

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.

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

The deferrable behaviour can be specified by using the below key words following DEFERRABLE 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

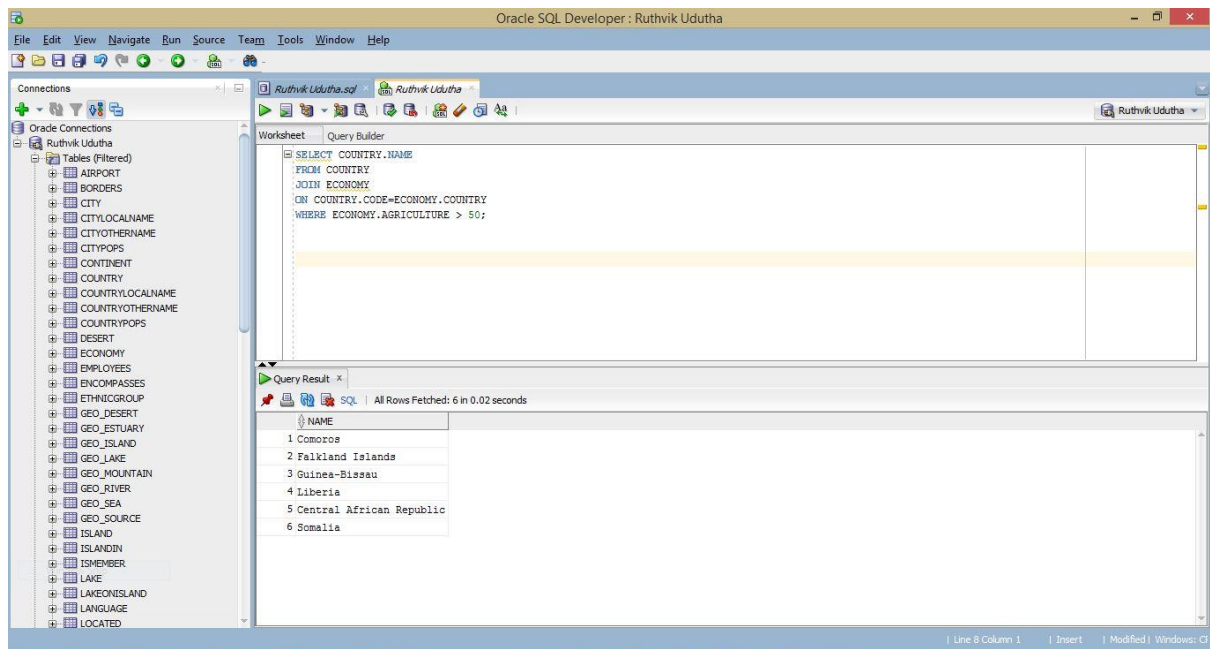
FROM

country

JOIN economy ON country.code = economy.country

WHERE

economy.agriculture > 50;



2.

SELECT

name

FROM

(

SELECT

name,

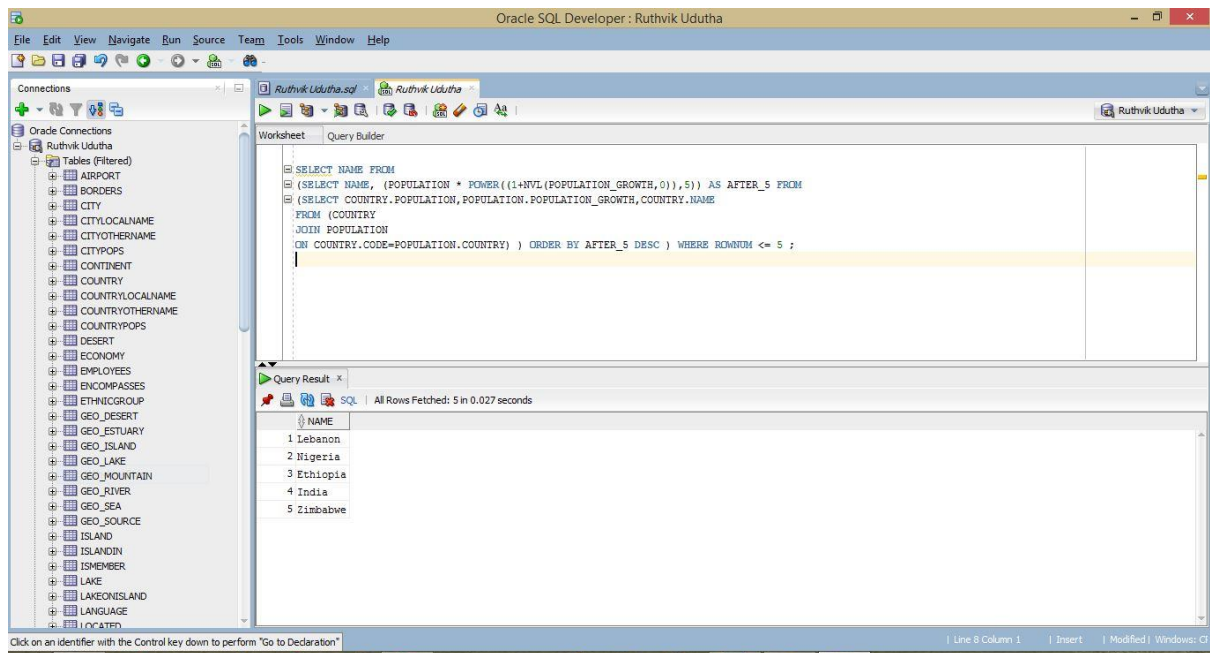
(population * power((1 + nvl(population_growth, 0)), 5)) AS after_5

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

```



```

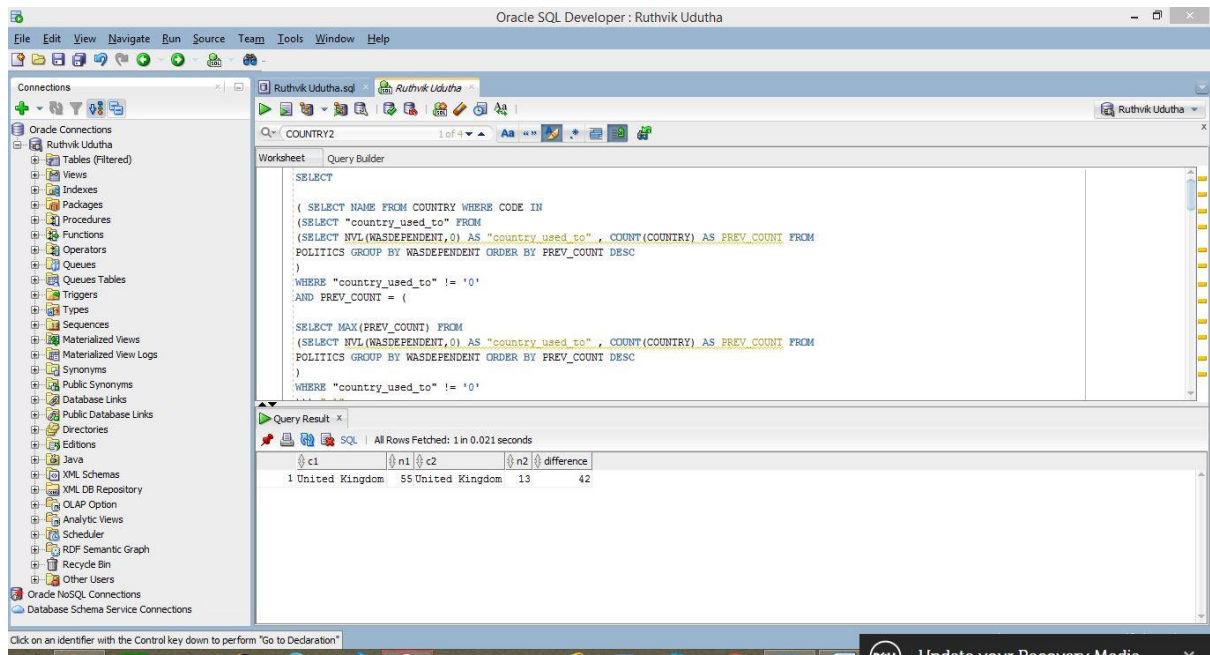
        wasdependent
    ORDER BY
        prev_count DESC
    )
WHERE
    "country_used_to" != '0'
AND prev_count = (
    SELECT
        MAX(prev_count)
    FROM
        (
            SELECT
                nvl(wasdependent, 0) AS "country_used_to",
                COUNT(country) AS prev_count
            FROM
                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

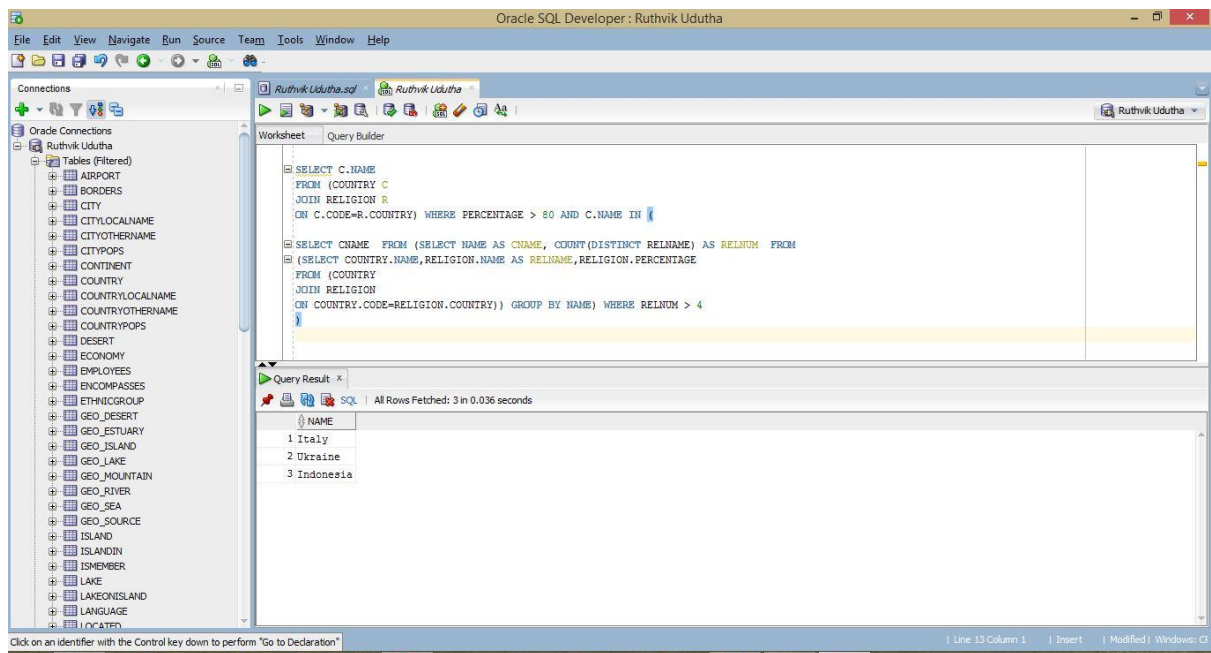
)

WHERE

relnum > 4

)

```



5.

SELECT

SUM(length)

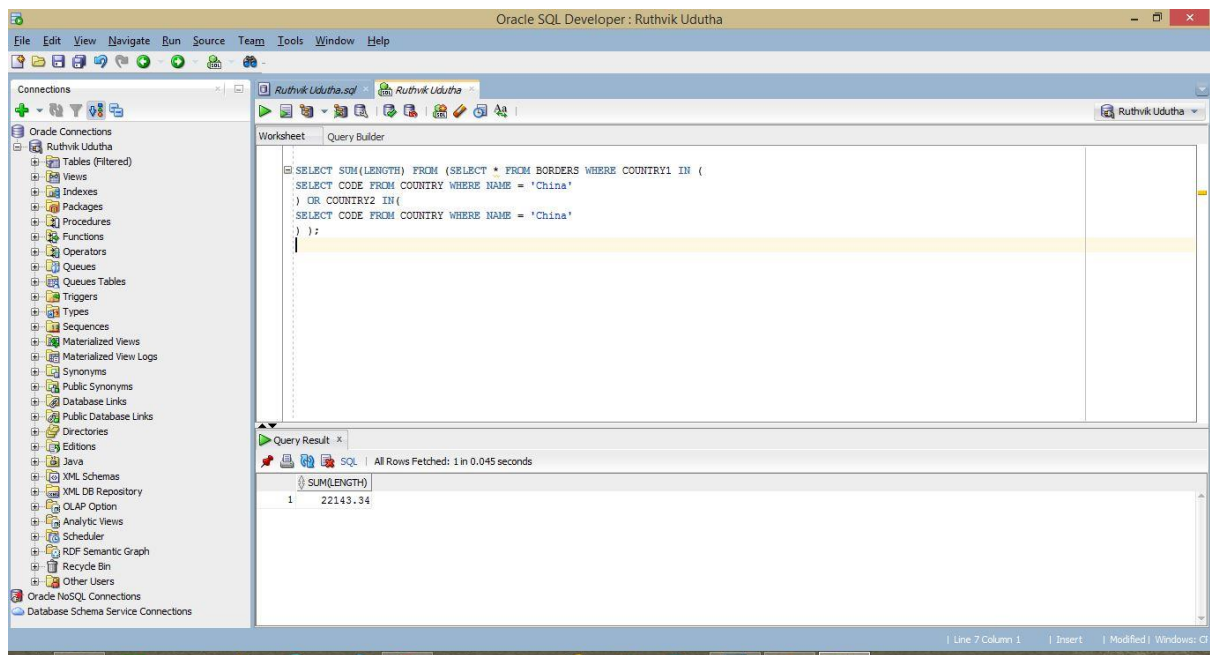
FROM

(

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

```

The screenshot shows the Oracle SQL Developer interface with the following components:

- Connections:** A tree view on the left showing the 'Ruthvik Udutha' connection selected.
- Worksheet:** The main area containing the SQL query:


```

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

```
- Query Result:** A table at the bottom showing the results of the query:

	NAME	NUM_FOLLOWERS
1	Muslim	1689585993.314
2	Hindu	1026774738.276
3	Roman Catholic	993705497.062
4	Protestant	407003149.583
5	Buddhist	307601717.816

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 > (

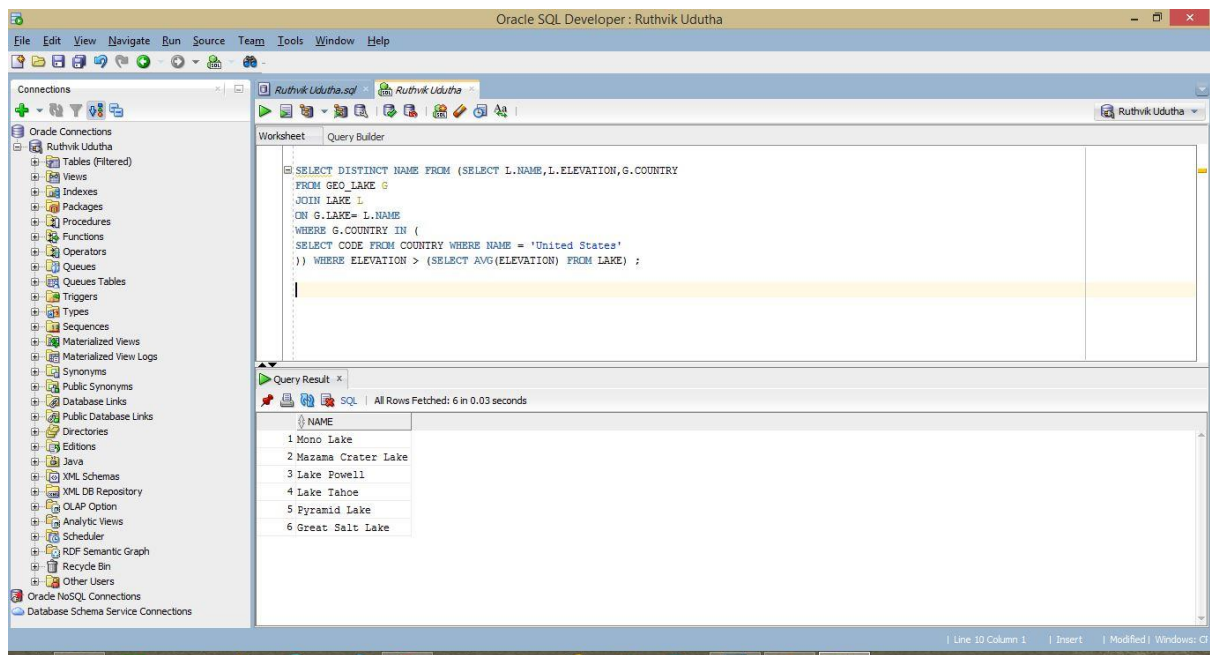
SELECT

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

)

)

)

```

The screenshot shows the Oracle SQL Developer interface. The main window displays a SQL query in the Worksheet tab. The query is a complex join and aggregation query. The Query Result window shows the results of the query, which are two rows of data.

Query:

```

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)
)
)

```

Query Result:

	MOUNTAIN_NAME	PROVINCE	POPULATION_DENSITY
1	Gede	Java Barat	1308.775277722814257851146224948412810583
2	Ciremai	Java Barat	1308.775277722814257851146224948412810583

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

The screenshot shows the Oracle SQL Developer interface. The main window displays a SQL query in the Worksheet tab. The query is as follows:

```
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
```

Below the query, the Query Result window shows the results of the query. The results are as follows:

PROVINCE	ISLAND_COUNT	COUNTRY	GDP
1 Illes Balears	4	E	1356000
2 Canarias	7	E	1356000
3 Niedersachsen	7	D	3593000
4 Schleswig-Holstein	6	D	3593000
5 Sicilia	3	I	2068000
6 Calabria	7	I	2068000
7 Sakhalin	4	R	2113000
8 Scotland	18	GB	2490000
9 Nunavut	6	CDN	1825000
10 Ontario	3	CDN	1825000
11 Hawaii	7	USA	16720000
12 New York	3	USA	16720000
13 California	5	USA	16720000

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

) JOIN r.name = l.river

ORDER BY

    r.length DESC

)

WHERE

ROWNUM <= 2

```

The screenshot shows the Oracle SQL Developer interface. The main window displays a query in the SQL editor:

```

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

```

The query results are displayed in the Query Result window, showing 2 rows fetched in 0.017 seconds:

	NAME	LENGTH
1	Zaire	4374
2	Niger	4184

The left pane shows the database schema with various tables listed, including AIRPORT, BORDERS, CITY, CITYLOCALNAME, CITYOTHERNAME, CITYPOPS, CONTINENT, COUNTRY, COUNTRYLOCALNAME, COUNTRYOTHERNAME, COUNTRYPOPS, DESERT, ECONOMY, EMPLOYEES, ENCOMPASSES, ETHNICGROUP, GEO_DESERT, GEO_ESTUARY, GEO_ISLAND, GEO_LAKE, GEO_MOUNTAIN, GEO_RIVER, GEO_SEA, GEO_SOURCE, ISLAND, ISLANDIN, ISMEMBER, LAKE, LAKEONISLAND, LANGUAGE, and LOCATED.

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
            )
    )
```

```

    )
)
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
)

```

The screenshot shows the Oracle SQL Developer interface with the following components:

- Connections:** A tree view on the left showing the 'Ruthvik Udutha' connection and a list of database tables including AIRPORT, BORDERS, CITY, CITYLOCALNAME, CITYOTHERNAME, CITYPOPS, CONTINENT, COUNTRY, COUNTRYLOCALNAME, COUNTRYOTHERNAME, COUNTRYPOPS, DESERT, ECONOMY, EMPLOYEES, ENCOMPASSES, ETHNICGROUP, GEO_DESERT, GEO_ESTUARY, GEO_ISLAND, GEO_LAKE, GEO_MOUNTAIN, GEO_RIVER, GEO_SEA, GEO_SOURCE, ISLAND, ISLANDIN, ISLANDIN, ISLANDIN, LAKE, LAKEONISLAND, LANGUAGE, and LOCATED.
- Worksheet:** The main area containing the SQL query:


```

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
  )
)

```
- Query Result:** A table showing the results of the query, with a single column 'NAME' and 15 rows of country names:

NAME
1 Kazakhstan
2 Cameroon
3 Switzerland
4 Nigeria
5 Tanzania
6 Zambia
7 Hungary
8 France
9 Burundi
10 Russia
11 Austria
12 Iran
13 Sweden
14 Cote d'Ivoire
15 Zaire
- Status Bar:** At the bottom, it indicates 'Line 9 Column 2', 'Insert' mode, and 'Modified | Windows: C'.

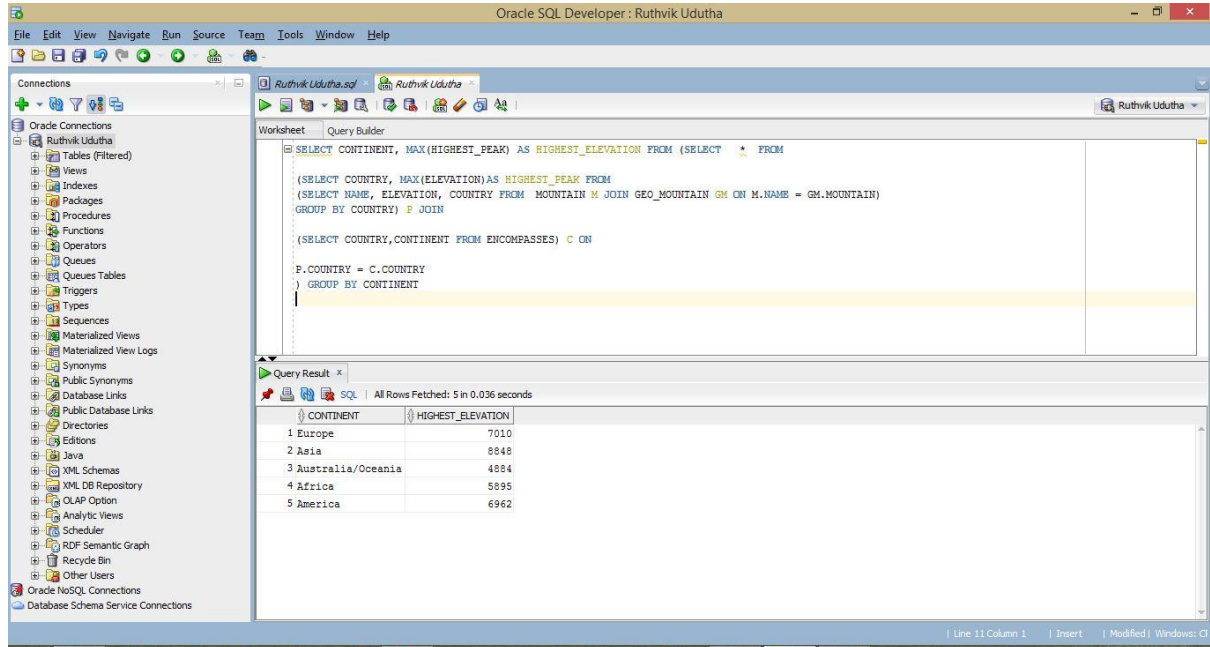
13.

```
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

((

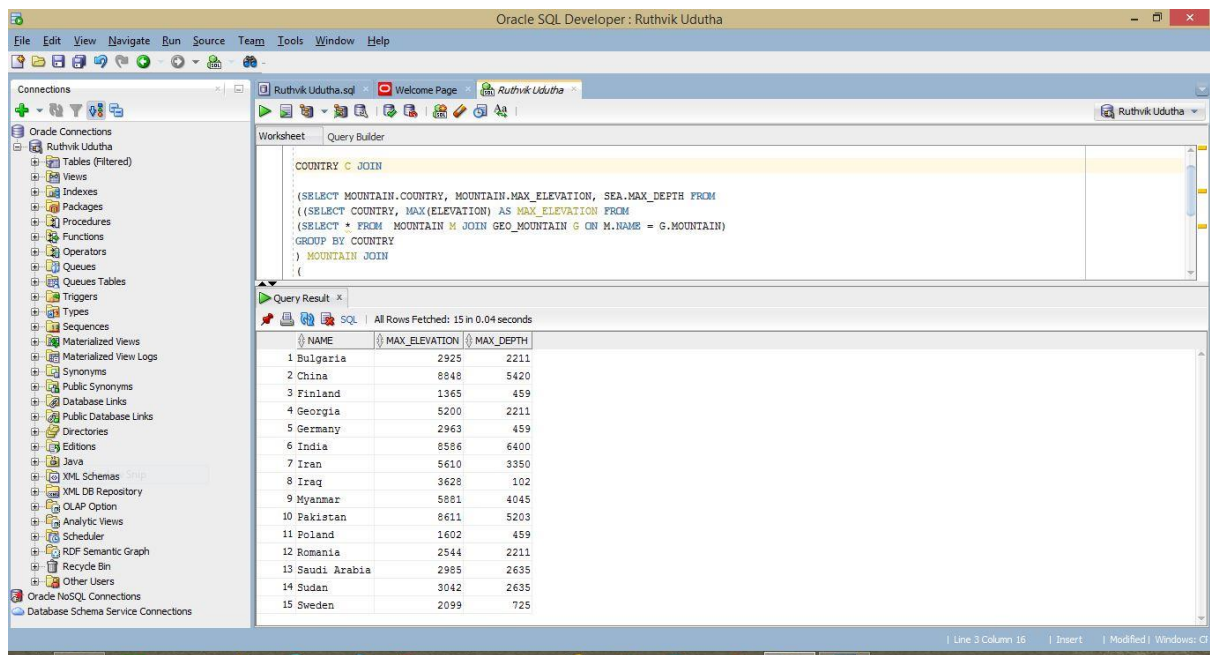
SELECT

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

```

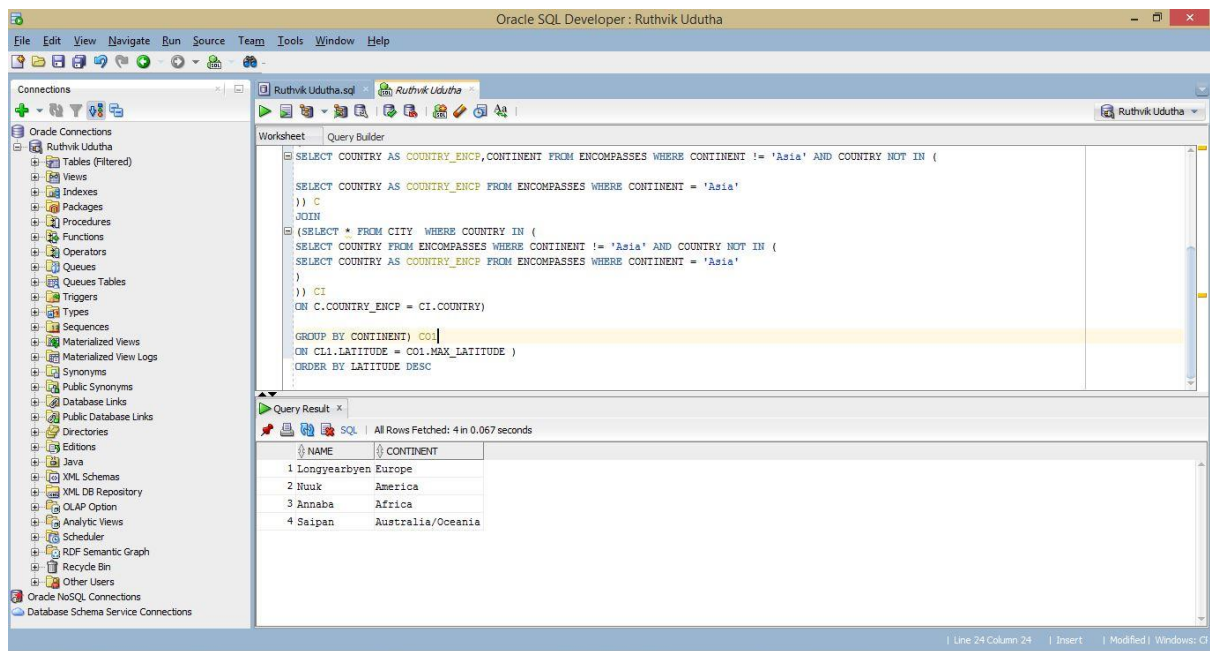
        country AS country_encp
    FROM
        encompasses
    WHERE
        continent = 'Asia'
    )
) cl1
JOIN (
    SELECT
        continent,
        MAX(latitude) AS max_latitude
    FROM
        (
            SELECT
                ci.country,
                ci.name AS city_name,
                c.continent,
                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

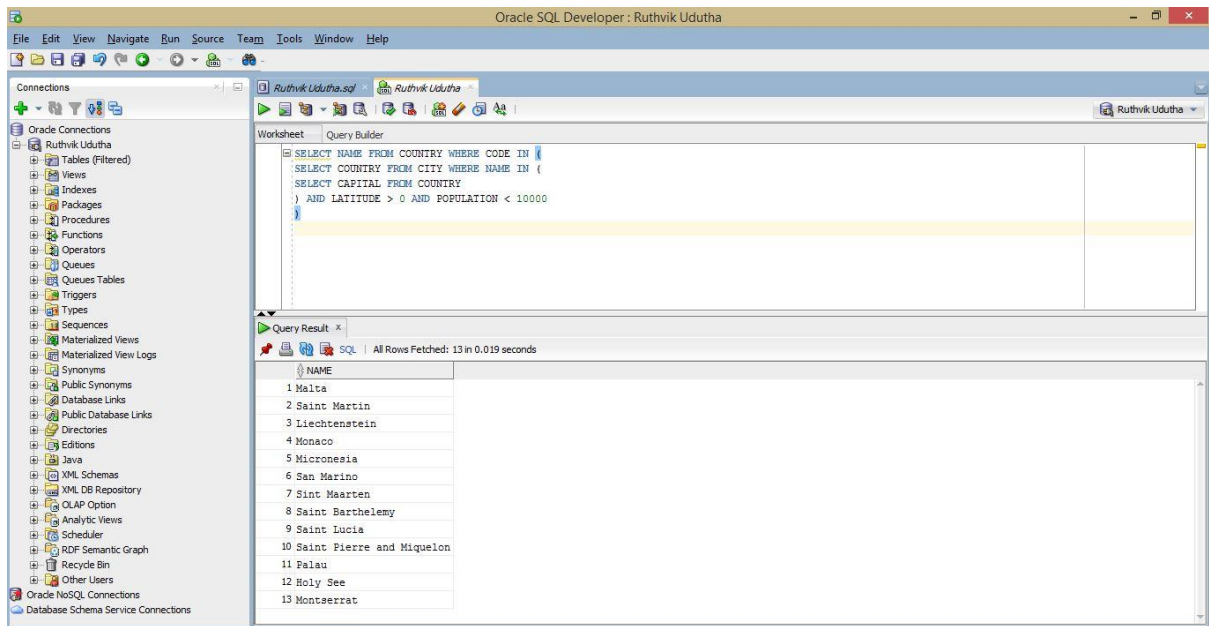
country

)

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

)

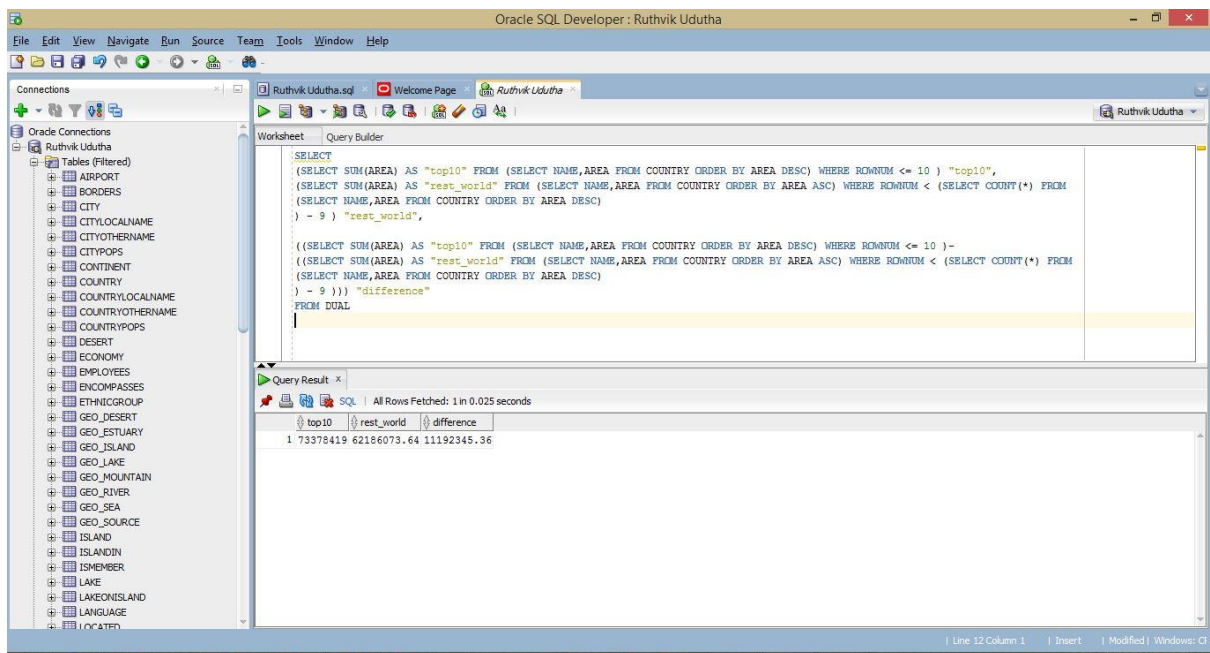
) - 9

)) "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

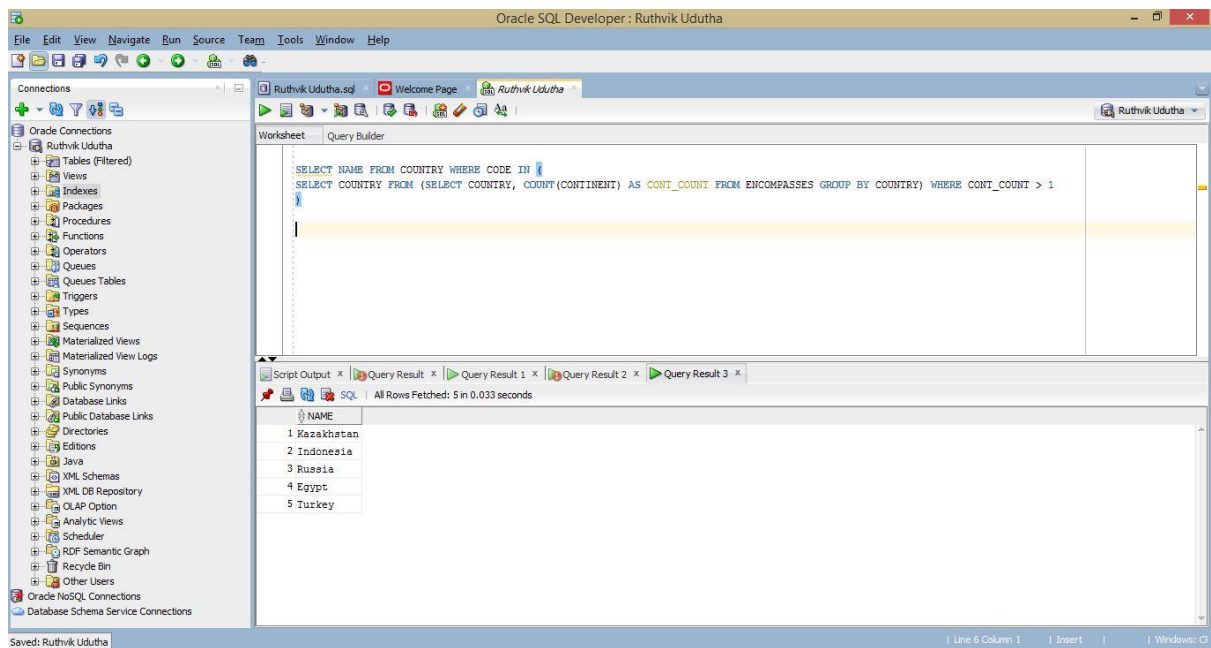
)

WHERE

    cont_count > 1

)

```



19.

```

SELECT

    i.name,

    i.area

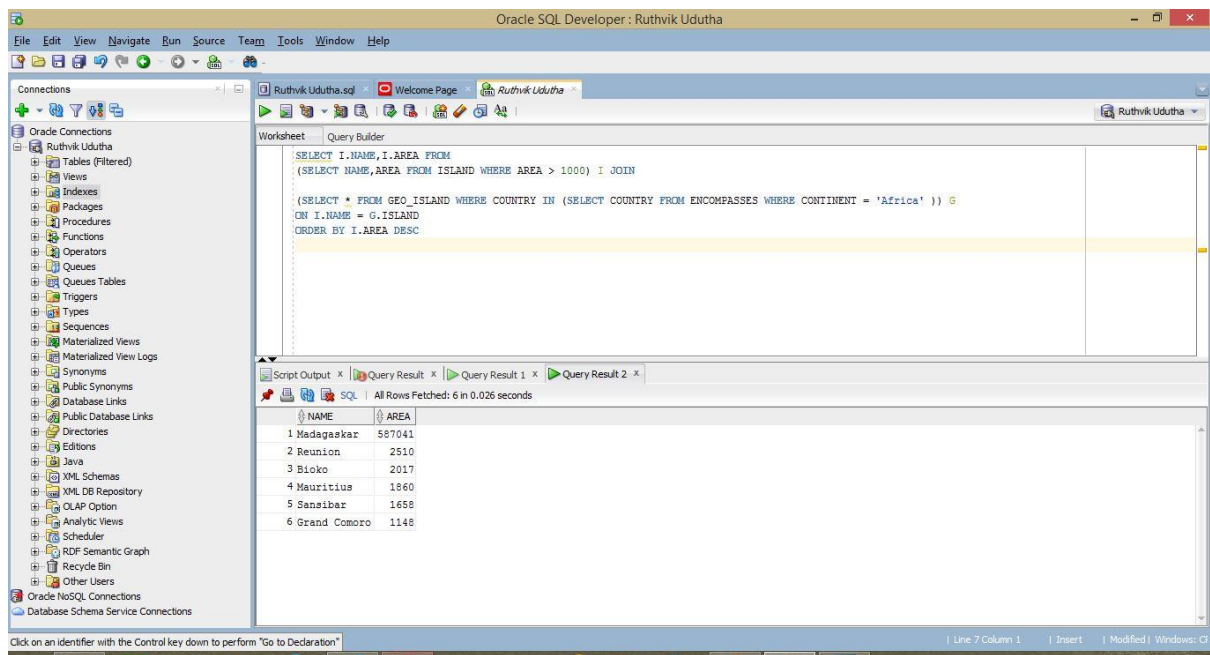
FROM

(

    SELECT

```

```
        name,
        area
FROM
    island
WHERE
    area > 1000
)
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

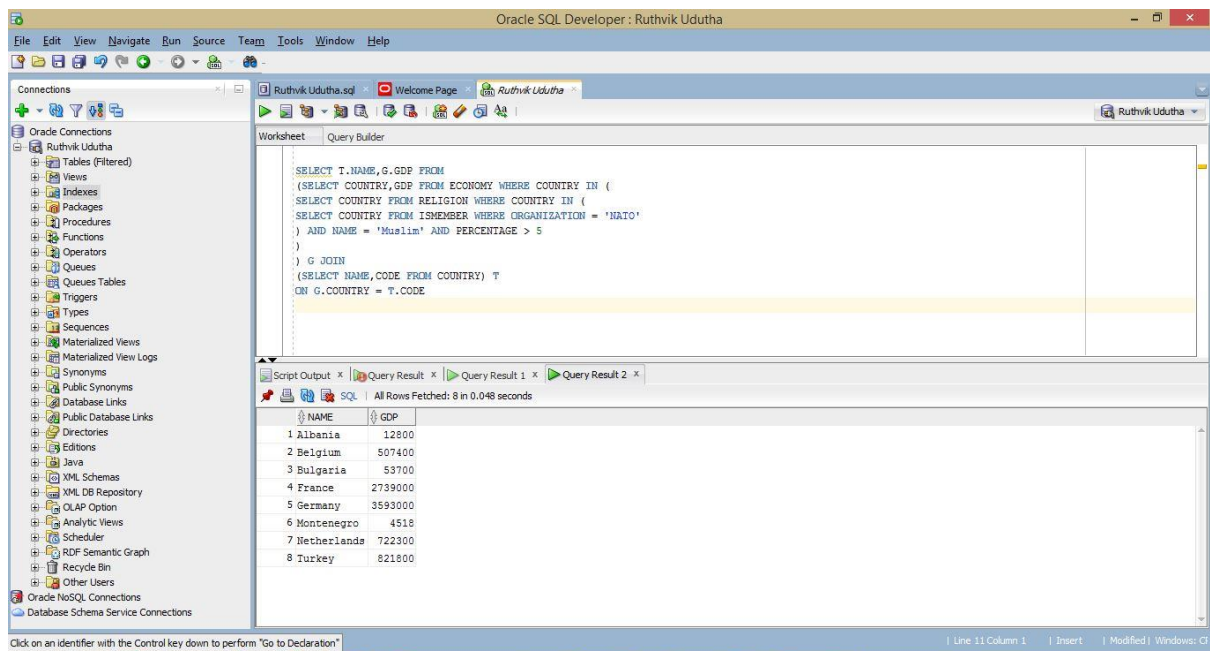
country IN (

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

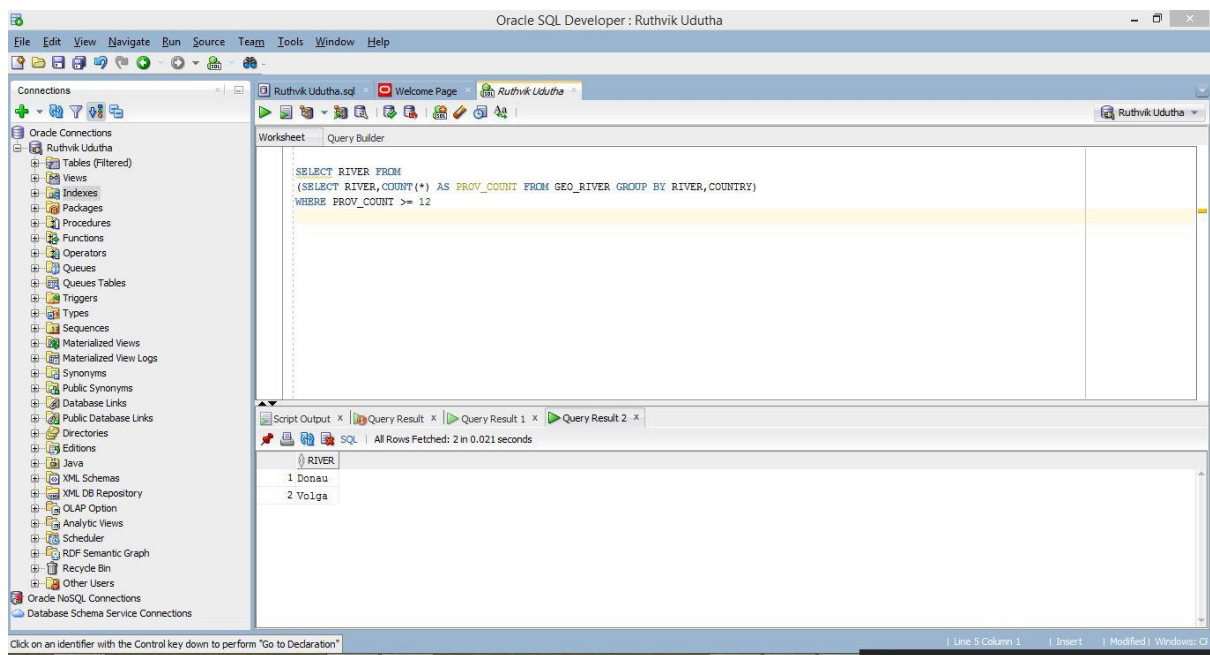
        river,

        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

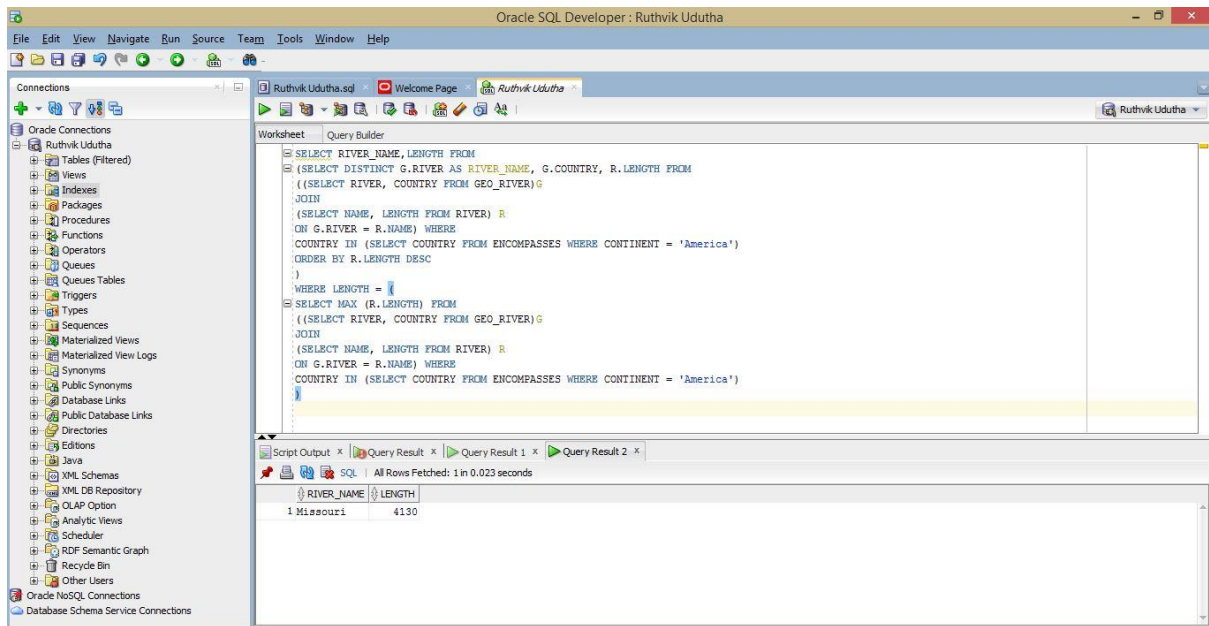
WHERE

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

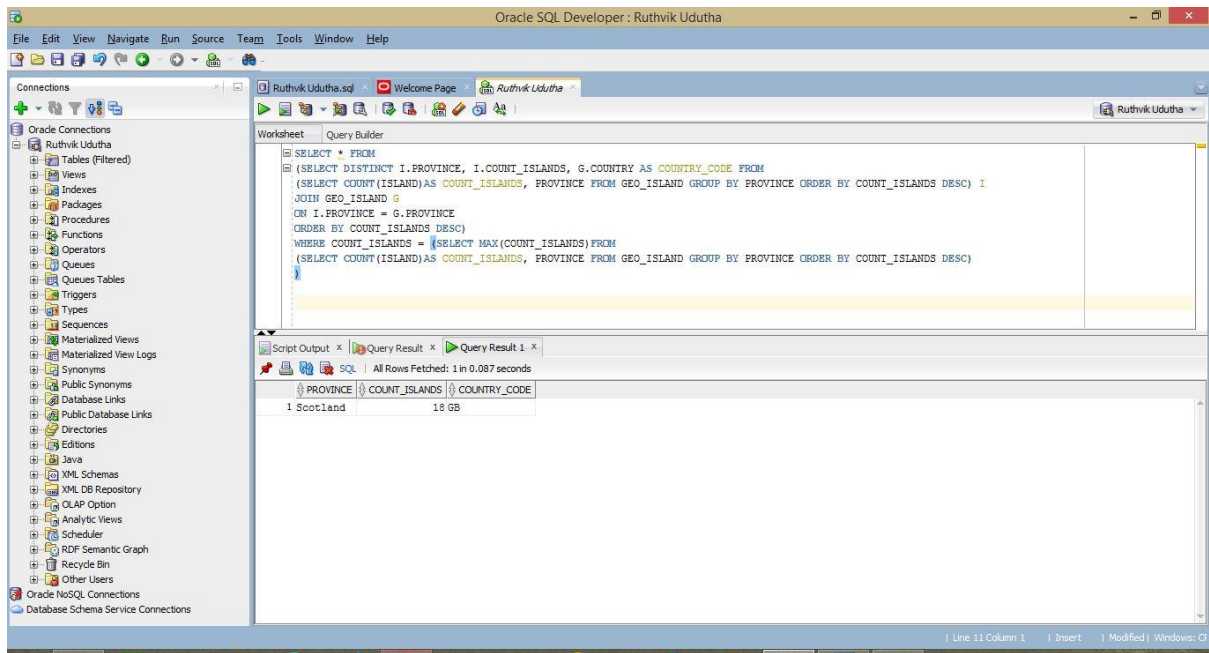
GROUP BY

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"

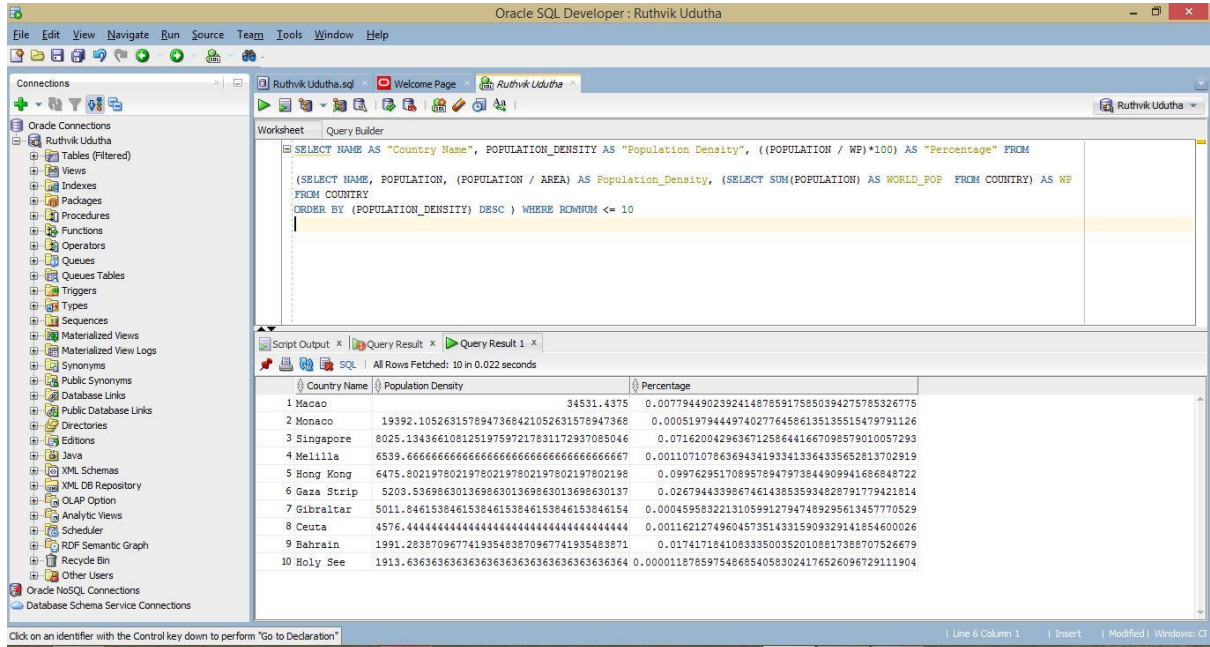
FROM

(
 SELECT
 name,
 population,
 (population / area) AS population_density,
 (
 SELECT
 SUM(population) AS world_pop
 FROM
 country
) AS wp
 FROM
 country
 ORDER BY

)

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 ORGANIZATIONS HAVING ASIAN MEMBER COUNTRIES

```

The screenshot shows the Oracle SQL Developer interface with the following components:

- Connections:** A tree view on the left showing the 'Ruthvik Udutha' connection selected.
- Worksheet:** The main area containing the SQL query:


```

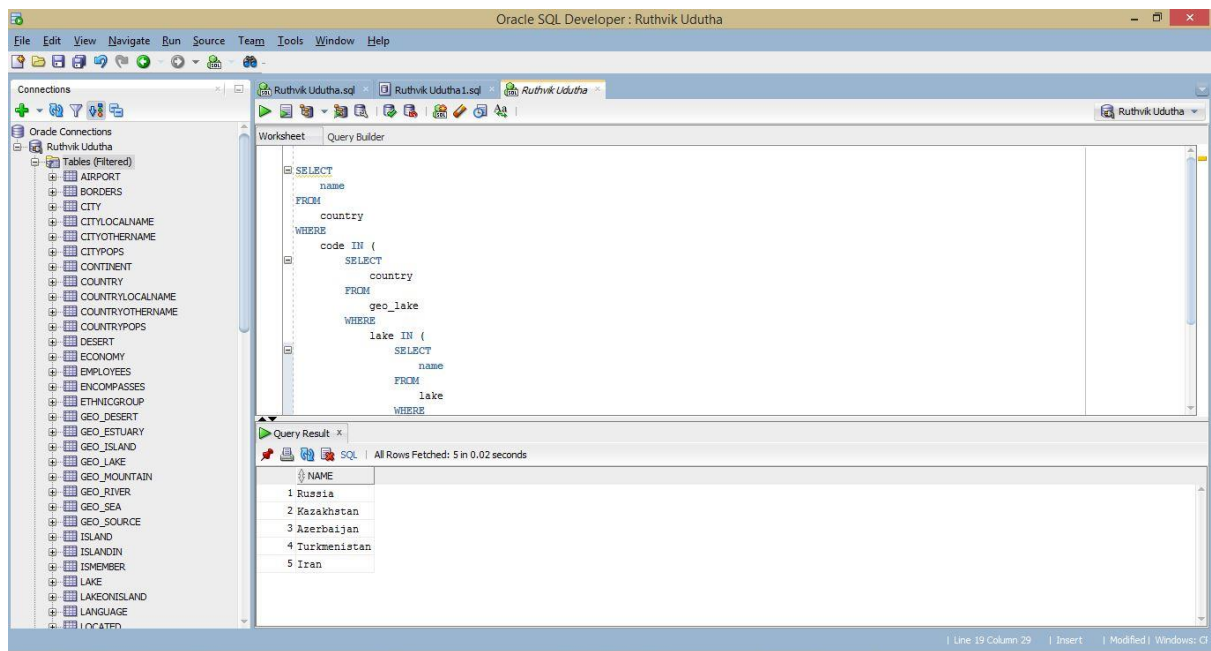
SELECT NAME FROM ORGANIZATION WHERE ABBREVIATION IN
(SELECT ABBREVIATION FROM (SELECT DISTINCT ABBREVIATION FROM (SELECT ABBREVIATION, NAME FROM ORGANIZATION) O
JOIN
(SELECT * FROM ISMEMBER WHERE COUNTRY IN
(SELECT COUNTRY FROM ENCOMPASSES WHERE CONTINENT = 'Asia')
) N
ON O.ABBREVIATION = N.ORGANIZATION
) 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
)
)
)

```
- Query Result:** A table at the bottom showing the results of the query:

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

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.

Drivers	<u>did</u>	dname	gender	age	
	<u>i</u>	P_n			
Reserve	<u>did</u>	<u>cid</u>	<u>day</u>	cost	
	<u>i</u>	<u>c</u>	02/14/2017		
Cars	<u>cid</u>	cname	model	color	rid
	<u>c</u>		Chevrolet	red	

2.

RentalCompay	<u>rid</u>	rname	revenue	rating
	<u>_r</u>			>6.5

IsMember	<u>did</u>	<u>rid</u>	join_time	member_type
	<u>_d</u>	<u>_r</u>		

Drivers	<u>did</u>	dname	gender	age
	<u>_d</u>	P._n		

3.

RentalCompay	<u>rid</u>	rname	revenue	rating	Conditions
	<u>_r</u>	<u>_rn</u>			

IsMember	<u>did</u>	<u>rid</u>	join_time	member_type
	<u>_i</u>	<u>_r</u>		

Drivers	<u>did</u>	dname	gender	age
	<u>_i</u>	P._n		<u>_a</u>
	<u>_m</u>			< <u>_a</u>

not				
-----	--	--	--	--

4.

Drivers	<u>did</u>	dname	gender	age	
	<u>_d</u>				
Reserve	<u>did</u>	<u>cid</u>	<u>day</u>		cost
	<u>_d</u>	<u>_c</u>			>2000
RentalCompay	<u>rid</u>	rname	revenue	rating	
	<u>_r</u>	Avis			
IsMember	<u>did</u>	<u>rid</u>	join_time	member_type	
	<u>_d</u>	<u>_r</u>		U.VIP	

5.

RentalCompay	<u>rid</u>	rname	revenue	rating	
	<u>_r</u>	P.G.			
IsMember	<u>did</u>	<u>rid</u>	join_time	member_type	
	MAX.CNT.UN.ALL_y	<u>_r</u>			

6.

Drivers	<u>did</u>	dname	gender	age	Condition		
	<u>_d</u>			<u>_a</u>	<u>_a</u> =(>21 and < 30)		
Reserve	<u>did</u>	<u>cid</u>	<u>day</u>		cost		
	<u>_d</u>	<u>_c</u>					
Cars	<u>cid</u>	cname	model	color	rid		
	<u>_c</u>		P.G.		<u>_r</u>		
RentalCompany		<u>rid</u>		rnane	revenue	rating	
		MAX.CNT.ALL_ <u>_R</u>					