Ruthvik Udutha **Assignment 3** Date:

(UFID : 3311-9706) (COP-5725 DBMS) 22th Oct,2019

Exercise 1

1.

[4 points] What is the meaning of these keywords?

For a database, all the constraints will be immediately applied. A constraint’s deferrable characteristic can be used to change this nature. Deferrable constraints are validated at transaction level and not statement level. There are two types of deferrable constraints : INITIALLY IMMEDIATE, INITIALLY DEFERRED.

DIFERRABLE : Allows the database constraint to change its default constraint behaviour.

The deferrable behaviour can be specified by using the below key words following DIFFERABLE key word.

INITIALLY DEFERRED : Unless we commit the data into the database, it won’t check for constraints

INITIALLY IMMEDIATE : Constraint is checked when we update, delete or add rows from a table.

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?

The Foreign key of country (Code, Capital, Province) is referencing to the attributes (Country, Name, Province) of the City table. Similarly we observe that attributes (Country, Province) of City are referencing to attributes (Country, Name) of Province, attributes of Province (Country) is referencing to attributes (Code) of Country and attributes (Capital, Country, CapProv) of Province are referencing to attributes (Name, Country, Province) of City.

The problem here is at the beginning when no table and rows were created and while we try to create and insert, we will face error due to referential integrity constraint. For example at the time of inserting into country, the error is thrown because City table is not created (and Referential integrity constraint exists on Country since attributes in Country are referencing to a few attributes in City). Similar reasoning can be applied to the other tables as well.

This problem is solved by using the “INITIALLY DEFERRED DEFERRABLE” key words. This allows us to change the database constraint behaviour. Now unless we commit the data into the database, it won’t check for referential constraints. Hence we solve the problem by committing the data into the data base after we fill in the tables : Country, City, Province. And from now on the constraints (referential integrity constraint) will be checked (i.e. after committing and not before)

**SQL Queries:**

**1.**

SELECT

country.name

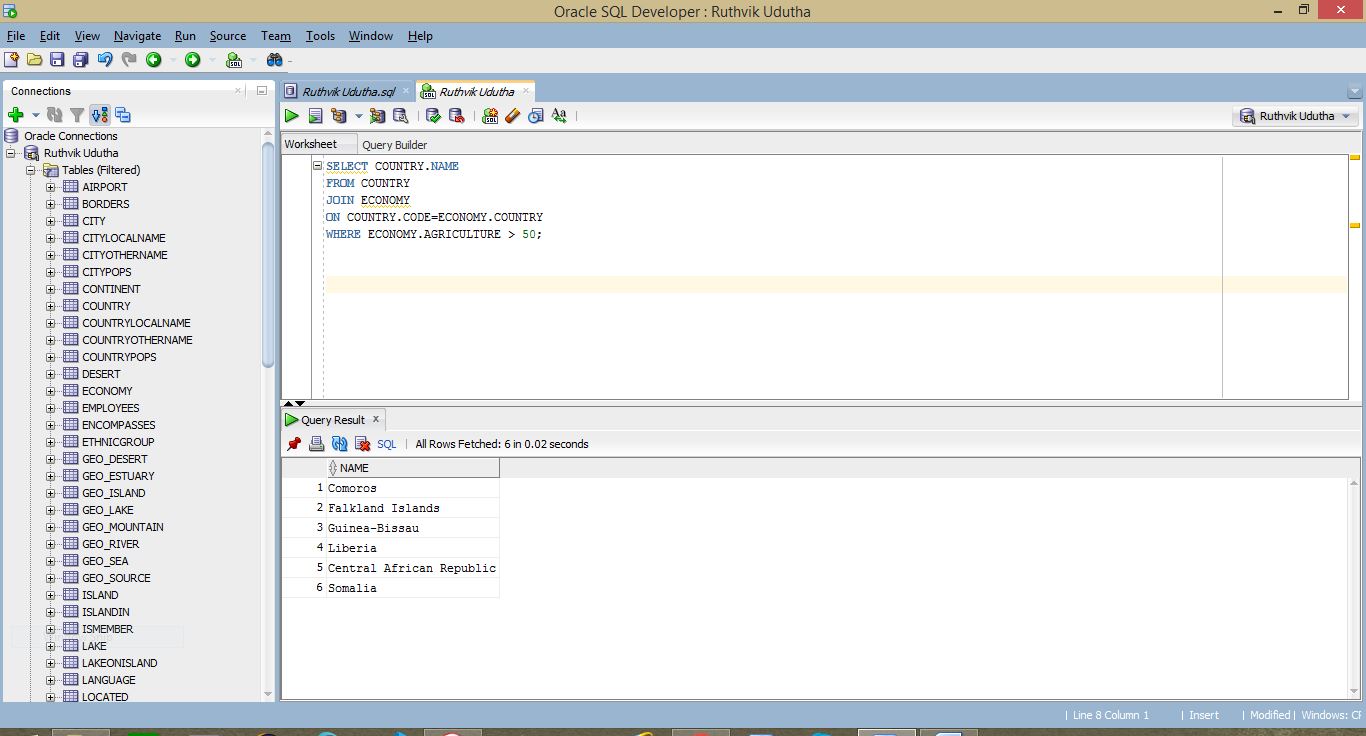
FROM

country

JOIN economy ON country.code = economy.country

WHERE

economy.agriculture > 50;



**2.**

SELECT

name

FROM

(

SELECT

name,

( population \* power((1 + nvl(population\_growth, 0)), 5) ) AS after\_5

FROM

(

SELECT

country.population,

population.population\_growth,

country.name

FROM

( country

JOIN population ON country.code = population.country )

)

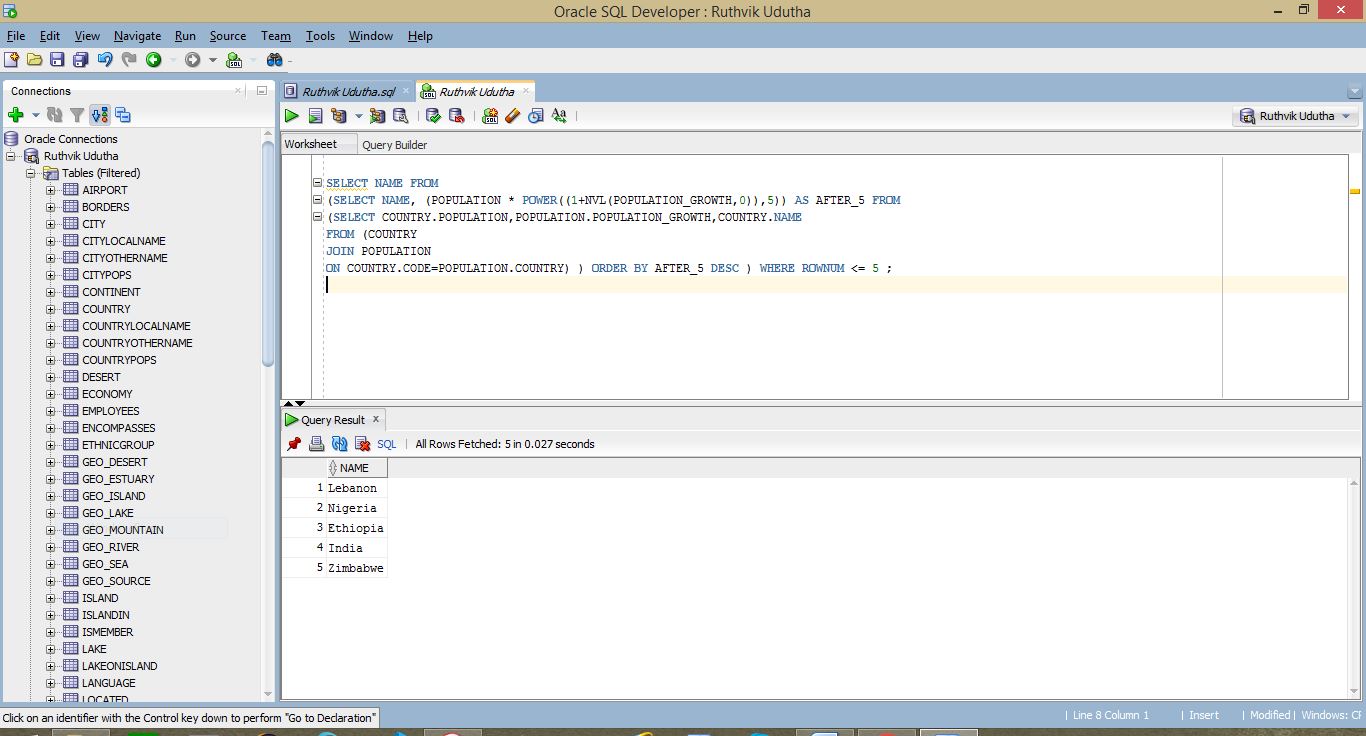
ORDER BY

after\_5 DESC

)

WHERE

ROWNUM <= 5;



**3.**

SELECT

(

SELECT

name

FROM

country

WHERE

code IN (

SELECT

"country\_used\_to"

FROM

(

SELECT

nvl(wasdependent, 0) AS "country\_used\_to",

COUNT(country) AS prev\_count

FROM

politics

GROUP BY

wasdependent

ORDER BY

prev\_count DESC

)

WHERE

"country\_used\_to" != '0'

AND prev\_count = (

SELECT

MAX(prev\_count)

FROM

(

SELECT

nvl(wasdependent, 0) AS "country\_used\_to",

COUNT(country) AS prev\_count

FROM

politics

GROUP BY

wasdependent

ORDER BY

prev\_count DESC

)

WHERE

"country\_used\_to" != '0'

)

)

) "c1",

(

SELECT

prev\_count

FROM

(

SELECT

nvl(wasdependent, 0) AS "country\_used\_to",

COUNT(country) AS prev\_count

FROM

politics

GROUP BY

wasdependent

ORDER BY

prev\_count DESC

)

WHERE

"country\_used\_to" != '0'

AND prev\_count = (

SELECT

MAX(prev\_count)

FROM

(

SELECT

nvl(wasdependent, 0) AS "country\_used\_to",

COUNT(country) AS prev\_count

FROM

politics

GROUP BY

wasdependent

ORDER BY

prev\_count DESC

)

WHERE

"country\_used\_to" != '0'

)

) "n1",

(

SELECT

name

FROM

country

WHERE

code IN (

SELECT

"country\_now"

FROM

(

SELECT

nvl(dependent, 0) AS "country\_now",

COUNT(country) AS now\_count

FROM

politics

GROUP BY

dependent

ORDER BY

now\_count DESC

)

WHERE

"country\_now" != '0'

AND now\_count = (

SELECT

MAX(now\_count)

FROM

(

SELECT

nvl(dependent, 0) AS "country\_now",

COUNT(country) AS now\_count

FROM

politics

GROUP BY

dependent

ORDER BY

now\_count DESC

)

WHERE

"country\_now" != '0'

)

)

) "c2",

(

SELECT

now\_count

FROM

(

SELECT

nvl(dependent, 0) AS "country\_now",

COUNT(country) AS now\_count

FROM

politics

GROUP BY

dependent

ORDER BY

now\_count DESC

)

WHERE

"country\_now" != '0'

AND now\_count = (

SELECT

MAX(now\_count)

FROM

(

SELECT

nvl(dependent, 0) AS "country\_now",

COUNT(country) AS now\_count

FROM

politics

GROUP BY

dependent

ORDER BY

now\_count DESC

)

WHERE

"country\_now" != '0'

)

) "n2",

( (

SELECT

prev\_count

FROM

(

SELECT

nvl(wasdependent, 0) AS "country\_used\_to",

COUNT(country) AS prev\_count

FROM

politics

GROUP BY

wasdependent

ORDER BY

prev\_count DESC

)

WHERE

"country\_used\_to" != '0'

AND prev\_count = (

SELECT

MAX(prev\_count)

FROM

(

SELECT

nvl(wasdependent, 0) AS "country\_used\_to",

COUNT(country) AS prev\_count

FROM

politics

GROUP BY

wasdependent

ORDER BY

prev\_count DESC

)

WHERE

"country\_used\_to" != '0'

)

) - (

SELECT

now\_count

FROM

(

SELECT

nvl(dependent, 0) AS "country\_now",

COUNT(country) AS now\_count

FROM

politics

GROUP BY

dependent

ORDER BY

now\_count DESC

)

WHERE

"country\_now" != '0'

AND now\_count = (

SELECT

MAX(now\_count)

FROM

(

SELECT

nvl(dependent, 0) AS "country\_now",

COUNT(country) AS now\_count

FROM

politics

GROUP BY

dependent

ORDER BY

now\_count DESC

)

WHERE

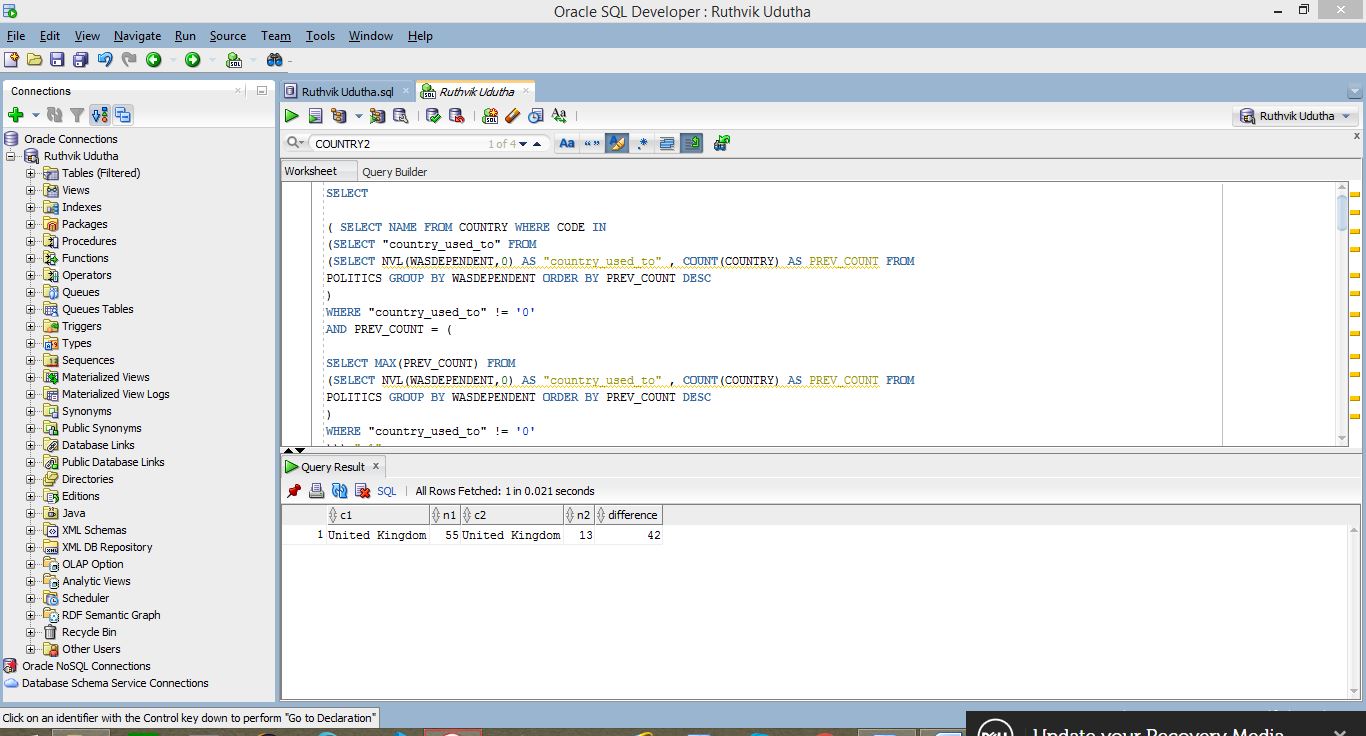
"country\_now" != '0'

)

) ) "difference"

FROM

dual



**4.**

SELECT

c.name

FROM

( country c

JOIN religion r ON c.code = r.country )

WHERE

percentage > 80

AND c.name IN (

SELECT

cname

FROM

(

SELECT

name AS cname,

COUNT(DISTINCT relname) AS relnum

FROM

(

SELECT

country.name,

religion.name AS relname,

religion.percentage

FROM

( country

JOIN religion ON country.code = religion.country )

)

GROUP BY

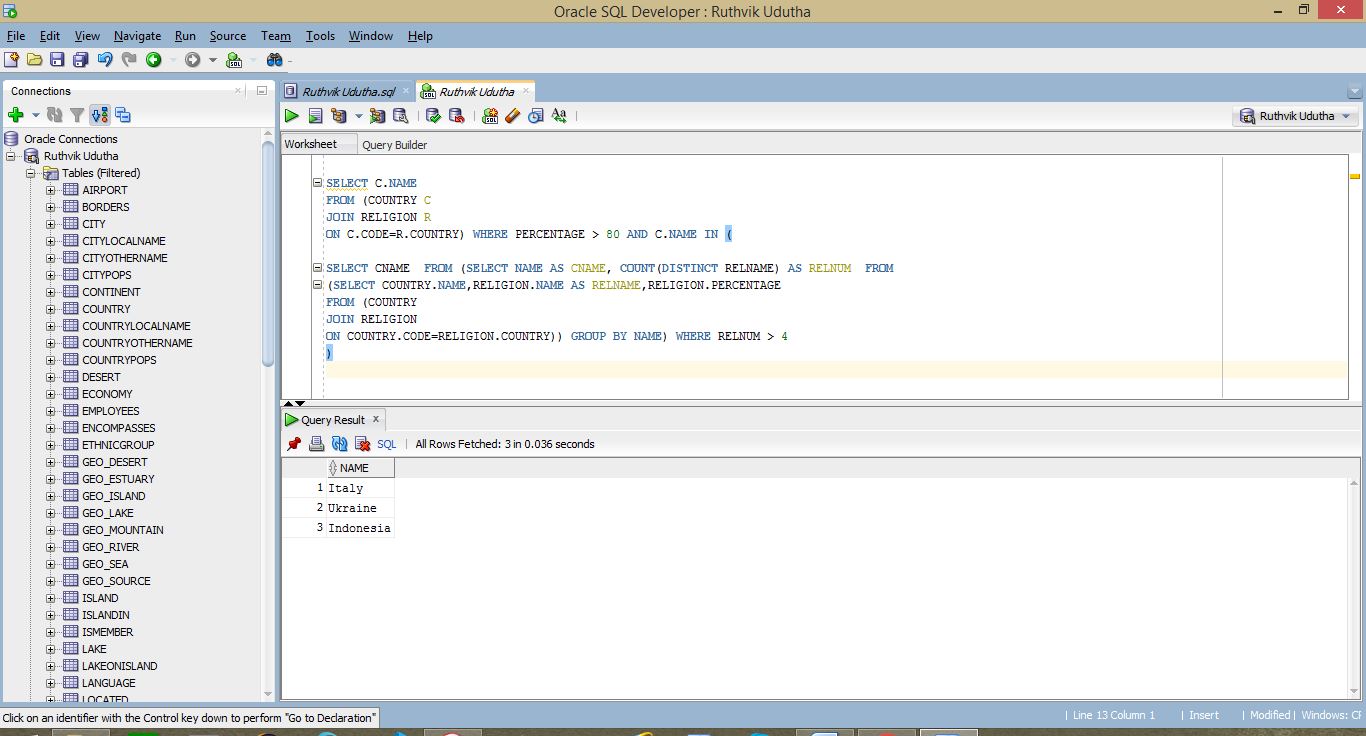
name

)

WHERE

relnum > 4

)



**5.**

SELECT

SUM(length)

FROM

(

SELECT

\*

FROM

borders

WHERE

country1 IN (

SELECT

code

FROM

country

WHERE

name = 'China'

)

OR country2 IN (

SELECT

code

FROM

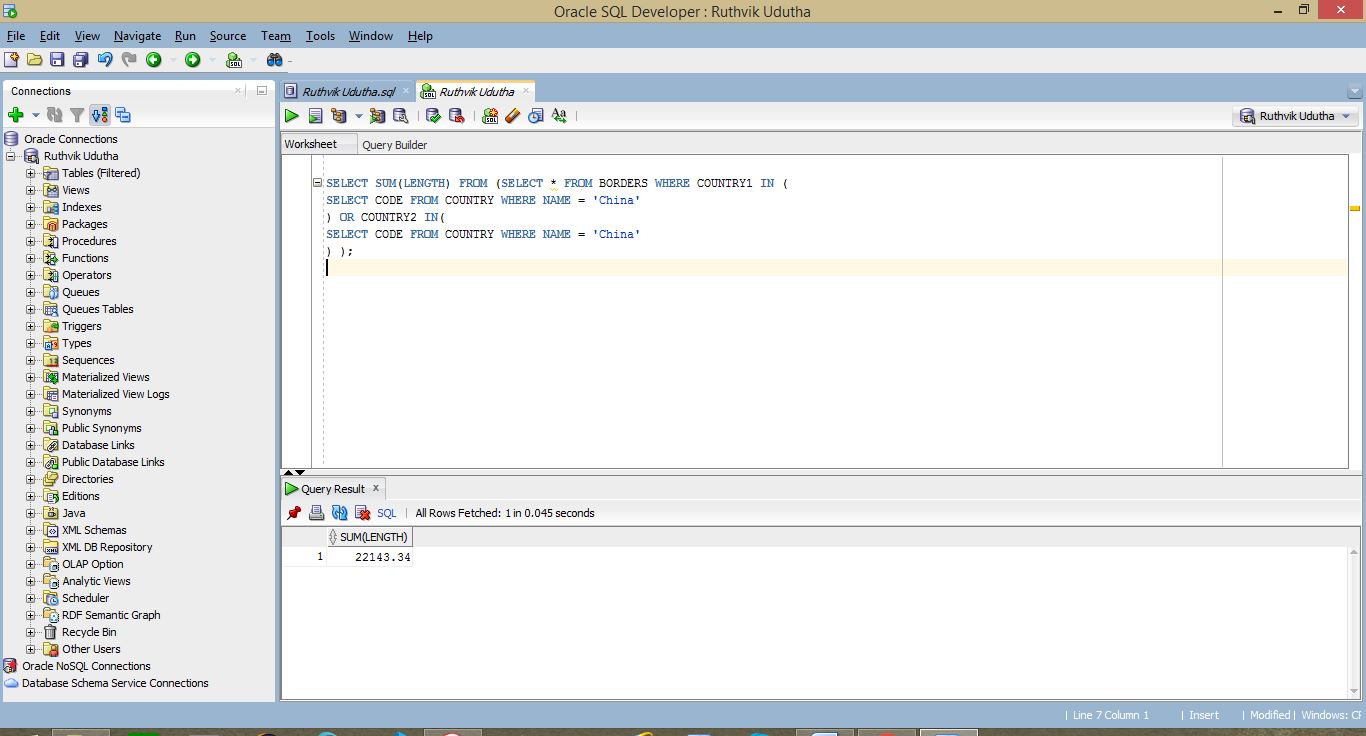
country

WHERE

name = 'China'

)

);



**6.**

SELECT

\*

FROM

(

SELECT

name,

SUM(pop\_per\_rel) AS num\_followers

FROM

(

SELECT

r.country,

r.name,

r.percentage,

c.population,

( ( percentage / 100 ) \* population ) AS pop\_per\_rel

FROM

(

SELECT

\*

FROM

religion

) r

JOIN (

SELECT

code,

population

FROM

country

) c ON r.country = c.code

)

GROUP BY

name

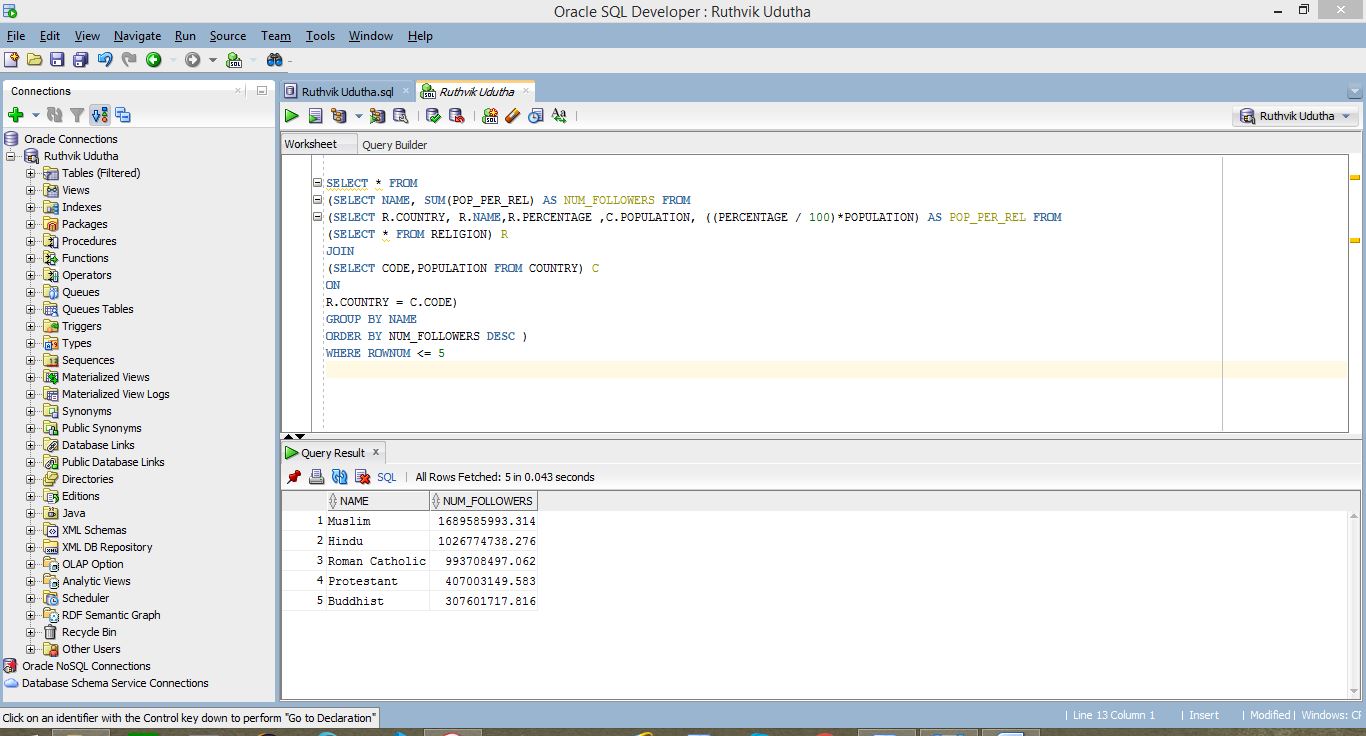
ORDER BY

num\_followers DESC

)

WHERE

ROWNUM <= 5



**7.**

SELECT DISTINCT

name

FROM

(

SELECT

l.name,

l.elevation,

g.country

FROM

geo\_lake g

JOIN lake l ON g.lake = l.name

WHERE

g.country IN (

SELECT

code

FROM

country

WHERE

name = 'United States'

)

)

WHERE

elevation > (

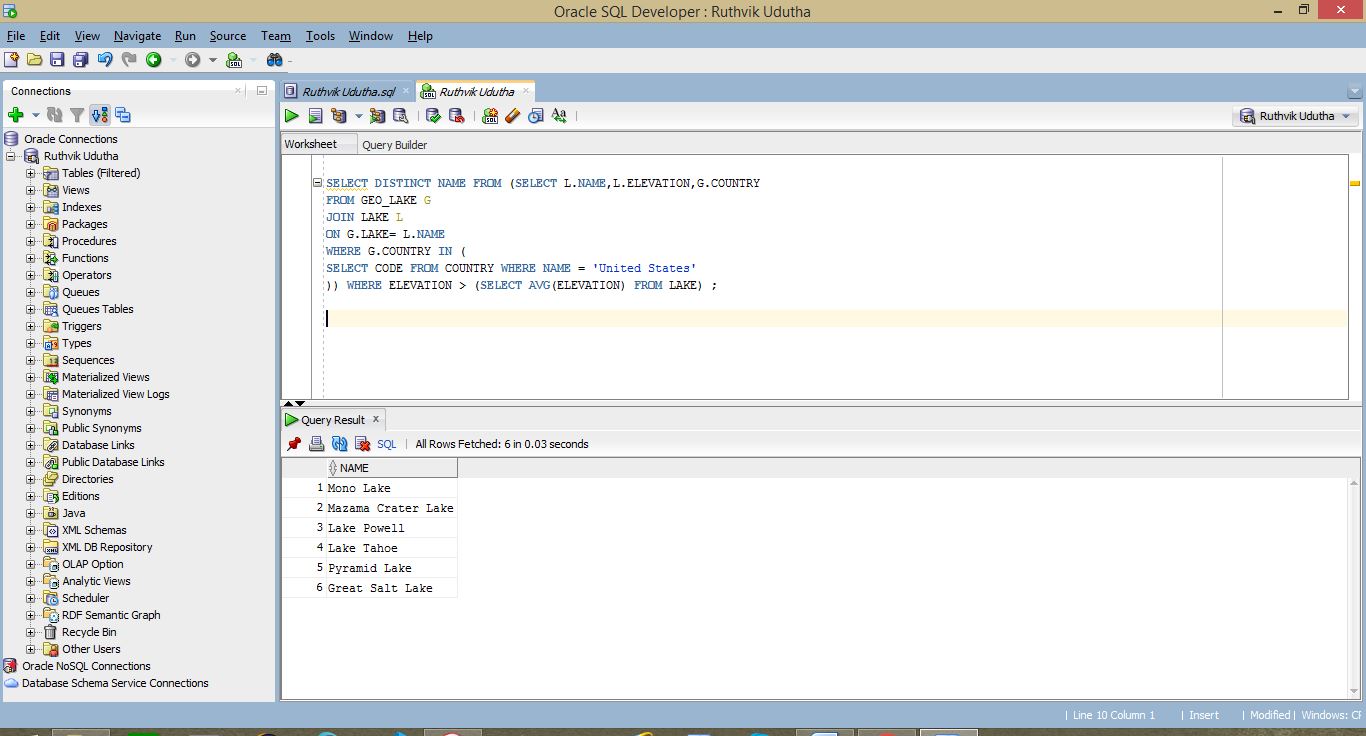
SELECT

AVG(elevation)

FROM

lake

);



**8.**

SELECT

\*

FROM

(

SELECT

t.name AS mountain\_name,

t.province,

j.population\_density

FROM

(

SELECT

m.name,

m.type,

g.province,

g.country

FROM

mountain m

JOIN geo\_mountain g ON m.name = g.mountain

WHERE

type = 'volcano'

) t

JOIN (

SELECT

name,

( population / area ) AS population\_density

FROM

province

) j ON t.province = j.name

ORDER BY

population\_density DESC

)

WHERE

population\_density IN (

(

SELECT

MAX(population\_density)

FROM

(

SELECT

t.name AS mountain\_name,

t.province,

j.population\_density

FROM

(

SELECT

m.name,

m.type,

g.province,

g.country

FROM

mountain m

JOIN geo\_mountain g ON m.name = g.mountain

WHERE

type = 'volcano'

) t

JOIN (

SELECT

name,

( population / area ) AS population\_density

FROM

province

) j ON t.province = j.name

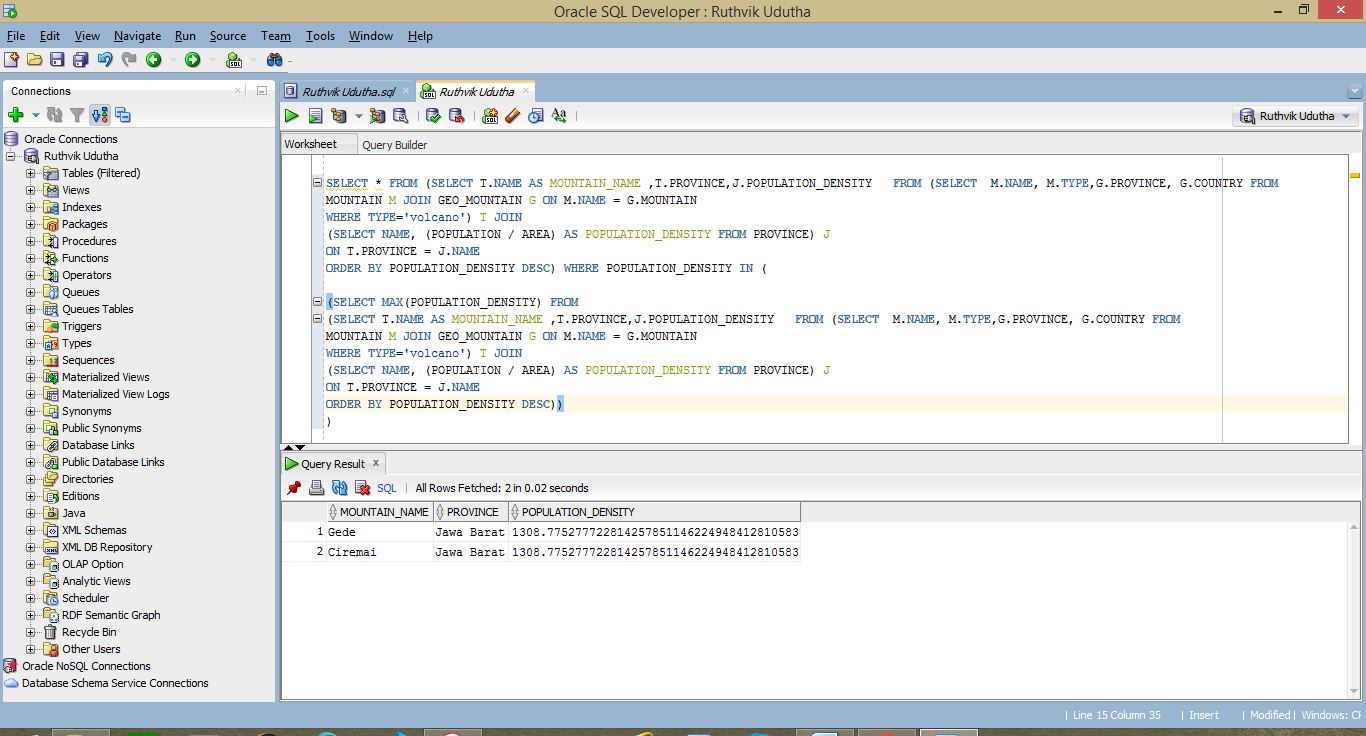
ORDER BY

population\_density DESC

)

)

)



**9.**

SELECT

\*

FROM

(

SELECT

t2.province,

t2.island\_count,

t2.country,

e.gdp

FROM

(

SELECT

t.province,

t.island\_count,

p.country

FROM

(

SELECT

\*

FROM

(

SELECT

province,

COUNT(island) AS island\_count

FROM

geo\_island

GROUP BY

province

)

WHERE

island\_count > 2

) t

JOIN province p ON t.province = p.name

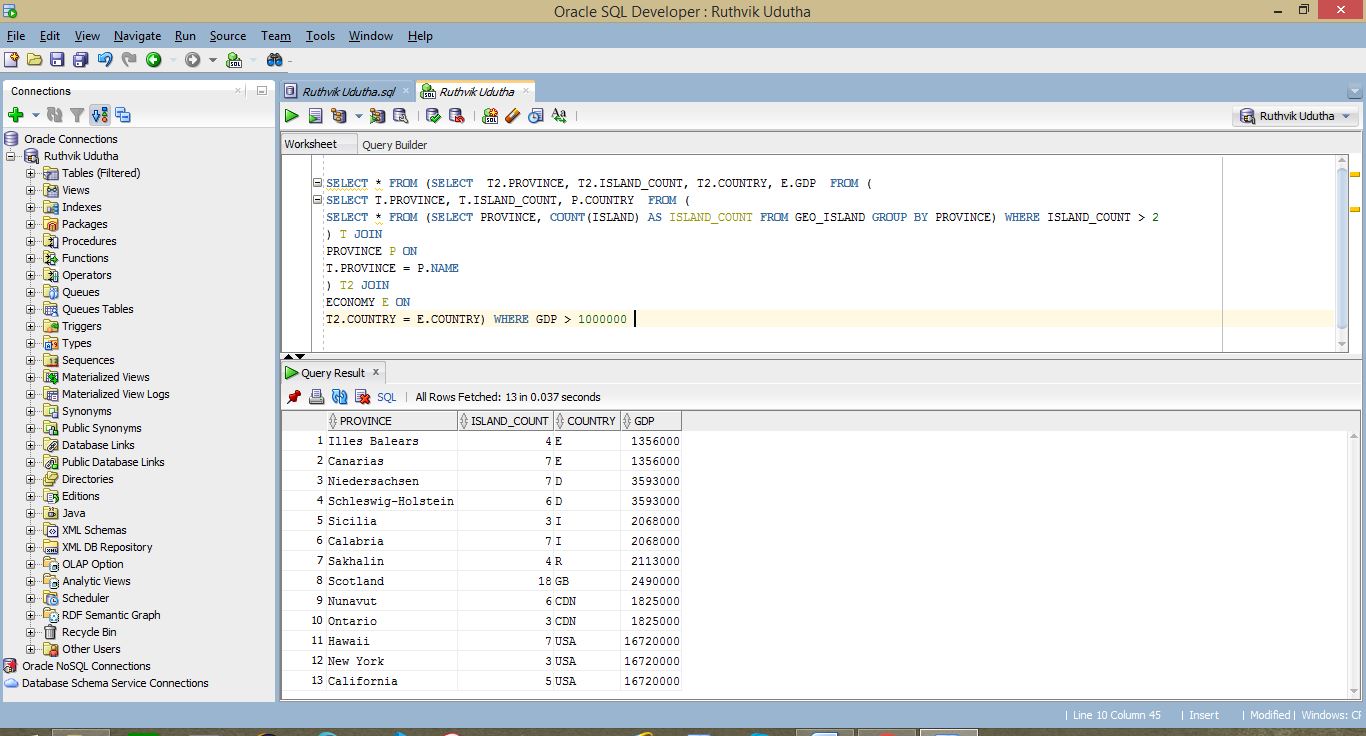
) t2

JOIN economy e ON t2.country = e.country

)

WHERE

gdp > 1000000



**10.**

SELECT

name,

length

FROM

(

SELECT

\*

FROM

(

SELECT

\*

FROM

river

WHERE

sea = 'Atlantic Ocean'

) r

JOIN (

SELECT

river,

COUNT(name) AS lake\_count

FROM

lake

GROUP BY

river

) l ON r.name = l.river

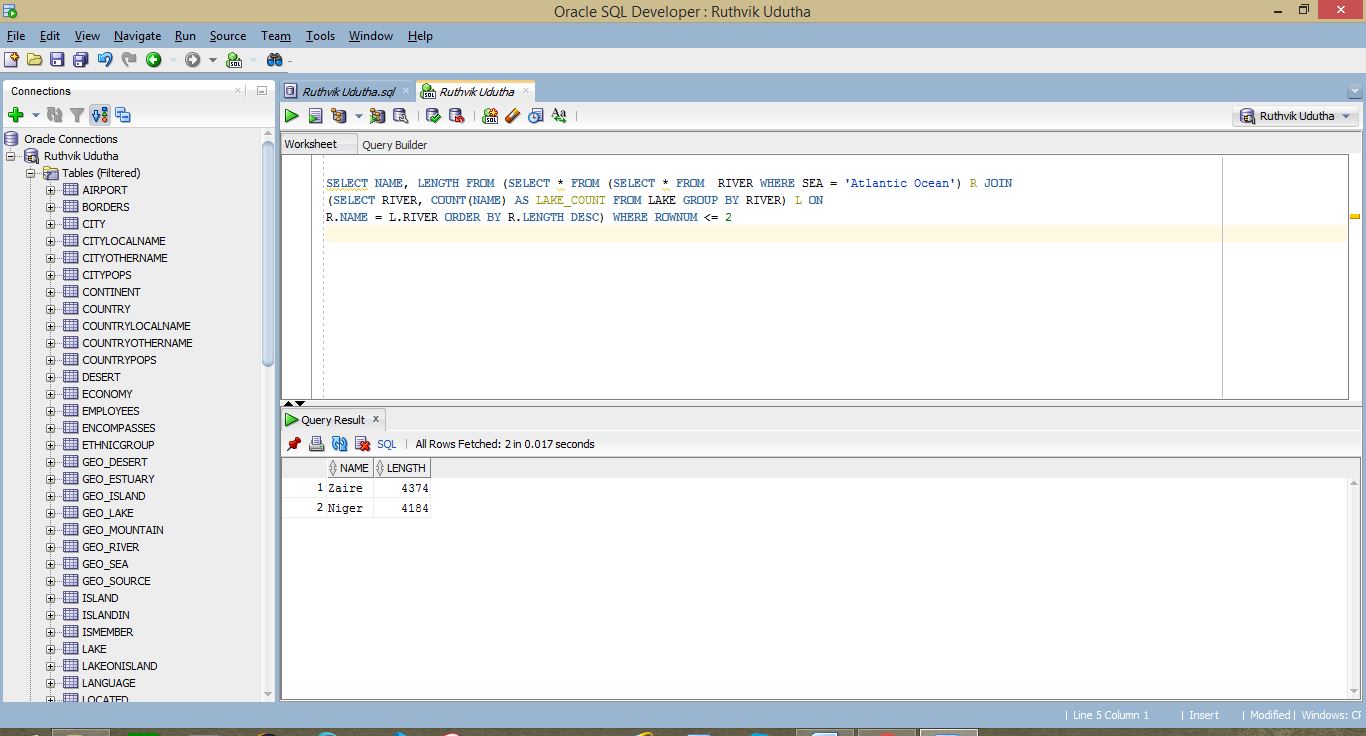
ORDER BY

r.length DESC

)

WHERE

ROWNUM <= 2



**11.**

SELECT

name

FROM

country

WHERE

code IN (

SELECT DISTINCT

r\_country

FROM

(

SELECT

lake,

country AS l\_country

FROM

geo\_lake

WHERE

lake IN (

SELECT

lake

FROM

(

SELECT

lake,

COUNT(province) AS province\_count

FROM

geo\_lake

GROUP BY

lake

)

WHERE

province\_count > 3

)

) l

JOIN (

SELECT

\*

FROM

(

SELECT

country AS r\_country,

COUNT(river) AS river\_count

FROM

geo\_river

GROUP BY

country

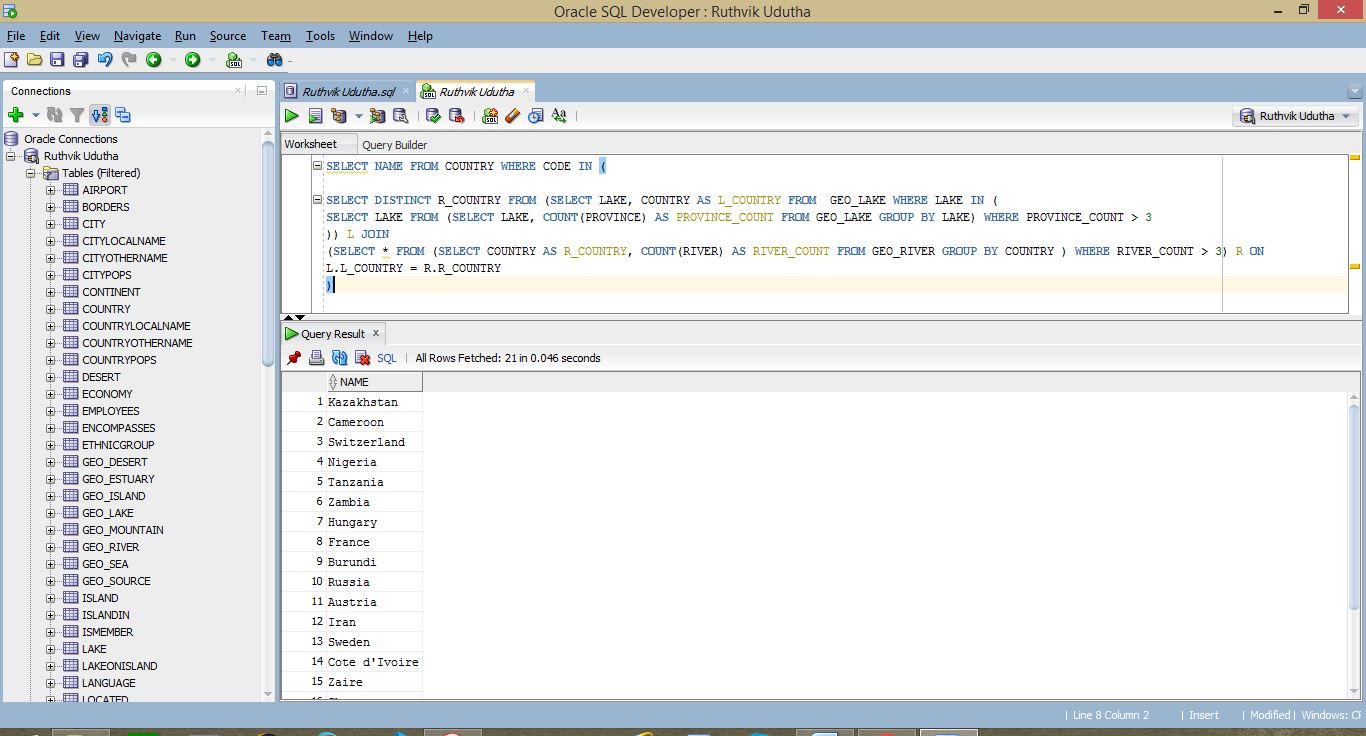
)

WHERE

river\_count > 3

) r ON l.l\_country = r.r\_country

)



**13.**

SELECT

continent,

MAX(highest\_peak) AS highest\_elevation

FROM

(

SELECT

\*

FROM

(

SELECT

country,

MAX(elevation) AS highest\_peak

FROM

(

SELECT

name,

elevation,

country

FROM

mountain m

JOIN geo\_mountain gm ON m.name = gm.mountain

)

GROUP BY

country

) p

JOIN (

SELECT

country,

continent

FROM

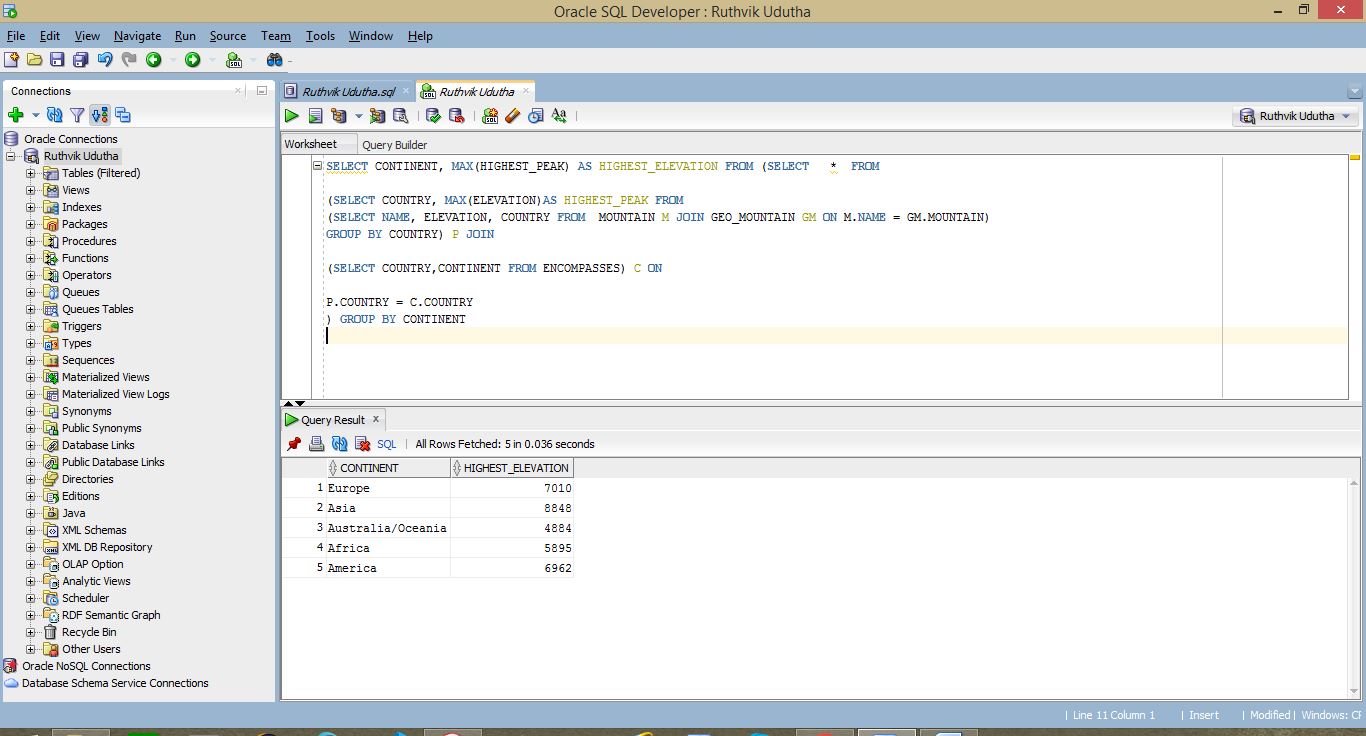
encompasses

) c ON p.country = c.country

)

GROUP BY

continent



**14.**

SELECT

c.name,

temp.max\_elevation,

temp.max\_depth

FROM

country c

JOIN (

SELECT

mountain.country,

mountain.max\_elevation,

sea.max\_depth

FROM

( (

SELECT

country,

MAX(elevation) AS max\_elevation

FROM

(

SELECT

\*

FROM

mountain m

JOIN geo\_mountain g ON m.name = g.mountain

)

GROUP BY

country

) mountain

JOIN (

SELECT

country,

MAX(depth) AS max\_depth

FROM

(

SELECT

\*

FROM

sea s

JOIN geo\_sea gs ON s.name = gs.sea

)

GROUP BY

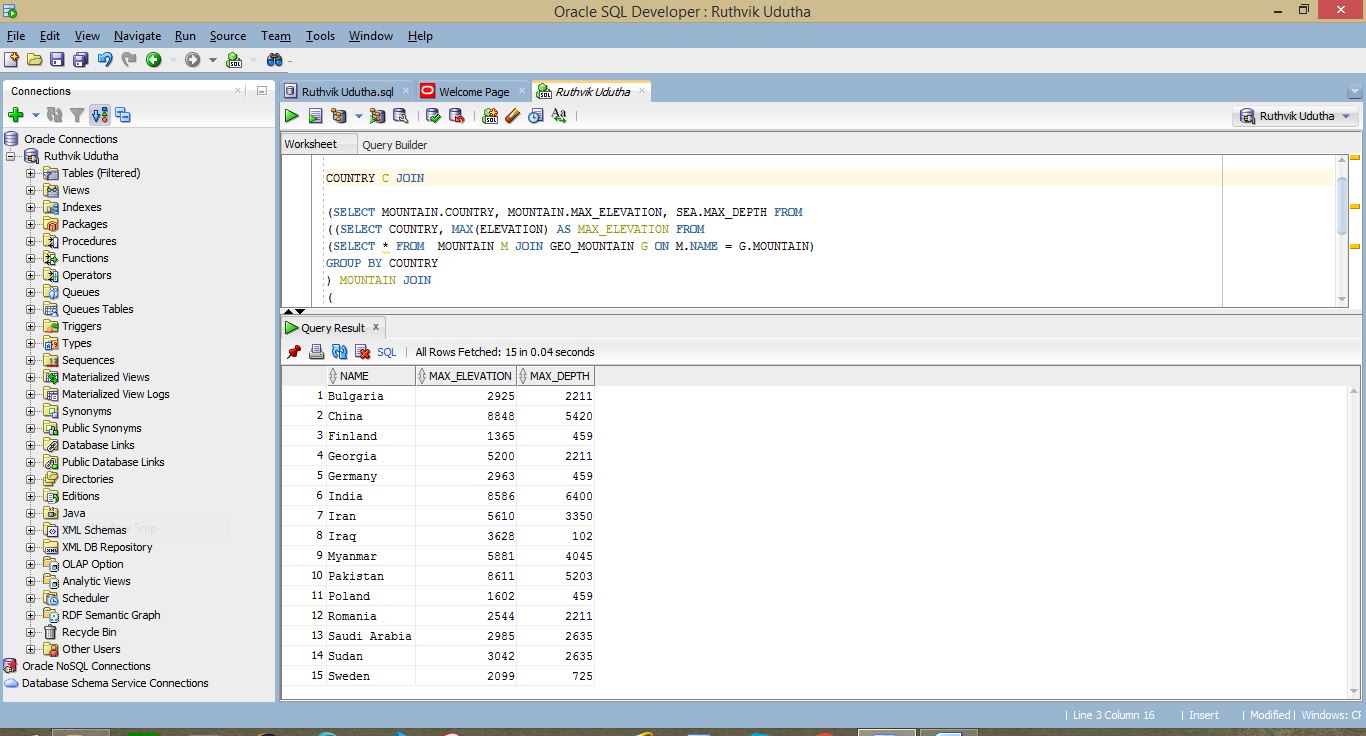
country

) sea ON mountain.country = sea.country )

WHERE

mountain.max\_elevation > sea.max\_depth

) temp ON c.code = temp.country



**15.**

SELECT

name,

continent

FROM

(

SELECT

cl1.name,

cl1.latitude,

co1.continent

FROM

(

SELECT

name,

latitude

FROM

city

WHERE

country NOT IN (

SELECT

country AS country\_encp

FROM

encompasses

WHERE

continent = 'Asia'

)

) cl1

JOIN (

SELECT

continent,

MAX(latitude) AS max\_latitude

FROM

(

SELECT

ci.country,

ci.name AS city\_name,

c.continent,

ci.latitude

FROM

(

SELECT

country AS country\_encp,

continent

FROM

encompasses

WHERE

continent != 'Asia'

AND country NOT IN (

SELECT

country AS country\_encp

FROM

encompasses

WHERE

continent = 'Asia'

)

) c

JOIN (

SELECT

\*

FROM

city

WHERE

country IN (

SELECT

country

FROM

encompasses

WHERE

continent != 'Asia'

AND country NOT IN (

SELECT

country AS country\_encp

FROM

encompasses

WHERE

continent = 'Asia'

)

)

) ci ON c.country\_encp = ci.country

)

GROUP BY

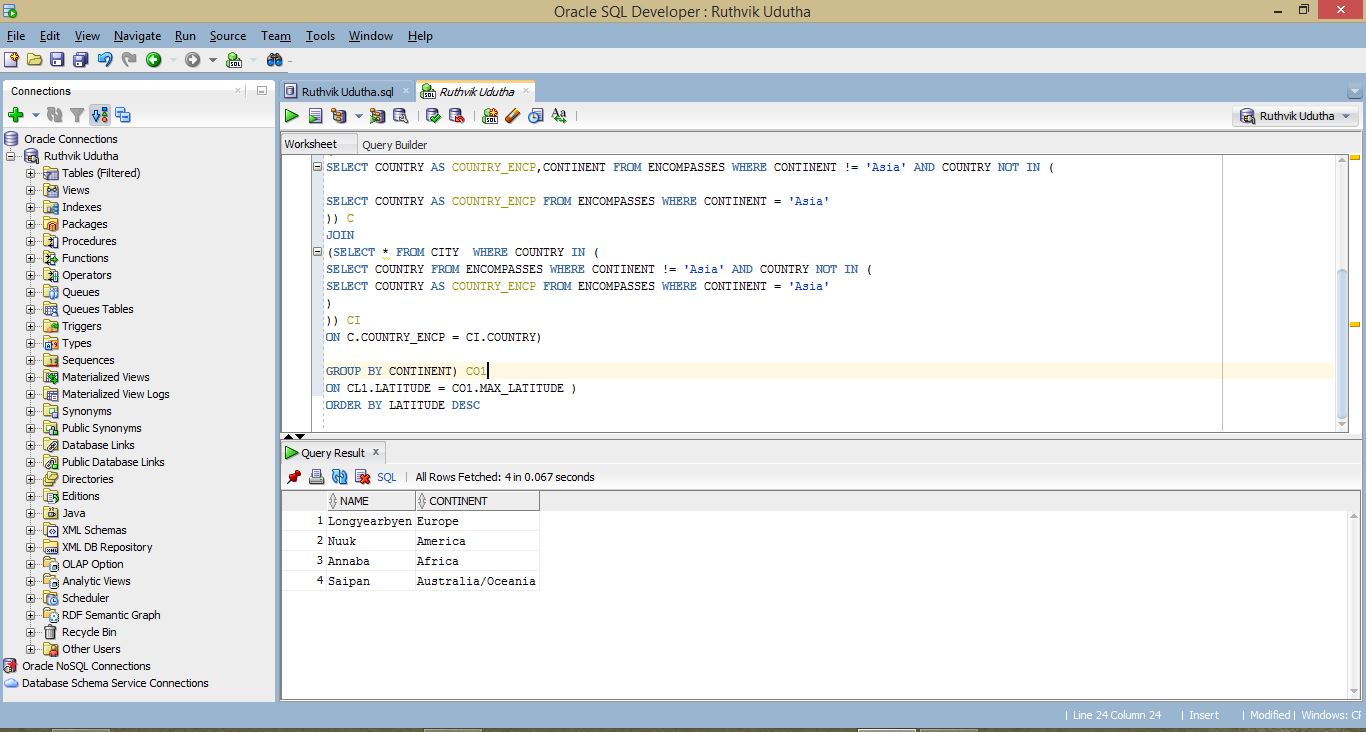
continent

) co1 ON cl1.latitude = co1.max\_latitude

)

ORDER BY

latitude DESC



**16.**

SELECT

name

FROM

country

WHERE

code IN (

SELECT

country

FROM

city

WHERE

name IN (

SELECT

capital

FROM

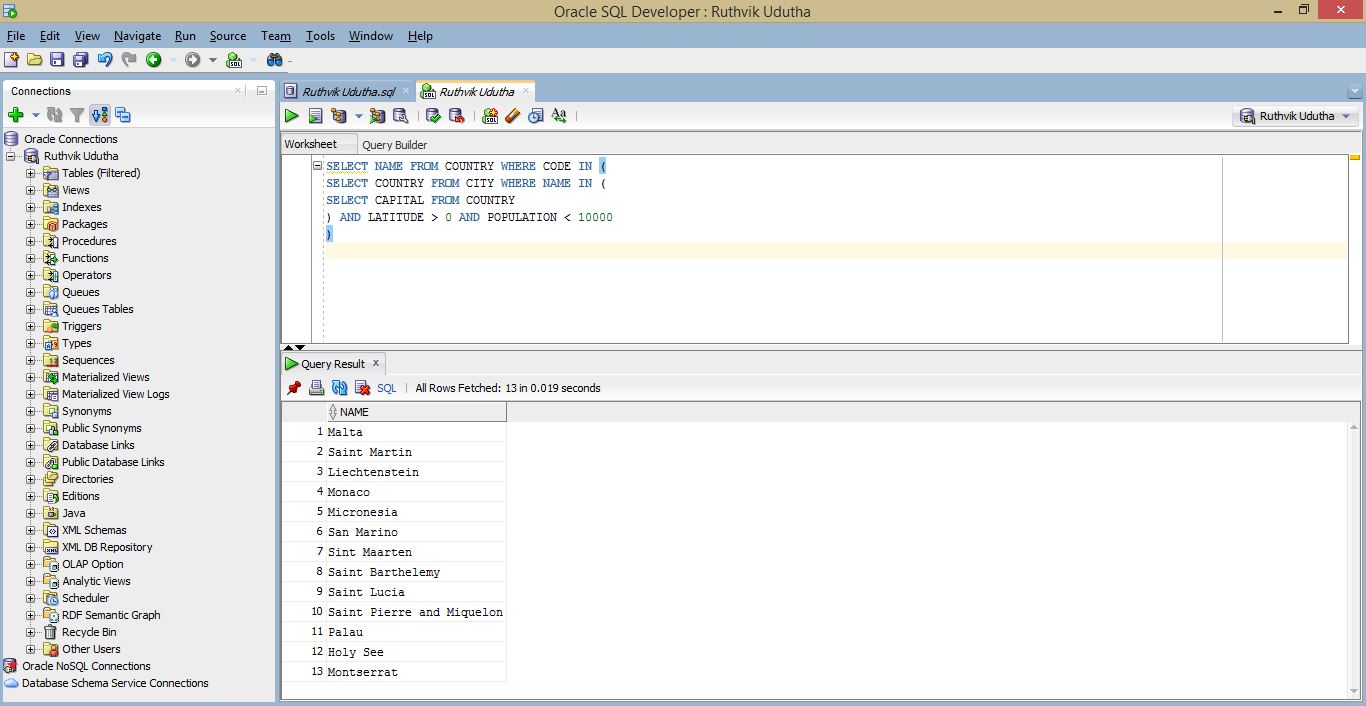
country

)

AND latitude > 0

AND population < 10000

)



**17.**

SELECT

(

SELECT

SUM(area) AS "top10"

FROM

(

SELECT

name,

area

FROM

country

ORDER BY

area DESC

)

WHERE

ROWNUM <= 10

) "top10",

(

SELECT

SUM(area) AS "rest\_world"

FROM

(

SELECT

name,

area

FROM

country

ORDER BY

area ASC

)

WHERE

ROWNUM < (

SELECT

COUNT(\*)

FROM

(

SELECT

name,

area

FROM

country

ORDER BY

area DESC

)

) - 9

) "rest\_world",

( (

SELECT

SUM(area) AS "top10"

FROM

(

SELECT

name,

area

FROM

country

ORDER BY

area DESC

)

WHERE

ROWNUM <= 10

) - ( (

SELECT

SUM(area) AS "rest\_world"

FROM

(

SELECT

name,

area

FROM

country

ORDER BY

area ASC

)

WHERE

ROWNUM < (

SELECT

COUNT(\*)

FROM

(

SELECT

name,

area

FROM

country

ORDER BY

area DESC

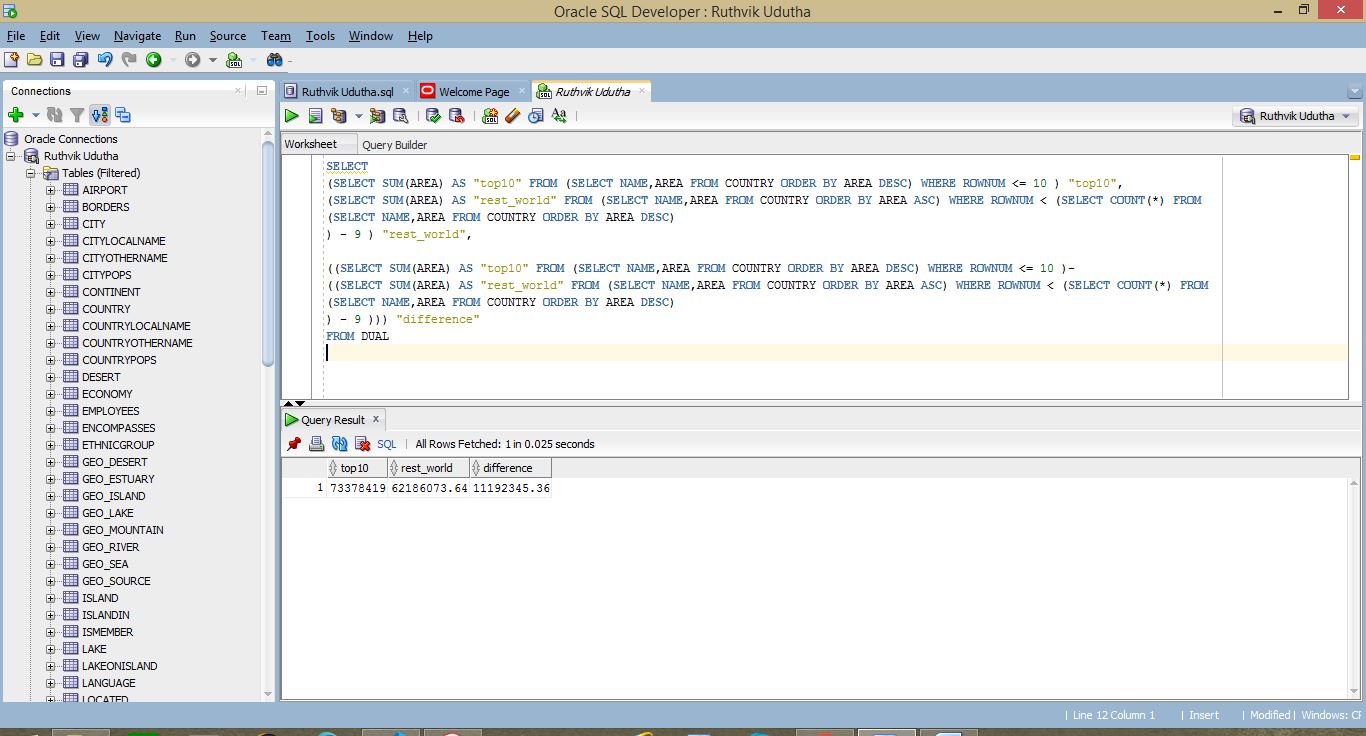
)

) - 9

) ) ) "difference"

FROM

dual



**18.**

SELECT

name

FROM

country

WHERE

code IN (

SELECT

country

FROM

(

SELECT

country,

COUNT(continent) AS cont\_count

FROM

encompasses

GROUP BY

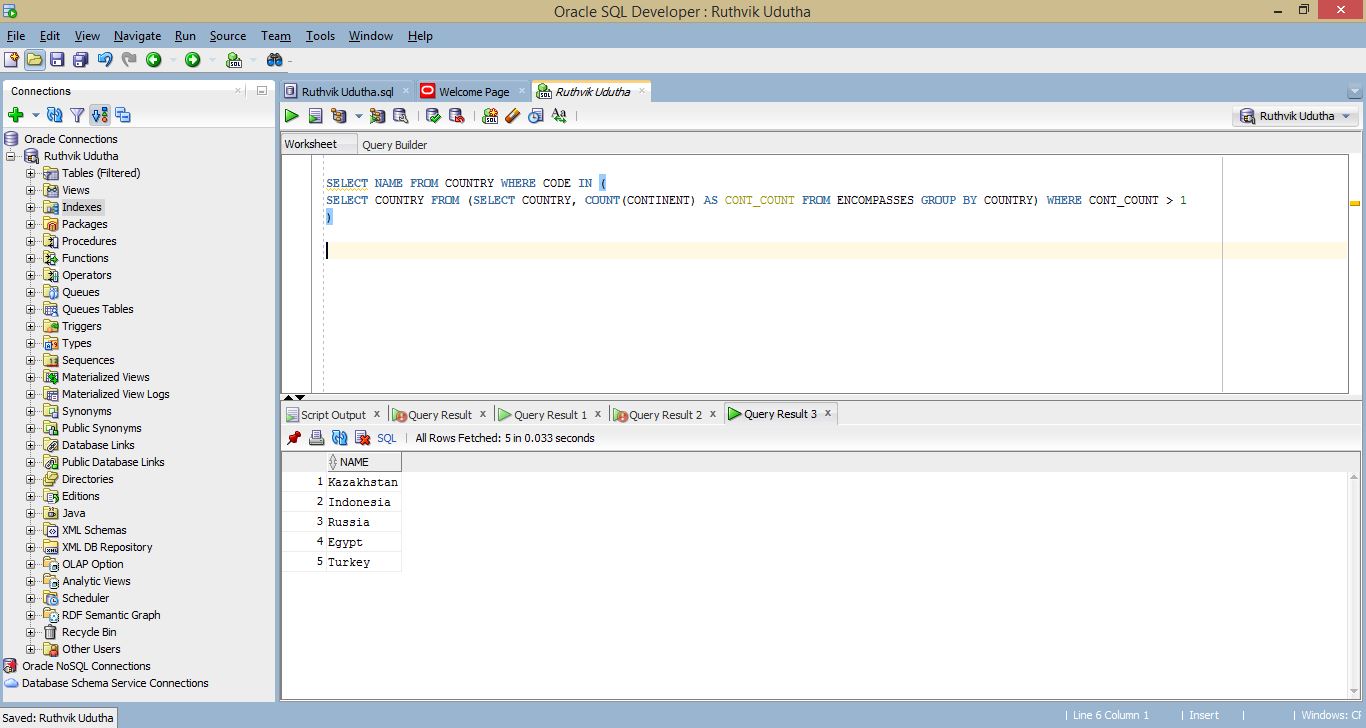
country

)

WHERE

cont\_count > 1

)



**19.**

SELECT

i.name,

i.area

FROM

(

SELECT

name,

area

FROM

island

WHERE

area > 1000

) i

JOIN (

SELECT

\*

FROM

geo\_island

WHERE

country IN (

SELECT

country

FROM

encompasses

WHERE

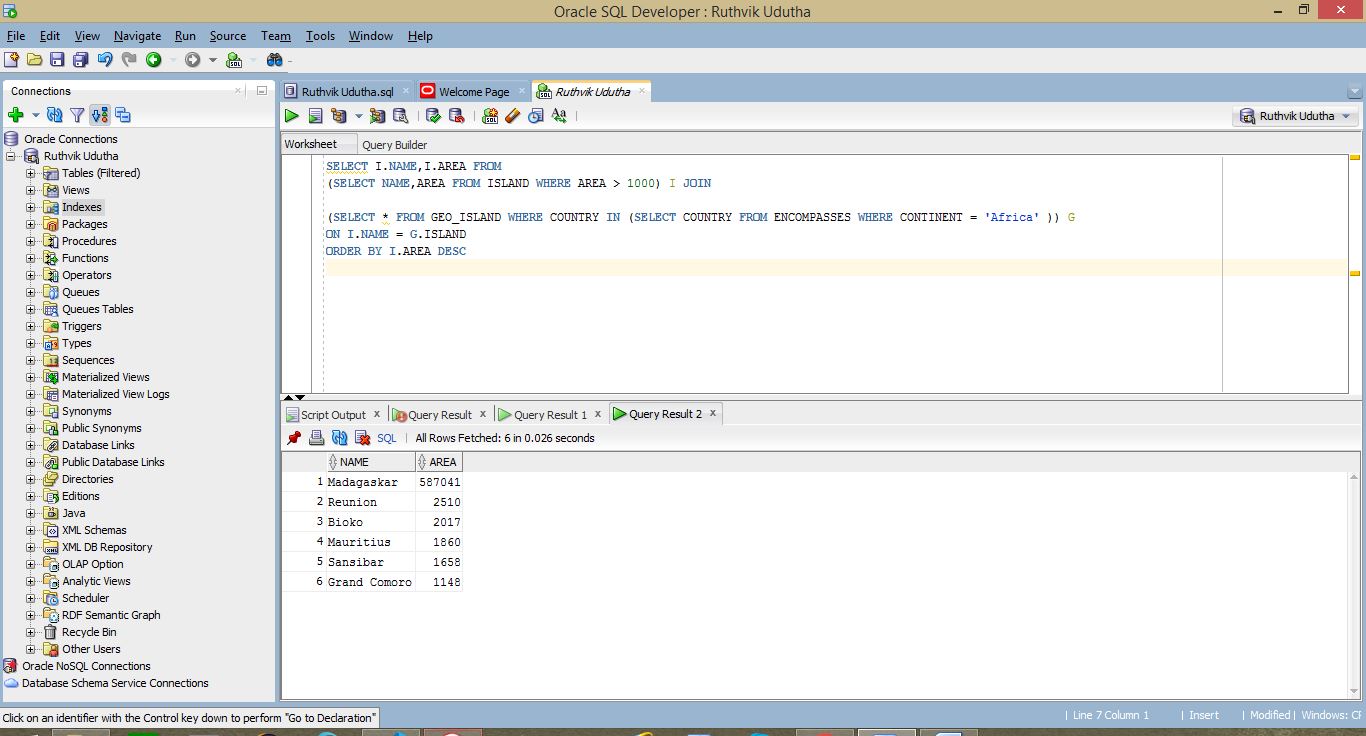
continent = 'Africa'

)

) g ON i.name = g.island

ORDER BY

i.area DESC



**20.**

SELECT

t.name,

g.gdp

FROM

(

SELECT

country,

gdp

FROM

economy

WHERE

country IN (

SELECT

country

FROM

religion

WHERE

country IN (

SELECT

country

FROM

ismember

WHERE

organization = 'NATO'

)

AND name = 'Muslim'

AND percentage > 5

)

) g

JOIN (

SELECT

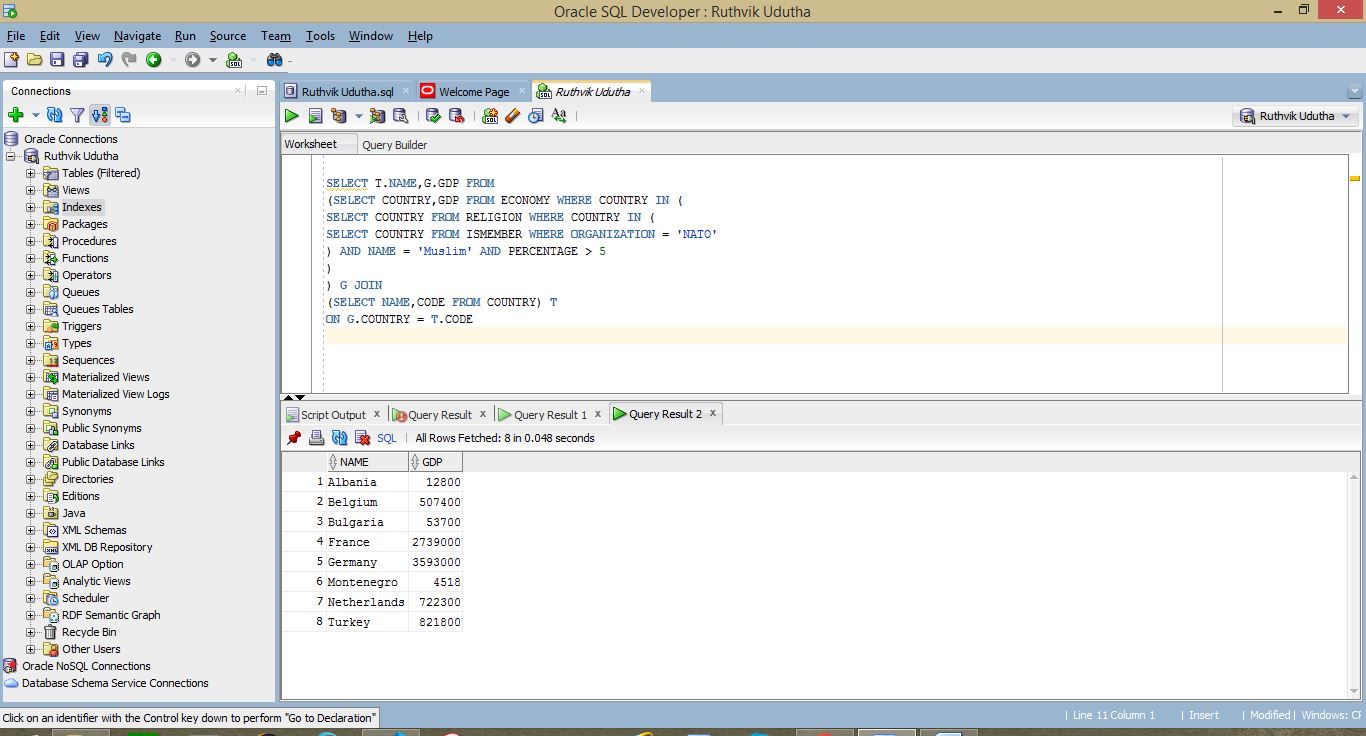
name,

code

FROM

country

) t ON g.country = t.code



**21.**

SELECT

river

FROM

(

SELECT

river,

COUNT(\*) AS prov\_count

FROM

geo\_river

GROUP BY

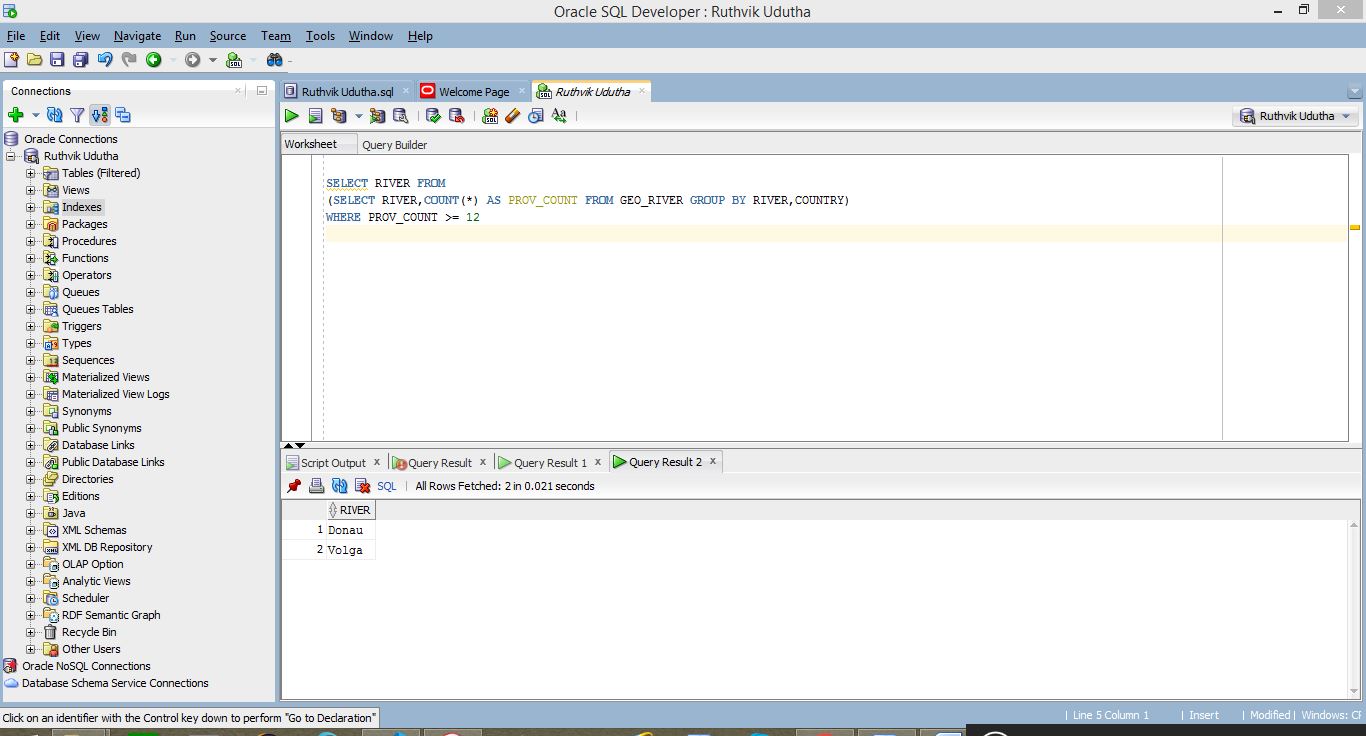
river,

country

)

WHERE

prov\_count >= 12



**22.**

SELECT

river\_name,

length

FROM

(

SELECT DISTINCT

g.river AS river\_name,

g.country,

r.length

FROM

( (

SELECT

river,

country

FROM

geo\_river

) g

JOIN (

SELECT

name,

length

FROM

river

) r ON g.river = r.name )

WHERE

country IN (

SELECT

country

FROM

encompasses

WHERE

continent = 'America'

)

ORDER BY

r.length DESC

)

WHERE

length = (

SELECT

MAX(r.length)

FROM

( (

SELECT

river,

country

FROM

geo\_river

) g

JOIN (

SELECT

name,

length

FROM

river

) r ON g.river = r.name )

WHERE

country IN (

SELECT

country

FROM

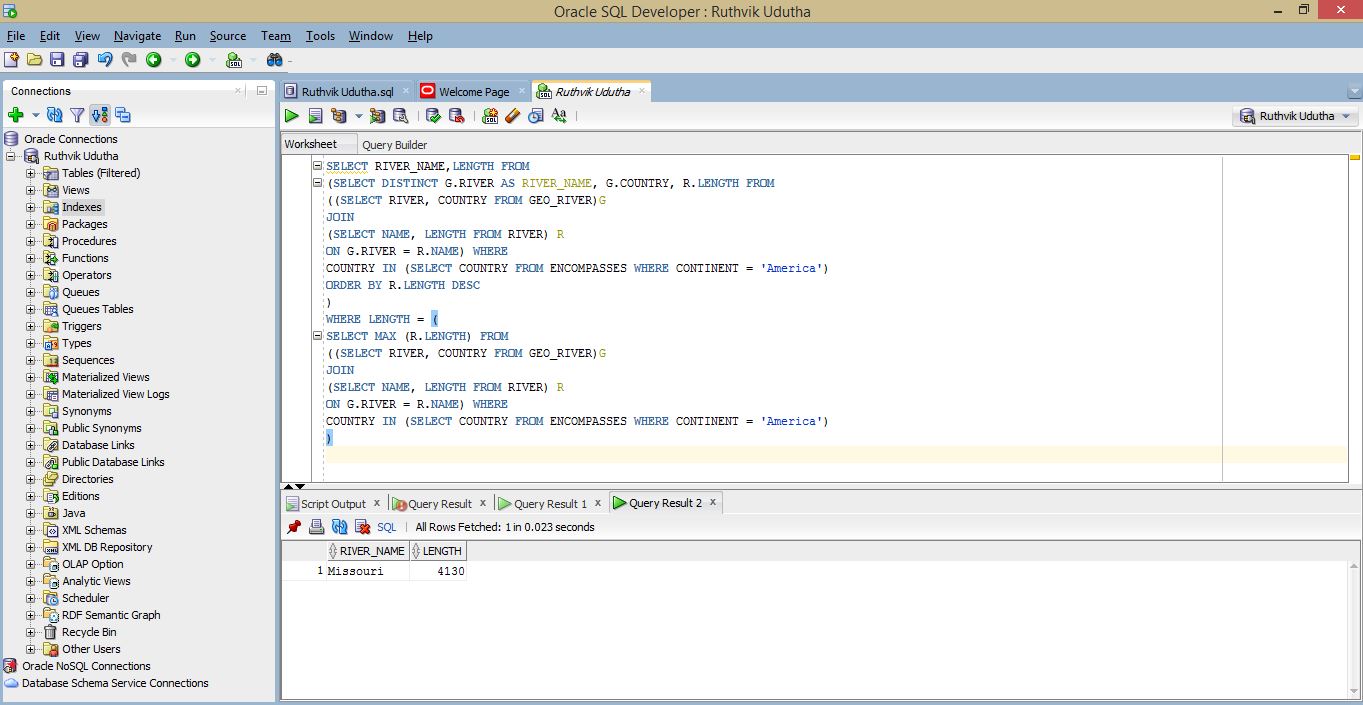
encompasses

WHERE

continent = 'America'

)

)



**23.**

SELECT

\*

FROM

(

SELECT DISTINCT

i.province,

i.count\_islands,

g.country AS country\_code

FROM

(

SELECT

COUNT(island) AS count\_islands,

province

FROM

geo\_island

GROUP BY

province

ORDER BY

count\_islands DESC

) i

JOIN geo\_island g

ON i.province = g.province

ORDER BY

count\_islands DESC

)

WHERE

count\_islands = (

SELECT

MAX(count\_islands)

FROM

(

SELECT

COUNT(island) AS count\_islands,

province

FROM

geo\_island

GROUP BY

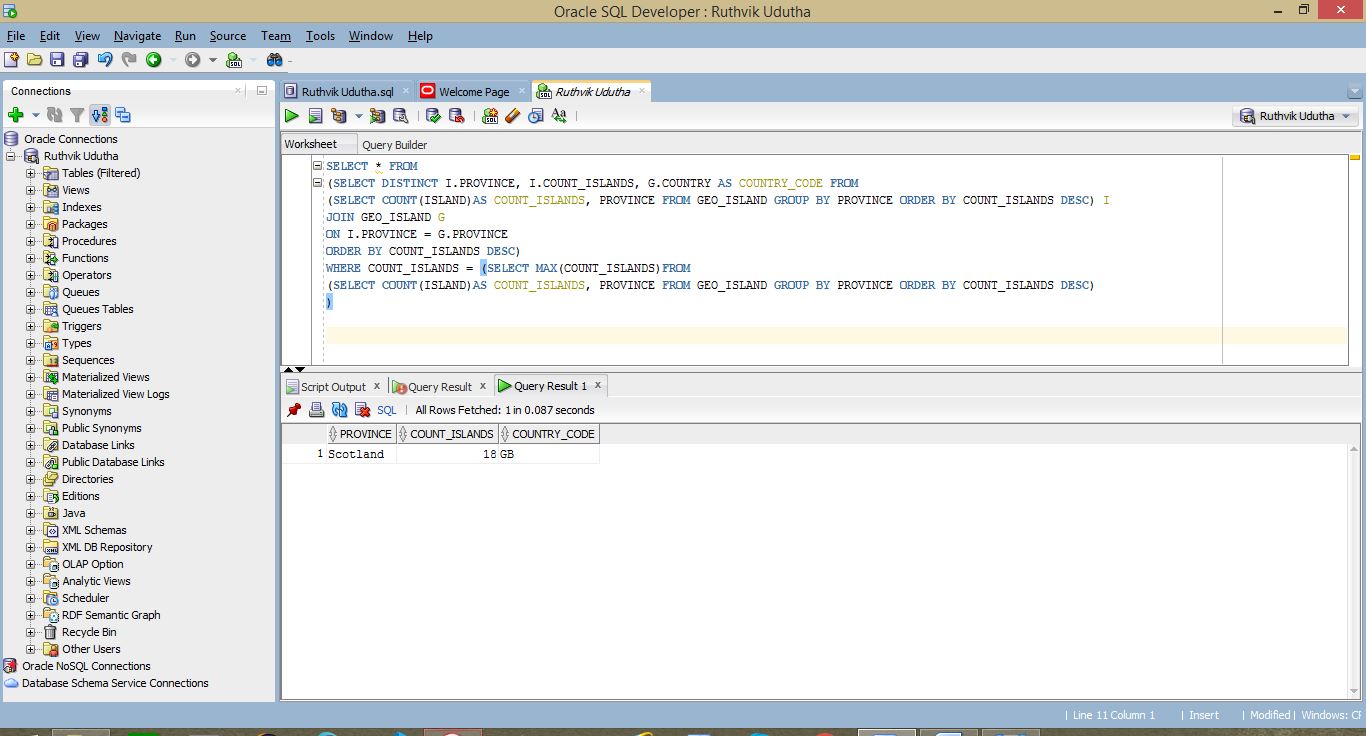
province

ORDER BY

count\_islands DESC

)

)



**24.**

SELECT

name AS "Country Name",

population\_density AS "Population Density",

( ( population / wp ) \* 100 ) AS "Percentage"

FROM

(

SELECT

name,

population,

( population / area ) AS population\_density,

(

SELECT

SUM(population) AS world\_pop

FROM

country

) AS wp

FROM

country

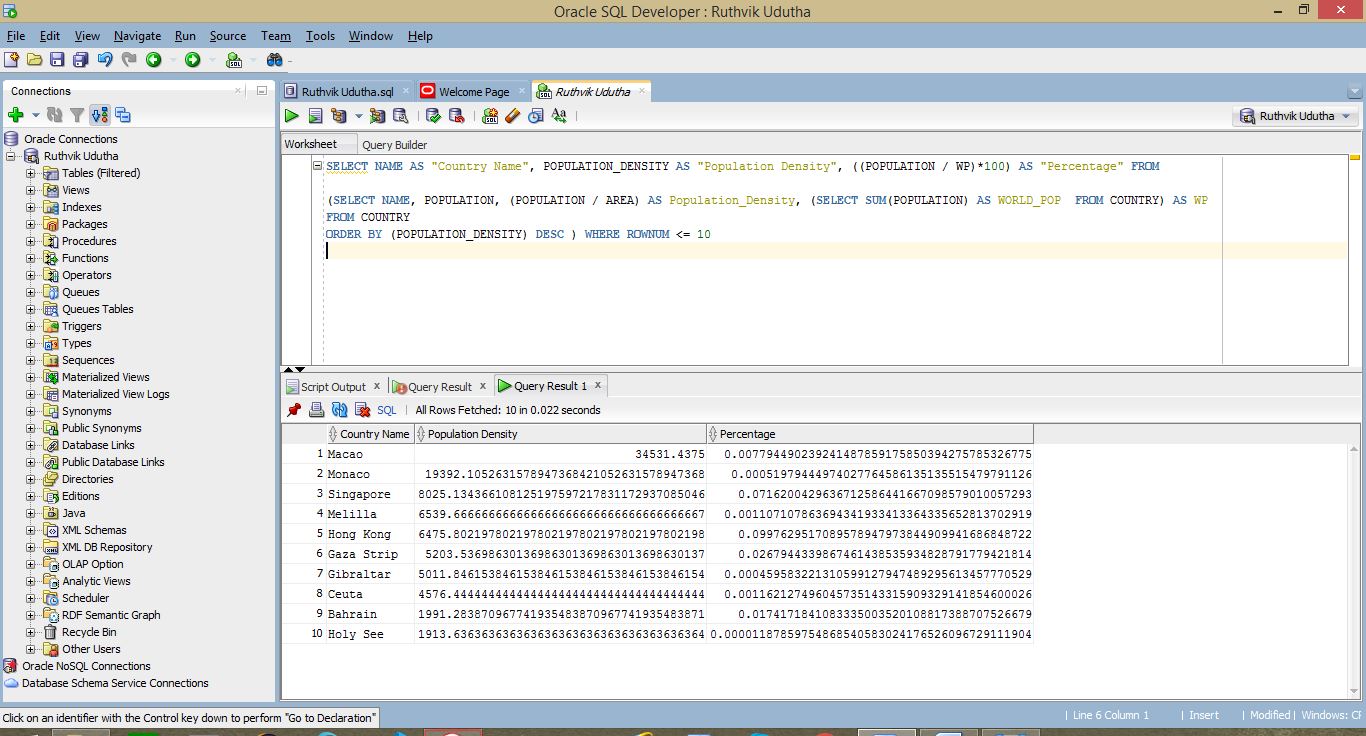
ORDER BY

( population\_density ) DESC

)

WHERE

ROWNUM <= 10



**25.**

SELECT

name

FROM

organization

WHERE

abbreviation IN (

SELECT

abbreviation

FROM

(

SELECT DISTINCT

abbreviation

FROM

(

SELECT

abbreviation,

name

FROM

organization

) o

JOIN (

SELECT

\*

FROM

ismember

WHERE

country IN (

SELECT

country

FROM

encompasses

WHERE

continent = 'Asia'

)

) n ON o.abbreviation = n.organization

)

WHERE

abbreviation NOT IN (

SELECT

non\_asia.abbreviation

FROM

(

SELECT DISTINCT

abbreviation

FROM

(

SELECT

abbreviation,

name

FROM

organization

) o

JOIN (

SELECT

\*

FROM

ismember

WHERE

country IN (

SELECT

country

FROM

encompasses

WHERE

continent = 'Asia'

)

) n ON o.abbreviation = n.organization

) asia

JOIN (

SELECT DISTINCT

abbreviation

FROM

(

SELECT

abbreviation,

name

FROM

organization

) o

JOIN (

SELECT

\*

FROM

ismember

WHERE

country IN (

SELECT

country

FROM

encompasses

WHERE

continent != 'Asia'

)

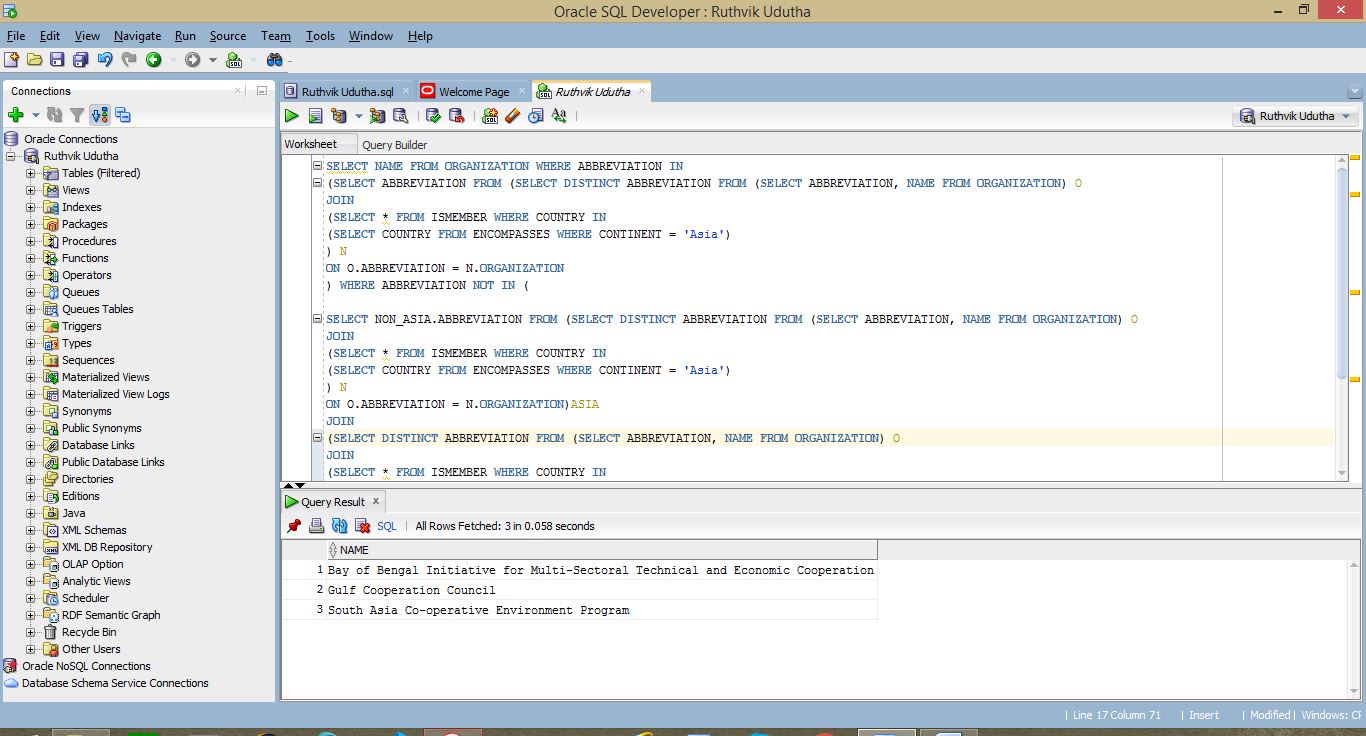
) n ON o.abbreviation = n.organization

) non\_asia ON asia.abbreviation = non\_asia.abbreviation

)

)

--NAME OF ORGANIZATONS HAVING ASIAN MEMBER COUNTRIES



**12.**

SELECT

name

FROM

country

WHERE

code IN (

SELECT

country

FROM

geo\_lake

WHERE

lake IN (

SELECT

name

FROM

lake

WHERE

area = (

SELECT

MAX(area)

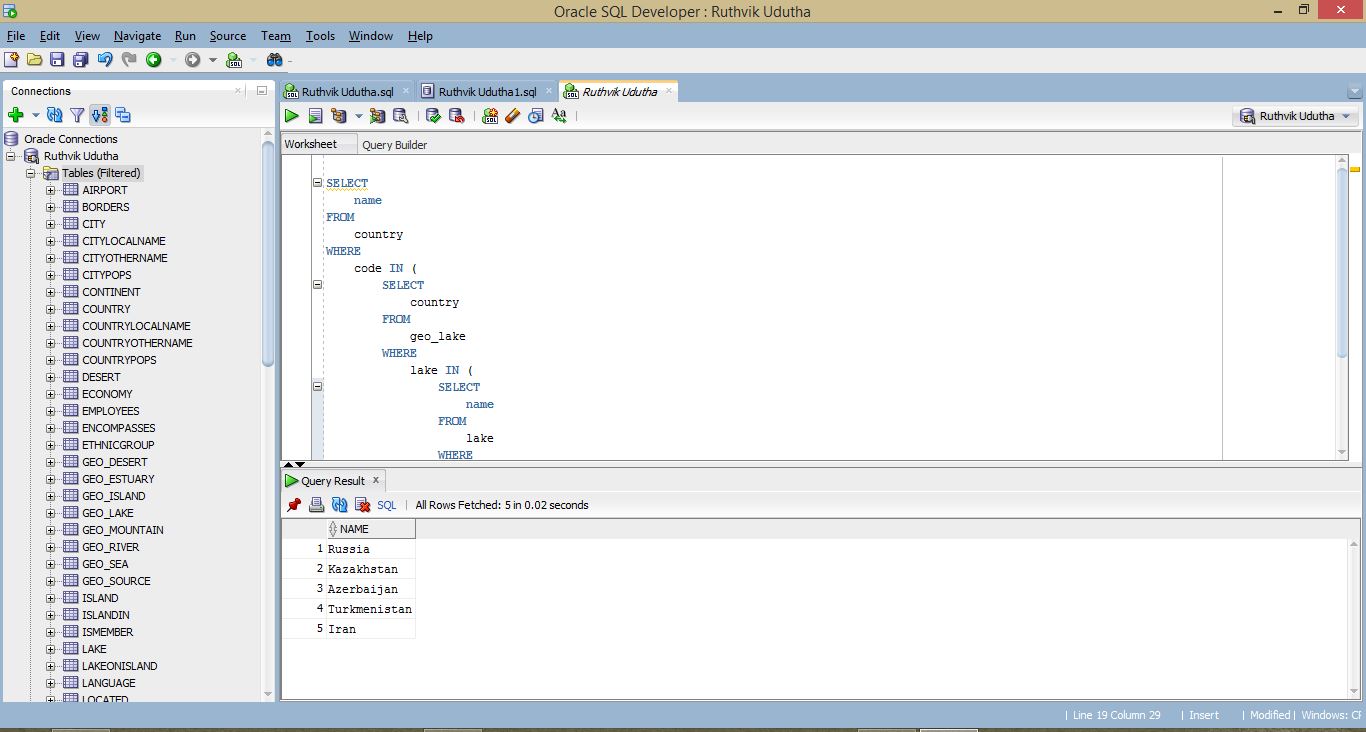
FROM

lake

)

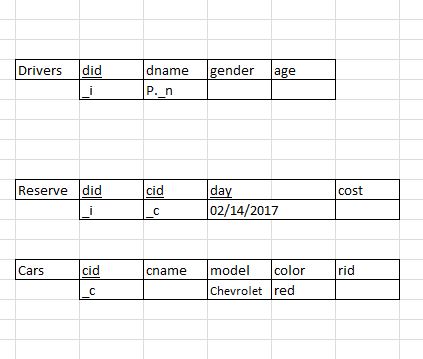
)

)

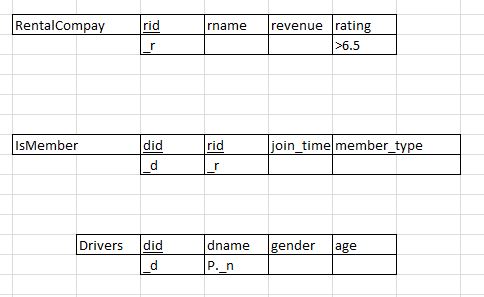


**Exercise 2 (QBE)**

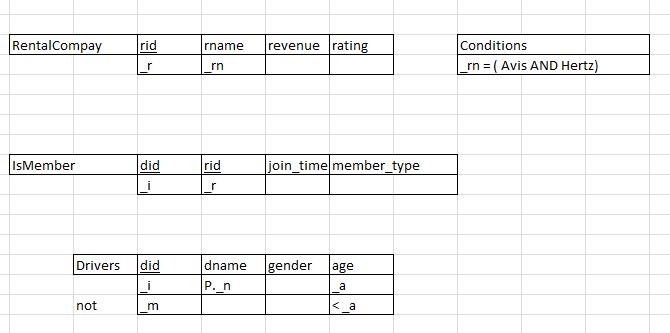
1.



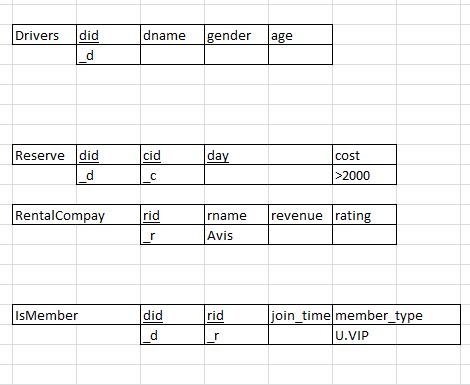
2.



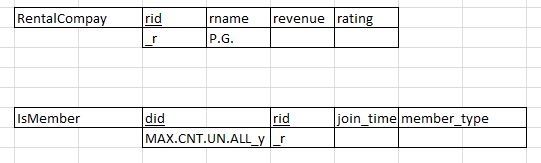
3.



4.



5.



6.

