Exercises: PostgreSQL Subqueries and Joins

This document defines the **exercise assignments** for the PostgreSQL course @ Software University.

Submit your solutions in the SoftUni Judge Contest.

In preparation for the upcoming assignments, you will engage with a database you are already familiar with, albeit with some modifications adjusted for these specific tasks. Begin by establishing a new database called ***subqueries_joins_booking_db*** *and accessing its designated query tool. Retrieve the* ***06-Exercises-Subqueries-Joins-booking_db.sql*** *file from the course instance and input it into the query section of your database tool. Following the import, run the queries outlined in the file. The schema and tables present in the* ***subqueries_joins_booking_db*** *database will be leveraged for the ensuing tasks.*

1. Booked for Nights

Perform a **JOIN operation** between the **"apartments"** table and the **"bookings"** table to retrieve only matching rows. The resulting columns should be renamed as **"Apartment Address"** for the concatenated **"address"** and **"address_2"** columns and **"Nights"** for the **"booked_for"** column. Conclude by arranging the outcome in ascending order based on the **"apartment_id"** column.

Submit your query for this task in the Judge system.

Example

Apartment Address	Nights
761 Nikita Mews Suite 306	19
016 Jules Harbor Apt. 583	6
690 Ullrich Ridges Apt. 303	4
92830 Predovic Meadow Suite 058	5
…	…
637 Melody Stream Apt. 498	4
85561 Johnson Expressway Suite 047	14

2. First 10 Apartments Booked At

Create an SQL query that selects the **first 10 apartments** in the **"apartments"** table, along with their corresponding booking date from the **"bookings"** table (**if available**). If a column in the **LEFT table has no booking date**, it should **still be included in the result set**.

- select the **"name"** column from the **"apartments"** table and rename it as **"Name"**
- select the **"country"** column from the **"apartments"** table and rename it as **"Country"**
- select the **"booked_at"** column from the **"bookings"** table, **convert it to a date format**, and rename it as **"Booked at"**

*** Note, that the **LEFT JOIN** is the same as the **LEFT OUTER JOIN** so you can use them **interchangeably**.

Submit your query for this task in the Judge system.

Example

Name	Country	Booked at
Wolff LLC	Bedfordshire	2021-03-18
O`Keefe - Wunsch	Buckinghamshire	2021-07-18
…	…	…
McDermott, Little and Waelchi	Buckinghamshire	[null]
Welch Group	Bedfordshire	2021-11-06
…	…	…
Ondricka - Cartwright	Bedfordshire	[null]
…	…	…
Ebert Inc	Berkshire	2021-05-18
O`Hara - Towne	Berkshire	[null]

3. First 10 Customers with Bookings

Write a SQL query that selects the **first 10 bookings** in the **"bookings"** table, along with their corresponding customer's full name from the **"customers"** table. If any column in the **RIGHT table** does **not have booking information** available, it should **still be included in the result set**.

- select the **" booking_id"** column from the **"bookings"** table, renaming it as **"Booking ID"**
- select the **"starts_at"** column from the **"bookings"** table, **convert it to a date format**, and rename it as **"Start Date"**
- select the **"apartment_id"** column from the **"bookings"** table, renaming it as **"Apartment ID"**
- select the **concatenated "first_name"** and **"last_name"** columns from the **"customers"** table, renaming the resulting column as **"Customer Name"**
- order the outcome in **ascending** order based on the **"Customer Name"** column

*** Note, that the **RIGHT JOIN** and **RIGHT OUTER JOIN** are the same therefore you can use them **interchangeably**.

Submit your query for this task in the Judge system.

Example

Booking ID	Start Date	Apartment ID	Customer Name
[null]	[null]	[null]	Abbey Oberbrunner
104	2021-11-19	16	Abby Konopelski
40	2022-11-18	83	Abel Brakus
…	…	…	…
120	2022-11-19	66	Addison Douglas
[null]	[null]	[null]	Adella Nicolas
…	…	…	…
55	2021-11-19	90	Aniya Terry

4. Booking Information

Retrieve booking information from the **three tables**, where **all records** should be returned, **regardless of matches**.

- select the **"booking_id"** column from the **"bookings"** table and rename it as **"Booking ID"**
- select the **"name"** column from the **"apartments"** table and rename it as **"Apartment Owner"**
- select the **"apartment_id"** column from the **"apartments"** table and rename it as **"Apartment ID"**
- select the **concatenated "first_name"** and **"last_name"** columns from the **"customers"** table, renaming the resulting column as **"Customer Name"**
- order the results in **ascending** order based on the **"Booking ID"**, **"Apartment Owner"** and **"Customer Name"** columns

*** Note, that the **FULL JOIN** and **FULL OUTER JOIN** are the same therefore you can use them **interchangeably**.

Submit your query for this task in the Judge system.

Example

Booking ID	Apartment Owner	Apartment ID	Customer Name
1	[null]	[null]	Vladimir Frami
2	Langworth and Sons	16	Cathrine Beier
3	Kiehn, Kovacek and Marvin	144	Eliza Considine
…	…	…	…
168	[null]	[null]	Eleanora Kuvalis
169	[null]	[null]	Myrtie Predovic
[null]	Abernathy, Herman and Marks	40	
[null]	Beer Inc	66	
…	…	…	…
[null]	[null]	[null]	Abbey Oberbrunner
[null]	[null]	[null]	Adella Nicolas
…	…	…	…
[null]	[null]	[null]	Wilfredo Hahn

5. Multiplication of Information**

Write a SQL query to fetch the booking **"booking_id"** and customer **"first_name"** from the **"bookings"** and **"customers"** tables, respectively. Use a **CROSS JOIN** to generate a **Cartesian product** of the two tables. Finally, sort the result set in **ascending order** based on the **"Customer Name"**.

The CROSS JOIN operation combines all rows from two or more tables, resulting in a new dataset where each row from the first table is paired with every row from the second table (and so on, for any additional tables). Due to the potentially substantial output in our specific task, there is no necessity to submit this particular operation to the Judge system

Example

Booking ID	Customer Name
80	Abbey
99	Abbey
55	Abbey
…	…
159	Abby
…	…
148	Augustine
30	Bernadine
166	Bernadine
…	…
111	Bulah
31	Cathrine
2	Cathrine
…	…

6. Unassigned Apartments

Create a SQL query to retrieve the **"booking_id"** of bookings and the corresponding **"companion_full_name"** from the **"customers"** table, where the **"apartment_id"** has not been assigned yet.

*** Note that in a **SQL JOIN** operation, if the columns being joined have the same name in both tables, you can use the **USING** syntax in the **JOIN** predicate instead of the **ON** clause.

Submit your query for this task in the Judge system.

Example

booking_id	apartment_id	companion_full_name
1	[null]	Josefina Abshire
4	[null]	Herman Jones
6	[null]	Sandra Langosh
…	…	…
148		Tara Johnson
168	[null]	Vincent Abbott

7. Bookings Made by Lead

Write a SQL query that selects the **"apartment_id"**, **"booked_for"** nights, customer's **"first_name"**, and **"country"** from the **"bookings"** and **"customers"** tables, respectively, by performing an **INNER JOIN.** Filter the results only to include bookings made by customers with a **"job_type"** of **"Lead"**.

Submit your query for this task in the Judge system.

Example

apartment_id	booked_for	first_name	country
46	14	Sammie	Borders
[null]	5	Sammie	Borders

8. Hahn's Bookings

Create a SQL query that **COUNT** the **number of bookings** made by customers whose **"last_name"** is **'Hahn'**. The output should show only the **count of bookings** and no other columns.

Submit your query for this task in the Judge system.

Example

count
10

9. Total Sum of Nights

Write a SQL query that retrieves the **"name"** of each apartment in the **"apartments"** table along with the **total sum of nights "booked_for"** for each apartment. **Group** the result by the apartment **"name"** and sort the result in **ascending** order based on the **"name"**.

Submit your query for this task in the Judge system.

Example

name	sum
Abernathy, Herman and Marks	43
Ankunding - Gibson	14
Bartoletti, O`Connell and Buckridge	11
…	…
Cartwright and Sons	6
…	…
D`Amore Inc	7
…	…
Wolf - Collier	15

Yundt - Bernier	45

10. Popular Vacation Destination

Create a SQL query to determine which **"country"** is a popular vacation destination during the summer season by:

- counting the number of bookings **"booking_id"** made for each **"country"** between **'2021-05-18 07:52:09.904+03'** and **'2021-09-17 19:48:02.147+03'** (exclusive)
- grouping the results by the **"country"**
- ordering the results in descending order based on the **"booking_count"**.

Submit your query for this task in the Judge system.

Example

country	booking_count
Berkshire	8
Cambridgeshire	8
Bedfordshire	7
Borders	7
Avon	6
Buckinghamshire	3

*To accomplish the upcoming exercises, you will be dealing with a database that you are already familiar with, but it has undergone some modifications specific to the upcoming tasks. Your first step is to create a fresh database named **subqueries_joins_geography_db** and access its query tool. Retrieve the **06-Exercises-Subqueries-Joins-geography_db.sql** file from the course instance and import it into the query tab of your newly created database. Once imported, proceed to execute the queries provided within the file. The schema and tables from the **subqueries_joins_geography_db** database will serve as the foundation for the subsequent tasks.*

11. Bulgaria's Peaks Higher than 2835 Meters

Retrieve the **"country_code"**, **"mountain_range"**, **"peak_name"** and **"elevation"** from the **"mountains"**, **"peaks"**, and **"mountains_countries"** tables using a SQL query. The query should only include rows where the peak **"elevation"** is greater than **2835 meters** and the **"country_code"** is **'BG'**. The results should be sorted in **descending** order based on peak **"elevation"**.

Submit your query for this task in the Judge system.

Example

country_code	mountain_range	peak_name	elevation
BG	Rila	Musala	2925
BG	Pirin	Vihren	2914
BG	Pirin	Kutelo	2908
BG	Rila	Malka Musala	2902

BG	Pirin	Banski Suhodol	2884
BG	Pirin	Golyam Polezhan	2851

12. Count Mountain Ranges

Create a SQL query that returns the number of **unique mountain ranges** for the countries with the **country codes** **'US'**, **'RU'**, and **'BG'**. The results should be grouped by **"country_code"** and ordered in **descending** order based on the **"mountain_range_count"**.

Submit your query for this task in the Judge system.

Example

country_code	mountain_range_count
BG	4
RU	1
US	1

13. Rivers in Africa

Write a SQL query that selects the **"country_name"** and **"river_name"** (if any) from the **"countries"**, **"countries_rivers"** and **"rivers"** tables, respectively, for the **first five countries in Africa**. If a country has no river, the **"river_name"** should be NULL. The result should be ordered in **ascending** order based on the **"country_name"**.

Submit your query for this task in the Judge system.

Example

country_name	river_name
Algeria	Niger
Angola	Congo
Benin	Niger
Botswana	[null]
Burkina Faso	Niger

14. Minimum Average Area Across Continents

Compute the **minimum average area** among all the continents, where the average area is **calculated as the average area of all the countries within each continent**.

Submit your query for this task in the Judge system.

Example

min_average_area
315685.370370370370

15. Countries Without Any Mountains

Create an SQL query to retrieve the **number of countries** that do **not have any mountains**.

Submit your query for this task in the Judge system.

Example

countries_without_mountains
231

16. Monasteries by Country **

To begin, create a table called **"monasteries"** with three columns:

- **"id"** - column should be a **PRIMARY KEY** and **automatically incremented**
- **"monastery_name"** - column should have a **maximum length of 255 characters**
- **"country_code"** - column should be **exactly two characters** long.

Then, insert the provided data into this table:

```
('Rila Monastery 'St. Ivan of Rila', 'BG'),
('Bachkovo Monastery 'Virgin Mary', 'BG'),
('Troyan Monastery 'Holy Mother''s Assumption', 'BG'),
('Kopan Monastery', 'NP'),
('Thrangu Tashi Yangtse Monastery', 'NP'),
('Shechen Tennyi Dargyeling Monastery', 'NP'),
('Benchen Monastery', 'NP'),
('Southern Shaolin Monastery', 'CN'),
('Dabei Monastery', 'CN'),
('Wa Sau Toi', 'CN'),
('Lhunshigyia Monastery', 'CN'),
('Rakya Monastery', 'CN'),
('Monasteries of Meteora', 'GR'),
('The Holy Monastery of Stavronikita', 'GR'),
('Taung Kalat Monastery', 'MM'),
('Pa-Auk Forest Monastery', 'MM'),
('Taktsang Palphug Monastery', 'BT'),
('Sümela Monastery', 'TR');
```

Next, modify the **"countries"** table by adding a **BOOLEAN** column called **"three_rivers"**. This column should have a **default value of false**, indicating that the country does not have three rivers. To update the **"three_rivers"** column for countries that have **more than three rivers running** through them, use a subquery **to count the number of rivers** in each country and compare the result to the value of **3**.

Finally, write a SQL query that selects the **"monastery_name"** and their respective **"country_name"** from the **"monasteries"** table, ordered **alphabetically by "monastery_name"**. The query should exclusively retrieve records for countries that have more than three rivers flowing within their borders.

Submit your query for this task in the Judge system.

Example

Monastery	Country
Bachkovo Monastery "Virgin Mary"	Bulgaria
Benchen Monastery	Nepal
Kopan Monastery	Nepal

Monasteries of Meteora	Greece
Pa-Auk Forest Monastery	Myanmar
…	…
Thrangu Tashi Yangtse Monastery	Nepal
Troyan Monastery “Holy Mother's Assumption”	Bulgaria

17. Monasteries by Continents and Countries**

Create a SQL query that **updates** the **"countries"** table by replacing **'Myanmar'** with **'Burma'**. Additionally, add a **new row** to the **"monasteries"** table with **'Hanga Abbey'** as the name and **'Tanzania'** as the **country code**. Another row should also be inserted into the **"monasteries"** table with **'Myin-Tin-Daik'** as the name and **'Myanmar'** as the **country code**.

To retrieve the number of monasteries in each country along with their corresponding **"continent_name"**, construct a query that joins the **"continents"**, **"countries"**, and **"monasteries"** tables. Choose exclusively those countries without the **"three_rivers"** column. Group the results by **"country_name"** and **"continent_name"**, and sort them by the **number of monasteries** in **descending** order. In the case of a tie, sort the results by **country name** in **ascending** order.

Submit your query for this task in the Judge system.

Example

Continent Name	Country Name	Monasteries Count
Asia	Nepal	4
Europe	Bulgaria	3
Asia	Burma	2
…	…	…
Asia	Bhutan	1
Africa	Tanzania	1
…	…	…
Oceania	American Samoa	0
…	…	…
Africa	Zambia	0
Africa	Zimbabwe	0

18. Retrieving Information about Indexes

Write a SQL query that selects the fields **"tablename"**, **"indexname"**, and **"indexdef"** from the **"pg_indexes"** table, with the condition that only indexes from the **"public"** schema are retrieved. Sort the results in ascending order based on the **"tablename"** field. If any of the values are equal, then sort the results by **"indexname"** in ascending order.

Submit your query for this task in the Judge system.

Example

tablename	indexname	indexdef
continents	continents_continent_code_key	CREATE UNIQUE INDEX continents_continent_code_key ON public.continents USING btree (continent_code)
…	…	…
countries	countries_country_code_key	CREATE UNIQUE INDEX countries_country_code_key ON public.countries USING btree (country_code)
…	…	…
currencies	currencies_currency_code_key	CREATE UNIQUE INDEX currencies_currency_code_key ON public.currencies USING btree (currency_code)
…	…	…
monasteries	monasteries_pkey	CREATE UNIQUE INDEX monasteries_pkey ON public.monasteries USING btree (id)
…	…	…
peaks	peaks_pkey	CREATE UNIQUE INDEX peaks_pkey ON public.peaks USING btree (id)
rivers	rivers_pkey	CREATE UNIQUE INDEX rivers_pkey ON public.rivers USING btree (id)

19. * Continents and Currencies

Write a SQL query to create a view called **"continent_currency_usage"** that shows the **"continent_code"**, **"currency_code"**, and a number of countries using the currency where **more than one country** on a continent uses the same currency. The column displaying the number of countries using the currency should be renamed as **"currency_usage"**. The data should be ordered by the **"currency_usage"** column in **descending order**.

*** Hint, to solve this problem, you should use a **subquery** with a **SELECT** statement that includes the **DENSE RANK()** function to assign a **rank** to each row within a **partition** of the result set. You will need to use the **GROUP BY** clause to group the **results by continent and currency**, and the **HAVING** clause to filter the results to only include those where multiple countries are using **the same currency**.

Submit your query for this task in the Judge system.

Example

continent_code	currency_code	currency_usage
EU	EUR	26
AF	XOF	8
NA	XCD	8
OC	USD	8
AS	AUD	2
AS	ILS	2

20. * The Highest Peak in Each Country

Create a SQL query to retrieve the **"peak_name"** and **"elevation"** of the **highest peak for each country**, along with the **"mountain_range"** it belongs to. In cases where there are **no peaks available** in some countries, display **"0"** as the **"elevation"**, **"(no highest peak)"** as the **"peak_name"**, and **"(no mountain)"** as the **"mountain_range"**. If there are multiple peaks in some countries with the same elevation, include all of them in the result. Sort the **"country_name"** in **alphabetical order**, and if there are countries with the same name, sort them by highest peak **"elevation"** in **descending order**.

*** Hint, one approach to tackle this problem is by utilizing a **Common Table Expression (CTE)** to create a temporary result set that can be referenced within another SQL statement, including **SELECT**. Name the CTE as **"row_number"**. The **ROW_NUMBER() function** will **assign a ranking to the peaks in each country** based on their elevation, with the highest peak receiving a **rank of 1**. Use the **COALESCE() function** to replace any null values in the **"peak_name"** and **"elevation"** columns, for countries where no peaks are available. You can use a **CASE** statement to display the **"mountain_range"** when the **"peak_name"** is not null, and **"(no mountain)"** when no peaks are available. Finally, filter the result set by the **"row_number"** column to only show the highest peak for each country.

Submit your query for this task in the Judge system.

Example

Country	Highest Peak Name	Highest Peak Elevation	Mountain
Afghanistan	(no highest peak)	0	(no mountain)
Aland	(no highest peak)	0	(no mountain)
…	…	…	…
Argentina	Aconcagua	6962	Andes
…	…	…	…
Brunei	(no highest peak)	0	(no mountain)
Bulgaria	Musala	2925	Rila
…	…	…	…
Canada	Mount Logan	5959	Saint Elias Mountains
…	…	…	…

United States	Mount McKinley	6194	Alaska Range
Uruguay	(no highest peak)	0	(no mountain)
…	…	…	…
Zambia	(no highest peak)	0	(no mountain)
Zimbabwe	(no highest peak)	0	(no mountain)

** This task is not required to be submitted to the Judge system and will not be considered in the final result.