Exercises: PostgreSQL Built-in Functions

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

Submit your solutions in the SoftUni Judge Contest.

Important: Throughout the course, you will receive different databases that may have similar names and structures but contain different data specific to each exercise. To ensure proper execution and avoid conflicts, it is crucial to create a new database for each exercise and import the provided file with the corresponding records. By following this approach, you can accurately work on each exercise and avoid any interference or data overlap between different exercises.

*Begin by **creating a database** called **geography_db** and then **launch its query tool**. After that, **download** the **03-Exercises-Built-in-Functions-geography_db.sql** file from the course instance, **import it into the query tab of your database**, and **execute the queries provided in the file**. Once you've executed the queries, take some time to familiarize yourself with the **geography_db** database and get to know its schema and tables.*

1. River Info

Create a **view** named **"view_river_info"** that concatenates the **"river_name"**, **"outflow"** and **"length"** columns from the **"rivers"** table in the following format:

```
'The river', ' ', river_name, ' ', 'flows into the', ' ', outflow, ' ', 'and is', ' ',
                        "length", ' ', 'kilometers long.'
```

The resulting column should be named **"River Information"**, and the rows should be ordered by the **"river_name"** field in **ascending alphabetical order**.

Submit your query for this task in the Judge system.

Example

River Information
The river Amazon flows into the Atlantic Ocean and is 6400 kilometers long.
The river Amur flows into the Sea of Okhotsk and is 4444 kilometers long.
The river Brahmaputra flows into the Ganges and is 2948 kilometers long.
The river Congo flows into the Atlantic Ocean and is 4700 kilometers long.
The river Danube flows into the Black Sea and is 2888 kilometers long.
…
The river Yenisei flows into the Kara Sea and is 5539 kilometers long.
The river Yukon flows into the Bering Sea and is 3185 kilometers long.

2. Concatenate Geography Data

Create a **view** named **"view_continents_countries_currencies_details"**. To do so, follow these steps:

- from the **"continents"** table, **combine** the **"continent_name"** and **"continent_code"** with a **colon** (:) **separator**, and name the resulting column **"Continent Details"**

- from the **"countries"** table, select the **"country_name"**, **"capital"**, and **"area_in_sq_km"** fields. You can add a hyphen (-) **between** the fields and **append 'km2' to the end** of **"area_in_sq_km"** to avoid confusion with other data. Name the resulting columns **"Country Information"**
- in the last column, **combine** the **"description"** and **"currency_code"** fields of the **"currencies"** table, using the following format: **description (currency_code)**. Name the resulting column **"Currencies"**
- sort the result by the **"Country Information"** and **"Currencies"** fields in **ascending alphabetical order**

Submit your query for this task in the Judge system.

Example

Continent Details	Country Information	Currencies
Asia: AS	Afghanistan - Kabul - 647500 - km2	Afghanistan Afghani (AFN)
Europe: EU	Aland - Mariehamn - 1580 - km2	Euro Member Countries (EUR)
Europe: EU	Albania - Tirana - 28748 - km2	Albania Lek (ALL)
…	…	…
North America: NA	Bahamas - Nassau - 13940 - km2	Bahamas Dollar (BSD)
Asia: AS	Bahrain - Manama - 665 - km2	Bahrain Dinar (BHD)
…	…	…
Asia: AS	Cambodia - Phnom Penh - 181040 - km2	Cambodia Riel (KHR)
Africa: AF	Cameroon - Yaoundé - 475440 - km2	Communauté Financière Africaine (BEAC) CFA Franc BEAC (XAF)
…	…	…
Africa: AF	Democratic Republic of the Congo - Kinshasa - 2345410 - km2	Congo/Kinshasa Franc (CDF)
…	…	…
Africa: AF	Zambia - Lusaka - 752614 - km2	Zambia Kwacha (ZMW)
Africa: AF	Zimbabwe - Harare - 390580 - km2	Zimbabwe Dollar (ZWD)

3. Capital Code

Add a **new column** to the **"countries"** table named **"capital_code"**, by generating the code by using the **SUBSTRING()** function to extract the **first 2 letters** from the **"capital"** field.
Choose whichever SQL syntax you prefer to use for the query.

Submit your query for this task in the Judge system.

Example

Before update

id	…	capital	…
1	…	Andorra la Vella	…

2	…	Abu Dhabi	…
3	…	Kabul	…
4	…	St. John`s	…
…	…	…	…
162	…	Zambia Kwacha	…
163	…	Zimbabwe Dollar	…

After update

id	…	capital	…	capital_code
1	…	Andorra la Vella	…	An
2	…	Abu Dhabi	…	Ab
3	…	Kabul	…	Ka
4	…	St. John`s	…	St
…	…	…	…	…
249	…	Zambia	…	Lu
250	…	Zimbabwe	…	Ha

4. (Descr)iption

Develop an SQL query that **removes** a portion of the **"description"** column from the **"currencies"** table. The query should extract the string **starting from the 5th character** and return the rest of the string.

Submit your query for this task in the Judge system.

Example

substring
ed Arab Emirates Dirham
anistan Afghani
nia Lek
erlands Antilles Guilder
…
ia Kwacha
abwe Dollar

5. Substring River Length

Compose an SQL query to fetch the **"river_length"** from the **"River Information"** column within the **"view_river_info"** view. Ensure that only the numerical value is selected from the string, with a **maximum of four digits, ranging from 0 to 9**.

*** Note that you can use the following regex expression **'([0-9]{1,4})'** to **find the number in the sentence**.

Submit your query for this task in the Judge system.

Example

river_length
6400
4444
2948
4700
…
5539
3185

6. Replace A

To write a SQL query that replaces letters in the **"mountain_range"** column of the **"mountains"** table, please follow these steps:

- replace all occurrences of **"a"** with **"@"**. Name the resulting column **"replace_a"**
- replace all occurrences of **"A"** with **"$"**. Name the resulting column **"replace_A"**

*** Note, the PostgreSQL **REPLACE()** function is **case-sensitive**. This means that if you use the function to replace a specific string or character, it will only replace those occurrences that match the case of the original string.

Submit your query for this task in the Judge system.

Example

replace_a	replace_A
Al@sk@ R@nge	$laska Range
Alborz	$lborz
Andes	$ndes
B@lk@n Mount@ins	Balkan Mountains
C@uc@sus	Caucasus
…	…
J@y@wij@y@ Mount@ins	Jayawijaya Mountains
…	…
Str@ndz@	Strandza
Monte Ros@	Monte Rosa

7. Translate

You may notice that the **"capital"** names in the **"countries"** table include letters that are not found in the English alphabet. To address this, you can employ the **TRANSLATE()** function to convert the non-English characters **'áãåçéíñóú'** to their corresponding English letters. Name the resulting column **"translated_name"**.

Submit your query for this task in the Judge system.

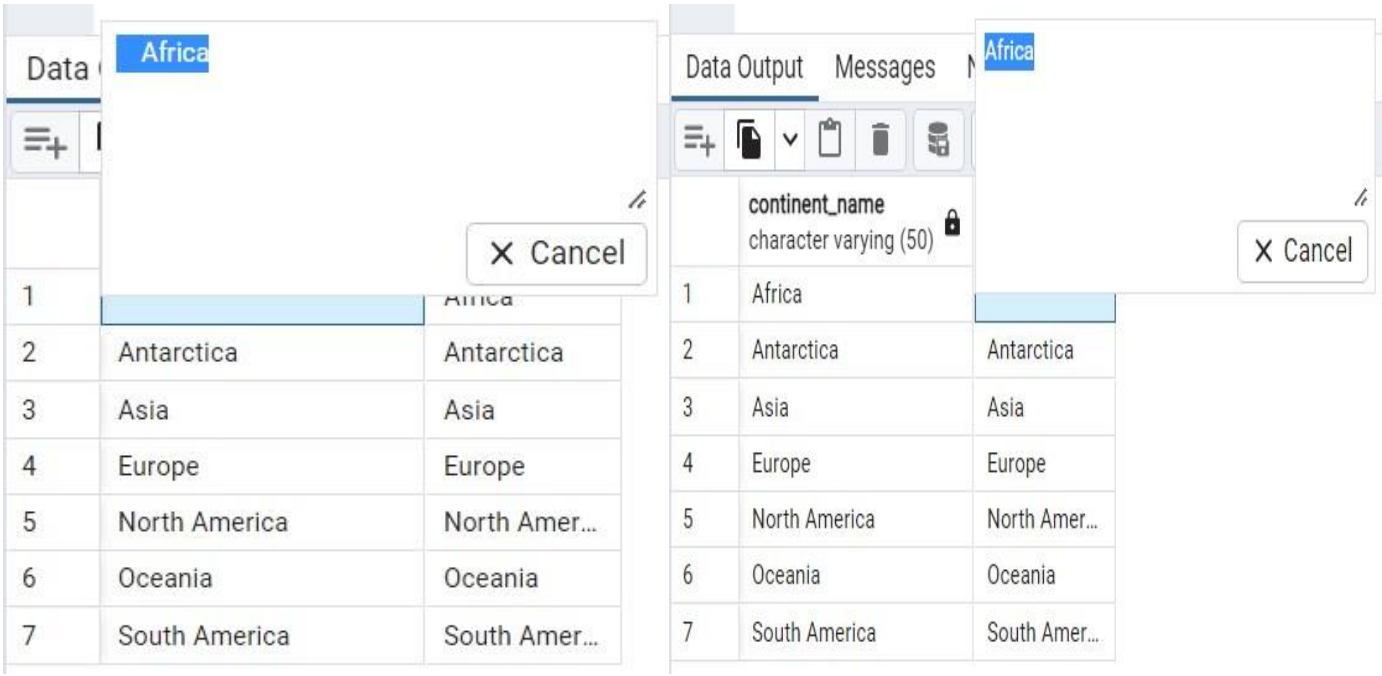
Example

capital	translated_name
Baghdad	Baghdad
Andorra la Vella	Andorra la Vella
Abu Dhabi	Abu Dhabi
…	…
Yaoundé	Yaounde
Beijing	Beijing
Bogotá	Bogota
San José	San Jose
…	…
El Aaiún	El Aaiun
…	…
Hagåtña	Hagatna
…	…
São Tomé	Sao Tome
…	…
Lomé	Lome
…	…
Lusaka	Lusaka
Harare	Harare

8. LEADING

If you open the records in the **"continents"** table, you will find that there are additional spaces added to the front of some of the **"continent_name"** values. Use the **TRIM()** function with the appropriate **flag** to remove them.

*** Please be aware that the accurate method to confirm the correctness of your query is by double-clicking on the field to select the value. The additional blue background color before or after the continent's name will signify any empty spaces, as illustrated in the screenshots below.



Submit your query for this task in the Judge system.

Example

continent_name	trim
'····Africa'	'Africa'
…	…
'····Asia'	'Asia'
…	…
'····North America'	'North America'
…	…
'····South America'	'South America'
…	…

9. TRAILING

The **TRIM()** function also has **another flag**, which can help you remove trailing spaces from the **"continent_name"** values.

Submit your query for this task in the Judge system.

Example

continent_name	trim
…	…
'Antarctica····'	'Antarctica'
…	…
'Europe····'	'Europe'

…	…
'Oceania····'	'Oceania'
…	…

10. LTRIM & RTRIM

The **TRIM()** function has a **shortened version** that can remove both **spaces and characters**. Write an SQL query to remove the **"m"** character as follows:

- remove the **'M'** character from the **left side** of the **"peak_name"** column within the **"peaks"** table, and assign the name **"Left Trim"** to the resulting new column
- remove the **'m'** character from the **right side** of the **"peak_name"** column within the **"peaks"** table, and assign the name **"Right Trim"** to the resulting new column

*** Note that the PostgreSQL **TRIM()** function and its **equivalent functions** are **case-sensitive**.

Submit your query for this task in the Judge system.

Example

Left Trim	Right Trim
Aconcagua	Aconcagua
Botev	Botev
Carstensz Pyramid	Carstensz Pyramid
…	…
onte Pissis	Monte Pissis
ount Giluwe	Mount Giluwe
ount Kenya	Mount Kenya
…	…
akalu	Makalu
Kom	Ko

11. Character Length and Bits

Combine the **"mountain_range"** column from the **"mountains"** table and the **"peak_name"** column from the **"peaks"** table into a single field called **"Mountain Information"**. Find the **number of characters** in the newly created text field and name the new column **"Characters Length"**. Additionally, express the **length in bits** and name the column **"Bits of a String"**.

Submit your query for this task in the Judge system.

Example

Mountain Information	Characters Length	Bits of a String
Andes Aconcagua	15	120
Balkan Mountains Botev	22	176

The Sudirman Range Carstensz Pyramid	36	296
Alborz Damavand	15	120
…	…	…
Balkan Mountains Vezhen	23	184
Balkan Mountains Kom	20	160

12. Length of a Number

Measure the length of the **"population"** numbers in the **"countries"** table. In this case, use the **CAST()** function to **convert the number into a string** and then use the **LENGTH()** function.

Submit your query for this task in the Judge system.

Example

population	length
29671605	8
84000	5
4975593	7
29121286	8
…	…
13460305	8
11651858	8

13. Positive and Negative LEFT

Write a SQL query to select the **FIRST 4 characters** from the **"peak_name"** column and name the new column **"Positive Left"**. Also, select **all characters except the LAST 4** from the **"peak_name"** column and name the new column **"Negative Left"**.
Submit your query for this task in the Judge system.

Example

peak_name	Positive Left	Negative Left
Aconcagua	Acon	Aconc
Botev	Bote	B
Carstensz Pyramid	Cars	Carstensz Pyr
Damavand	Dama	Dama
Dykh-Tau	Dykh	Dykh
Elbrus	Elbr	El
…	…	…
Vezhen	Vezh	Ve

Kom	Kom	

14. Positive and Negative RIGHT

Write a SQL query to select the **LAST 4 characters** from the **"peak_name"** column and name the new column **"Positive Right"**. Also, select **all characters except the FIRST 4** from the **"peak_name"** column and name the new column **"Negative Right"**.

Submit your query for this task in the Judge system.

Example

peak_name	Positive Right	Negative Right
Aconcagua	agua	cagua
Botev	otev	v
Carstensz Pyramid	amid	tensz Pyramid
Damavand	vand	vand
Dykh-Tau	-Tau	-Tau
Elbrus	brus	us
…	…	…
Vezhen	zhen	en
Kom	Kom	

15. Update iso_code

As some of the values in the **"iso_code"** column of the **"countries"** table are null, update them by taking the **first three characters** from the **"country_name"** column and converting them to **uppercase**.

Submit your query for this task in the Judge system.

Example

Before update

id	…	iso_code
1	…	[null]
2	…	ARE
3	…	[null]
4	…	ATG
5	…	AIA
6	…	ALB
7	…	[null]
…	…	…
249	…	ZMB

250	…	ZWE

After update

id	…	iso_code
1	…	AND
2	…	ARE
3	…	AFG
4	…	ATG
5	…	AIA
6	…	ALB
7	…	ARM
…	…	…
249	…	ZMB
250	…	ZWE

16. REVERSE country_code

Create a SQL query to update the values in the **"country_code"** column of the **"countries"** table. The update should convert the values to **lowercase** and **reverse** the string.

Submit your query for this task in the Judge system.

Example

Before update

id	country_code
1	AD
2	AE
3	AF
4	AG
…	…
16	AZ
17	BA
…	…
249	ZM
250	ZW

After update

id	country_code
1	da

2	ea
3	fa
4	ga
…	…
16	za
17	ab
…	…
249	mz
250	wz

17. Elevation --->> Peak Name

Write an SQL query to select the **"elevation"** and **"peak_name"** columns from the **"peaks"** table where the **"elevation"** is **greater than or equal to 4884**. Concatenate them with a **single space**, use the **REPEAT()** function to create an arrow between them **"--->>"**, and name the new column **"Elevation --->> Peak Name"**.

Submit your query for this task in the Judge system.

Example

Elevation --->> Peak Name
6962 --->> Aconcagua
4884 --->> Carstensz Pyramid
5610 --->> Damavand
5205 --->> Dykh-Tau
5642 --->> Elbrus
…
8462 --->> Makalu
8201 --->> Cho Oyu

*To complete the upcoming exercises, it is necessary to **create a new database** named* **booking_db** *and open its query tool. **Download** the **03-Exercises-Built-in-Functions-booking_db.sql** file from the course instance and **import it into the query tab** of your database. After importing, **execute the queries** in the file. Use the schema and tables available in the* **booking_db** *database for the tasks that follow.*

18. Arithmetical Operators

Let's apply our understanding of mathematical operators in SQL. To begin, **create a fresh table** named **"bookings_calculation"**. You can achieve this by selecting the **"booked_for"** values from the **"bookings"** table where the **"apartment_id" equals 93**. The **"booked_for"** column signifies the number of nights the apartment is booked.

Next, alter the table by **adding two new columns**:

- **"multiplication"** column with a **NUMERIC** data type

- **"modulo"** column, also of **NUMERIC** data type

For the final step, proceed to **calculate the earnings earned by the owner for each night**, following these instructions:

- populate the **"multiplication"** column by **multiplying** the **"booked_for"** values by **50**
- fill the **"modulo"** column with values representing the remainder when **"booked_for"** is **divided** by **50**

Submit your query for this task in the Judge system.

Example

booked_for	multiplication	modulo
9	450	9
3	150	3
9	450	9
10	500	10
1	50	1
9	450	9

19. ROUND vs TRUNC

Create a SQL query that retrieves the **"latitude"** column from the **"apartments"** table. Apply the **ROUND()** function to it with a **precision of 2** decimal places, and then apply the **TRUNC()** function with the same precision. Finally, compare and measure the **differences in the output produced by the two functions**.

Submit your query for this task in the Judge system.

Example

latitude	round	trunc
38.1941	38.19	38.19
5.9271	5.93	5.92
-10.6776	-10.68	-10.67
82.1055	82.11	82.10
-32.5269	-32.53	-32.52
…	…	…
41.7393	41.74	41.73
-67.5961	-67.60	-67.59

20. Absolute Value

Write an SQL query to select the **"longitude"** column from the **"apartments"** table and apply the **ABS()** function to it to find its absolute value.

Submit your query for this task in the Judge system.

Example

longitude	abs
58.3150	58.3150
-95.1855	95.1855
-105.1542	105.1542
-142.8697	142.8697
45.3129	45.3129
…	…
-57.1891	57.1891
-96.6056	96.6056

21. Billing Day**

To generate payment documents for reservations made for apartments, follow these steps:

- firstly, **add a new column** to the **"bookings"** table called **"billing_day"** with the data type of **"TIMESTAMPTZ"** and **set its default value** to **"CURRENT_TIMESTAMP"**
- after that, create a SQL query that retrieves the **"billing_day"** column from the **"bookings"** table and **formats** it as **"DD 'Day' MM 'Month' YYYY 'Year' HH24:MI:SS"**, naming the resulting column **"Billing Day"**.

! The example result shown below is generated using the **CURRENT_TIMESTAMP** and it is purely for illustrative purposes.

Therefore, there is no need to submit this task to the Judge system.

Example

…	billing_day
	2023-07-17 23:15:59.3749+03
	2023-07-17 23:15:59.3749+03
	…

Billing Day
17 Day 07 Month 2023 Year 23:15:59
17 Day 07 Month 2023 Year 23:15:59
…

22. EXTRACT Booked At

Create a SQL query to retrieve the **YEAR**, **MONTH**, **DAY**, **HOUR**, **MINUTE**, and **SECOND** values from the **"booked_at"** column. Use the **CEILING()** function to **round up** the **SECOND** value to the nearest whole number.

*** Note that the **"booked_at"** column is stored as **"TIMESTAMPTZ"** (timestamp with time zone) data type. When extracting hours from this column, please be aware that the extraction considers your account's time zone

information, which may result in different hour values based on the time zone. To ensure consistent results for this task, utilize the **"AT TIME ZONE"** function to convert the timestamp to the **UTC** time zone before extracting the hour. This approach will help ensure uniformity in the results.

Submit your query for this task in the Judge system.

Example

YEAR	MONTH	DAY	HOUR	MINUTE	SECOND
2021	10	16	8	8	60
2022	10	17	12	31	6
2023	10	15	0	16	45
2021	1	10	3	25	38
2020	5	17	20	1	3
…	…	…	…	…	…
2021	1	18	11	58	37
2021	4	18	12	50	35

23. Early Birds**

Compose a SQL query to determine the **"user_id"** of customers who prefer booking **10 months in advance**. Achieve this by computing the time **difference between** the **"starts_at"** and **"booked_at"** columns in the **"bookings"** table, and storing the resultant values in a new column named **"Early Birds"**. Afterward, apply a filter to select only the rows where the **"Early Birds"** value is **greater than or equal to 10 months**, and retrieve the corresponding **"user_id"**.

*** As a suggestion, it is worth noting that the **WHERE** clause allows you to use the **INTERVAL '10 months'** to indicate a time **period of 10 months** when used in conjunction with the **AGE()** function for computing the time difference between two dates.

There is no need to submit the solution to the Judge system for this problem.

Example

user_id	Early Birds
63	10 mons 8 days 19:30:20.531
53	10 mons 1 day 00:35:07.551
18	10 mons 12:26:33.623
22	10 mons 03:34:53.748
…	…
7	10 mons 1 day 03:56:17.546
18	10 mons 12:26:33.623

24. Match or Not

Retrieve the **"companion_full_name"** and **"email"** columns from the **"users"** table where the following conditions are met:

- the **"companion_full_name"** column should **contain the substring '%aNd%'** in a **case-insensitive manner**
- the **"email"** column should **NOT contain** the substring **'%@gmail'** in a **case-sensitive manner**

Submit your query for this task in the Judge system.

Example

companion_full_name	email
Ms. Brandy Rice	Rosalind_Hudson@hotmail.com
Terrell Blanda IV	Macie87@hotmail.com
Andrew Gottlieb PhD	Alfred.Purdy2@gmail.com
Sandra Langosh	Thalia_Wehner78@gmail.com

25. * COUNT by Initial

To generate a report displaying the **user count grouped by their initials**, you can utilize an SQL query. This query involves selecting the **initial two characters** from the **"first_name"** column of the **"users"** table and storing them in a newly created column named **"initials"**. Afterward, you can use the **GROUP BY** clause to group the users based on their initials. Then, employ the **COUNT()** function to retrieve the number of users in each group, and name the resulting column **"user_count"**. Finally, you can arrange the resulting data in **descending order** according to the **"user_count"** column. In cases where multiple groups have the same count, you can further sort them **alphabetically** based on their **"initials"**.

Submit your query for this task in the Judge system.

Example

initials	user_count
Ar	7
Ma	7
Ja	5
El	4
…	…
Ab	3
…	…
Vl	1

Wi	1

26. * SUM

To calculate the **total value of bookings for the apartment**, you can use an SQL query that retrieves the **"booked_for"** column from the **"bookings**" table and applies the **SUM()** function to it. Then, you can add a filter to select only the rows where the **"apartment_id"** is equal to **90**.

Submit your query for this task in the Judge system.

Example

total_value
50

27. * Average Value

Create an SQL query that utilizes the **AVG()** function to calculate the average value of the **"multiplication"** column in the **"bookings_calculation"** table.

Submit your query for this task in the Judge system.

Example

avarage_value
341.6666666666666667

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