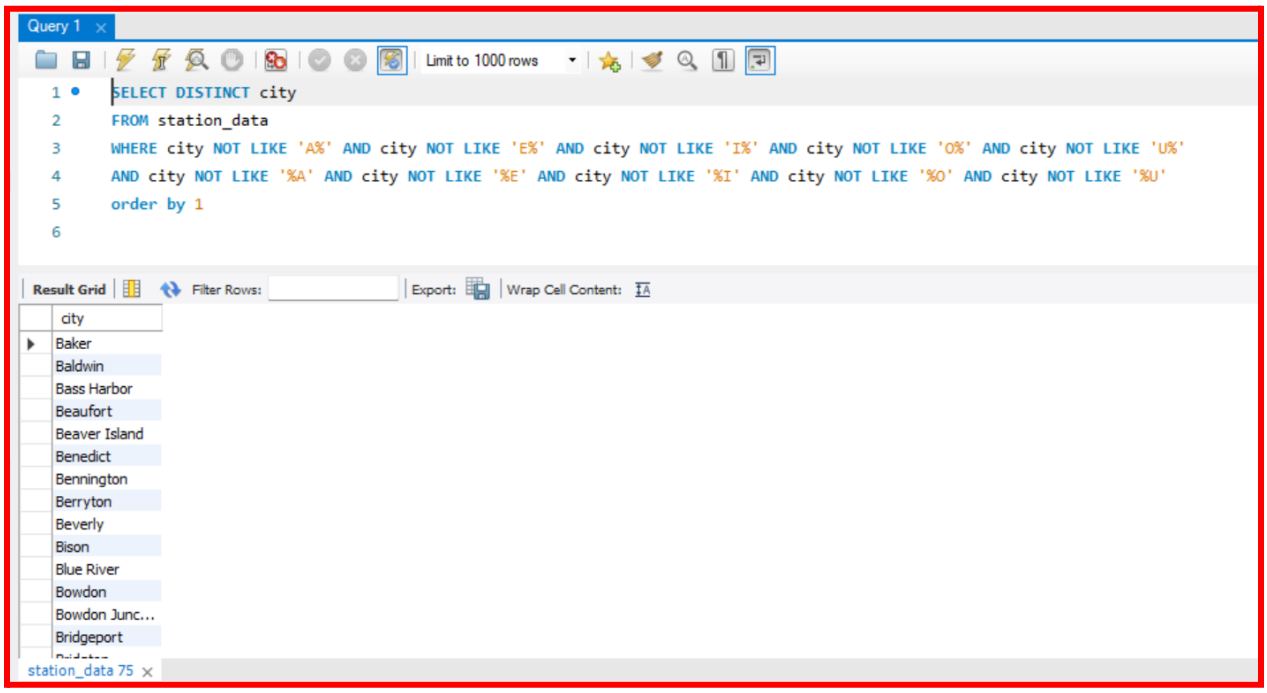
Q14. Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.


Query 1
Limit to 10
1 SELECT DISTINCT city
2 FROM station_data
3 WHERE city not RLIKE '[aeiouAEIOU]$'
4 order by 1
5
Result Grid
Filter Rows:
Export:
city
Addison
Agency
Alanson
Albany
Albion
Algonac
Allerton
Alton
Andover
Anthony
Arlington
Arrows...
Athens
Auburn
station_data 54

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.



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.



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.

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.
Return the result table in any order.
The query result format is in the following example.

Input:
Product table:

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

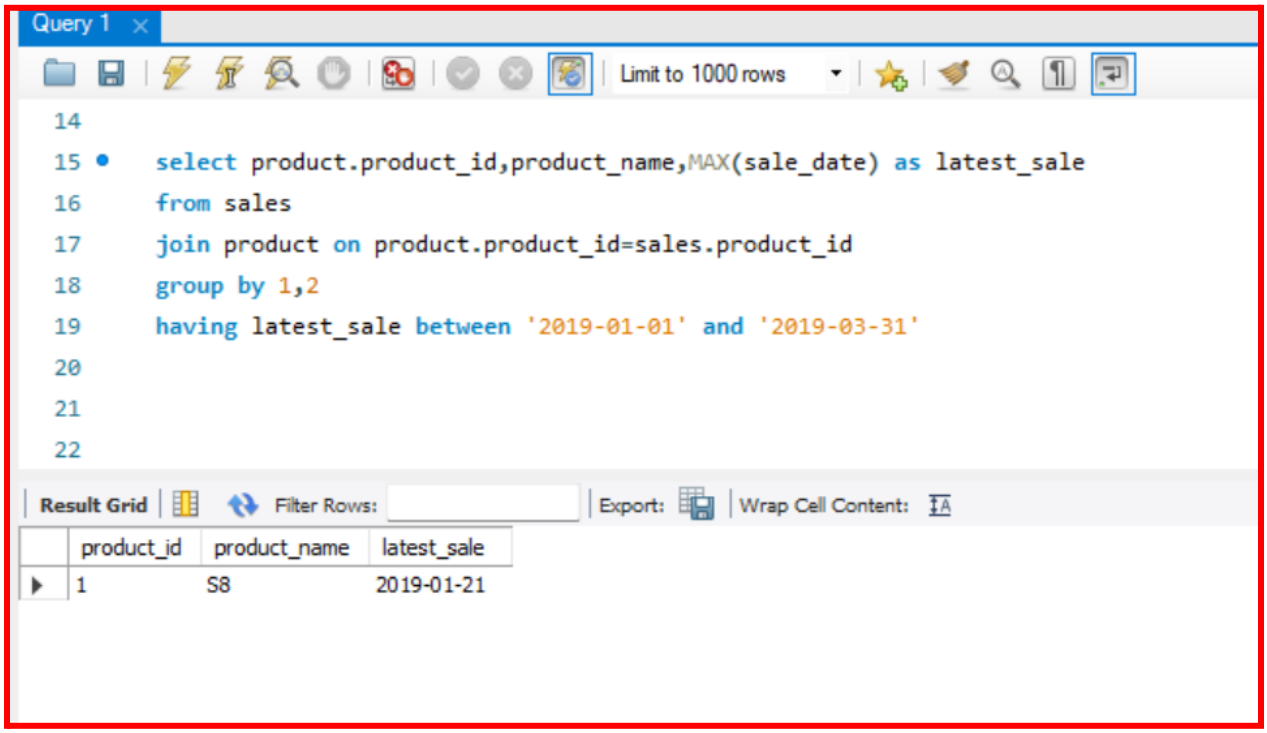
Sales table:

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

Output:

product_id	product_name
1	S8

Explanation:
The product with id 1 was only sold in the spring of 2019.
The product with id 2 was sold in the spring of 2019 but was also sold after the spring of 2019.
The product with id 3 was sold after spring 2019.
We return only product 1 as it is the product that was only sold in the spring of 2019.


Query 1
Limit to 1000 rows
14
15 select product.product_id,product_name,MAX(sale_date) as latest_sale
16 from sales
17 join product on product.product_id=sales.product_id
18 group by 1,2
19 having latest_sale between '2019-01-01' and '2019-03-31'
20
21
22
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
product_id
product_name
latest_sale
1
S8
2019-01-21

Q18.

Table: Views

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

Output:

id
4
7

```
18
19 •  select distinct views.author_id from views
20     join views as v on v.viewer_id=views.author_id
21     order by 1 asc
22
23
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

author_id
4
7

Q19.

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.
Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.
The query result format is in the following example.

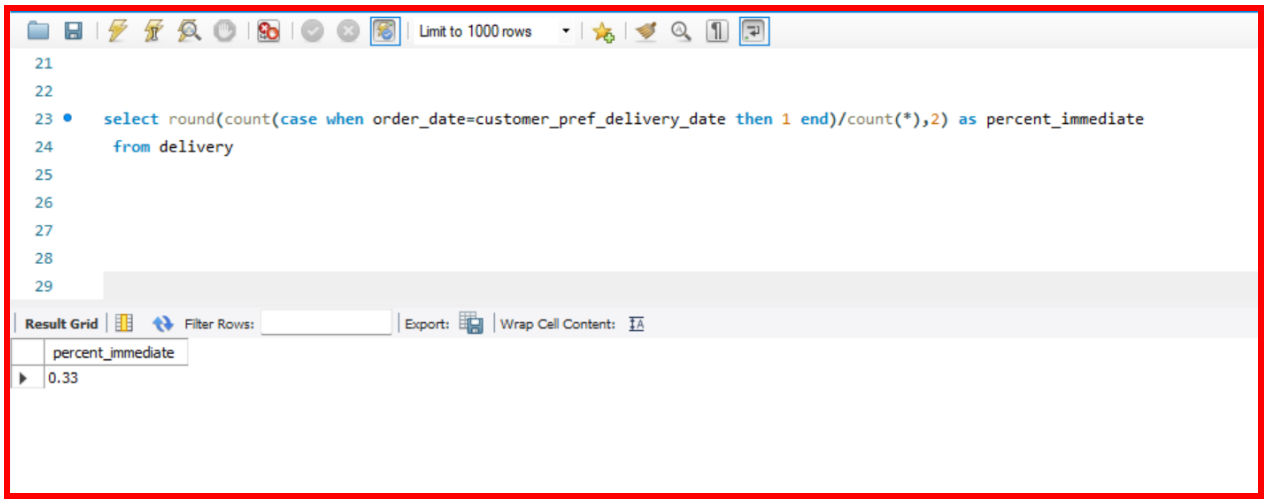
Input:
Delivery table:

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

Output:

immediate_percentage
33.33

Explanation: The orders with delivery id 2 and 3 are immediate while the others are scheduled.



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

Output:

ad_id	ctr
1	66.67
3	50
2	33.33
5	0

Explanation:
for ad_id = 1, ctr = (2/(2+1)) * 100 = 66.67
for ad_id = 2, ctr = (1/(1+2)) * 100 = 33.33
for ad_id = 3, ctr = (1/(1+1)) * 100 = 50.00
for ad_id = 5, ctr = 0.00, Note that ad_id = 5 has no clicks or views.
Note that we do not care about Ignored Ads.

```sql
select ad_id,
case
when (count(case when action='Viewed' then 1 end)+count(case when action='Clicked' then 1 end))=0 then 0
else  count(case when action='Clicked' then 1 end)/(count(case when action='Viewed' then 1 end)+count(case when action='Clicked' then 1 end
))*100 end as ctr
from ads
group by 1
order by 1
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

ad_id	ctr
1	66.6667
2	33.3333
3	50.0000
5	0

Result Grid

Form Editor

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

Output:

employee_id	team_size
1	3
2	3
3	3
4	1
5	2
6	2

Explanation:
Employees with Id 1,2,3 are part of a team with team_id = 8.
An employee with Id 4 is part of a team with team_id = 7.
Employees with Id 5,6 are part of a team with team_id = 9.

```
77 •  select employee_id,count(*) over(partition by team_id) as team_size
78    from employee
79    order by 1
80
81
82
83
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee_id	team_size
1	3
2	3
3	3
4	1
5	2
6	2

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.

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.

Return result table in any order.
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

Output:

country_name	weather_type
USA	Cold
Australia	Cold
Peru	Hot
Morocco	Hot
China	Warm

Explanation:
Average weather_state in the USA in November is (15) / 1 = 15 so the weather type is Cold.
Average weather_state in Australia in November is (-2 + 0 + 3) / 3 = 0.333 so the weather type is Cold.
Average weather_state in Peru in November is (25) / 1 = 25 so the weather type is Hot.
The average weather_state in China in November is (16 + 18 + 21) / 3 = 18.333 so the weather type is warm.
Average weather_state in Morocco in November is (25 + 27 + 31) / 3 = 27.667 so the weather type is Hot.
We know nothing about the average weather_state in Spain in November so we do not include it in the result table.

```
125  select country_name,
126  case
127  when avg(weather_state)>=25 then 'Hot'
128  when avg(weather_state)<=15 then 'Cold'
129  else 'Warm' end as weather_type
130  from weather
131  join countries on countries.country_id=weather.country_id
132  where extract(month from day)=11 and extract(year from day)=2019
133  group by 1
134
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

country_name	weather_type
USA	Cold
AUSTRALIA	Cold
CHINA	Warm
PERU	Hot
MOROCCO	Hot

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.

Write an SQL query to find the average selling price for each product. average_price should be rounded to 2 decimal places.
Return the result table in any order.
The query result format is in the following example.

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

Output:

product_id	average_price
1	6.96
2	16.96

Explanation:
Average selling price = Total Price of Product / Number of products sold.
Average selling price for product 1 = ((100 * 5) + (15 * 20)) / 115 = 6.96
Average selling price for product 2 = ((200 * 15) + (30 * 30)) / 230 = 16.96

```sql
select prices.product_id,round(sum(price*units)/sum(units),2) as sold_price from units_sold
join prices on prices.product_id=units_sold .product_id and purchase_date between start_date and end_date
group by 1
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	sold_price
1	6.96
2	16.96

Q25.

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.

Write an SQL query to report the device that is first logged in for each player.
Return the result table in any order.
The query result format is in the following example.

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

Output:

player_id	device_id
1	2
2	3
3	1

```sql
-- 25
select player_id,min(event_date) as first_login
from activity
group by 1
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

player_id	first_login
1	2016-03-01
2	2017-06-25
3	2016-03-02

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.

Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount.
Return result table in any order.
The query result format is in the following example.

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

Output:

product_name	unit
Leetcode Solutions	130
Leetcode Kit	100

Explanation:
Products with product_id = 1 is ordered in February a total of (60 + 70) = 130.
Products with product_id = 2 is ordered in February a total of 80.
Products with product_id = 3 is ordered in February a total of (2 + 3) = 5.
Products with product_id = 4 was not ordered in February 2020.
Products with product_id = 5 is ordered in February a total of (50 + 50) = 100.

```sql
-- Ans 27
select orders.product_id,product_name,product_category,sum(unit) as units
from products
join orders on orders.product_id=products.product_id
and extract(month from order_date)=2
group by 1,2,3
having sum(unit)>=100
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	product_category	units
1	LEETCODE SOLUTIONS	BOOK	130
5	LEETCODE KIT	T-SHIRT	100

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.

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'.

Return the result table in any order.
The query result format is in the following example.

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

Output:

user_id	name	mail
1	Winston	winston@leetcode.com
3	Annabelle	bella-@leetcode.com
4	Sally	sally.come@leetcode.com

Explanation:
The mail of user 2 does not have a domain.
The mail of user 5 has the # sign which is not allowed.
The mail of user 6 does not have the leetcode domain.
The mail of user 7 starts with a period.

```sql
-- Ans 27
select *
from users
where  mail regexp '^[A-Za-z][A-Za-z0-9\_\.\-]*@leetcode\.com$'
```

Result Grid | Filter Rows: | Edit: | Export/Import:

user_id	name	mail
1	WINSTON	winston@leetcode.com
3	ANNABELLE	bella-@leetcode.com
4	SALLY	sally.come@leetcode.com
NULL	NULL	NULL

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.

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.
Return the result table in any order.
The query result format is in the following example.

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

Output:

customer_id	name
1	Winston

Explanation:
Winston spent $300 (300 * 1) in June and $100 (10 * 1 + 45 * 2) in July 2020.
Jonathan spent $600 (300 * 2) in June and $20 (2 * 10) in July 2020.
Moustafa spent $110 (10 * 2 + 45 * 2) in June and $0 in July 2020.

Ans 28

```
select customer_id,name from (
select customers.customer_id,name,extract(month from
order_date),sum(quantity*price) as amount_spent,row_number() over(partition
by customers.customer_id) as instances
from orders
join customers on customers.customer_id=orders.customer_id
join products on products.product_id=orders.product_id
and extract(month from order_date) in (6,7)
group by 1,2,3
having amount_spent>=100
)b
where instances=2
```

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.

Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020.
Return the result table in any order.
The query result format is in the following example.

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

Output:

title
Aladdin

Explanation:
"Leetcode Movie" is not a content for kids.
"Alg. for Kids" is not a movie.
"Database Sols" is not a movie
"Alladin" is a movie, content for kids and was streamed in June 2020.
"Cinderella" was not streamed in June 2020.

```
403     -- Ans 29
404 •   select distinct title from tv_program
405     join content on content.content_id=tv_program.content_id
406     and extract(month from program_date)=6 and extract(year from program_date)=2020
407     and kids_content='Y'
408
409
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title
ALADDIN

Q30.

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.

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

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

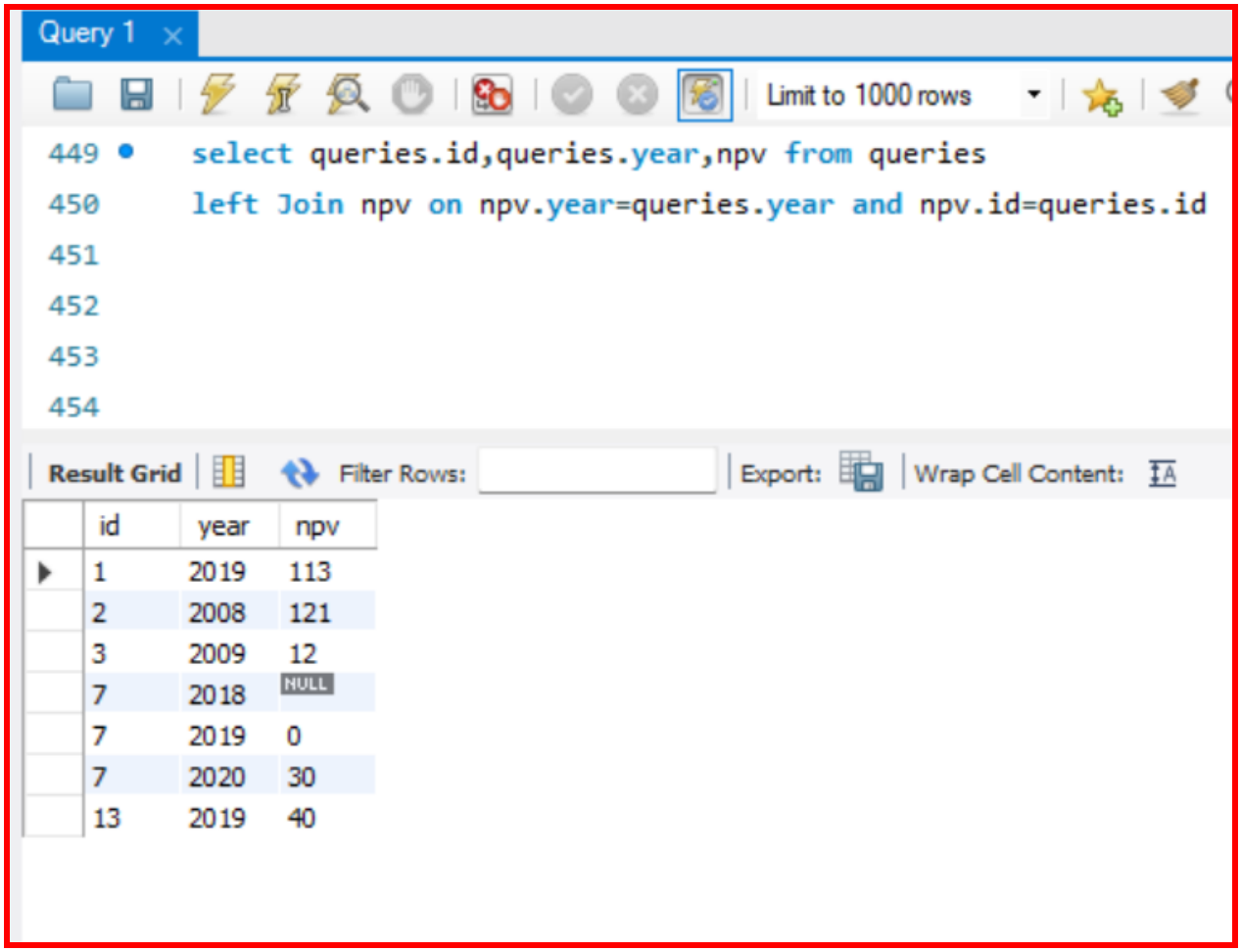
Queries table:

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

Output:

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

Explanation:
The npv value of (7, 2018) is not present in the NPV table, we consider it 0.
The npv values of all other queries can be found in the NPV table.


Query 1
Limit to 1000 rows
449 select queries.id,queries.year,npv from queries
450 left Join npv on npv.year=queries.year and npv.id=queries.id
451
452
453
454
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
id year npv
1 2019 113
2 2008 121
3 2009 12
7 2018 NULL
7 2019 0
7 2020 30
13 2019 40

Q31.

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.

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

Input:
NPV table:

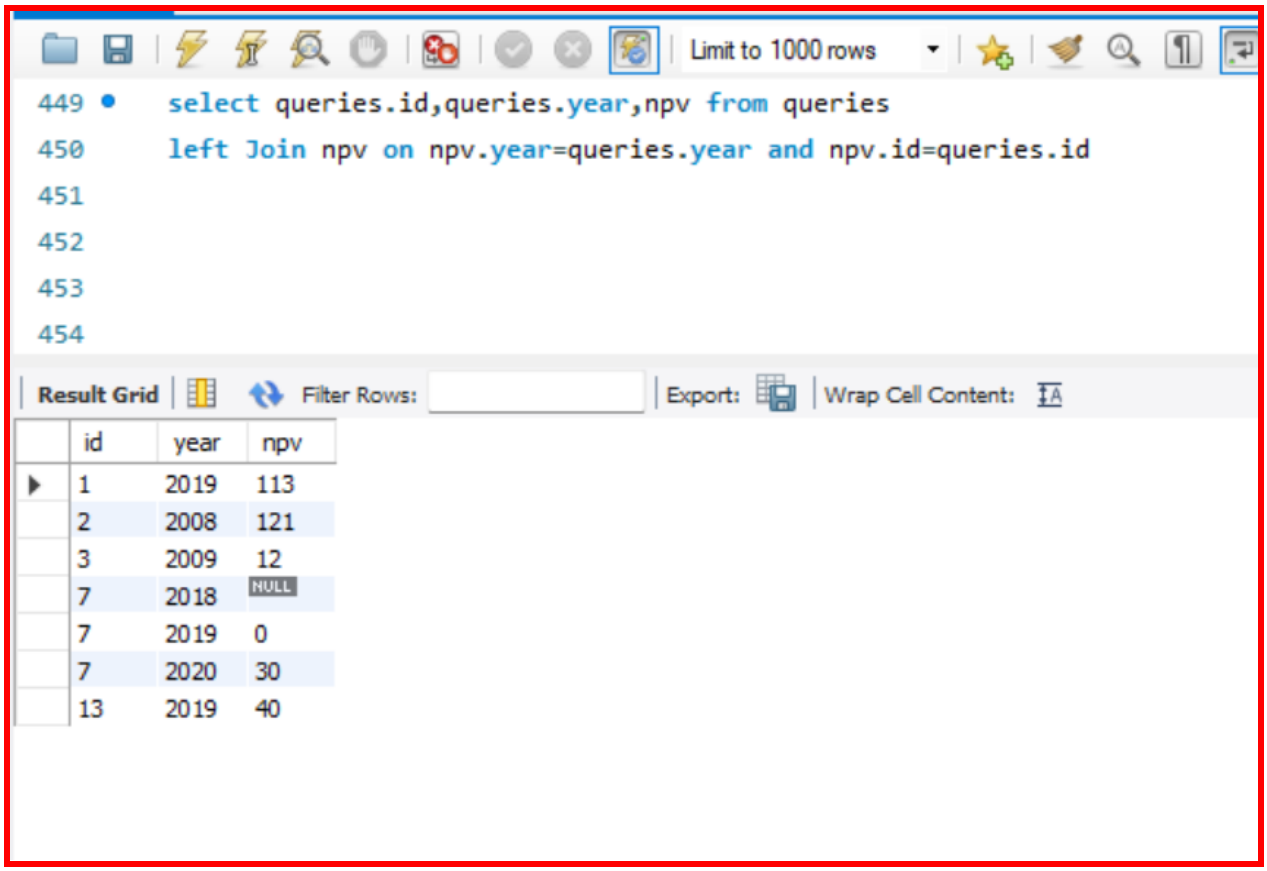
id	year	npv
1	2018	100
7	2020	30
13	2019	40
1	2019	113
2	2008	121
3	2009	12
11	2020	99
7	2019	0

Queries table:

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

Output:

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


Limit to 1000 rows
449 select queries.id,queries.year,npv from queries
450 left Join npv on npv.year=queries.year and npv.id=queries.id
451
452
453
454
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
id year npv
1 2019 113
2 2008 121
3 2009 12
7 2018 NULL
7 2019 0
7 2020 30
13 2019 40

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.

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.
Return the result table in any order.
The query result format is in the following example.

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

Output:

unique_id	name
null	Alice
null	Bob
2	Meir
3	Winston
1	Jonathan

Explanation:
Alice and Bob do not have a unique ID, We will show null instead.
The unique ID of Meir is 2.
The unique ID of Winston is 3.
The unique ID of Jonathan is 1.

```
475
476 •   select employees.id,name,unique_id from employees
477     left join employees_uni on employees_uni.id=employees.id
478
479
480
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	unique_id
1	ALICE	NULL
3	JONATHAN	1
7	BOB	NULL
11	MEIR	2
90	WINSTON	3

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".

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.
The query result format is in the following example.

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

Output:

name	travelled_distance
Elvis	450
Lee	450
Bob	317
Jonathan	312
Alex	222
Alice	120
Donald	0

Explanation:
Elvis and Lee travelled 450 miles, Elvis is the top traveller as his name is alphabetically smaller than Lee.
Bob, Jonathan, Alex, and Alice have only one ride and we just order them by the total distances of the ride.
Donald did not have any rides, the distance travelled by him is 0.

```
524 •  select user_id,name,sum(distance) as distance_travelled from rides
525    join users on users.id=rides.user_id
526    group by 1,2
527    order by 3 desc,2 asc
528
529
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

user_id	name	distance_travelled
19	ELVIS	450
7	LEE	450
2	BOB	317
13	JONATHON	312
3	ALEX	222
1	ALICE	120

Question 34 Same as Question 26

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.

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.

The query result format is in the following example.

-- Find the name of the user who has rated the greatest number of movies

```
584 •  select users.user_id,name,count(*) as rating from movie_rating
585    join users on users.user_id=movie_rating.user_id
586    group by 1,2
587    order by 3 desc ,2 asc
588    limit 1
589
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch

user_id	name	rating
1	DANIEL	3

-- Find the movie name with the highest average rating in February 2020.

```
590 •  select title,avg(rating) as avg_rating from movie_rating
591    join movies on movies.movie_id=movie_rating.movie_id and extract(month from created_at)=2
592    group by 1
593    order by 2 desc ,1 asc
594    limit 1
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

title	avg_rating
FROZEN 2	3.5000

Question 36

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".

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.
The query result format is in the following example.

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

Output:

name	travelled_distance
Elvis	450
Lee	450
Bob	317
Jonathan	312
Alex	222
Alice	120
Donald	0

Explanation:
Elvis and Lee travelled 450 miles, Elvis is the top traveller as his name is alphabetically smaller than Lee.
Bob, Jonathan, Alex, and Alice have only one ride and we just order them by the total distances of the ride.
Donald did not have any rides, the distance travelled by him is 0.

Answer:Same as Question 33

Q37.

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.

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.
Return the result table in any order.

The query result format is in the following example.

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

Output:

unique_id	name
null	Alice
null	Bob
2	Meir
3	Winston
1	Jonathan

Explanation:
Alice and Bob do not have a unique ID, We will show null instead.
The unique ID of Meir is 2.
The unique ID of Winston is 3.
The unique ID of Jonathan is 1.

Answer:Same as Question 32

Question 38

Table: Departments

Column Name	Type
id	int
name	varchar

id is the primary key of this table.
The table has information about the id of each department of a university.

Table: Students

Column Name	Type
id	int
name	varchar
department_id	int

id is the primary key of this table.
The table has information about the id of each student at a university and the id of the department he/she studies at.
Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist.
Return the result table in any order.
The query result format is in the following example.

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

```
635
636 •   select students.id,students.name from students
637     left join departments on departments.id=students.department_id
638     where departments.id is null
639     order by 2
640
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name
7	DAIANA
4	JASMINE
2	JOHN
3	STEVE

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

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.
Return the result table in any order.
The query result format is in the following example.

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

Output:

person1	person2	call_count	total_duration
1	2	2	70
1	3	1	20
3	4	4	999

```
661 •  select
662    case
663    when from_id<to_id then from_id else to_id end as from_person,
664    case
665    when to_id>from_id then to_id else from_id end as to_person,sum(duration) as call_duration
666    from calls
667    group by 1,2
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

from_person	to_person	call_duration
1	2	70
1	3	20
3	4	999

Q40.

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.

Write an SQL query to find the average selling price for each product. average_price should be rounded to 2 decimal places.
Return the result table in any order.
The query result format is in the following example.

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

Output:

product_id	average_price
1	6.96
2	16.96

Explanation:
Average selling price = Total Price of Product / Number of products sold.
Average selling price for product 1 = ((100 * 5) + (15 * 20)) / 115 = 6.96
Average selling price for product 2 = ((200 * 15) + (30 * 30)) / 230 = 16.96

Answers: SAME AS Question 23

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.

Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse.
Return the result table in any order.
The query result format is in the following example.

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

Output:

warehouse_name	volume
LCHouse1	12250
LCHouse2	20250
LCHouse3	800

```
715 •   select name,sum(units*width*length*height) as volume from warehouse
716     join products on products.product_id=warehouse.product_id
717     group by 1
718
719
720
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	volume
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.

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.
The query result format is in the following example.

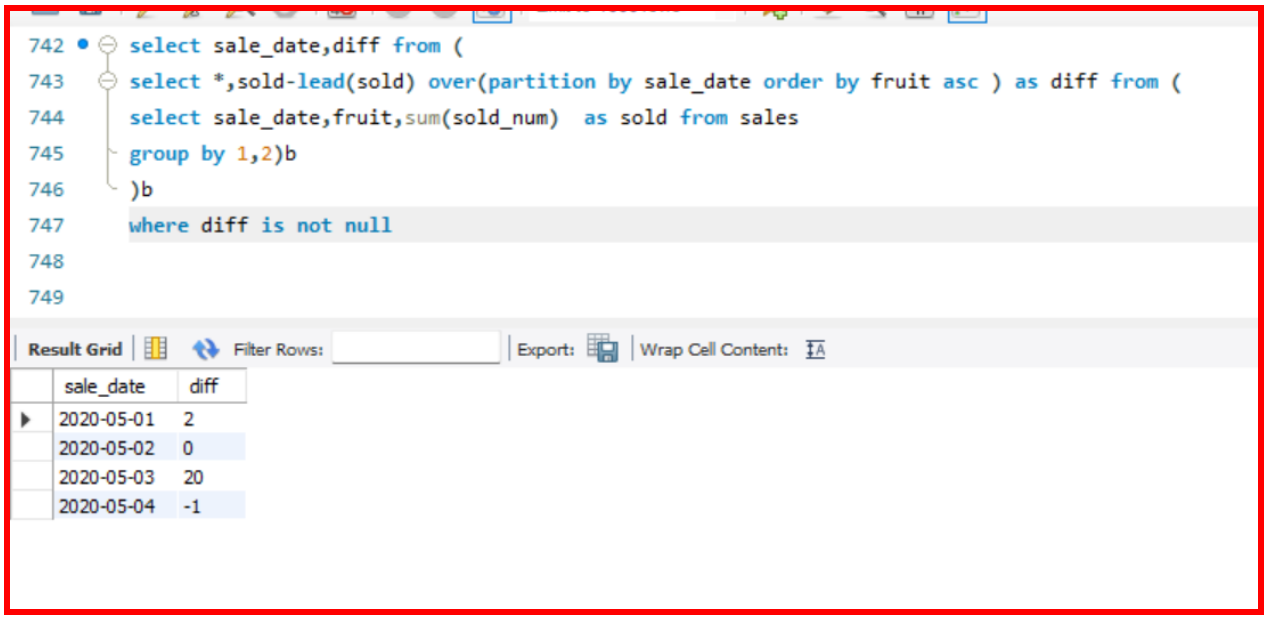
Input:
Sales table:

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

Output:

sale_date	diff
2020-05-01	2
2020-05-02	0
2020-05-03	20
2020-05-04	-1

Explanation:
Day 2020-05-01, 10 apples and 8 oranges were sold (Difference 10 - 8 = 2).
Day 2020-05-02, 15 apples and 15 oranges were sold (Difference 15 - 15 = 0).
Day 2020-05-03, 20 apples and 0 oranges were sold (Difference 20 - 0 = 20).
Day 2020-05-04, 15 apples and 16 oranges were sold (Difference 15 - 16 = -1).



```
742  select sale_date,diff from (
743  select *,sold-lead(sold) over(partition by sale_date order by fruit asc ) as diff from (
744  select sale_date,fruit,sum(sold_num)  as sold from sales
745  group by 1,2)b
746  )b
747  where diff is not null
748
749
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sale_date	diff
2020-05-01	2
2020-05-02	0
2020-05-03	20
2020-05-04	-1

Question 43

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.

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.
The query result format is in the following example.

Input:
Activity table:

player_id	device_id	event_date	games_played
1	2	2016-03-01	5
1	2	2016-03-02	6
2	3	2017-06-25	1
3	1	2016-03-02	0
3	4	2018-07-03	5

Output:

fraction
0.33

Explanation:
Only the player with id 1 logged back in after the first day he had logged in so the answer is 1/3 = 0.33

```
775 •  select
776    round(count(case when date_diff = 1 then  player_id end)/count(distinct player_id),2) as fraction
777    from (
778    select player_id,event_date,datediff(lead(event_date) over(partition by player_id),event_date)
779    as date_diff
780     from activity
781    )b
782
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fraction
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.

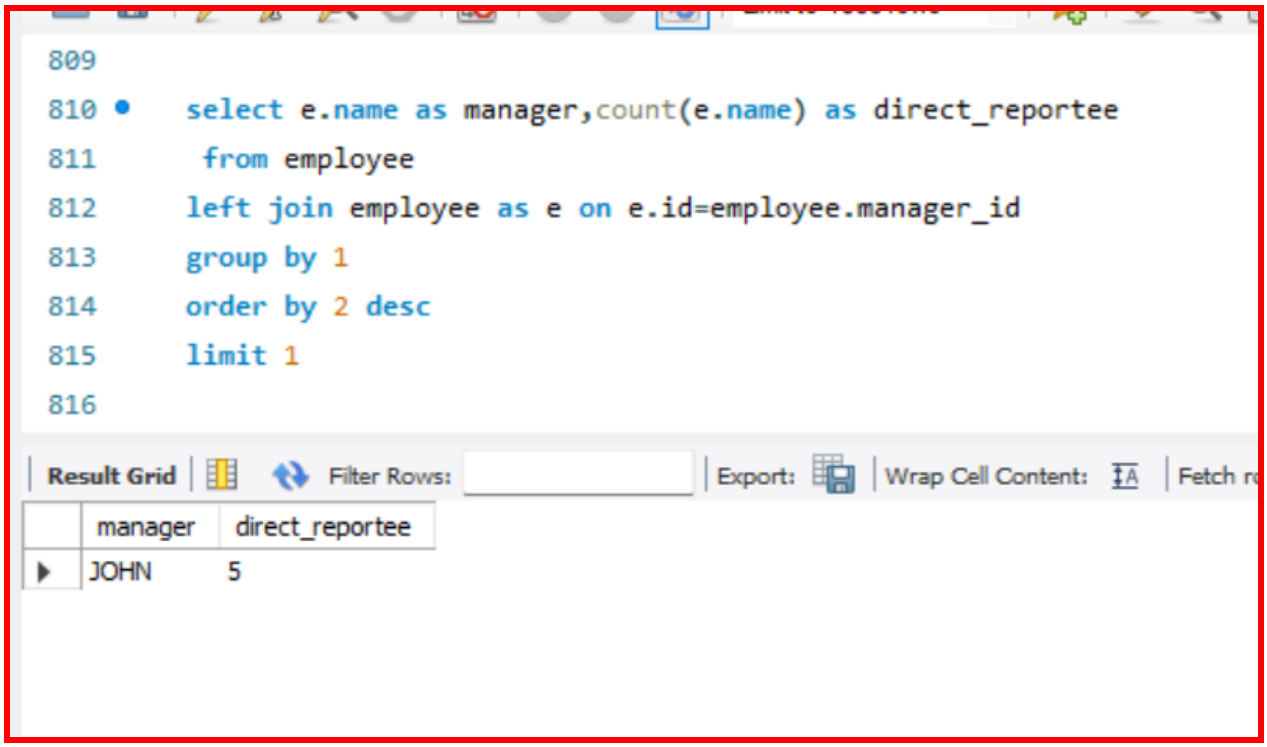
Write an SQL query to report the managers with at least five direct reports.
Return the result table in any order.
The query result format is in the following example.

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

Output:

name
John



```
809
810 •   select e.name as manager,count(e.name) as direct_reportee
811      from employee
812     left join employee as e on e.id=employee.manager_id
813     group by 1
814     order by 2 desc
815     limit 1
816
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch r

manager	direct_reportee
JOHN	5

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.

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.
The query result format is in the following example.

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

```
853 •  select department_name,count(case when student.dept_id is not null then 1 end) as students_majoring
854    from department
855    left join student on student.dept_id=department.dept_id
856    group by 1
857
858
859
860
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

department_name	students_majoring
ENGINEERING	2
SCIENCE	1
LAW	0

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.

Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table.
Return the result table in any order.
The query result format is in the following example.

Input:
Customer table:

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

Product table:

product_key
5
6

Output:

customer_id
1
3

```sql
with products as (
select *,count(*) over () as total_products from product
)
select customer_id,total_products,count(distinct case when customer.product_key is not null then customer.product_key end)
distinct_products
from products
left join customer on customer.product_key=products.product_key
group by 1,2
having distinct_products=total_products

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	total_products	distinct_products
1	2	2
3	2	2

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.

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.
Return the result table in any order.
The query result format is in the following example.

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

Output:

project_id	employee_id
1	1
1	3
2	1

Explanation:
Both employees with id 1 and 3 have the most experience among the employees of the first project.
For the second project, the employee with id 1 has the most experience.

```
934  select * from (
935  select project_id,name,experience_years,dense_rank() over (partition by project.project_id order by experience_years desc) as ranking
936  from project
937  left join employee on employee.employee_id=project.employee_id
938  )b
939  where ranking=1
940
941
942
943
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

project_id	name	experience_years	ranking
1	KHALED	3	1
1	JOHN	3	1
2	KHALED	3	1

Q48.

Table: Books

Column Name	Type
book_id	int
name	varchar
available_from	date

book_id is the primary key of this table.

Table: Orders

Column Name	Type
order_id	int
book_id	int
quantity	int
dispatch_date	date

order_id is the primary key of this table.
book_id is a foreign key to the Books table.

Write an SQL query that reports the books that have sold less than 10 copies in the last year, excluding books that have been available for less than one month from today. Assume today is 2019-06-23.
Return the result table in any order.
The query result format is in the following example.

Input:

Books table:

book_id	name	available_from
1	"Kalila And Demna"	2010-01-01
2	"28 Letters"	2012-05-12
3	"The Hobbit"	2019-06-10
4	"13 Reasons Why"	2019-06-01
5	"The Hunger Games"	2008-09-21

```
987 •  select name,books.book_id,sum(case when extract(year from '2019-06-23')-extract(year from dispatch_date)=1 then quantity else 0 end) as
       total_books_sold_last_year
988     from orders
989    left join books on books.book_id=orders.book_id
990    where datediff('2019-06-23',available_from)>30
991    group by 1,2
992    having total_books_sold_last_year<=10
993
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	book_id	total_books_sold_last_year
Kalila And Demna	1	3
The Hunger Games	5	0

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.

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.
The query result format is in the following example.

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

Output:

student_id	course_id	grade
1	2	99
2	2	95
3	3	82

```
1020
1021 • select * from (
1022   select *,dense_rank() over(partition by student_id order by grade desc,course_id asc) as ranking
1023   from enrollments
1024   )b
1025   where ranking=1
1026
1027
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	course_id	grade	ranking
1	2	99	1
2	2	95	1
3	3	82	1

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.

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.
Return the result table in any order.
The query result format is in the following example.

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

Output:

group_id	player_id
1	15
2	35
3	40

Method 1

```
with base as (
select matches.*,players.group_id as first_group,p.group_id as second_group
from matches
left join players on players.player_id=matches.first_player
left join players as p on p.player_id=matches.second_player
)

select * from (
select *,row_number() over (partition by player_group order by total_goals
desc) as ranking
 from (
select player_group,player,sum(goals) as total_goals from (
(select first_group as player_group,first_player as
player,sum(first_player_goals) as goals from base
group by 1,2)
union all
(select second_group as player_group,second_player as
player,sum(second_player_goals) as goals from base
group by 1,2)
)b
group by 1,2
)b
```

```
)b
where ranking=1
```

Method 2