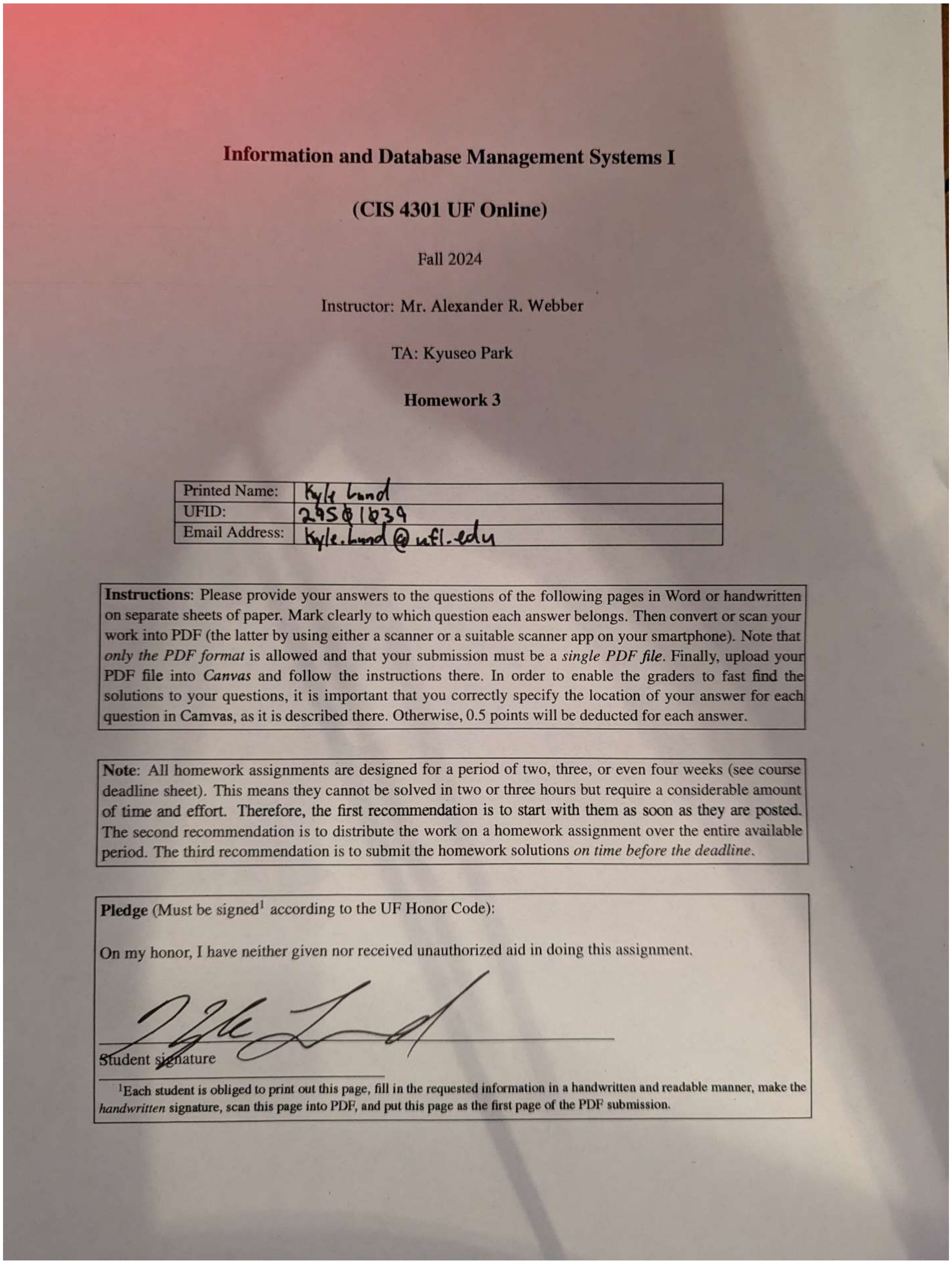


Information and Database Management Systems I

(CIS 4301 UF Online)

Fall 2024

Instructor: Mr. Alexander R. Webber

TA: Kyuseo Park

Homework 3

Printed Name:	Kyle Lund
UFID:	29501639
Email Address:	kyle.lund@ufl.edu

Instructions: Please provide your answers to the questions of the following pages in Word or handwritten on separate sheets of paper. Mark clearly to which question each answer belongs. Then convert or scan your work into PDF (the latter by using either a scanner or a suitable scanner app on your smartphone). Note that *only the PDF format* is allowed and that your submission must be a *single PDF file*. Finally, upload your PDF file into *Canvas* and follow the instructions there. In order to enable the graders to fast find the solutions to your questions, it is important that you correctly specify the location of your answer for each question in Camvas, as it is described there. Otherwise, 0.5 points will be deducted for each answer.

Note: All homework assignments are designed for a period of two, three, or even four weeks (see course deadline sheet). This means they cannot be solved in two or three hours but require a considerable amount of time and effort. Therefore, the first recommendation is to start with them as soon as they are posted. The second recommendation is to distribute the work on a homework assignment over the entire available period. The third recommendation is to submit the homework solutions *on time before the deadline*.

Pledge (Must be signed[1] according to the UF Honor Code):

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

[illegible]

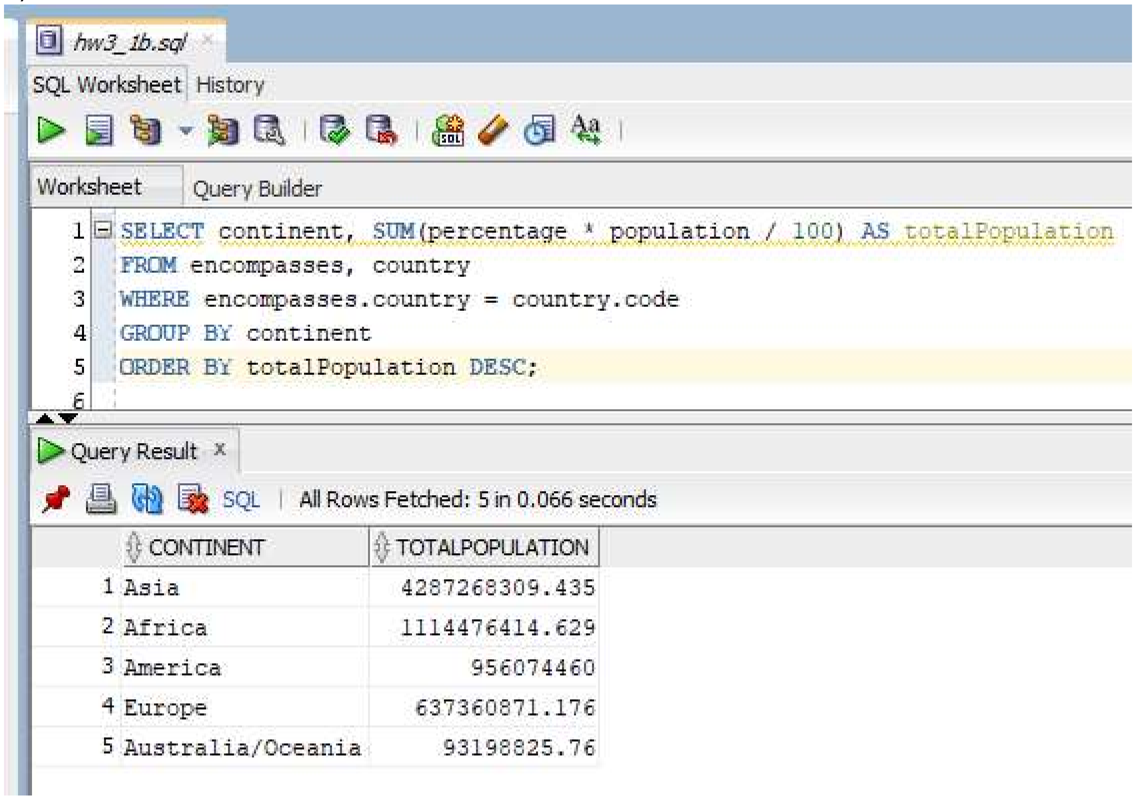
Student signature

[1]Each student is obliged to print out this page, fill in the requested information in a handwritten and readable manner, make the *handwritten* signature, scan this page into PDF, and put this page as the first page of the PDF submission.

Question 1
a)
1) As stated here: “DEFERRABLE is a constraint attribute in SQL that specifies that the enforcement of the constraint can be postponed until the end of a transaction. This enables a transaction to insert data [in]to a database even though it initially violates a constraint, provided that the violation is corrected before the transaction commits. When a constraint is declared as DEFERRABLE, a temporary violation is allowed in a specific scope of the transaction, known as the deferrable time.”

“By further specifying INITIALLY DEFERRED, it enforces that the constraint checking will always be deferred till the transaction’s end. The data insertion doesn’t violate the foreign key constraint as the check is performed after both insertions are executed.” This is contrasted with the INITIALLY IMMEDIATE option which causes constraint validation to happen immediately unless deferred is specifically requested.

2) The problem arises from adding foreign key constraints. Specifically in this instance the foreign keys in the tables create circular references to each other. If these constraints were not deferable, we would attempt to add a row to one of these tables, only to find that a required entry in another table, as specified by the foreign keys, does not exist, as we haven’t had a chance to create it yet. Making these constraints deferable allows us to enter the data into each of the tables and have the foreign key constraints checked at the end of the transaction, once we’ve had a chance to enter in all of the relevant data.

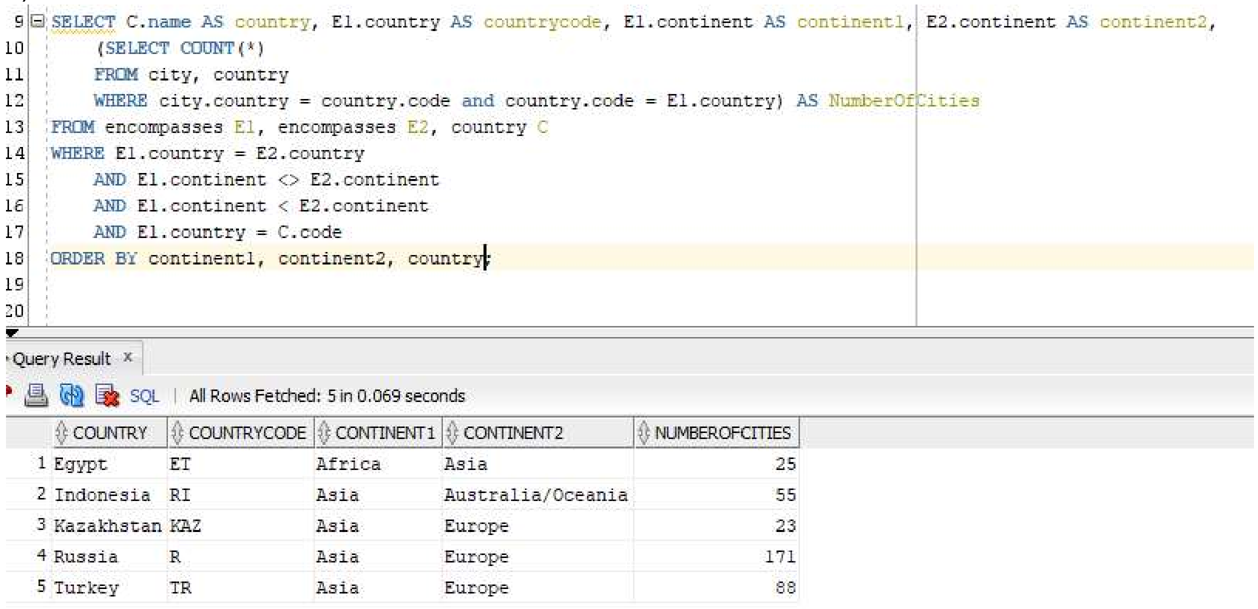
b)
1)


```
SELECT continent, SUM(percentage * population / 100) AS totalPopulation
FROM encompasses, country
WHERE encompasses.country = country.code
GROUP BY continent
ORDER BY totalPopulation DESC;
```

All Rows Fetched: 5 in 0.066 seconds

	CONTINENT	TOTALPOPULATION
1	Asia	4287268309.435
2	Africa	1114476414.629
3	America	956074460
4	Europe	637360871.176
5	Australia/Oceania	93198825.76

2)



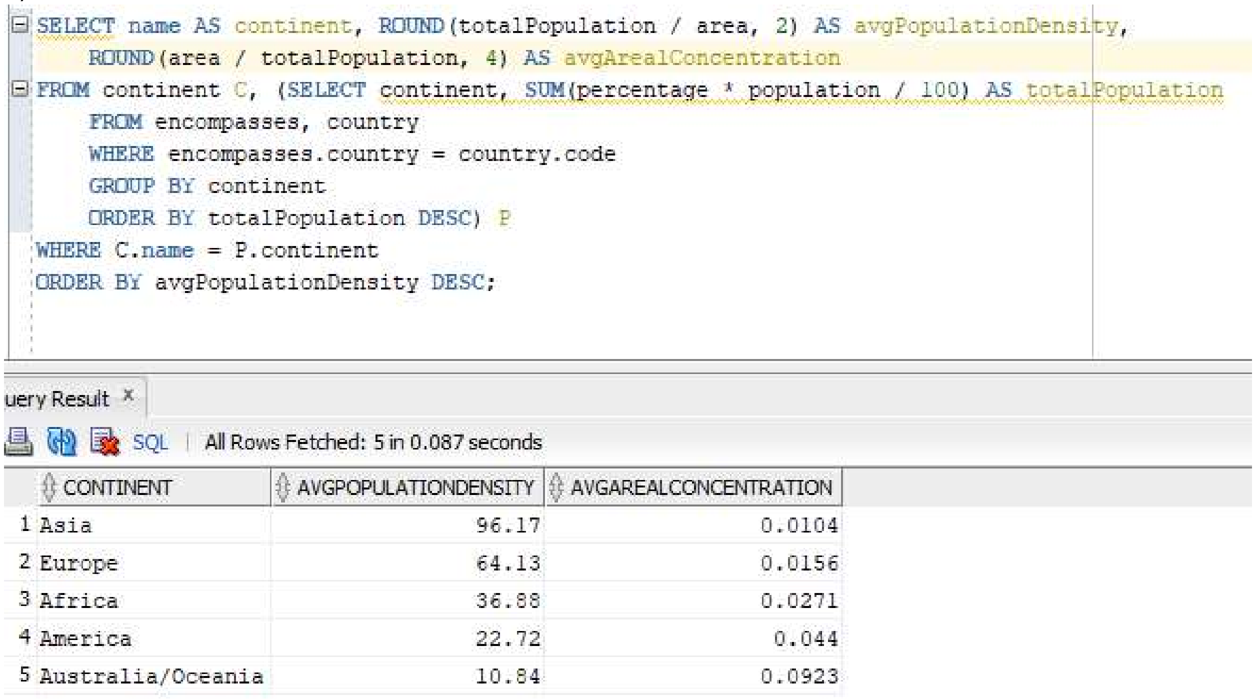
```sql
SELECT C.name AS country, E1.country AS countrycode, E1.continent AS continent1, E2.continent AS continent2,
    (SELECT COUNT(*)
    FROM city, country
    WHERE city.country = country.code and country.code = E1.country) AS NumberOfCities
FROM encompasses E1, encompasses E2, country C
WHERE E1.country = E2.country
    AND E1.continent <> E2.continent
    AND E1.continent < E2.continent
    AND E1.country = C.code
ORDER BY continent1, continent2, country;
```

Query Result

All Rows Fetched: 5 in 0.069 seconds

	COUNTRY	COUNTRYCODE	CONTINENT1	CONTINENT2	NUMBEROFCITIES
1	Egypt	ET	Africa	Asia	25
2	Indonesia	RI	Asia	Australia/Oceania	55
3	Kazakhstan	KAZ	Asia	Europe	23
4	Russia	R	Asia	Europe	171
5	Turkey	TR	Asia	Europe	88

3)



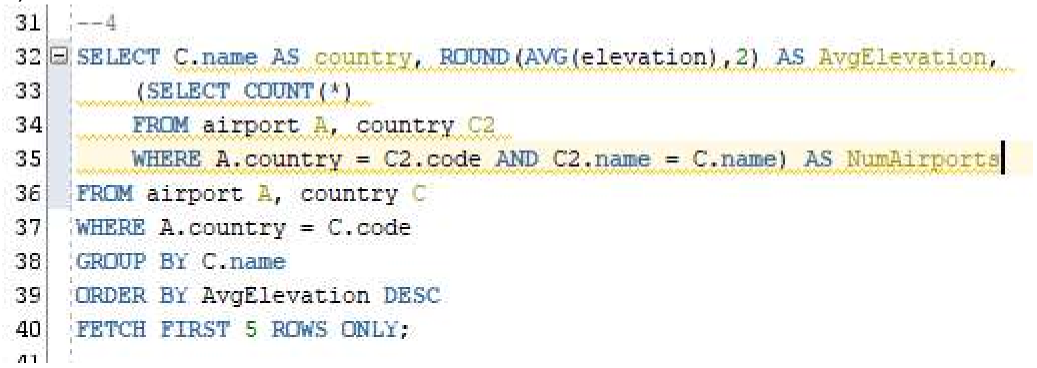
```sql
SELECT name AS continent, ROUND(totalPopulation / area, 2) AS avgPopulationDensity,
    ROUND(area / totalPopulation, 4) AS avgArealConcentration
FROM continent C, (SELECT continent, SUM(percentage * population / 100) AS totalPopulation
    FROM encompasses, country
    WHERE encompasses.country = country.code
    GROUP BY continent
    ORDER BY totalPopulation DESC) P
WHERE C.name = P.continent
ORDER BY avgPopulationDensity DESC;
```

uery Result

All Rows Fetched: 5 in 0.087 seconds

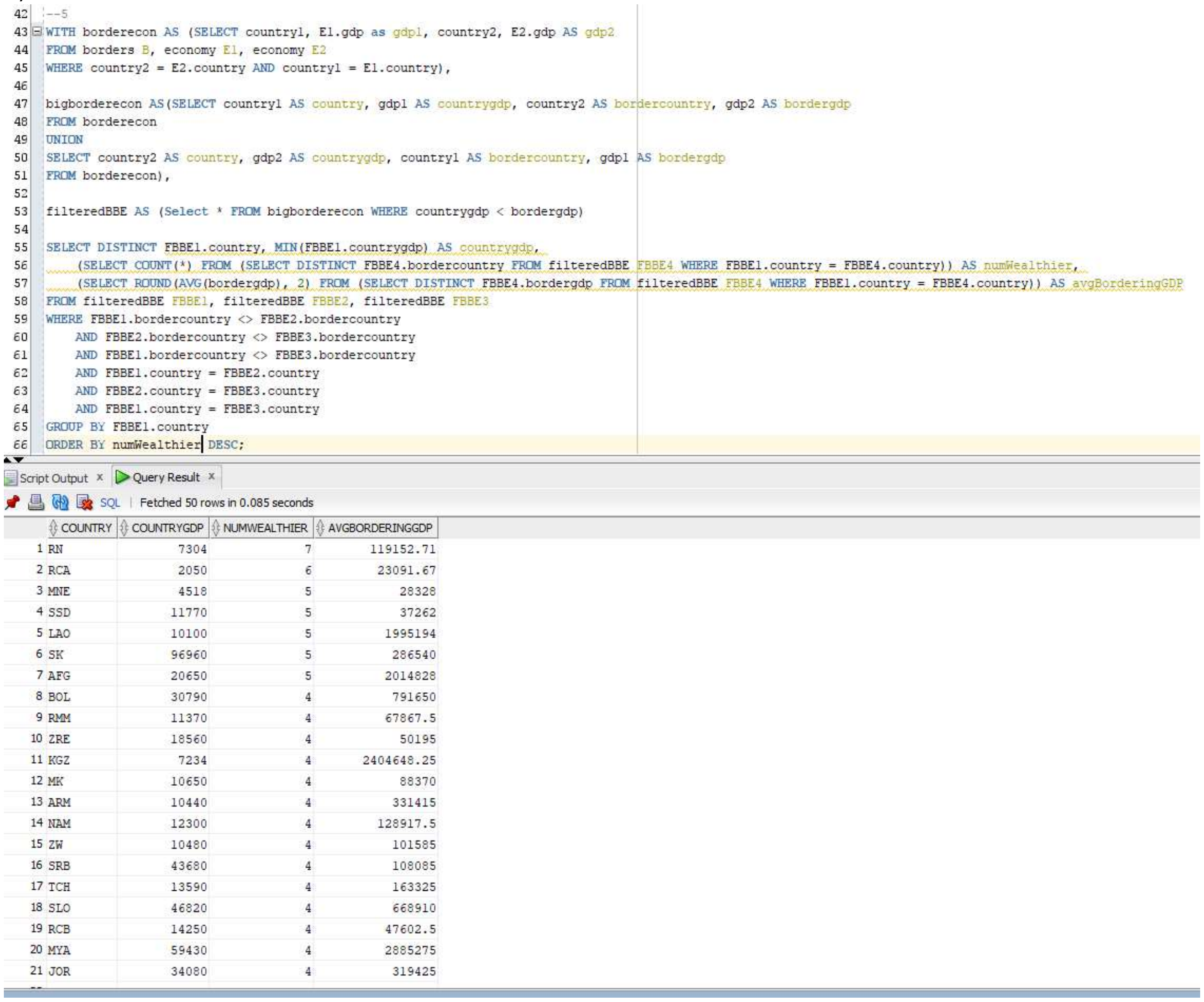
	CONTINENT	AVGPOPULATIONDENSITY	AVGAREALCONCENTRATION
1	Asia	96.17	0.0104
2	Europe	64.13	0.0156
3	Africa	36.88	0.0271
4	America	22.72	0.044
5	Australia/Oceania	10.84	0.0923

4)



```sql
--4
SELECT C.name AS country, ROUND(AVG(elevation),2) AS AvgElevation,
    (SELECT COUNT(*)
    FROM airport A, country C2
    WHERE A.country = C2.code AND C2.name = C.name) AS NumAirports
FROM airport A, country C
WHERE A.country = C.code
GROUP BY C.name
ORDER BY AvgElevation DESC
FETCH FIRST 5 ROWS ONLY;
```

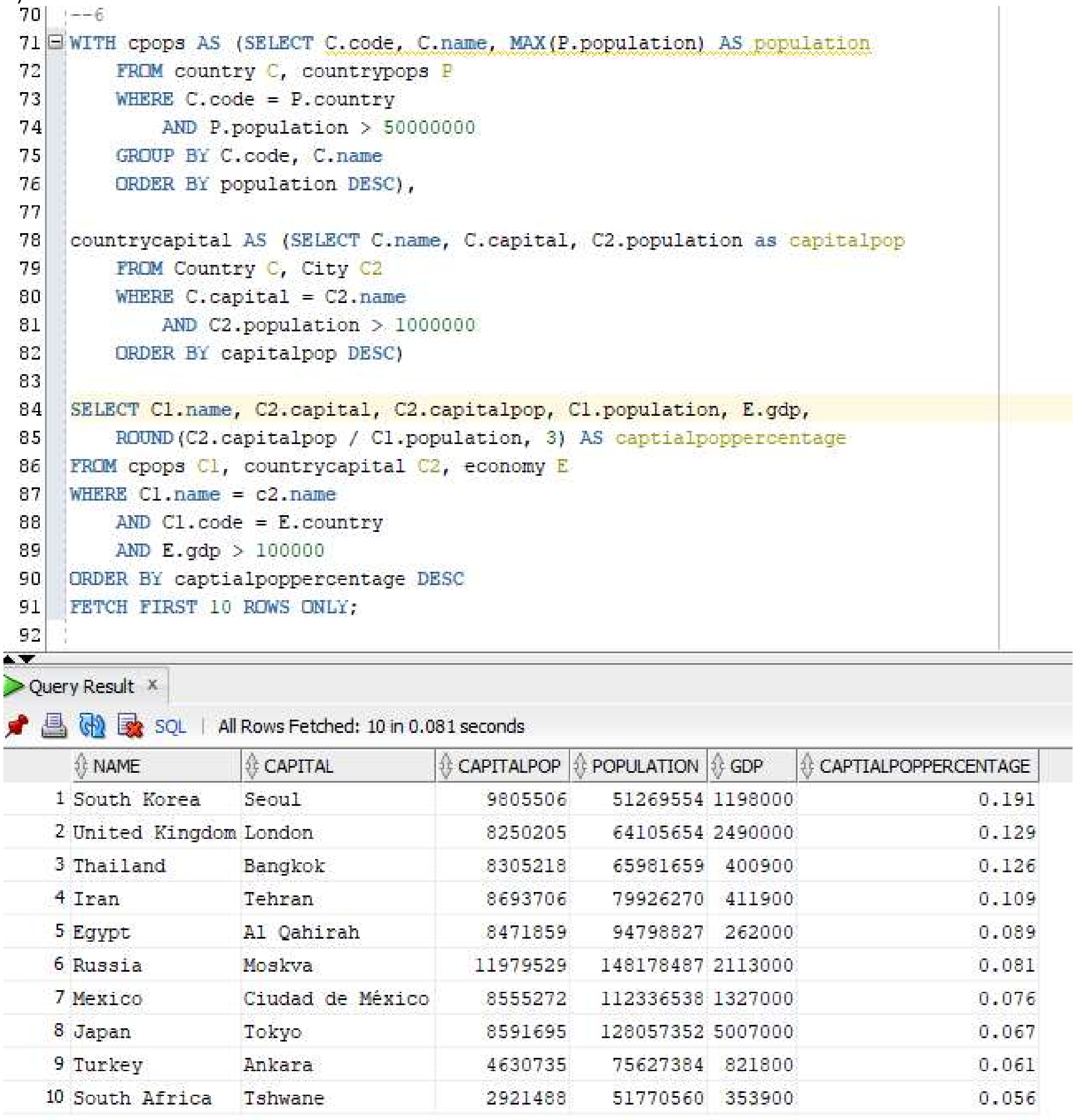
5)



```sql
--5
WITH borderecon AS (SELECT country1, E1.gdp as gdp1, country2, E2.gdp AS gdp2
FROM borders B, economy E1, economy E2
WHERE country2 = E2.country AND country1 = E1.country),

bigborderecon AS(SELECT country1 AS country, gdp1 AS countrygdp, country2 AS bordercountry, gdp2 AS bordergdp
FROM borderecon
UNION
SELECT country2 AS country, gdp2 AS countrygdp, country1 AS bordercountry, gdp1 AS bordergdp
FROM borderecon),

filteredBBE AS (Select * FROM bigborderecon WHERE countrygdp < bordergdp)

SELECT DISTINCT FBBE1.country, MIN(FBBE1.countrygdp) AS countrygdp,
    (SELECT COUNT(*) FROM (SELECT DISTINCT FBBE4.bordercountry FROM filteredBBE FBBE4 WHERE FBBE1.country = FBBE4.country)) AS numWealthier,
    (SELECT ROUND(AVG(bordergdp), 2) FROM (SELECT DISTINCT FBBE4.bordergdp FROM filteredBBE FBBE4 WHERE FBBE1.country = FBBE4.country)) AS avgBorderingGDP
FROM filteredBBE FBBE1, filteredBBE FBBE2, filteredBBE FBBE3
WHERE FBBE1.bordercountry <> FBBE2.bordercountry
    AND FBBE2.bordercountry <> FBBE3.bordercountry
    AND FBBE1.bordercountry <> FBBE3.bordercountry
    AND FBBE1.country = FBBE2.country
    AND FBBE2.country = FBBE3.country
    AND FBBE1.country = FBBE3.country
GROUP BY FBBE1.country
ORDER BY numWealthier DESC;
```

```
Fetched 50 rows in 0.085 seconds

     COUNTRY  COUNTRYGDP  NUMWEALTHIER  AVGBORDERINGGDP
   1 RN             7304             7        119152.71
   2 RCA            2050             6         23091.67
   3 MNE            4518             5            28328
   4 SSD           11770             5            37262
   5 LAO           10100             5          1995194
   6 SK            96960             5           286540
   7 AFG           20650             5          2014828
   8 BOL           30790             4           791650
   9 RMM           11370             4          67867.5
  10 ZRE           18560             4            50195
  11 KGZ            7234             4       2404648.25
  12 MK            10650             4            88370
  13 ARM           10440             4           331415
  14 NAM           12300             4         128917.5
  15 ZW            10480             4           101585
  16 SRB           43680             4           108085
  17 TCH           13590             4           163325
  18 SLO           46820             4           668910
  19 RCB           14250             4          47602.5
  20 MYA           59430             4          2885275
  21 JOR           34080             4           319425
```

6)



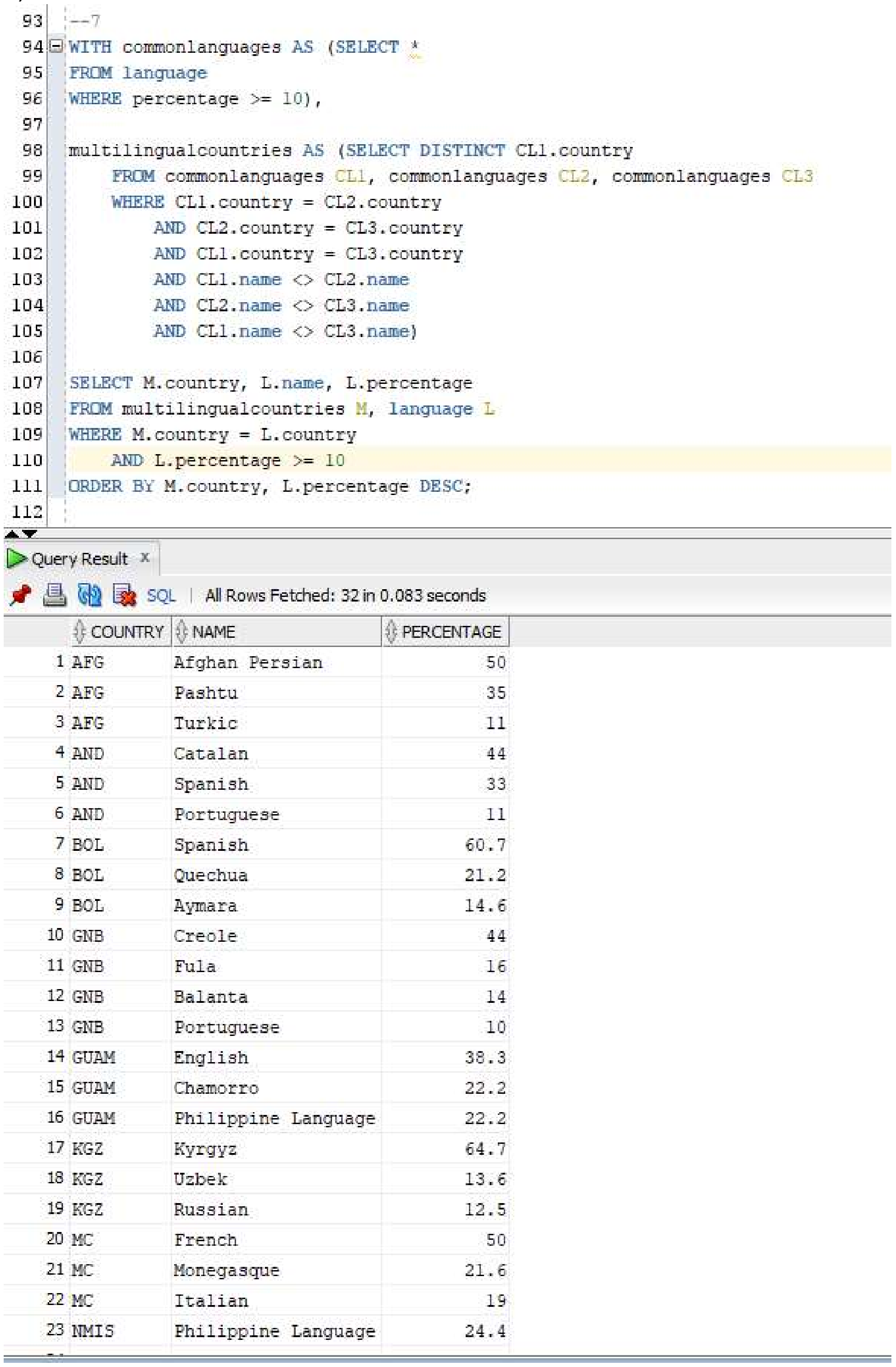
```
--6
WITH cpops AS (SELECT C.code, C.name, MAX(P.population) AS population
    FROM country C, countrypops P
    WHERE C.code = P.country
        AND P.population > 50000000
    GROUP BY C.code, C.name
    ORDER BY population DESC),

countrycapital AS (SELECT C.name, C.capital, C2.population as capitalpop
    FROM Country C, City C2
    WHERE C.capital = C2.name
        AND C2.population > 1000000
    ORDER BY capitalpop DESC)

SELECT C1.name, C2.capital, C2.capitalpop, C1.population, E.gdp,
    ROUND(C2.capitalpop / C1.population, 3) AS captialpoppercentage
FROM cpops C1, countrycapital C2, economy E
WHERE C1.name = c2.name
    AND C1.code = E.country
    AND E.gdp > 100000
ORDER BY captialpoppercentage DESC
FETCH FIRST 10 ROWS ONLY;
```

Query Result

SQL | All Rows Fetched: 10 in 0.081 seconds

	NAME	CAPITAL	CAPITALPOP	POPULATION	GDP	CAPTIALPOPPERCENTAGE
1	South Korea	Seoul	9805506	51269554	1198000	0.191
2	United Kingdom	London	8250205	64105654	2490000	0.129
3	Thailand	Bangkok	8305218	65981659	400900	0.126
4	Iran	Tehran	8693706	79926270	411900	0.109
5	Egypt	Al Qahirah	8471859	94798827	262000	0.089
6	Russia	Moskva	11979529	148178487	2113000	0.081
7	Mexico	Ciudad de México	8555272	112336538	1327000	0.076
8	Japan	Tokyo	8591695	128057352	5007000	0.067
9	Turkey	Ankara	4630735	75627384	821800	0.061
10	South Africa	Tshwane	2921488	51770560	353900	0.056

7)



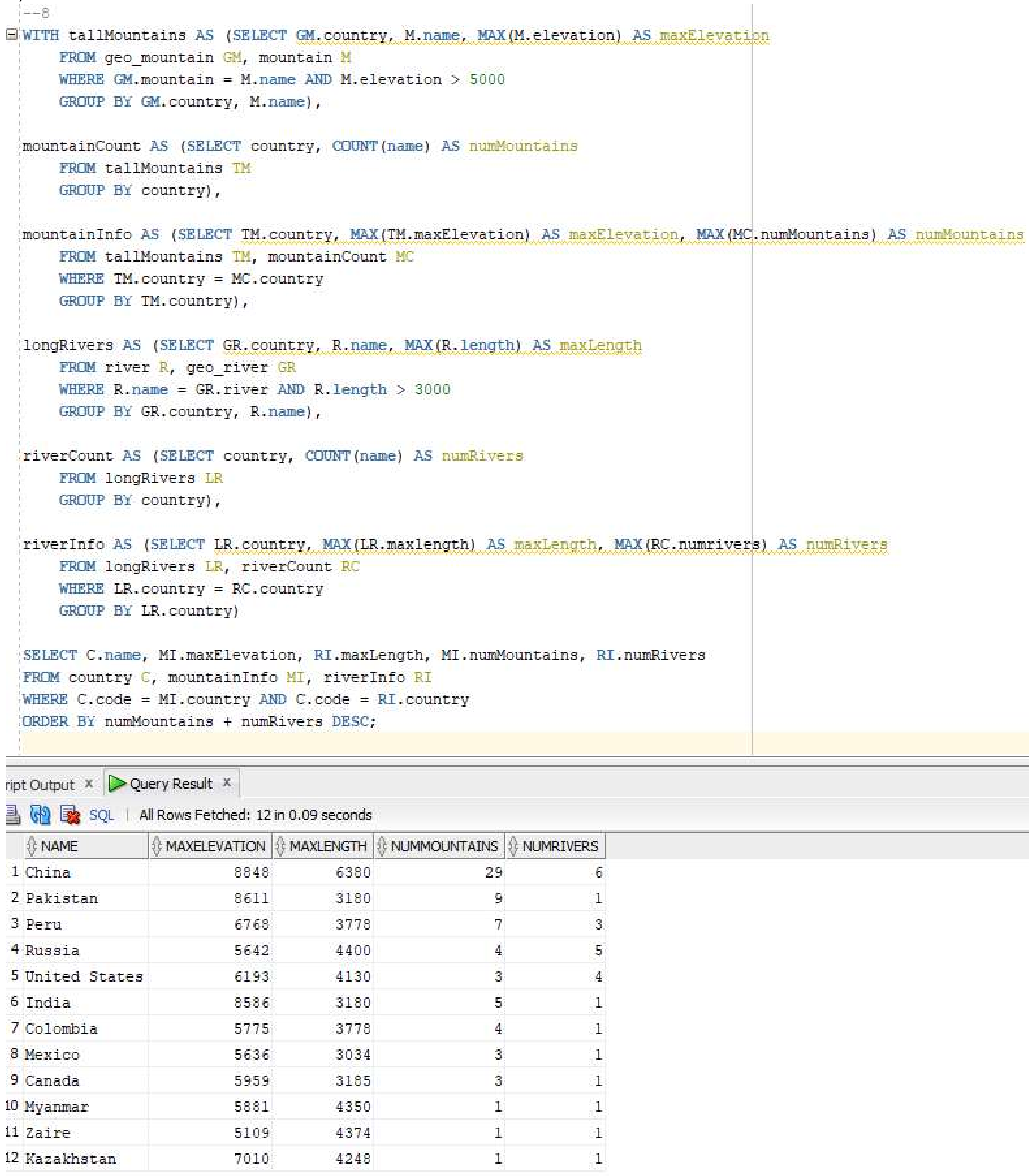
```sql
--7
WITH commonlanguages AS (SELECT *
FROM language
WHERE percentage >= 10),

multilingualcountries AS (SELECT DISTINCT CL1.country
    FROM commonlanguages CL1, commonlanguages CL2, commonlanguages CL3
    WHERE CL1.country = CL2.country
        AND CL2.country = CL3.country
        AND CL1.country = CL3.country
        AND CL1.name <> CL2.name
        AND CL2.name <> CL3.name
        AND CL1.name <> CL3.name)

SELECT M.country, L.name, L.percentage
FROM multilingualcountries M, language L
WHERE M.country = L.country
    AND L.percentage >= 10
ORDER BY M.country, L.percentage DESC;
```

Query Result

SQL | All Rows Fetched: 32 in 0.083 seconds

	COUNTRY	NAME	PERCENTAGE
1	AFG	Afghan Persian	50
2	AFG	Pashtu	35
3	AFG	Turkic	11
4	AND	Catalan	44
5	AND	Spanish	33
6	AND	Portuguese	11
7	BOL	Spanish	60.7
8	BOL	Quechua	21.2
9	BOL	Aymara	14.6
10	GNB	Creole	44
11	GNB	Fula	16
12	GNB	Balanta	14
13	GNB	Portuguese	10
14	GUAM	English	38.3
15	GUAM	Chamorro	22.2
16	GUAM	Philippine Language	22.2
17	KGZ	Kyrgyz	64.7
18	KGZ	Uzbek	13.6
19	KGZ	Russian	12.5
20	MC	French	50
21	MC	Monegasque	21.6
22	MC	Italian	19
23	NMIS	Philippine Language	24.4

8)

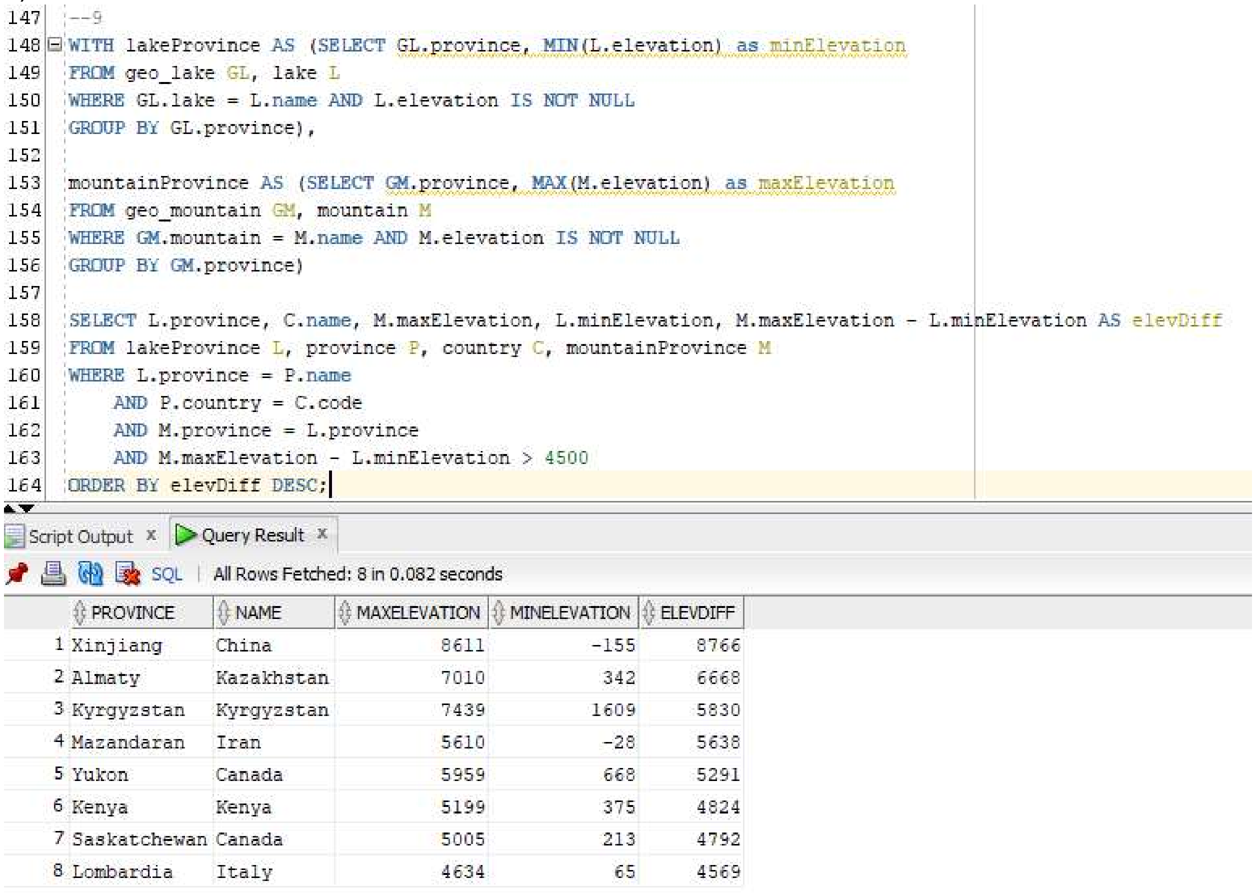


```
--8
WITH tallMountains AS (SELECT GM.country, M.name, MAX(M.elevation) AS maxElevation
    FROM geo_mountain GM, mountain M
    WHERE GM.mountain = M.name AND M.elevation > 5000
    GROUP BY GM.country, M.name),

mountainCount AS (SELECT country, COUNT(name) AS numMountains
    FROM tallMountains TM
    GROUP BY country),

mountainInfo AS (SELECT TM.country, MAX(TM.maxElevation) AS maxElevation, MAX(MC.numMountains) AS numMountains
    FROM tallMountains TM, mountainCount MC
    WHERE TM.country = MC.country
    GROUP BY TM.country),

longRivers AS (SELECT GR.country, R.name, MAX(R.length) AS maxLength
    FROM river R, geo_river GR
    WHERE R.name = GR.river AND R.length > 3000
    GROUP BY GR.country, R.name),

riverCount AS (SELECT country, COUNT(name) AS numRivers
    FROM longRivers LR
    GROUP BY country),

riverInfo AS (SELECT LR.country, MAX(LR.maxlength) AS maxLength, MAX(RC.numrivers) AS numRivers
    FROM longRivers LR, riverCount RC
    WHERE LR.country = RC.country
    GROUP BY LR.country)

SELECT C.name, MI.maxElevation, RI.maxLength, MI.numMountains, RI.numRivers
FROM country C, mountainInfo MI, riverInfo RI
WHERE C.code = MI.country AND C.code = RI.country
ORDER BY numMountains + numRivers DESC;
```

All Rows Fetched: 12 in 0.09 seconds

	NAME	MAXELEVATION	MAXLENGTH	NUMMOUNTAINS	NUMRIVERS
1	China	8848	6380	29	6
2	Pakistan	8611	3180	9	1
3	Peru	6768	3778	7	3
4	Russia	5642	4400	4	5
5	United States	6193	4130	3	4
6	India	8586	3180	5	1
7	Colombia	5775	3778	4	1
8	Mexico	5636	3034	3	1
9	Canada	5959	3185	3	1
10	Myanmar	5881	4350	1	1
11	Zaire	5109	4374	1	1
12	Kazakhstan	7010	4248	1	1

9)

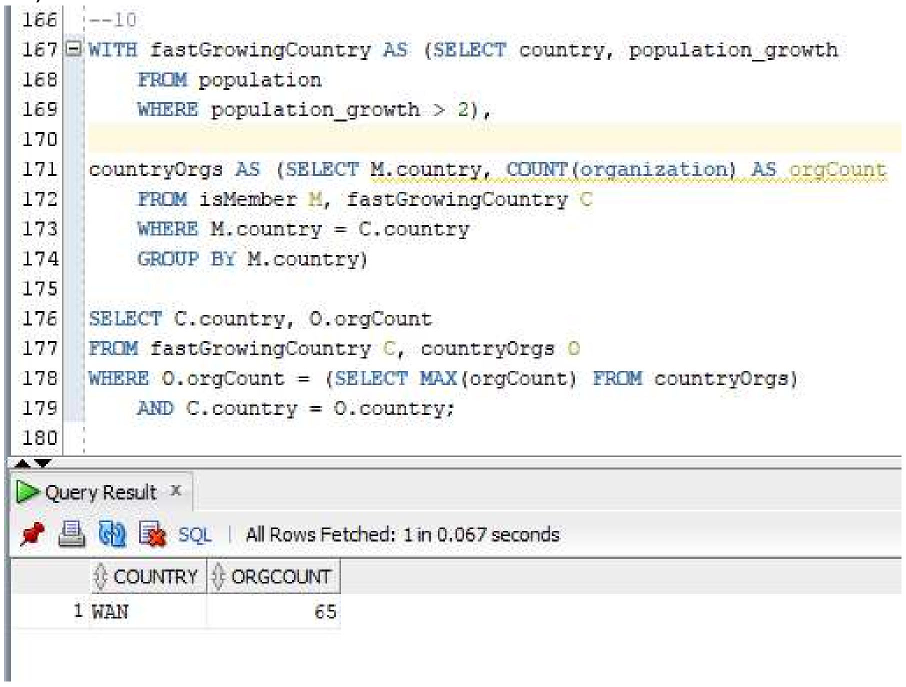


```sql
--9
WITH lakeProvince AS (SELECT GL.province, MIN(L.elevation) as minElevation
FROM geo_lake GL, lake L
WHERE GL.lake = L.name AND L.elevation IS NOT NULL
GROUP BY GL.province),

mountainProvince AS (SELECT GM.province, MAX(M.elevation) as maxElevation
FROM geo_mountain GM, mountain M
WHERE GM.mountain = M.name AND M.elevation IS NOT NULL
GROUP BY GM.province)

SELECT L.province, C.name, M.maxElevation, L.minElevation, M.maxElevation - L.minElevation AS elevDiff
FROM lakeProvince L, province P, country C, mountainProvince M
WHERE L.province = P.name
    AND P.country = C.code
    AND M.province = L.province
    AND M.maxElevation - L.minElevation > 4500
ORDER BY elevDiff DESC;
```

All Rows Fetched: 8 in 0.082 seconds

	PROVINCE	NAME	MAXELEVATION	MINELEVATION	ELEVDIFF
1	Xinjiang	China	8611	-155	8766
2	Almaty	Kazakhstan	7010	342	6668
3	Kyrgyzstan	Kyrgyzstan	7439	1609	5830
4	Mazandaran	Iran	5610	-28	5638
5	Yukon	Canada	5959	668	5291
6	Kenya	Kenya	5199	375	4824
7	Saskatchewan	Canada	5005	213	4792
8	Lombardia	Italy	4634	65	4569

10)

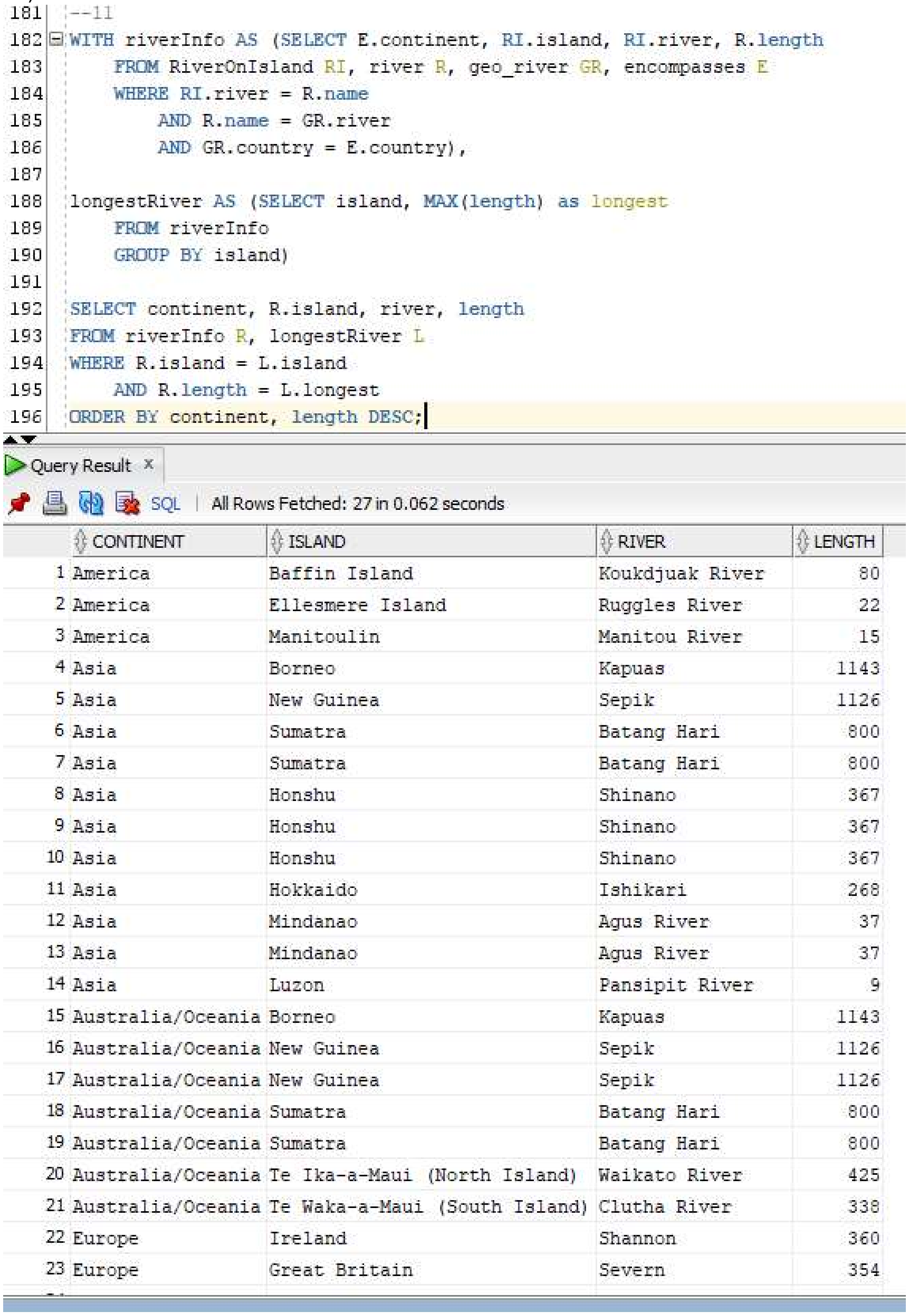


```sql
--10
WITH fastGrowingCountry AS (SELECT country, population_growth
    FROM population
    WHERE population_growth > 2),

countryOrgs AS (SELECT M.country, COUNT(organization) AS orgCount
    FROM isMember M, fastGrowingCountry C
    WHERE M.country = C.country
    GROUP BY M.country)

SELECT C.country, O.orgCount
FROM fastGrowingCountry C, countryOrgs O
WHERE O.orgCount = (SELECT MAX(orgCount) FROM countryOrgs)
    AND C.country = O.country;
```

All Rows Fetched: 1 in 0.067 seconds

	COUNTRY	ORGCOUNT
1	WAN	65

11)



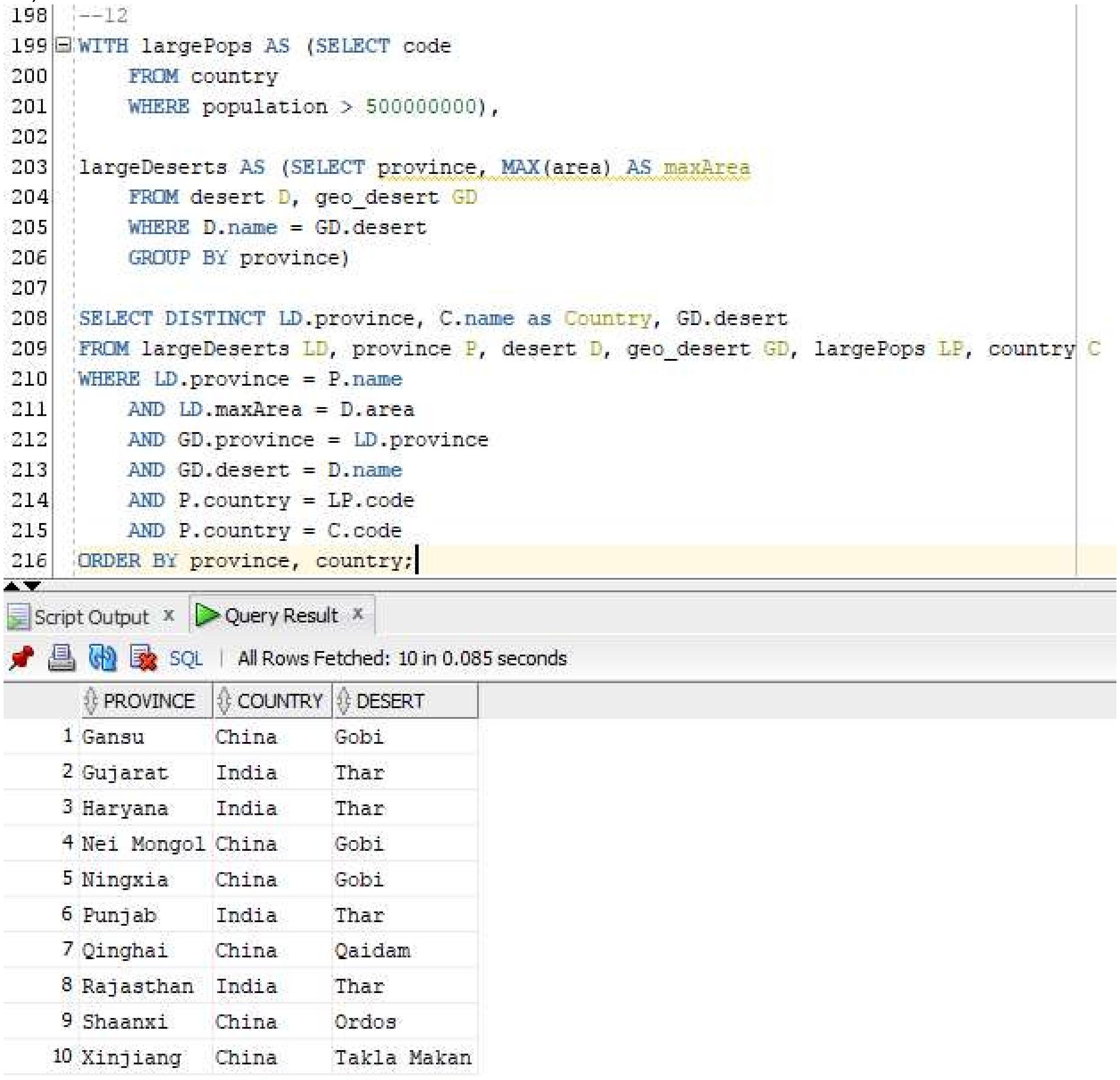
```sql
--11
WITH riverInfo AS (SELECT E.continent, RI.island, RI.river, R.length
    FROM RiverOnIsland RI, river R, geo_river GR, encompasses E
    WHERE RI.river = R.name
        AND R.name = GR.river
        AND GR.country = E.country),

longestRiver AS (SELECT island, MAX(length) as longest
    FROM riverInfo
    GROUP BY island)

SELECT continent, R.island, river, length
FROM riverInfo R, longestRiver L
WHERE R.island = L.island
    AND R.length = L.longest
ORDER BY continent, length DESC;
```

Query Result ×

SQL | All Rows Fetched: 27 in 0.062 seconds

	CONTINENT	ISLAND	RIVER	LENGTH
1	America	Baffin Island	Koukdjuak River	80
2	America	Ellesmere Island	Ruggles River	22
3	America	Manitoulin	Manitou River	15
4	Asia	Borneo	Kapuas	1143
5	Asia	New Guinea	Sepik	1126
6	Asia	Sumatra	Batang Hari	800
7	Asia	Sumatra	Batang Hari	800
8	Asia	Honshu	Shinano	367
9	Asia	Honshu	Shinano	367
10	Asia	Honshu	Shinano	367
11	Asia	Hokkaido	Ishikari	268
12	Asia	Mindanao	Agus River	37
13	Asia	Mindanao	Agus River	37
14	Asia	Luzon	Pansipit River	9
15	Australia/Oceania	Borneo	Kapuas	1143
16	Australia/Oceania	New Guinea	Sepik	1126
17	Australia/Oceania	New Guinea	Sepik	1126
18	Australia/Oceania	Sumatra	Batang Hari	800
19	Australia/Oceania	Sumatra	Batang Hari	800
20	Australia/Oceania	Te Ika-a-Maui (North Island)	Waikato River	425
21	Australia/Oceania	Te Waka-a-Maui (South Island)	Clutha River	338
22	Europe	Ireland	Shannon	360
23	Europe	Great Britain	Severn	354

12)

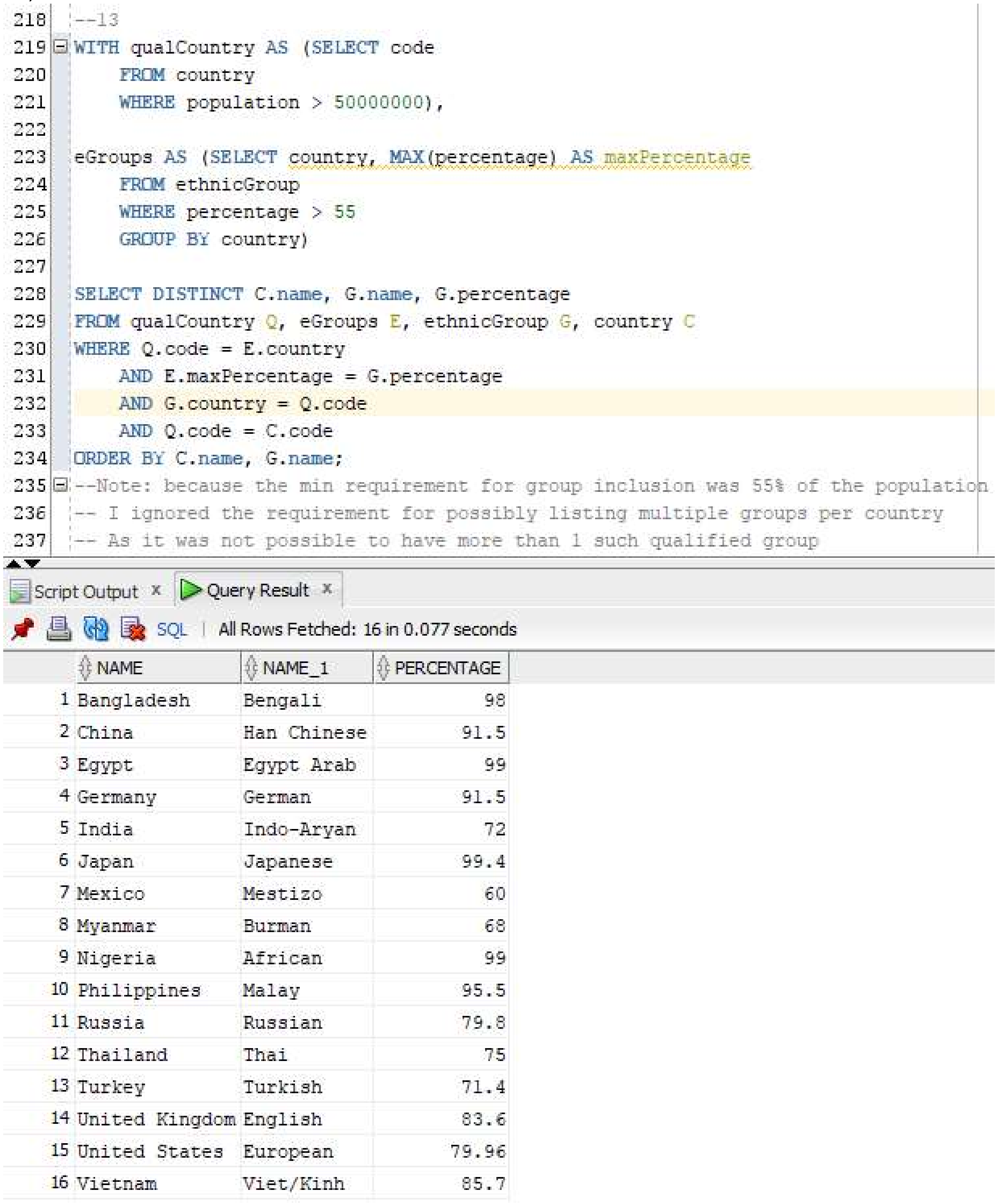


```
--12
WITH largePops AS (SELECT code
    FROM country
    WHERE population > 500000000),

largeDeserts AS (SELECT province, MAX(area) AS maxArea
    FROM desert D, geo_desert GD
    WHERE D.name = GD.desert
    GROUP BY province)

SELECT DISTINCT LD.province, C.name as Country, GD.desert
FROM largeDeserts LD, province P, desert D, geo_desert GD, largePops LP, country C
WHERE LD.province = P.name
    AND LD.maxArea = D.area
    AND GD.province = LD.province
    AND GD.desert = D.name
    AND P.country = LP.code
    AND P.country = C.code
ORDER BY province, country;
```

All Rows Fetched: 10 in 0.085 seconds

	PROVINCE	COUNTRY	DESERT
1	Gansu	China	Gobi
2	Gujarat	India	Thar
3	Haryana	India	Thar
4	Nei Mongol	China	Gobi
5	Ningxia	China	Gobi
6	Punjab	India	Thar
7	Qinghai	China	Qaidam
8	Rajasthan	India	Thar
9	Shaanxi	China	Ordos
10	Xinjiang	China	Takla Makan

13)



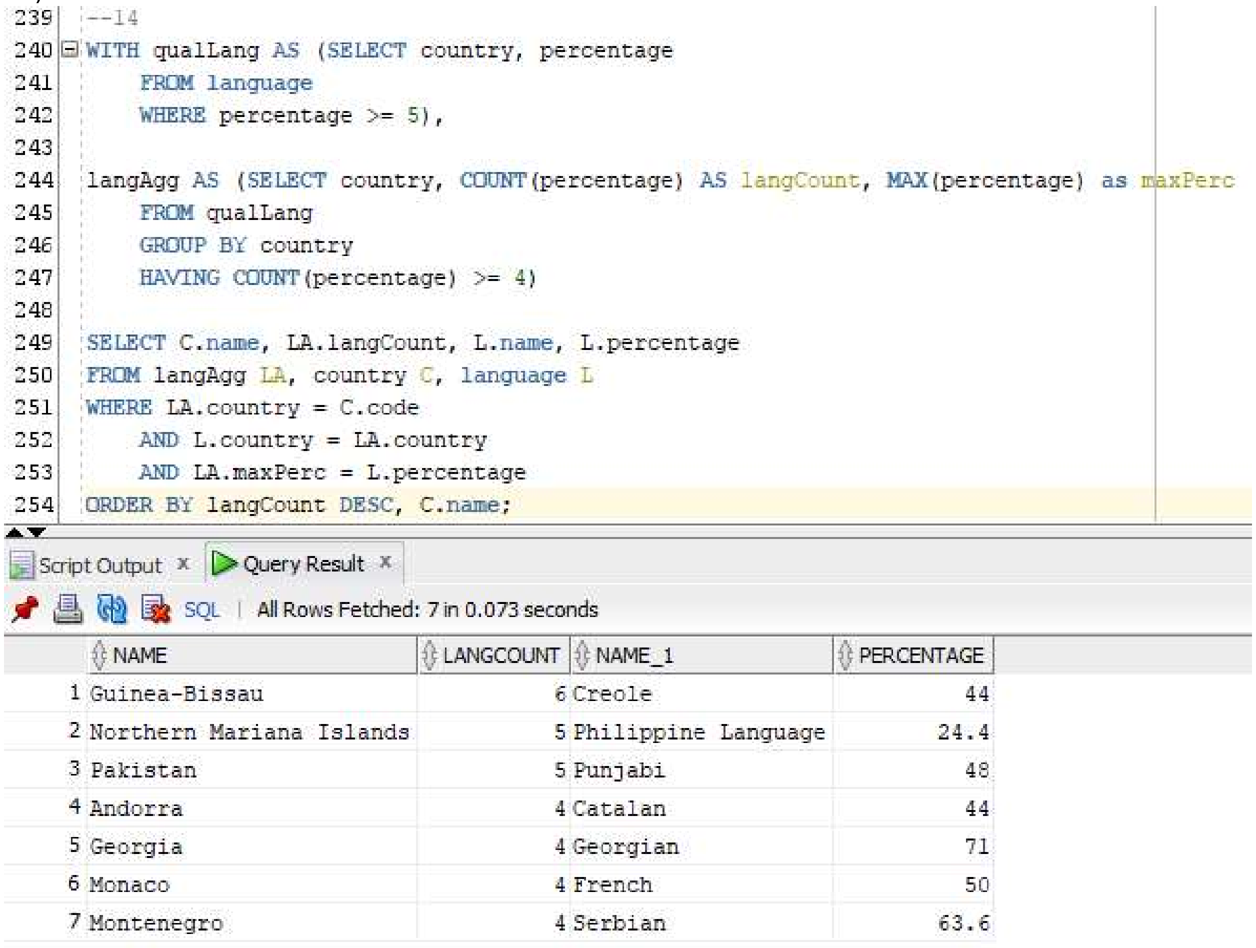
```
--13
WITH qualCountry AS (SELECT code
    FROM country
    WHERE population > 50000000),

eGroups AS (SELECT country, MAX(percentage) AS maxPercentage
    FROM ethnicGroup
    WHERE percentage > 55
    GROUP BY country)

SELECT DISTINCT C.name, G.name, G.percentage
FROM qualCountry Q, eGroups E, ethnicGroup G, country C
WHERE Q.code = E.country
    AND E.maxPercentage = G.percentage
    AND G.country = Q.code
    AND Q.code = C.code
ORDER BY C.name, G.name;
--Note: because the min requirement for group inclusion was 55% of the population
-- I ignored the requirement for possibly listing multiple groups per country
-- As it was not possible to have more than 1 such qualified group
```

Script Output × Query Result ×

SQL | All Rows Fetched: 16 in 0.077 seconds

	NAME	NAME_1	PERCENTAGE
1	Bangladesh	Bengali	98
2	China	Han Chinese	91.5
3	Egypt	Egypt Arab	99
4	Germany	German	91.5
5	India	Indo-Aryan	72
6	Japan	Japanese	99.4
7	Mexico	Mestizo	60
8	Myanmar	Burman	68
9	Nigeria	African	99
10	Philippines	Malay	95.5
11	Russia	Russian	79.8
12	Thailand	Thai	75
13	Turkey	Turkish	71.4
14	United Kingdom	English	83.6
15	United States	European	79.96
16	Vietnam	Viet/Kinh	85.7

14)

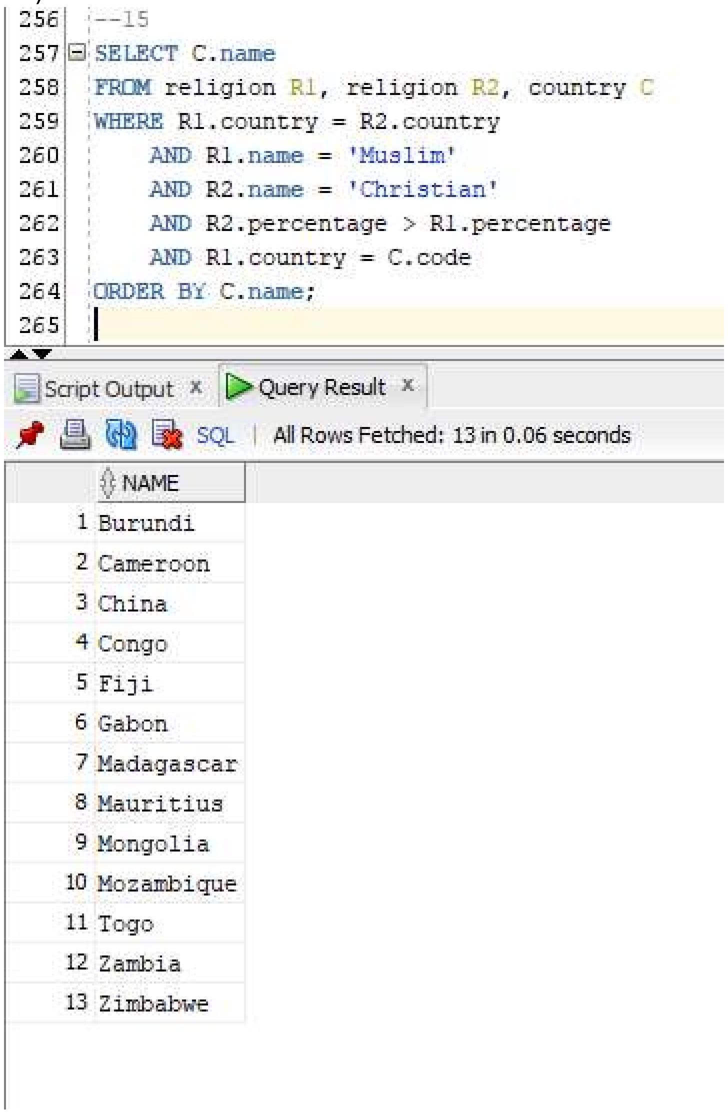


```
--14
WITH qualLang AS (SELECT country, percentage
    FROM language
    WHERE percentage >= 5),

langAgg AS (SELECT country, COUNT(percentage) AS langCount, MAX(percentage) as maxPerc
    FROM qualLang
    GROUP BY country
    HAVING COUNT(percentage) >= 4)

SELECT C.name, LA.langCount, L.name, L.percentage
FROM langAgg LA, country C, language L
WHERE LA.country = C.code
    AND L.country = LA.country
    AND LA.maxPerc = L.percentage
ORDER BY langCount DESC, C.name;
```

All Rows Fetched: 7 in 0.073 seconds

	NAME	LANGCOUNT	NAME_1	PERCENTAGE
1	Guinea-Bissau	6	Creole	44
2	Northern Mariana Islands	5	Philippine Language	24.4
3	Pakistan	5	Punjabi	48
4	Andorra	4	Catalan	44
5	Georgia	4	Georgian	71
6	Monaco	4	French	50
7	Montenegro	4	Serbian	63.6

15)



```
--15
SELECT C.name
FROM religion R1, religion R2, country C
WHERE R1.country = R2.country
    AND R1.name = 'Muslim'
    AND R2.name = 'Christian'
    AND R2.percentage > R1.percentage
    AND R1.country = C.code
ORDER BY C.name;
```

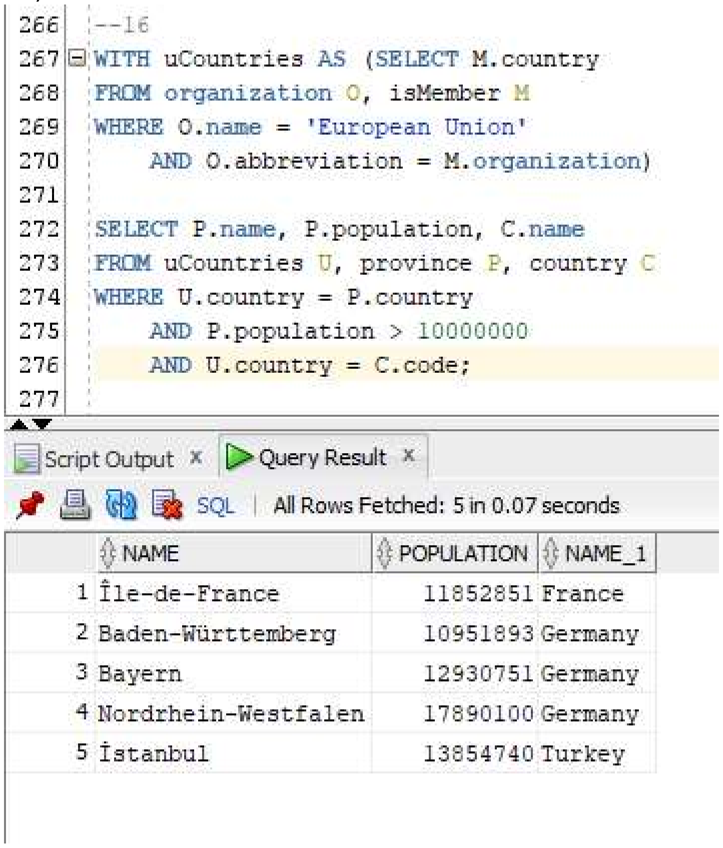
Script Output × Query Result ×

SQL | All Rows Fetched: 13 in 0.06 seconds

```
   NAME
 1 Burundi
 2 Cameroon
 3 China
 4 Congo
 5 Fiji
 6 Gabon
 7 Madagascar
 8 Mauritius
 9 Mongolia
10 Mozambique
11 Togo
12 Zambia
13 Zimbabwe
```

Without additional guidance I took the question literally, only looking at those countries with the specific ‘Christian’ religion. In the event we were supposed to look at every denomination of Christian, I would have used an “AND R2.name IN (‘denomination1’, ‘denomination2’, ...) construct (plus a whole lot of googling to determine how many of the possible religions are some variant of Christianity).

16)

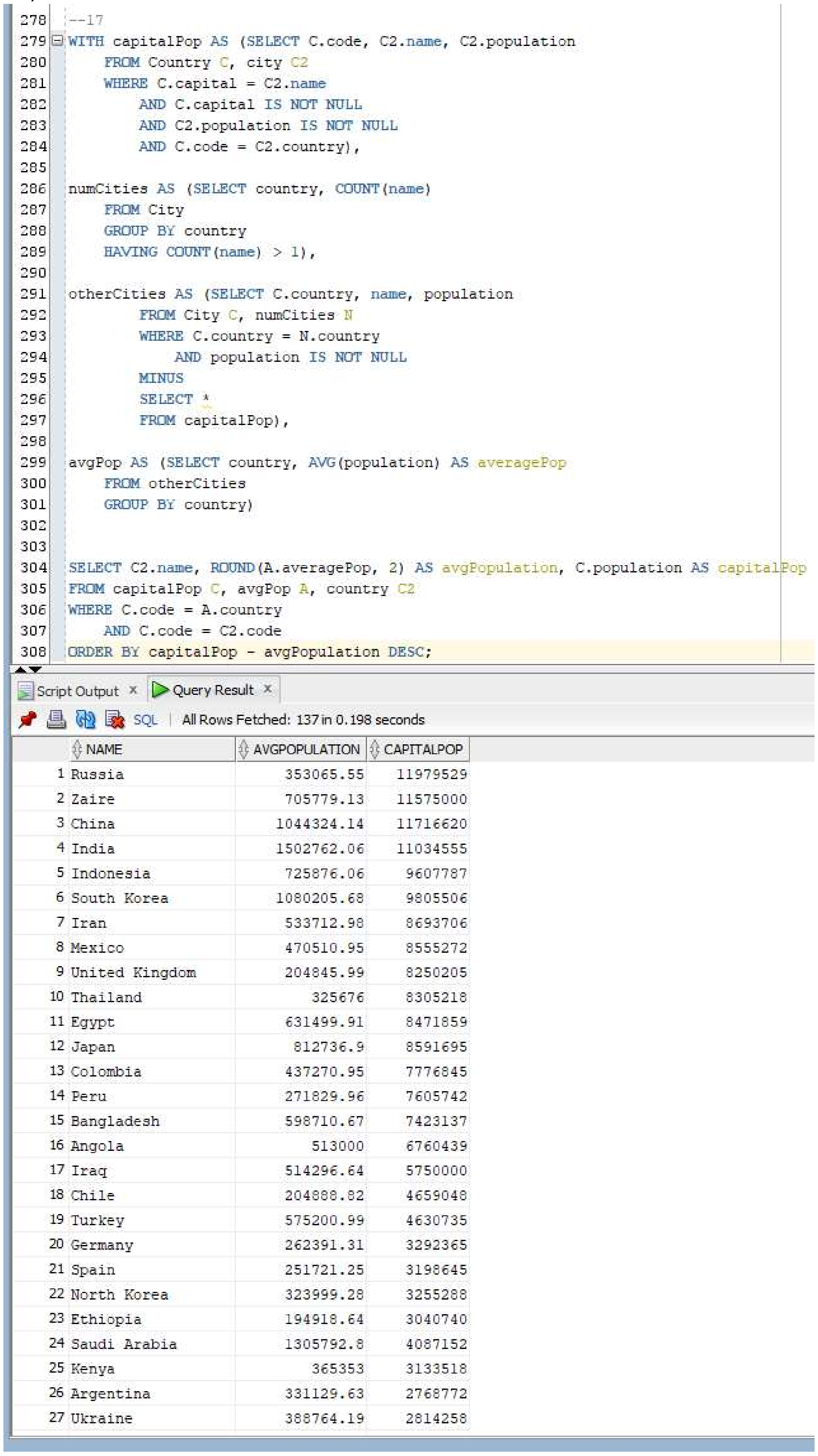


```
--16
WITH uCountries AS (SELECT M.country
FROM organization O, isMember M
WHERE O.name = 'European Union'
    AND O.abbreviation = M.organization)

SELECT P.name, P.population, C.name
FROM uCountries U, province P, country C
WHERE U.country = P.country
    AND P.population > 10000000
    AND U.country = C.code;
```

All Rows Fetched: 5 in 0.07 seconds

	NAME	POPULATION	NAME_1
1	Île-de-France	11852851	France
2	Baden-Württemberg	10951893	Germany
3	Bayern	12930751	Germany
4	Nordrhein-Westfalen	17890100	Germany
5	İstanbul	13854740	Turkey

17)

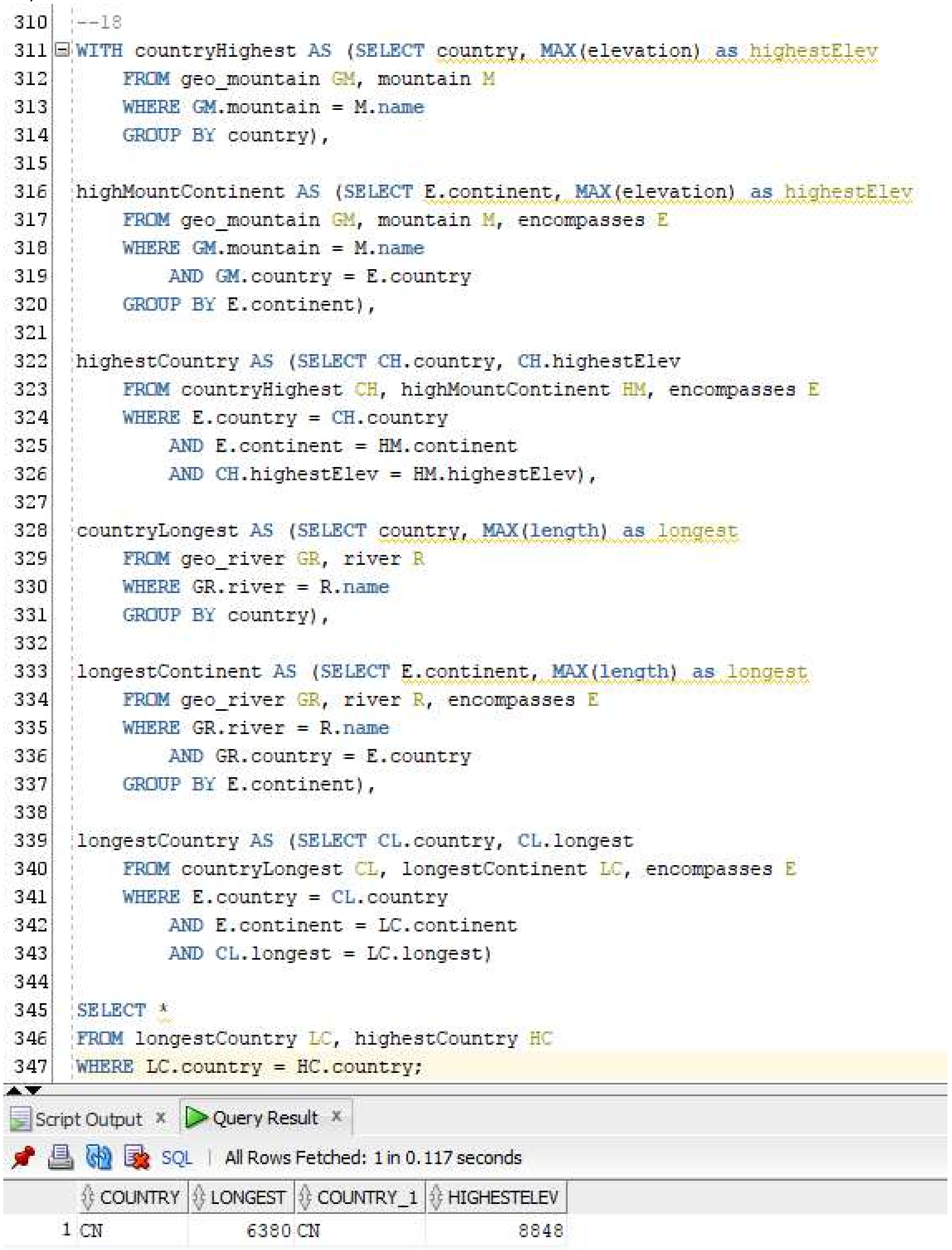


```sql
--17
WITH capitalPop AS (SELECT C.code, C2.name, C2.population
    FROM Country C, city C2
    WHERE C.capital = C2.name
        AND C.capital IS NOT NULL
        AND C2.population IS NOT NULL
        AND C.code = C2.country),

numCities AS (SELECT country, COUNT(name)
    FROM City
    GROUP BY country
    HAVING COUNT(name) > 1),

otherCities AS (SELECT C.country, name, population
        FROM City C, numCities N
        WHERE C.country = N.country
            AND population IS NOT NULL
        MINUS
        SELECT *
        FROM capitalPop),

avgPop AS (SELECT country, AVG(population) AS averagePop
    FROM otherCities
    GROUP BY country)


SELECT C2.name, ROUND(A.averagePop, 2) AS avgPopulation, C.population AS capitalPop
FROM capitalPop C, avgPop A, country C2
WHERE C.code = A.country
    AND C.code = C2.code
ORDER BY capitalPop - avgPopulation DESC;
```

All Rows Fetched: 137 in 0.198 seconds

	NAME	AVGPOPULATION	CAPITALPOP
1	Russia	353065.55	11979529
2	Zaire	705779.13	11575000
3	China	1044324.14	11716620
4	India	1502762.06	11034555
5	Indonesia	725876.06	9607787
6	South Korea	1080205.68	9805506
7	Iran	533712.98	8693706
8	Mexico	470510.95	8555272
9	United Kingdom	204845.99	8250205
10	Thailand	325676	8305218
11	Egypt	631499.91	8471859
12	Japan	812736.9	8591695
13	Colombia	437270.95	7776845
14	Peru	271829.96	7605742
15	Bangladesh	598710.67	7423137
16	Angola	513000	6760439
17	Iraq	514296.64	5750000
18	Chile	204888.82	4659048
19	Turkey	575200.99	4630735
20	Germany	262391.31	3292365
21	Spain	251721.25	3198645
22	North Korea	323999.28	3255288
23	Ethiopia	194918.64	3040740
24	Saudi Arabia	1305792.8	4087152
25	Kenya	365353	3133518
26	Argentina	331129.63	2768772
27	Ukraine	388764.19	2814258

18)



```sql
--18
WITH countryHighest AS (SELECT country, MAX(elevation) as highestElev
    FROM geo_mountain GM, mountain M
    WHERE GM.mountain = M.name
    GROUP BY country),

highMountContinent AS (SELECT E.continent, MAX(elevation) as highestElev
    FROM geo_mountain GM, mountain M, encompasses E
    WHERE GM.mountain = M.name
        AND GM.country = E.country
    GROUP BY E.continent),

highestCountry AS (SELECT CH.country, CH.highestElev
    FROM countryHighest CH, highMountContinent HM, encompasses E
    WHERE E.country = CH.country
        AND E.continent = HM.continent
        AND CH.highestElev = HM.highestElev),

countryLongest AS (SELECT country, MAX(length) as longest
    FROM geo_river GR, river R
    WHERE GR.river = R.name
    GROUP BY country),

longestContinent AS (SELECT E.continent, MAX(length) as longest
    FROM geo_river GR, river R, encompasses E
    WHERE GR.river = R.name
        AND GR.country = E.country
    GROUP BY E.continent),

longestCountry AS (SELECT CL.country, CL.longest
    FROM countryLongest CL, longestContinent LC, encompasses E
    WHERE E.country = CL.country
        AND E.continent = LC.continent
        AND CL.longest = LC.longest)

SELECT *
FROM longestCountry LC, highestCountry HC
WHERE LC.country = HC.country;
```

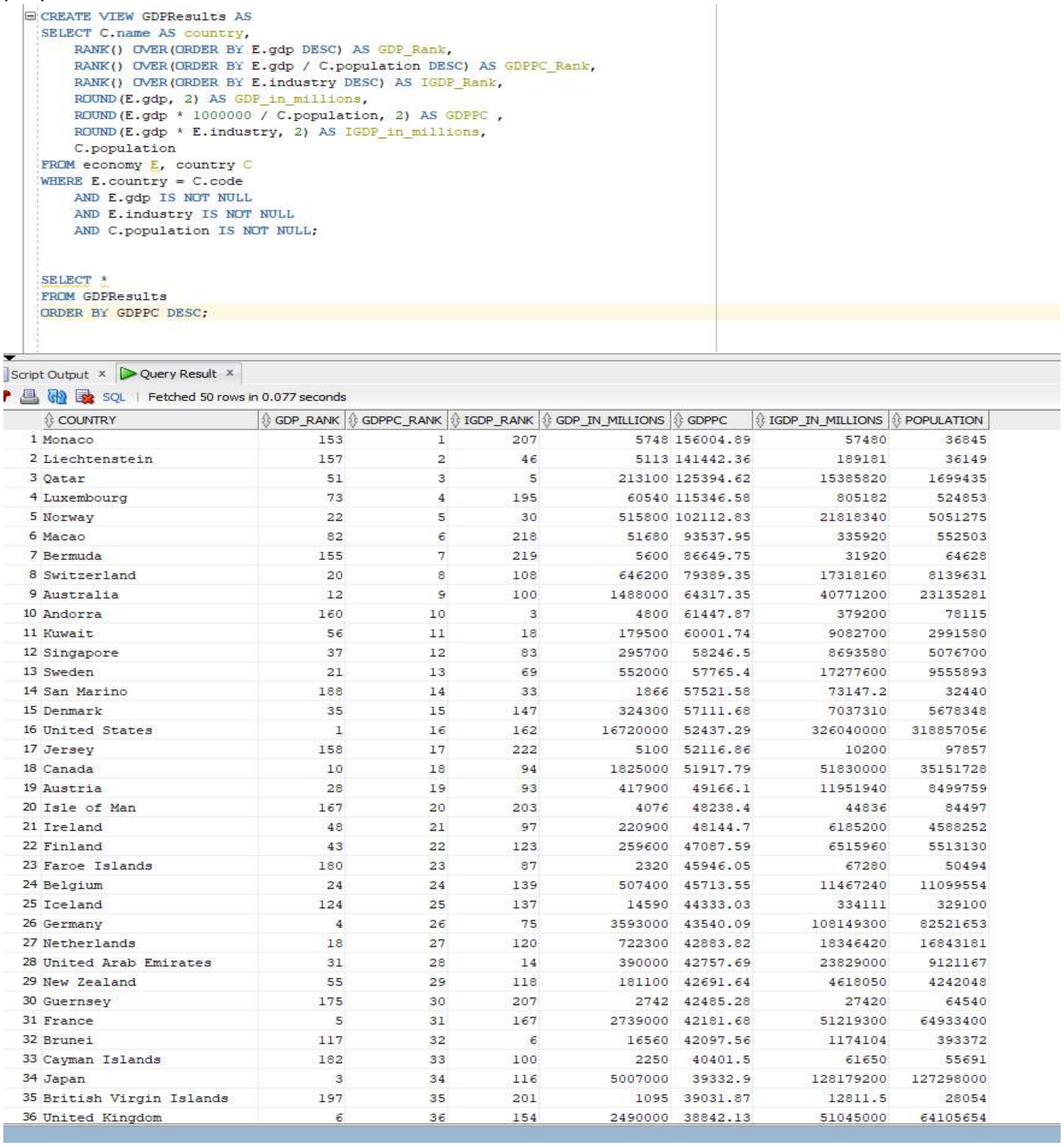
Script Output × Query Result ×

SQL | All Rows Fetched: 1 in 0.117 seconds

	COUNTRY	LONGEST	COUNTRY_1	HIGHESTELEV
1	CN	6380	CN	8848

Question 2

To create persistent queries, I utilized the CREATE VIEW functionality for all of the specified tables (GDPResults, GDPOfCountry, GDPPerCapita, GDPByIndustryPercentage). In order to create the rank order for GDP, GDP per capita, and Industrial GDP Percentage, I utilized the RANK() OVER(ORDER BY [insert column name here] DESC). The RANK function allows ties, and it wasn't specified whether we should allow this or not. In the event that we need to create a strict 1-N ranking, the DENSE_RANK function needs to be used. The final item of note in the creation of the GDPResults view is that the rounding to two decimal places was done via the ROUND function. This is the result of the creation of the GDPResults view, arbitrarily ordered by GDP per capita for viewing purposes:

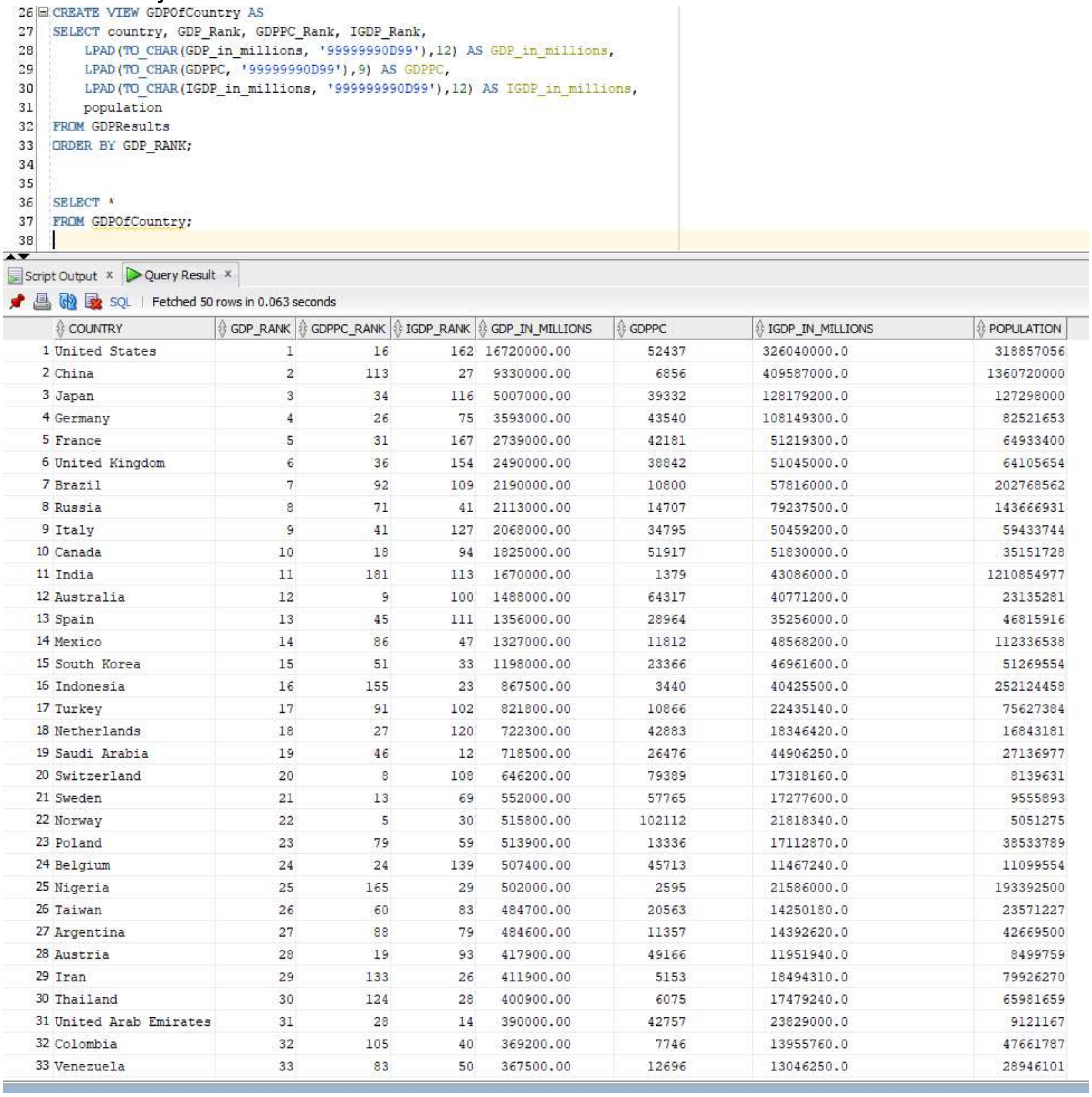


```
CREATE VIEW GDPResults AS
SELECT C.name AS country,
    RANK() OVER(ORDER BY E.gdp DESC) AS GDP_Rank,
    RANK() OVER(ORDER BY E.gdp / C.population DESC) AS GDPPC_Rank,
    RANK() OVER(ORDER BY E.industry DESC) AS IGDP_Rank,
    ROUND(E.gdp, 2) AS GDP_in_millions,
    ROUND(E.gdp * 1000000 / C.population, 2) AS GDPPC ,
    ROUND(E.gdp * E.industry, 2) AS IGDP_in_millions,
    C.population
FROM economy E, country C
WHERE E.country = C.code
    AND E.gdp IS NOT NULL
    AND E.industry IS NOT NULL
    AND C.population IS NOT NULL;


SELECT *
FROM GDPResults
ORDER BY GDPPC DESC;
```

	COUNTRY	GDP_RANK	GDPPC_RANK	IGDP_RANK	GDP_IN_MILLIONS	GDPPC	IGDP_IN_MILLIONS	POPULATION
1	Monaco	153	1	207	5748	156004.89	57480	36845
2	Liechtenstein	157	2	46	5113	141442.36	189181	36149
3	Qatar	51	3	5	213100	125394.62	15385820	1699435
4	Luxembourg	73	4	195	60540	115346.58	805182	524853
5	Norway	22	5	30	515800	102112.83	21818340	5051275
6	Macao	82	6	218	51680	93537.95	335920	552503
7	Bermuda	155	7	219	5600	86649.75	31920	64628
8	Switzerland	20	8	108	646200	79389.35	17318160	8139631
9	Australia	12	9	100	1488000	64317.35	40771200	23135281
10	Andorra	160	10	3	4800	61447.87	379200	78115
11	Kuwait	56	11	18	179500	60001.74	9082700	2991580
12	Singapore	37	12	83	295700	58246.5	8693580	5076700
13	Sweden	21	13	69	552000	57765.4	17277600	9555893
14	San Marino	188	14	33	1866	57521.58	73147.2	32440
15	Denmark	35	15	147	324300	57111.68	7037310	5678348
16	United States	1	16	162	16720000	52437.29	326040000	318857056
17	Jersey	158	17	222	5100	52116.86	10200	97857
18	Canada	10	18	94	1825000	51917.79	51830000	35151728
19	Austria	28	19	93	417900	49166.1	11951940	8499759
20	Isle of Man	167	20	203	4076	48238.4	44836	84497
21	Ireland	48	21	97	220900	48144.7	6185200	4588252
22	Finland	43	22	123	259600	47087.59	6515960	5513130
23	Faroe Islands	180	23	87	2320	45946.05	67280	50494
24	Belgium	24	24	139	507400	45713.55	11467240	11099554
25	Iceland	124	25	137	14590	44333.03	334111	329100
26	Germany	4	26	75	3593000	43540.09	108149300	82521653
27	Netherlands	18	27	120	722300	42883.82	18346420	16843181
28	United Arab Emirates	31	28	14	390000	42757.69	23829000	9121167
29	New Zealand	55	29	118	181100	42691.64	4618050	4242048
30	Guernsey	175	30	207	2742	42485.28	27420	64540
31	France	5	31	167	2739000	42181.68	51219300	64933400
32	Brunei	117	32	6	16560	42097.56	1174104	393372
33	Cayman Islands	182	33	100	2250	40401.5	61650	55691
34	Japan	3	34	116	5007000	39332.9	128179200	127298000
35	British Virgin Islands	197	35	201	1095	39031.87	12811.5	28054
36	United Kingdom	6	36	154	2490000	38842.13	51045000	64105654

The three sub-views that pulled from this data were created in a similar fashion, the only difference is that these views ordered the columns differently, and the data is ordered in the view by the specified parameter. Vertical alignment of the column values was accomplished by converting the numerical value to a string using the TO_CHAR function, and the result of this was used as input into the LPAD (left pad) function. The results of the three sub views are shown below:

GDPOfCountry

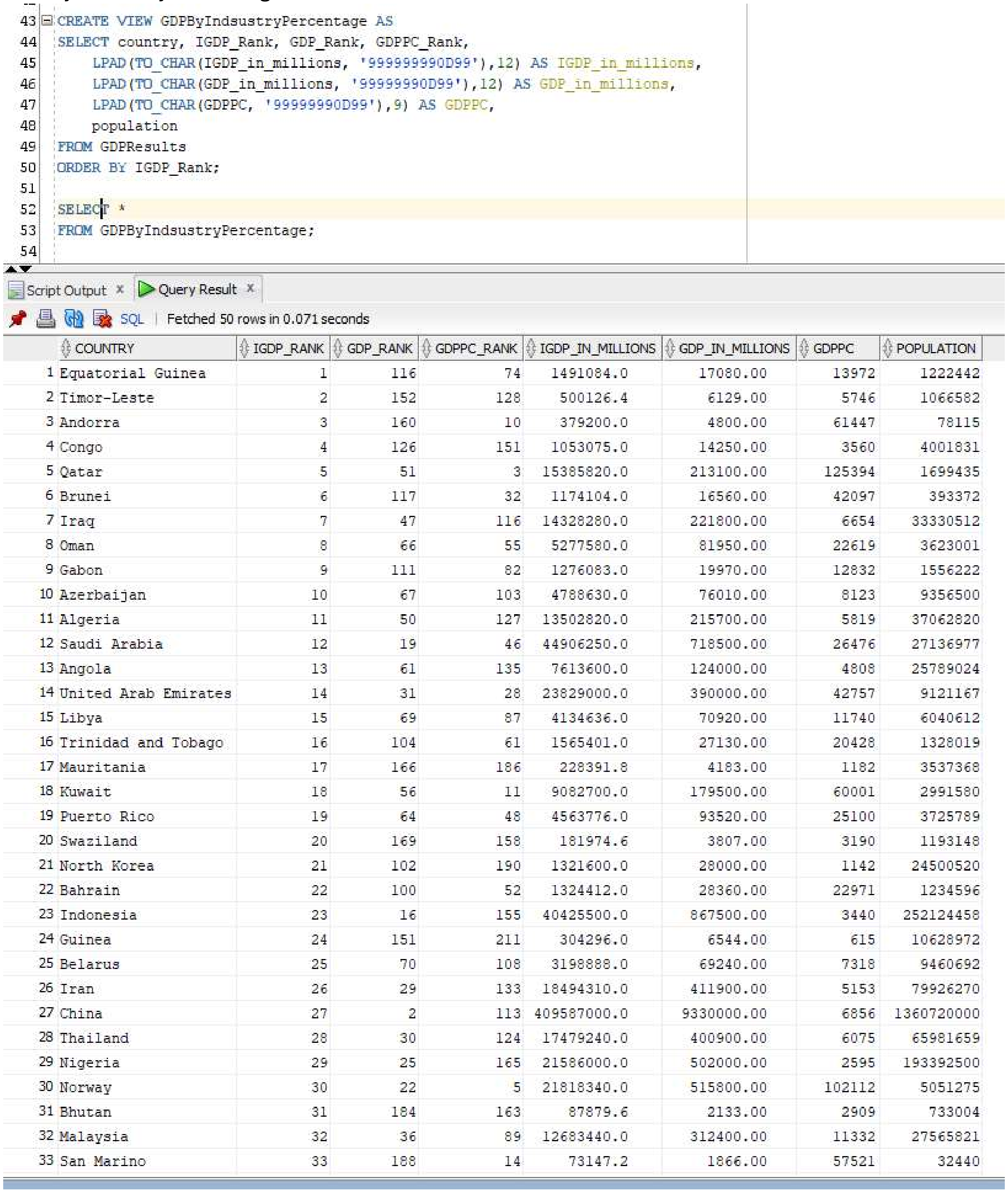


```sql
CREATE VIEW GDPOfCountry AS
SELECT country, GDP_Rank, GDPPC_Rank, IGDP_Rank,
    LPAD(TO_CHAR(GDP_in_millions, '99999990D99'),12) AS GDP_in_millions,
    LPAD(TO_CHAR(GDPPC, '99999990D99'),9) AS GDPPC,
    LPAD(TO_CHAR(IGDP_in_millions, '999999990D99'),12) AS IGDP_in_millions,
    population
FROM GDPResults
ORDER BY GDP_RANK;


SELECT *
FROM GDPOfCountry;
```

Fetched 50 rows in 0.063 seconds

	COUNTRY	GDP_RANK	GDPPC_RANK	IGDP_RANK	GDP_IN_MILLIONS	GDPPC	IGDP_IN_MILLIONS	POPULATION
1	United States	1	16	162	16720000.00	52437	326040000.0	318857056
2	China	2	113	27	9330000.00	6856	409587000.0	1360720000
3	Japan	3	34	116	5007000.00	39332	128179200.0	127298000
4	Germany	4	26	75	3593000.00	43540	108149300.0	82521653
5	France	5	31	167	2739000.00	42181	51219300.0	64933400
6	United Kingdom	6	36	154	2490000.00	38842	51045000.0	64105654
7	Brazil	7	92	109	2190000.00	10800	57816000.0	202768562
8	Russia	8	71	41	2113000.00	14707	79237500.0	143666931
9	Italy	9	41	127	2068000.00	34795	50459200.0	59433744
10	Canada	10	18	94	1825000.00	51917	51830000.0	35151728
11	India	11	181	113	1670000.00	1379	43086000.0	1210854977
12	Australia	12	9	100	1488000.00	64317	40771200.0	23135281
13	Spain	13	45	111	1356000.00	28964	35256000.0	46815916
14	Mexico	14	86	47	1327000.00	11812	48568200.0	112336538
15	South Korea	15	51	33	1198000.00	23366	46961600.0	51269554
16	Indonesia	16	155	23	867500.00	3440	40425500.0	252124458
17	Turkey	17	91	102	821800.00	10866	22435140.0	75627384
18	Netherlands	18	27	120	722300.00	42883	18346420.0	16843181
19	Saudi Arabia	19	46	12	718500.00	26476	44906250.0	27136977
20	Switzerland	20	8	108	646200.00	79389	17318160.0	8139631
21	Sweden	21	13	69	552000.00	57765	17277600.0	9555893
22	Norway	22	5	30	515800.00	102112	21818340.0	5051275
23	Poland	23	79	59	513900.00	13336	17112870.0	38533789
24	Belgium	24	24	139	507400.00	45713	11467240.0	11099554
25	Nigeria	25	165	29	502000.00	2595	21586000.0	193392500
26	Taiwan	26	60	83	484700.00	20563	14250180.0	23571227
27	Argentina	27	88	79	484600.00	11357	14392620.0	42669500
28	Austria	28	19	93	417900.00	49166	11951940.0	8499759
29	Iran	29	133	26	411900.00	5153	18494310.0	79926270
30	Thailand	30	124	28	400900.00	6075	17479240.0	65981659
31	United Arab Emirates	31	28	14	390000.00	42757	23829000.0	9121167
32	Colombia	32	105	40	369200.00	7746	13955760.0	47661787
33	Venezuela	33	83	50	367500.00	12696	13046250.0	28946101

GDPByIndsustryPercentage



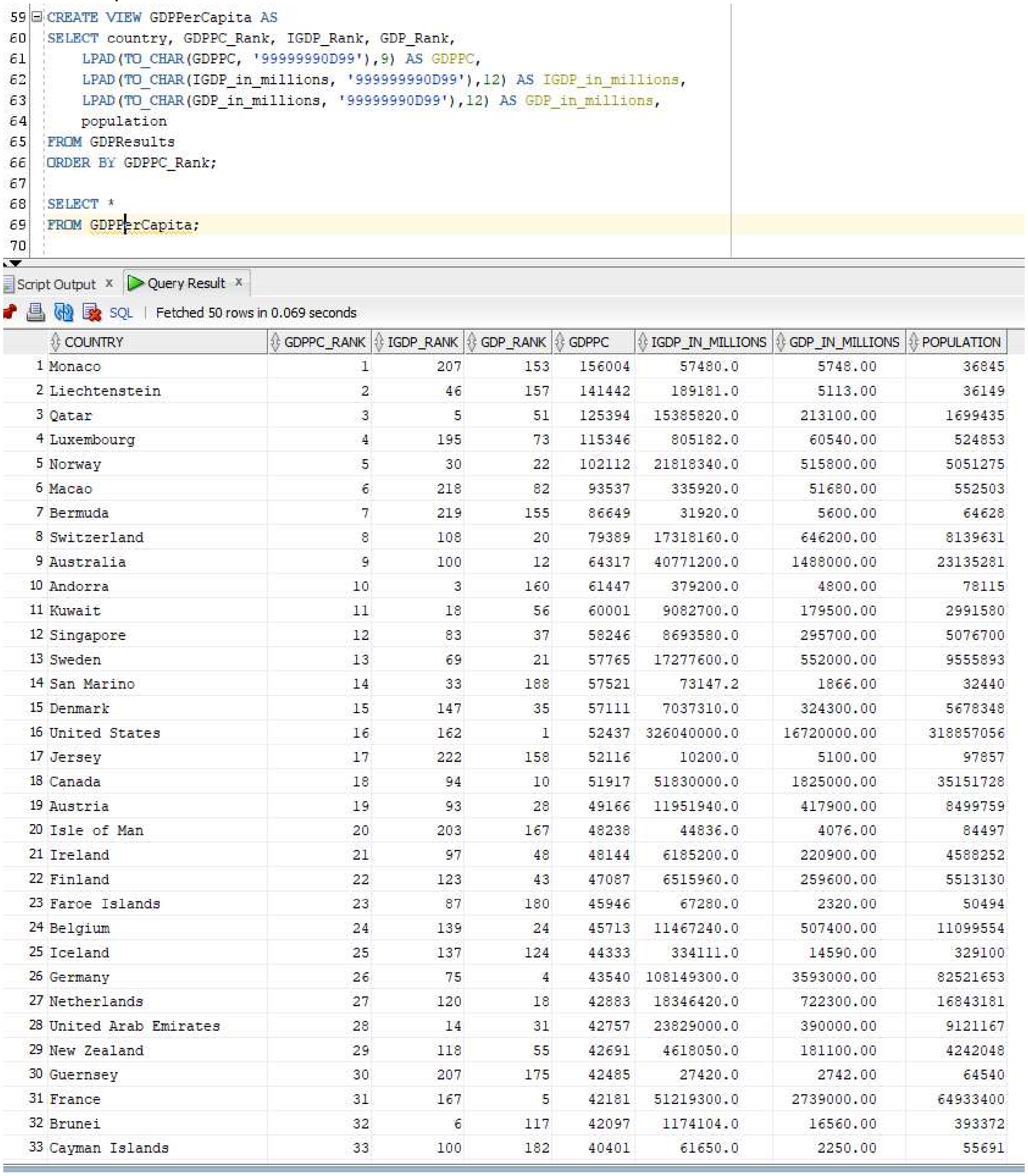
```
CREATE VIEW GDPByIndsustryPercentage AS
SELECT country, IGDP_Rank, GDP_Rank, GDPPC_Rank,
    LPAD(TO_CHAR(IGDP_in_millions, '999999990D99'),12) AS IGDP_in_millions,
    LPAD(TO_CHAR(GDP_in_millions, '99999990D99'),12) AS GDP_in_millions,
    LPAD(TO_CHAR(GDPPC, '99999990D99'),9) AS GDPPC,
    population
FROM GDPResults
ORDER BY IGDP_Rank;

SELECT *
FROM GDPByIndsustryPercentage;
```

Script Output × Query Result ×

SQL | Fetched 50 rows in 0.071 seconds

	COUNTRY	IGDP_RANK	GDP_RANK	GDPPC_RANK	IGDP_IN_MILLIONS	GDP_IN_MILLIONS	GDPPC	POPULATION
1	Equatorial Guinea	1	116	74	1491084.0	17080.00	13972	1222442
2	Timor-Leste	2	152	128	500126.4	6129.00	5746	1066582
3	Andorra	3	160	10	379200.0	4800.00	61447	78115
4	Congo	4	126	151	1053075.0	14250.00	3560	4001831
5	Qatar	5	51	3	15385820.0	213100.00	125394	1699435
6	Brunei	6	117	32	1174104.0	16560.00	42097	393372
7	Iraq	7	47	116	14328280.0	221800.00	6654	33330512
8	Oman	8	66	55	5277580.0	81950.00	22619	3623001
9	Gabon	9	111	82	1276083.0	19970.00	12832	1556222
10	Azerbaijan	10	67	103	4788630.0	76010.00	8123	9356500
11	Algeria	11	50	127	13502820.0	215700.00	5819	37062820
12	Saudi Arabia	12	19	46	44906250.0	718500.00	26476	27136977
13	Angola	13	61	135	7613600.0	124000.00	4808	25789024
14	United Arab Emirates	14	31	28	23829000.0	390000.00	42757	9121167
15	Libya	15	69	87	4134636.0	70920.00	11740	6040612
16	Trinidad and Tobago	16	104	61	1565401.0	27130.00	20428	1328019
17	Mauritania	17	166	186	228391.8	4183.00	1182	3537368
18	Kuwait	18	56	11	9082700.0	179500.00	60001	2991580
19	Puerto Rico	19	64	48	4563776.0	93520.00	25100	3725789
20	Swaziland	20	169	158	181974.6	3807.00	3190	1193148
21	North Korea	21	102	190	1321600.0	28000.00	1142	24500520
22	Bahrain	22	100	52	1324412.0	28360.00	22971	1234596
23	Indonesia	23	16	155	40425500.0	867500.00	3440	252124458
24	Guinea	24	151	211	304296.0	6544.00	615	10628972
25	Belarus	25	70	108	3198888.0	69240.00	7318	9460692
26	Iran	26	29	133	18494310.0	411900.00	5153	79926270
27	China	27	2	113	409587000.0	9330000.00	6856	1360720000
28	Thailand	28	30	124	17479240.0	400900.00	6075	65981659
29	Nigeria	29	25	165	21586000.0	502000.00	2595	193392500
30	Norway	30	22	5	21818340.0	515800.00	102112	5051275
31	Bhutan	31	184	163	87879.6	2133.00	2909	733004
32	Malaysia	32	36	89	12683440.0	312400.00	11332	27565821
33	San Marino	33	188	14	73147.2	1866.00	57521	32440

GDPPerCapita



```sql
CREATE VIEW GDPPerCapita AS
SELECT country, GDPPC_Rank, IGDP_Rank, GDP_Rank,
    LPAD(TO_CHAR(GDPPC, '99999990D99'),9) AS GDPPC,
    LPAD(TO_CHAR(IGDP_in_millions, '999999990D99'),12) AS IGDP_in_millions,
    LPAD(TO_CHAR(GDP_in_millions, '99999990D99'),12) AS GDP_in_millions,
    population
FROM GDPResults
ORDER BY GDPPC_Rank;

SELECT *
FROM GDPPerCapita;
```

Fetched 50 rows in 0.069 seconds

	COUNTRY	GDPPC_RANK	IGDP_RANK	GDP_RANK	GDPPC	IGDP_IN_MILLIONS	GDP_IN_MILLIONS	POPULATION
1	Monaco	1	207	153	156004	57480.0	5748.00	36845
2	Liechtenstein	2	46	157	141442	189181.0	5113.00	36149
3	Qatar	3	5	51	125394	15385820.0	213100.00	1699435
4	Luxembourg	4	195	73	115346	805182.0	60540.00	524853
5	Norway	5	30	22	102112	21818340.0	515800.00	5051275
6	Macao	6	218	82	93537	335920.0	51680.00	552503
7	Bermuda	7	219	155	86649	31920.0	5600.00	64628
8	Switzerland	8	108	20	79389	17318160.0	646200.00	8139631
9	Australia	9	100	12	64317	40771200.0	1488000.00	23135281
10	Andorra	10	3	160	61447	379200.0	4800.00	78115
11	Kuwait	11	18	56	60001	9082700.0	179500.00	2991580
12	Singapore	12	83	37	58246	8693580.0	295700.00	5076700
13	Sweden	13	69	21	57765	17277600.0	552000.00	9555893
14	San Marino	14	33	188	57521	73147.2	1866.00	32440
15	Denmark	15	147	35	57111	7037310.0	324300.00	5678348
16	United States	16	162	1	52437	326040000.0	16720000.00	318857056
17	Jersey	17	222	158	52116	10200.0	5100.00	97857
18	Canada	18	94	10	51917	51830000.0	1825000.00	35151728
19	Austria	19	93	28	49166	11951940.0	417900.00	8499759
20	Isle of Man	20	203	167	48238	44836.0	4076.00	84497
21	Ireland	21	97	48	48144	6185200.0	220900.00	4588252
22	Finland	22	123	43	47087	6515960.0	259600.00	5513130
23	Faroe Islands	23	87	180	45946	67280.0	2320.00	50494
24	Belgium	24	139	24	45713	11467240.0	507400.00	11099554
25	Iceland	25	137	124	44333	334111.0	14590.00	329100
26	Germany	26	75	4	43540	108149300.0	3593000.00	82521653
27	Netherlands	27	120	18	42883	18346420.0	722300.00	16843181
28	United Arab Emirates	28	14	31	42757	23829000.0	390000.00	9121167
29	New Zealand	29	118	55	42691	4618050.0	181100.00	4242048
30	Guernsey	30	207	175	42485	27420.0	2742.00	64540
31	France	31	167	5	42181	51219300.0	2739000.00	64933400
32	Brunei	32	6	117	42097	1174104.0	16560.00	393372
33	Cayman Islands	33	100	182	40401	61650.0	2250.00	55691

Conclusion
Two of the final views, specifically the one focusing on overall GDP and the one focusing on GDP per capita, revealed nothing unexpected. The GDP rankings were dominated by large, mostly western countries. Though I do have to admit that Brazil at rank seven was a surprise to me, but this surprise can be entirely attributed to my lack of background knowledge on Brazil's economy. The GDP per capita ranking was dominated by small, wealthy nations of two general varieties. The first variety are those countries with access to large amounts of natural resources relative to their small population. This includes countries like Qatar and Norway. The other variety are those countries that are widely considered to be tax havens, such as Monaco, Switzerland, and Bermuda. The third view, however, was surprising. My initial expectations were that the list would be somewhat similar to the GDP rankings, dominated by relatively wealthy, western countries. But I failed to consider the gradual transition from industry to services that countries make as they get wealthier and was surprised that most countries on this list were relatively poor and generally considered to be in the 'developing countries' category.

Question 3
A) Not possible; the lowest nested subquery (SELECT * FROM works WHERE employee.enum = works.enum AND works.pnum = projects.pnum) attempts to join tables (employee, projects) that it has not imported, resulting in an error.

B) Not possible; there is no GROUP BY functionality in relational algebra. Some googling has suggested that there have been attempted to extend the set of relational algebra operations (I saw one potential symbol for GROUP BY. It was very squiggly), but nothing that has been introduced in this course allows this functionality.