Database Management Systems

(COP 5725)

Spring 2020

Instructor: Dr. Markus Schneider

TA: Kyuseo Park

Homework 3

Name:	Xiao Hu
UFID:	69363179
Email Address:	xiao.hu@ufl.edu

Pledge (Must be signed according to UF Honor Code)

On my honor, I have neither given nor received unauthorized aid in doing this assignment.

Xiao Hu

Signature

For scoring use only:

	Maximum	Received
Exercise 1	85	
Exercise 2	15	
Total	100	

Exercise 1 (SQL Queries) [85 points]

We are given a geostatistical database about countries, continents, rivers, etc. The following information is available in Canvas together with this homework assignment for download:

- An ER diagram of the geostatistical database in PDF format (*HW3Ex1-geostatistical-database-ER-diagram.pdf*).
- An informal description of the database schema in PDF format (*HW3Ex1-geostatistical-database-schema-explanation.pdf*).
- A text file that contains *create table* commands to create the database schema (*HW3Ex1-geostatistical-database-schema.sql*).
- A text file hat contains *insert* commands for about 47,800 tuples to fill the database tables (*HW3Ex1-geostatistical-database-input-data.sql*).
- A text file that contains *drop table* commands to delete the database schema and the data in the database (*HW3Ex1-geostatistical-database-drop-tables.sql*).

In a first step, use the CISE Oracle DBMS and the Oracle SQL Developer software to create the database schema and fill the database with data. This will also help you learn about the system environment for your group project. In particular, the use of MySQL, PostgreSQL, and other database systems is not allowed.

In a second step, look at the database schema in the file *HW3Ex1-geostatistical-database-schema.sql*. From lines 38 to 52 you will find the following lines:

```
ALTER TABLE Country
  ADD CONSTRAINT FK_CountryREFCity
  FOREIGN KEY (Code, Capital, Province)
  REFERENCES City(Country, Name, Province)
  INITIALLY DEFERRED DEFERRABLE;

ALTER TABLE City
  ADD CONSTRAINT FK_CityREFProvince
  FOREIGN KEY (Country, Province)
  REFERENCES Province(Country, Name)
  INITIALLY DEFERRED DEFERRABLE;

ALTER TABLE Province
  ADD CONSTRAINT FK_ProvinceREFCountry
  FOREIGN KEY (Country)
  REFERENCES Country(Code)
  INITIALLY DEFERRED DEFERRABLE;

ALTER TABLE Province
  ADD CONSTRAINT FK_ProvinceREFCity
  FOREIGN KEY (Capital, Country, CapProv)
  REFERENCES City(Name, Country, Province)
  INITIALLY DEFERRED DEFERRABLE;
```

Your task is to explore this scenario by using the Internet. The keywords `INITIALLY DEFERRED  DEFERRABLE` are non-standard SQL. They are supported by several database systems such as Oracle and PostgreSQL. Answer the following questions:

1. [4 points] What is the meaning of these keywords?
2. [6 points] Why is the action indicated by the keyword `INITIALLY DEFERRED DEFERRABLE` needed in the scenario above? What is the problem? How is the problem solved?

1. initial deferred means that only check the deferred constraint at the point the transaction is commited. Deferrable means that checking a constraint can at the end of a transaction.

2. As for the foreign key of the table city and province, exists the cycle foreign key constrains. So we can not insert the data to table province and city unless use the keyword to let the insert clause check the constrain after inserting the data.

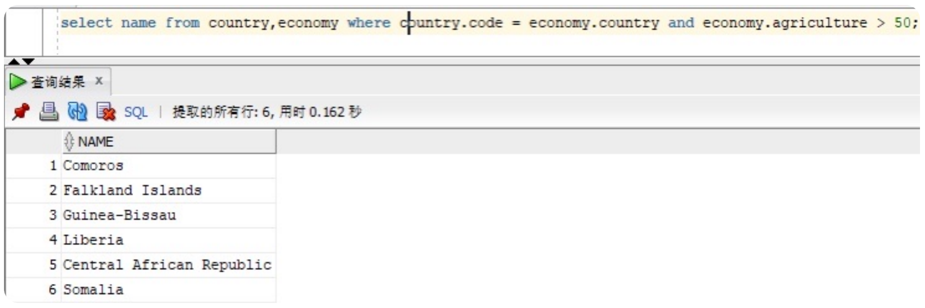
1. ~~[4 points] What is the meaning of these keywords?~~
2. ~~[6 points] Why is the action indicated by the keyword `INITIALLY DEFERRED DEFERRABLE` needed in the scenario above? What is the problem? How is the problem solved?~~

In a third step, write SQL queries for the colloquial queries below and **show the results by providing screenshots for both your SQL queries and query results**. The screenshots must be embedded (as images) into the PDF file that contains your solutions to this whole assignment. In order to increase readability, the SQL queries should be written in a structured manner, all SQL keywords should be fully capitalized, and the table and attribute names should be written in the same way as in the schema file.

1. [1 point] Find the names of countries where agriculture takes more than 50% of its gross domestic product (GPD).
2. [3 points] List the top five countries that will have the largest population after five years. [Assume that the population in five years is equal to the population this year * (1 + growth rate)^5. The population growth in the database schema is in percentage and should be divided by 100. Use the new attributes *Country*, *Population after 5 years*, and *Rank* for the resulting table schema.
3. [4 points] Find the country c1 that *used to* have the maximum number n1 of countries/areas depending on it. Further, find the country c2 that *now* has the maximum number n2 of countries/areas depending on it. Output c1, n1, c2, n2, and the difference between n1 and n2.
4. [4 points] List the country names that have more than 4 different kinds of religion and at least one religion takes more than 80%.
5. [3 points] Compute the total length of the border that China shares with its neighboring countries.
6. [4 points] Find the top five popular religions and the numbers of their believers in the world.
7. [3 points] Find the names of the lakes in the United States with an elevation that is above the average elevation of all lakes world-wide.
8. [4 points] Find the largest population density (population/area) of provinces that have mountains of the "volcano" type. Output the province name, mountain name, and the population density.
9. [3 points] Find the provinces that are located on more than 2 islands and whose country's GDP is greater than 1000000.
10. [3 points] Find the two longest rivers that flow through at least one lake and that finally flow into the Atlantic Ocean. Output the name and the length of the rivers.
11. [4 points] Determine the names of countries that have more than three rivers and that have lakes next to more than three provinces.
12. [4 points] Find the names of those countries that are bounded by the largest lake.
13. [2 points] Find the height of the highest mountain for each continent.

14. [3 points] Find the countries whose depth of the deepest sea is less than the elevation of the highest mountain. Display the country name, depth of its deepest sea, and the elevation of the highest mountain.
15. [4 points] Find the northernmost cities of each continent (except Asia). Display the names of these cities and their continent. List cities that are northern of other cities in the result table first.
16. [1 point] Find all countries whose capitals have positive latitudes and less than 10000 inhabitants.
17. [4 points] Find what is larger. Is it the sum of the areas of the 10 largest countries (attribute *top10*) or the sum of the areas of the remaining countries (attribute *rest_world*)? What is their difference (attribute *difference*)? Display the values for the attributes *top10*, *rest_world*, and *difference*.
18. [2 points] Find all countries that cross continental boundaries.
19. [2 points] Display each island in Africa and its area if the area is larger than 1000 square kilometers. The output should be in descending order of the size of the areas.
20. [3 points] List the names and GDPs of those countries which are members of the NATO and more than 5 percent of their population are Muslims.
21. [1 point] Find names of rivers which cross at least 12 provinces in the same country.
22. [2 points] Find the name and length of the longest river on the American continent.
23. [3 points] Find the provinces that have the largest number of islands in the world. Output the country code, the province, and the number of islands.
24. [3 points] List the 10 country names (attribute "Country Name") with the highest population density (attribute "Population Density") as well as the percentage of the world population (attribute "Percentage") each one contains.
25. [5 points] List the names of organizations that have only Asian countries as members.

1.



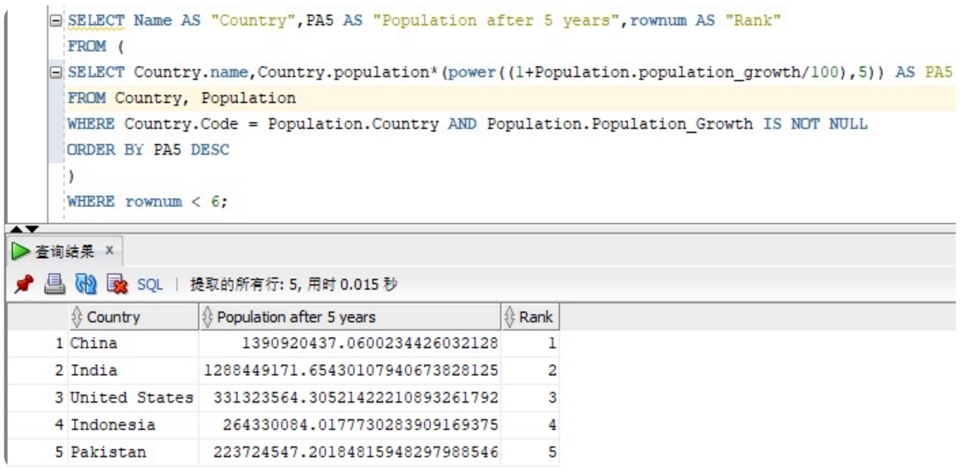
```
select name from country,economy where country.code = economy.country and economy.agriculture > 50;
```

查询結果 ×

SQL | 提取的所有行: 6, 用时 0.162 秒

```
  NAME
1 Comoros
2 Falkland Islands
3 Guinea-Bissau
4 Liberia
5 Central African Republic
6 Somalia
```

2.



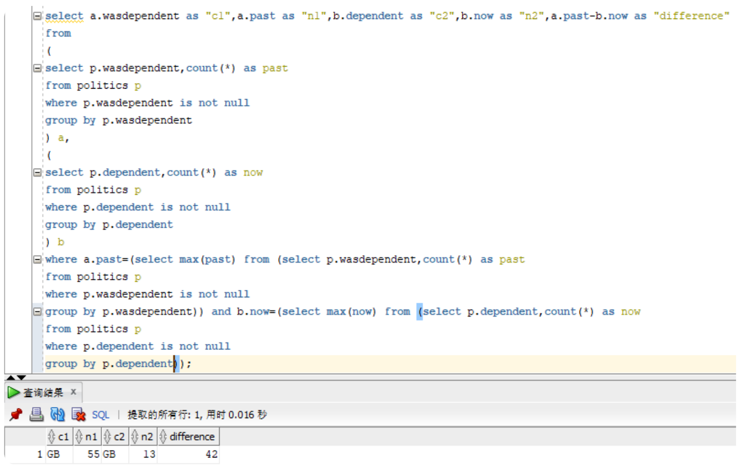
```
SELECT Name AS "Country",PA5 AS "Population after 5 years",rownum AS "Rank"
FROM (
SELECT Country.name,Country.population*(power((1+Population.population_growth/100),5)) AS PA5
FROM Country, Population
WHERE Country.Code = Population.Country AND Population.Population_Growth IS NOT NULL
ORDER BY PA5 DESC
)
WHERE rownum < 6;
```

查询結果 ×

SQL | 提取的所有行: 5, 用时 0.015 秒

```
  Country        Population after 5 years             Rank
1 China             1390920437.0600234426032128          1
2 India            1288449171.65430107940673828125      2
3 United States     331323564.30521422210893261792       3
4 Indonesia         264330084.0177730283909169375       4
5 Pakistan          223724547.20184815948297988546      5
```

3.



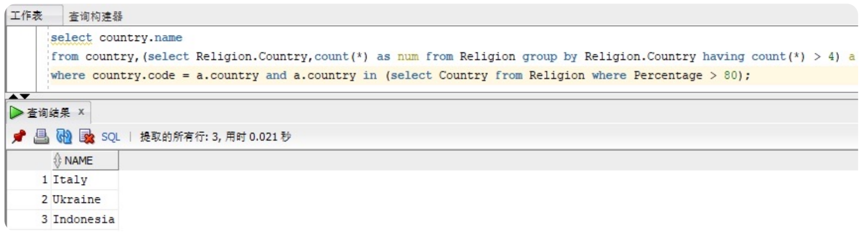
```
select a.wasdependent as "c1",a.past as "n1",b.dependent as "c2",b.now as "n2",a.past-b.now as "difference"
from
(
select p.wasdependent,count(*) as past
from politics p
where p.wasdependent is not null
group by p.wasdependent
) a,
(
select p.dependent,count(*) as now
from politics p
where p.dependent is not null
group by p.dependent
) b
where a.past=(select max(past) from (select p.wasdependent,count(*) as past
from politics p
where p.wasdependent is not null
group by p.wasdependent)) and b.now=(select max(now) from (select p.dependent,count(*) as now
from politics p
where p.dependent is not null
group by p.dependent));
```

查询結果

SQL | 提取的所有行: 1, 用时 0.016 秒

```
   c1  n1 c2  n2 difference
 1 GB  55 GB  13         42
```

4.



工作表 查询构建器

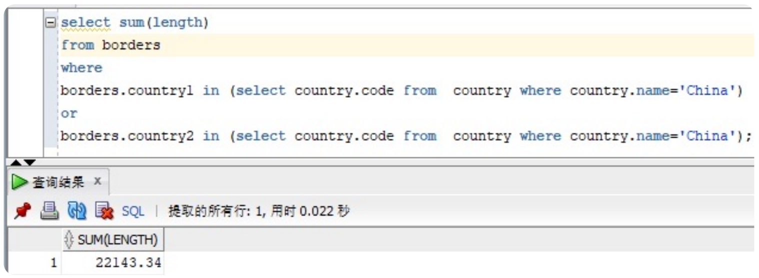
```
select country.name
from country,(select Religion.Country,count(*) as num from Religion group by Religion.Country having count(*) > 4) a
where country.code = a.country and a.country in (select Country from Religion where Percentage > 80);
```

查询結果

SQL | 提取的所有行: 3, 用时 0.021 秒

```
  NAME
1 Italy
2 Ukraine
3 Indonesia
```

5.



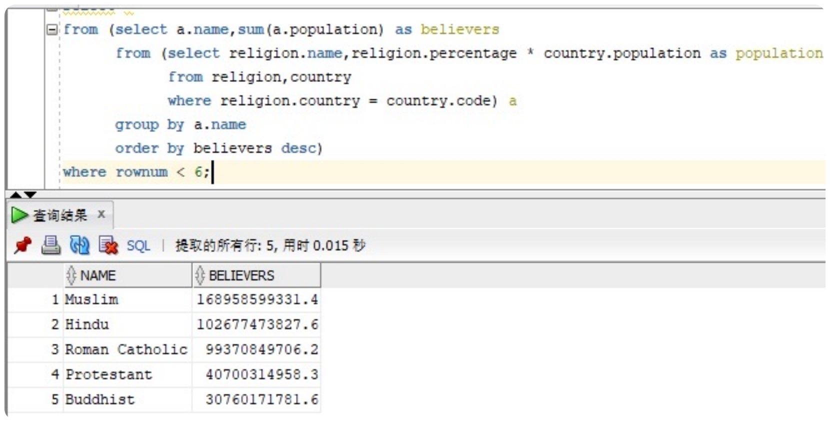
```
select sum(length)
from borders
where
borders.country1 in (select country.code from  country where country.name='China')
or
borders.country2 in (select country.code from  country where country.name='China');
```

查询結果

SQL | 提取的所有行: 1, 用时 0.022 秒

```
  SUM(LENGTH)
1    22143.34
```

6.



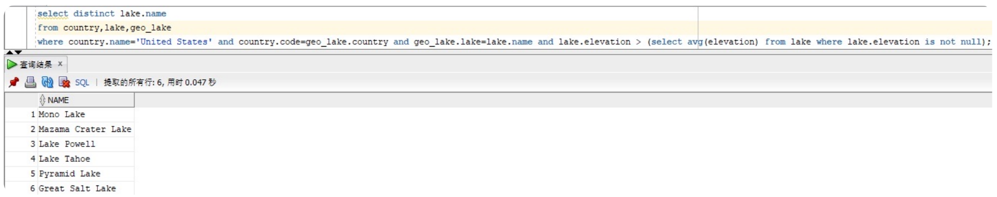
```
from (select a.name,sum(a.population) as believers
        from (select religion.name,religion.percentage * country.population as population
                from religion,country
                where religion.country = country.code) a
        group by a.name
        order by believers desc)
where rownum < 6;
```

查询結果

SQL | 提取的所有行: 5, 用时 0.015 秒

```
  NAME            BELIEVERS
1 Muslim          168958599331.4
2 Hindu           102677473827.6
3 Roman Catholic   99370849706.2
4 Protestant       40700314958.3
5 Buddhist         30760171781.6
```

7.



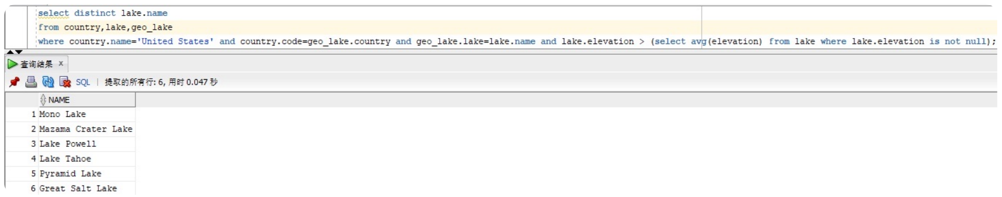
```
select distinct lake.name
from country,lake,geo_lake
where country.name='United States' and country.code=geo_lake.country and geo_lake.lake=lake.name and lake.elevation > (select avg(elevation) from lake where lake.elevation is not null);
```

查询結果

SQL | 提取的所有行: 6, 用时 0.047 秒

```
  NAME
1 Mono Lake
2 Mazama Crater Lake
3 Lake Powell
4 Lake Tahoe
5 Pyramid Lake
6 Great Salt Lake
```

8.



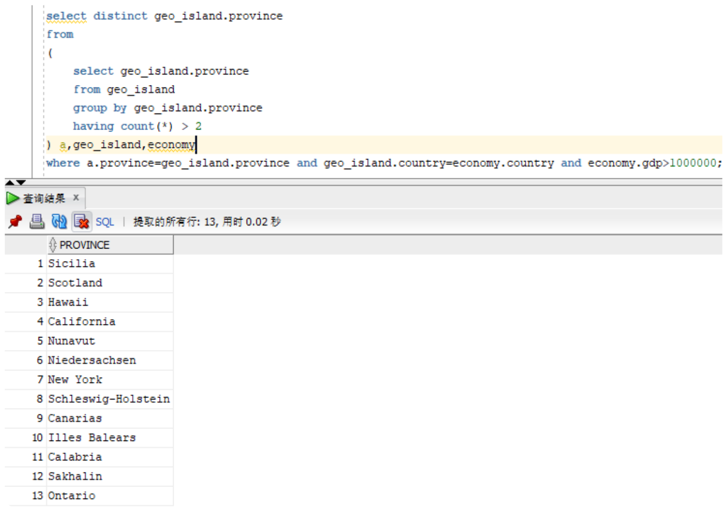
```
select distinct lake.name
from country,lake,geo_lake
where country.name='United States' and country.code=geo_lake.country and geo_lake.lake=lake.name and lake.elevation > (select avg(elevation) from lake where lake.elevation is not null);
```

查询結果 ×

SQL | 提取的所有行: 6, 用时 0.047 秒

```
  NAME
1 Mono Lake
2 Mazama Crater Lake
3 Lake Powell
4 Lake Tahoe
5 Pyramid Lake
6 Great Salt Lake
```

9.



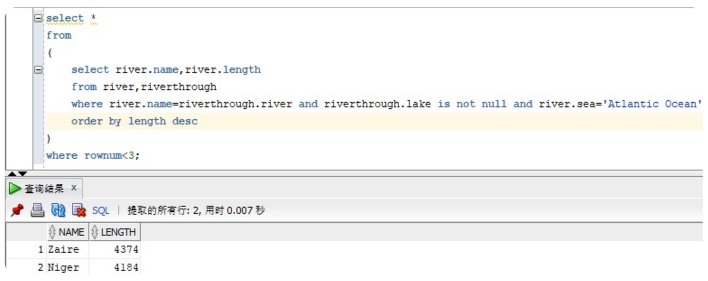
```
select distinct geo_island.province
from
(
    select geo_island.province
    from geo_island
    group by geo_island.province
    having count(*) > 2
) a,geo_island,economy
where a.province=geo_island.province and geo_island.country=economy.country and economy.gdp>1000000;
```

查询結果 ×

SQL | 提取的所有行: 13, 用时 0.02 秒

```
   PROVINCE
 1 Sicilia
 2 Scotland
 3 Hawaii
 4 California
 5 Nunavut
 6 Niedersachsen
 7 New York
 8 Schleswig-Holstein
 9 Canarias
10 Illes Balears
11 Calabria
12 Sakhalin
13 Ontario
```

10.



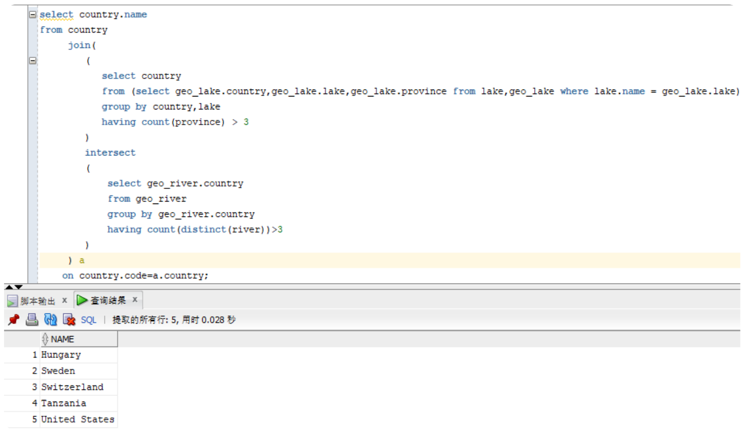
```
select *
from
(
    select river.name,river.length
    from river,riverthrough
    where river.name=riverthrough.river and riverthrough.lake is not null and river.sea='Atlantic Ocean'
    order by length desc
)
where rownum<3;
```

查询結果 ×

SQL | 提取的所有行: 2, 用时 0.007 秒

```
  NAME   LENGTH
1 Zaire    4374
2 Niger    4184
```

11.



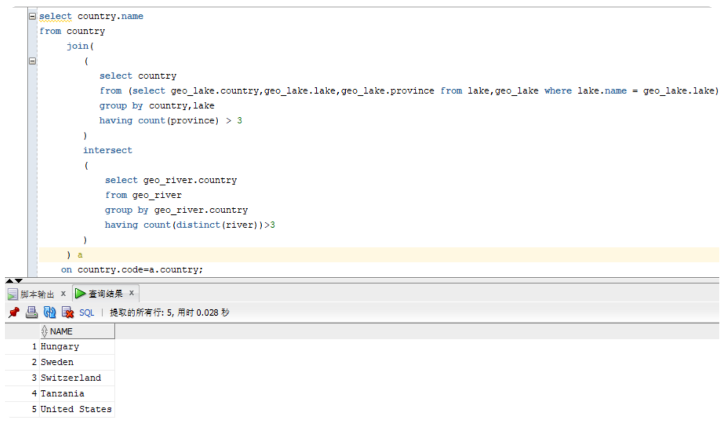
```
select country.name
from country
    join(
        (
            select country
            from (select geo_lake.country,geo_lake.lake,geo_lake.province from lake,geo_lake where lake.name = geo_lake.lake)
            group by country,lake
            having count(province) > 3
        )
        intersect
        (
             select geo_river.country
             from geo_river
             group by geo_river.country
             having count(distinct(river))>3
        )
    ) a
   on country.code=a.country;
```

脚本输出 × 查询結果 ×

SQL | 提取的所有行: 5, 用时 0.028 秒

```
  NAME
1 Hungary
2 Sweden
3 Switzerland
4 Tanzania
5 United States
```

12.



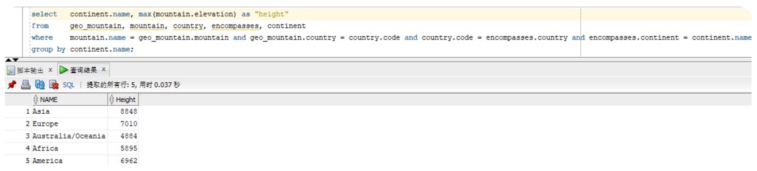
```sql
select country.name
from country
    join(
        (
            select country
            from (select geo_lake.country,geo_lake.lake,geo_lake.province from lake,geo_lake where lake.name = geo_lake.lake)
            group by country,lake
            having count(province) > 3
        )
        intersect
        (
             select geo_river.country
             from geo_river
             group by geo_river.country
             having count(distinct(river))>3
        )
    ) a
    on country.code=a.country;
```

脚本输出 × 查询结果 ×

SQL | 提取的所有行: 5, 用时 0.028 秒

	NAME
1	Hungary
2	Sweden
3	Switzerland
4	Tanzania
5	United States

13.



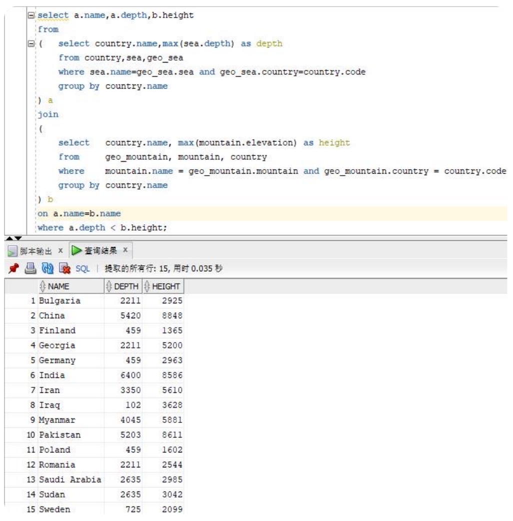
```sql
select   continent.name, max(mountain.elevation) as "height"
from     geo_mountain, mountain, country, encompasses, continent
where    mountain.name = geo_mountain.mountain and geo_mountain.country = country.code and country.code = encompasses.country and encompasses.continent = continent.name
group by continent.name;
```

脚本输出 × 查询结果 ×

SQL | 提取的所有行: 5, 用时 0.037 秒

	NAME	Height
1	Asia	8848
2	Europe	7010
3	Australia/Oceania	4884
4	Africa	5895
5	America	6962

14.



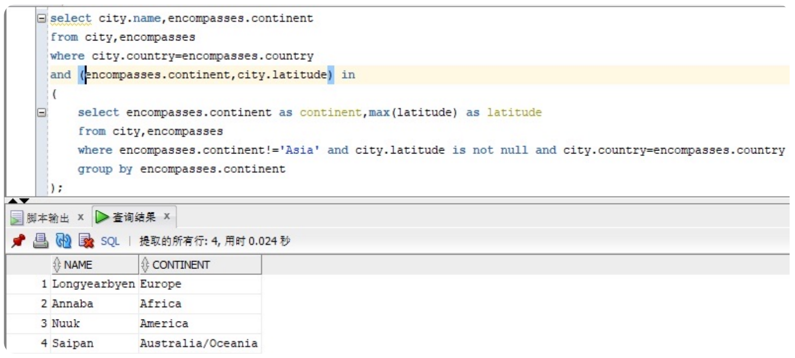
```sql
select a.name,a.depth,b.height
from
(   select country.name,max(sea.depth) as depth
    from country,sea,geo_sea
    where sea.name=geo_sea.sea and geo_sea.country=country.code
    group by country.name
) a
join
(
    select   country.name, max(mountain.elevation) as height
    from     geo_mountain, mountain, country
    where    mountain.name = geo_mountain.mountain and geo_mountain.country = country.code
    group by country.name
) b
on a.name=b.name
where a.depth < b.height;
```

脚本输出 × 查询结果 ×

SQL | 提取的所有行: 15, 用时 0.035 秒

	NAME	DEPTH	HEIGHT
1	Bulgaria	2211	2925
2	China	5420	8848
3	Finland	459	1365
4	Georgia	2211	5200
5	Germany	459	2963
6	India	6400	8586
7	Iran	3350	5610
8	Iraq	102	3628
9	Myanmar	4045	5881
10	Pakistan	5203	8611
11	Poland	459	1602
12	Romania	2211	2544
13	Saudi Arabia	2635	2985
14	Sudan	2635	3042
15	Sweden	725	2099

15.



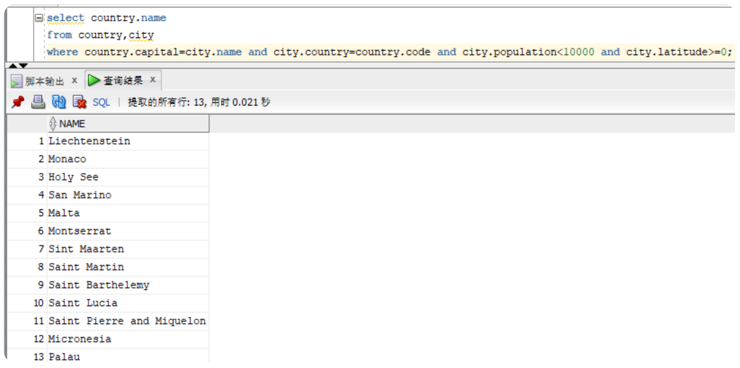
```sql
select city.name,encompasses.continent
from city,encompasses
where city.country=encompasses.country
and (encompasses.continent,city.latitude) in
(
    select encompasses.continent as continent,max(latitude) as latitude
    from city,encompasses
    where encompasses.continent!='Asia' and city.latitude is not null and city.country=encompasses.country
    group by encompasses.continent
);
```

脚本输出 × 查询结果 ×

SQL | 提取的所有行: 4, 用时 0.024 秒

	NAME	CONTINENT
1	Longyearbyen	Europe
2	Annaba	Africa
3	Nuuk	America
4	Saipan	Australia/Oceania

16.



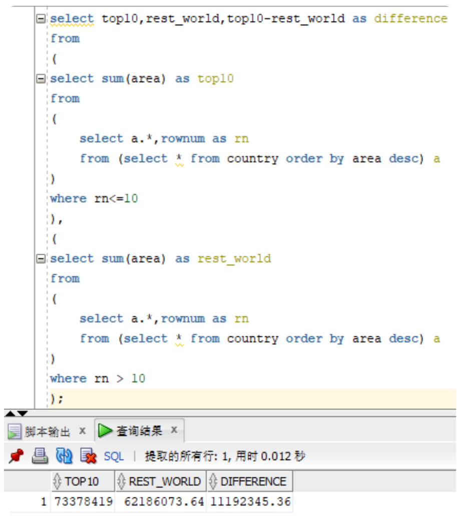
```
select country.name
from country,city
where country.capital=city.name and city.country=country.code and city.population<10000 and city.latitude>=0;
```

脚本输出 x 查询結果 x

SQL | 提取的所有行: 13, 用时 0.021 秒

```
   NAME
 1 Liechtenstein
 2 Monaco
 3 Holy See
 4 San Marino
 5 Malta
 6 Montserrat
 7 Sint Maarten
 8 Saint Martin
 9 Saint Barthelemy
10 Saint Lucia
11 Saint Pierre and Miquelon
12 Micronesia
13 Palau
```

17.



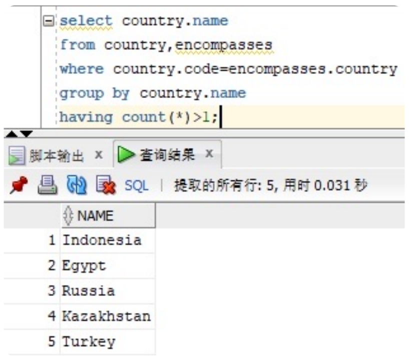
```
select top10,rest_world,top10-rest_world as difference
from
(
select sum(area) as top10
from
(
    select a.*,rownum as rn
    from (select * from country order by area desc) a
)
where rn<=10
),
(
select sum(area) as rest_world
from
(
    select a.*,rownum as rn
    from (select * from country order by area desc) a
)
where rn > 10
);
```

脚本输出 x 查询結果 x

SQL | 提取的所有行: 1, 用时 0.012 秒

```
  TOP10    REST_WORLD  DIFFERENCE
1 73378419 62186073.64 11192345.36
```

18.



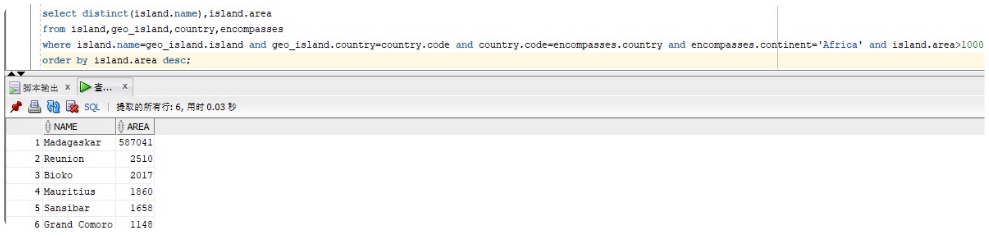
```
select country.name
from country,encompasses
where country.code=encompasses.country
group by country.name
having count(*)>1;
```

脚本输出 x 查询結果 x

SQL | 提取的所有行: 5, 用时 0.031 秒

```
  NAME
1 Indonesia
2 Egypt
3 Russia
4 Kazakhstan
5 Turkey
```

19.



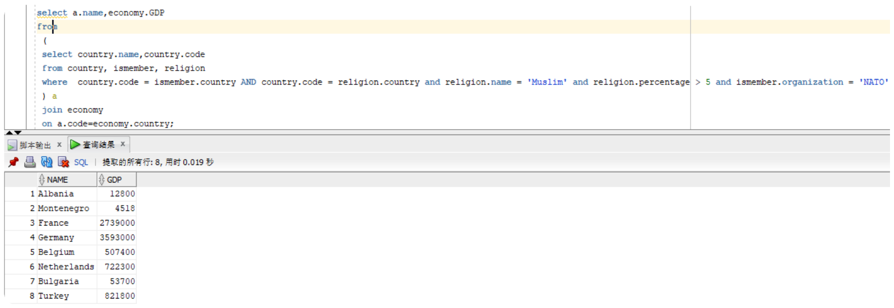
```
select distinct(island.name),island.area
from island,geo_island,country,encompasses
where island.name=geo_island.island and geo_island.country=country.code and country.code=encompasses.country and encompasses.continent='Africa' and island.area>1000
order by island.area desc;
```

脚本输出 x 查... x

SQL | 提取的所有行: 6, 用时 0.03 秒

```
  NAME          AREA
1 Madagaskar    587041
2 Reunion         2510
3 Bioko           2017
4 Mauritius       1860
5 Sansibar        1658
6 Grand Comoro    1148
```

20.



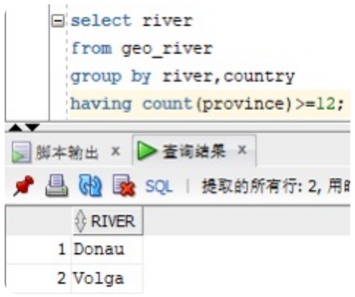
```
select a.name,economy.GDP
from
 (
 select country.name,country.code
 from country, ismember, religion
 where  country.code = ismember.country AND country.code = religion.country and religion.name = 'Muslim' and religion.percentage > 5 and ismember.organization = 'NATO'
 ) a
 join economy
 on a.code=economy.country;
```

脚本输出 × 查询结果 ×

SQL | 提取的所有行: 8, 用时 0.019 秒

	NAME	GDP
1	Albania	12800
2	Montenegro	4518
3	France	2739000
4	Germany	3593000
5	Belgium	507400
6	Netherlands	722300
7	Bulgaria	53700
8	Turkey	821800

21.



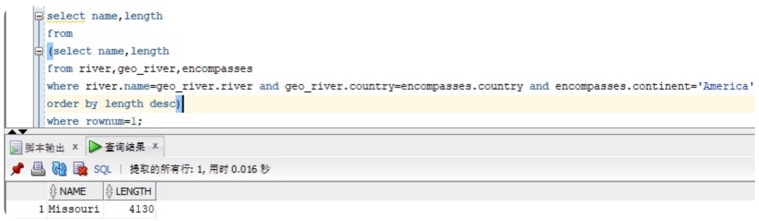
```
select river
from geo_river
group by river,country
having count(province)>=12;
```

脚本输出 × 查询结果 ×

SQL | 提取的所有行: 2, 用

	RIVER
1	Donau
2	Volga

22、



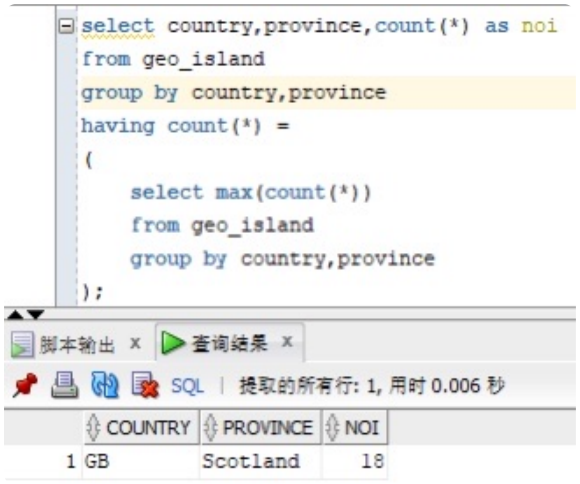
```
select name,length
from
(select name,length
from river,geo_river,encompasses
where river.name=geo_river.river and geo_river.country=encompasses.country and encompasses.continent='America'
order by length desc)
where rownum=1;
```

脚本输出 × 查询结果 ×

SQL | 提取的所有行: 1, 用时 0.016 秒

	NAME	LENGTH
1	Missouri	4130

23.



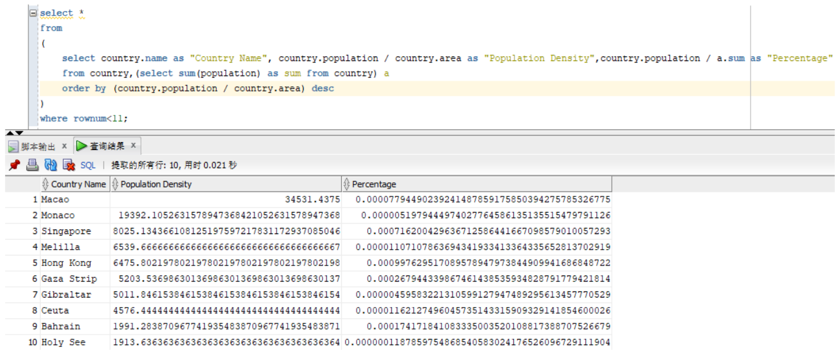
```
select country,province,count(*) as noi
from geo_island
group by country,province
having count(*) =
(
    select max(count(*))
    from geo_island
    group by country,province
);
```

脚本输出 × 查询结果 ×

SQL | 提取的所有行: 1, 用时 0.006 秒

	COUNTRY	PROVINCE	NOI
1	GB	Scotland	18

24,



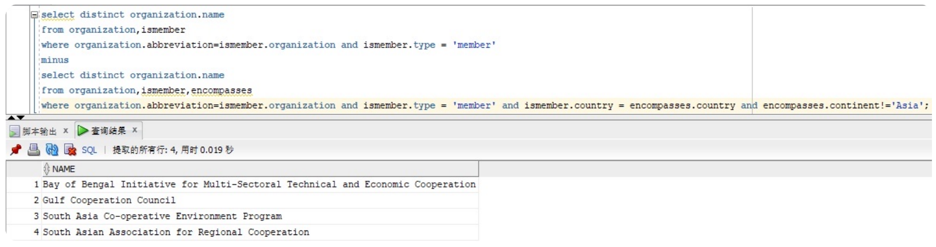
```sql
select *
from
(
    select country.name as "Country Name", country.population / country.area as "Population Density",country.population / a.sum as "Percentage"
    from country,(select sum(population) as sum from country) a
    order by (country.population / country.area) desc
)
where rownum<11;
```

脚本输出 × | 查询结果 ×

SQL | 提取的所有行: 10, 用时 0.021 秒

	Country Name	Population Density	Percentage
1	Macao	34531.4375	0.00007794490239241487859175850394275785326775
2	Monaco	19392.1052631578947368421052631578947368	0.0000051979444974027764586135135515479791126
3	Singapore	8025.134366108125197597217831172937085046	0.000716200429636712586441667098579010057293
4	Melilla	6539.666666666666666666666666666666666667	0.00001107107863694341933413364335652813702919
5	Hong Kong	6475.802197802197802197802197802197802198	0.000997629517089578947973844909941686848722
6	Gaza Strip	5203.53698630136986301369863013698630137	0.000267944339867461438535934828791779421814
7	Gibraltar	5011.846153846153846153846153846153846154	0.00000459583221310599127947489295613457770529
8	Ceuta	4576.444444444444444444444444444444444444	0.00001162127496045735143315909329141854600026
9	Bahrain	1991.283870967741935483870967741935483871	0.000174171841083335003520108817388707526679
10	Holy See	1913.636363636363636363636363636363636364	0.0000001187859754868540583024176526096729111904

25,



```sql
select distinct organization.name
from organization,ismember
where organization.abbreviation=ismember.organization and ismember.type = 'member'
minus
select distinct organization.name
from organization,ismember,encompasses
where organization.abbreviation=ismember.organization and ismember.type = 'member' and ismember.country = encompasses.country and encompasses.continent!='Asia';
```

脚本输出 × | 查询结果 ×

SQL | 提取的所有行: 4, 用时 0.019 秒

	NAME
1	Bay of Bengal Initiative for Multi-Sectoral Technical and Economic Cooperation
2	Gulf Cooperation Council
3	South Asia Co-operative Environment Program
4	South Asian Association for Regional Cooperation

Exercise 2 (QBE) [15 points]

Consider the following database schema:

Drivers (did, dname, gender, age)

Reserve (did, cid, day, cost)

Cars (cid, cname, model, color, rid)

RentalCompany (rid, rname, revenue, rating)

IsMember(did, rid, join_time, member_type)

Display the QBE tables that will answer the following questions.

1. [2 points] Find the names of drivers who have reserved a red car on day "02/14/2017" of model "Chevrolet".
2. [2 points] Find the names of all drivers that are members of a rental company whose rating is greater than 6.5.
3. [3 points] Find the youngest driver who is a member of both company 'Avis' and company 'Hertz'.
4. [2 points] Update the member type to 'VIP' for those drivers who were members of company 'Avis' and have spent more than 2000 in renting (reserving) cars from Avis.
5. [3 points] Find the rental company which has the largest number of members.
6. [3 points] Find the car model that is rented most frequently by drivers whose age is between 21 and 30 (not equal to 21 or 30).

1.

Drivers	did	dname	gender	age
	_x	P._n		

Reserve	did	cid	day	cost.
	_x	_y	02/14/2017	

Cars	cid	cname	model	color	rid
	_y		chevrolet	red	

2.

Drivers	did.	dname	gender	age.
	_x	P. _n		

ismember	did	rid	join_time	member_type
	_x	_y		

rentalcompany	rid	rname	revenue	rating
	_y			>6.5

3.

Driver	did	dname	gender	age
	P._id			P._age_
¬	_id2			<_age

ismember	did	rid	join_time	member_type
	_id	_x		
	_id	_y		
	_id2	_x		
	_id2	_y		

rentalcompany	rid	rname	revenue	rating
	_x	Avis		
	_y	Hertz.		

4.

driver	did	dname	gender	age
	_id.			

ismember	did	rid	join_time	member_type
U.	_id	_r		VIP

rentalcompany	rid	rname	revenue	rating.
	_r	Avis		

reserve	did	cid	day	cost.
	G. _id.			SUM.ALL._X

conditions
SUM.ALL._X>2000

5、

ismember.	did	rid	join_time	member_type
	CNT.ALL._id	G._x		
¬	>CNT.ALL._id	G._y		

rentalcompany	rid	rname	revenue	rating
P.	_X	_n		

6.

Driver	did	dname	gender	age
	_id			_a

car	cid	cname	model	color	rid
	CNT.UN.ALL._x		G. y		
¬	CNT.UN.ALL._X		G. z		

reserve	did	cid	day	cost.
	_id.	_X		

conditions
_a >21 and _a < 30