# Analysis of Countries Database Using SQL

In this project I have analyzed countries' database data. I have completed four project goals here.
Database Dump File Link - https://github.com/ikramulS/Analysis-of-Countries-Database-Using-SQL-.git

1. **Project Goal 1**
   Identify the cities from those countries that they are included in either economies or currencies but not in populations.

   **SQL Query**

```
Select  c1.name
  from cities AS c1
  WHERE country_code IN
(
    SELECT e.code
    FROM economies AS  e
    union
    SELECT c2.code
    FROM currencies AS c2
        Except        -- Exclude the code those are appearing in population
    SELECT p.country_code
    FROM populations AS p
);  -- Select all the country code  from economies and currencies except populations
```

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Dependents CountriesV1/postgres@PostgreSQL 14 *

No limit

CountriesV1/postgres@PostgreSQL 14

Query Editor Query History Scratch Pad

- Servers
  - PostgreSQL 14
    - Databases (2)
      - CountriesV1
        - Casts
        - Catalogs
        - Event Triggers
        - Extensions
        - Foreign Data Wrappers
        - Languages
        - Publications
        - Schemas (1)
          - public
            - Collations
            - Domains
            - FTS Configurations
            - FTS Dictionaries
            - FTS Parsers
            - FTS Templates
            - Foreign Tables
            - Functions
            - Materialized Views
            - Procedures
            - Sequences

```sql
Select  c1.name
  from cities AS c1
  -- semi join (Choose only records matching the result of multiple set theory clauses)
  WHERE country_code IN
(
    -- Select all the country code  from economies and currencies
    SELECT e.code
    FROM economies AS  e
    union
    SELECT c2.code
    FROM currencies AS c2
    -- Exclude the code those are appearing in population
    except
    SELECT p.country_code
    FROM populations AS p
);
```

**Output -**

Data Output | Explain | Messages | Notifications

| | name [PK] character varying |
|---|---|
| 1 | Bucharest |
| 2 | Kaohsiung |
| 3 | New Taipei City |
| 4 | Taichung |
| 5 | Tainan |
| 6 | Taipei |

-

These six cities  are not recorded in the ‘Population’ table.

2. **Project Goal 2**
   Identify which country had the max of gdp_percapita, and how high it was, using multiple subqueries., for each of the six continents listed in 2015.

**SQL Query**

```
SELECT name, continent, gdp_percapita
   FROM countries
          INNER JOIN economies  -- Join to economies
          using(code)  -- Match on code
    WHERE year = 2015  -- Where year is 2015
    and gdp_percapita in (
        SELECT MAX(gdp_percapita) AS max_gdp_percapita
        FROM (
             SELECT name, continent, gdp_percapita
             FROM countries
             INNER JOIN economies
             using(code)
             WHERE year = 2015) AS subquery  -- And inflation rate in subquery
(alias as subquery)


        GROUP BY continent)    -- Group by continent
          order by gdp_percapita desc;
```

pgAdmin 4

pgAdmin File Object Tools Help

Browser | Dashboard | Properties | SQL | Statistics | Dependencies | Dependents | CountriesV1/postgres@PostgreS

- Servers
  - PostgreSQL 14
    - Databases (2)
      - CountriesV1
        - Casts
        - Catalogs
        - Event Triggers
        - Extensions
        - Foreign Data Wrappers
        - Languages
        - Publications
        - Schemas (1)
          - public
            - Collations
            - Domains
            - FTS Configurations
            - FTS Dictionaries
            - FTS Parsers
            - FTS Templates
            - Foreign Tables
            - Functions
            - Materialized Views
            - Procedures
            - Sequences
            - Tables (9)

No limit

CountriesV1/postgres@PostgreSQL 14

Query Editor | Query History

```
-- Select fields
SELECT name, continent, gdp_percapita
  -- From countries
  FROM countries
    -- Join to economies
    INNER JOIN economies
    -- Match on code
    using(code)
  -- Where year is 2015
  WHERE year = 2015
    -- And inflation rate in subquery (alias as subquery)
    and gdp_percapita in (
        SELECT MAX(gdp_percapita) AS max_gdp_percapita
        FROM (
             SELECT name, continent, gdp_percapita
             FROM countries
             INNER JOIN economies
             using(code)
             WHERE year = 2015) AS subquery

      -- Group by continent
        GROUP BY continent)
        order by gdp_percapita desc;
```

**Output-**

Data Output | Explain | Messages | Notifications

| | name character varying | continent character varying | gdp_percapita real |
|---|---|---|---|
| 1 | Luxembourg | Europe | 100950.49 |
| 2 | Macao | Asia | 70214.9 |
| 3 | United States | North America | 56174.94 |
| 4 | Australia | Oceania | 51363.9 |
| 5 | Equatorial Guinea | Africa | 17286.92 |
| 6 | Uruguay | South America | 15317.58 |

Luxembourg has the highest gdp_percapita in the year of 2015 in Europe. In Asia it is Macao.

3. **Project Goal 3**
   Determining the top 10 capital cities in Europe and the Americas in terms of a calculated percentage using city_proper_pop and metroarea_pop in cities.

   **SQL Query -**

```
Select cities.name,
country_code,city_proper_pop,metroarea_pop,city_proper_pop/metroarea_pop*100 as
city_perc
from cities
inner join countries
on cities.country_code=countries.code
Where cities.name in(
  Select capital
  from countries
      Where (continent='Europe'
      OR continent like'%America%' )
)
  and metroarea_pop is not null
Order BY city_perc DESC
LIMIT 10
```

pgAdmin 4

pgAdmin File Object Tools Help

Browser | Dashboard | Properties | SQL | Statistics | Dependencies | Dependents | CountriesV1/postgres@PostgreSQL 14 *

- Servers (1)
  - PostgreSQL 14
    - Databases (2)
      - CountriesV1
        - Casts
        - Catalogs
        - Event Triggers
        - Extensions
        - Foreign Data Wrappers
        - Languages
        - Publications
        - Schemas (1)
          - public
            - Collations
            - Domains
            - FTS Configurations
            - FTS Dictionaries
            - FTS Parsers
            - FTS Templates
            - Foreign Tables
            - Functions
            - Materialized Views
            - Procedures
            - Sequences

No limit

CountriesV1/postgres@PostgreSQL 14

Query Editor | Query History

```
Select cities.name, country_code,city_proper_pop,metroarea_pop,city_proper_pop/metroarea_pop
from cities
inner join countries
on cities.country_code=countries.code
Where cities.name in(
  Select capital
  from countries
      Where (continent='Europe'
      OR continent like'%America%' )
)
  and metroarea_pop is not null
Order BY city_perc DESC
LIMIT 10
```

**Output-**

Data Output | Explain | Messages | Notifications

| | name [PK] character varying | country_code character varying | city_proper_pop real | metroarea_pop real | city_perc double precision |
|---|---|---|---|---|---|
| 1 | Lima | PER | 8.852e+06 | 1.075e+07 | 82.34418630599976 |
| 2 | Bogota | COL | 7.878783e+06 | 9.8e+06 | 80.3957462310791 |
| 3 | Moscow | RUS | 1.2197596e+07 | 1.617e+07 | 75.43349266052246 |
| 4 | Vienna | AUT | 1.863881e+06 | 2.6e+06 | 71.6877281665802 |
| 5 | Montevideo | URY | 1.305082e+06 | 1.947604e+06 | 67.00961589813232 |
| 6 | Caracas | VEN | 1.943901e+06 | 2.923959e+06 | 66.48181676864624 |
| 7 | Rome | ITA | 2.877215e+06 | 4.353775e+06 | 66.0855233669281 |
| 8 | Brasilia | BRA | 2.556149e+06 | 3.919864e+06 | 65.2101457118988 |
| 9 | London | GBR | 8.673713e+06 | 1.3879757e+07 | 62.491822242736816 |
| 10 | Budapest | HUN | 1.759407e+06 | 2.927944e+06 | 60.09018421173096 |

4. **Project Goal 4**
   Calculate the average fertility rate for each region in 2015.

   **SQL Query -**

   – Using SubQuery

```
select region,continent,avg(fertility_rate) as avg_fert_rate
from countries
inner join populations
on countries.code=populations.country_code and code in
(Select country_code
from populations
where year='2015')
Group by region,continent
order by avg_fert_rate
```

   –Using Join

```
/*SELECT region, continent, avg(fertility_rate) AS avg_fert_rate
  FROM countries AS c
    INNER JOIN populations AS p
      ON c.code = p.country_code
  WHERE year = 2015
GROUP BY region, continent
ORDER BY avg_fert_rate;*/
```

CountriesV1/postgres@PostgreSQL 14

Query Editor  Query History

```
select region,continent,avg(fertility_rate) as avg_fert_rate
from countries
inner join populations
on countries.code=populations.country_code and code in
(Select country_code
from populations
where year='2015')
Group by region,continent
order by avg_fert_rate
/*SELECT region, continent, avg(fertility_rate) AS avg_fert_rate
  FROM countries AS c
    INNER JOIN populations AS p
      ON c.code = p.country_code
  WHERE year = 2015
GROUP BY region, continent
ORDER BY avg_fert_rate;*/
```

**Output-**

Data Output | Explain | Messages | Notifications

| | region character varying | continent character varying | avg_fert_rate double precision |
|---|---|---|---|
| 1 | Southern Europe | Europe | 1.4261000037193299 |
| 2 | Eastern Europe | Europe | 1.490888900227017 |
| 3 | Baltic Countries | Europe | 1.603333314259847 |
| 4 | Eastern Asia | Asia | 1.6207143068313599 |
| 5 | Western Europe | Europe | 1.6325000077486038 |
| 6 | North America | North America | 1.7657500207424164 |
| 7 | British Islands | Europe | 1.875 |
| 8 | Nordic Countries | Europe | 1.8933333555857341 |
| 9 | Australia and New Zealand | Oceania | 1.9114999771118164 |

Australia and New Zealand region has the highest average fertility rate in 2015 whereas in Eastern Europe the average fertility rate is the lowest in 2015.