Built-in Functions

Functions and Wildcards in PostgreSQL

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Questions

Software University

sli.do

#python-db

Table of Contents

Functions in PostgreSQL

SQL Functions

- **String** Functions – manipulating text
  - e.g., concatenate column values
- **Math** Functions – calculations and working with aggregate data
  - e.g., perform geometric and currency operations
- **Date/Time** Functions
  - e.g., compute the length of a time span
- A great variety of Other Functions
  - have a look at the **official documentation**

String Functions

Concatenation

- Concatenating arguments

```
SELECT CONCAT(first_name,' ',last_name) AS full_name
FROM authors;
```

- Concatenating with a specific separator
  - Skips any **NULL** values after the separator argument

```
SELECT CONCAT_WS(', ', last_name, born)
AS summary
FROM authors;
```

Extracting Substring

- Extract a part of a string

```
SUBSTRING(string, start_position, length)
```

```
SELECT SUBSTRING('SoftUni', 5, 3)
```

```
Uni
```

- Example: get a short **summary** of a book

```
SELECT title,
       year_of_release,
       SUBSTRING(description, 1, 50) || '...' AS
summary
FROM books;
```

Extracting Substring

- Another way of writing the same function

```
SUBSTRING(string FROM start_position FOR length)
```

```
SELECT SUBSTRING('SoftUni' FROM 5 FOR 3)
```

```
Uni
```

- Example: get full names in the format "A. Christie"

```
SELECT
      CONCAT(SUBSTRING(first_name FROM 1 FOR 1),
                       '.', ' ', last_name)
AS full_name
FROM authors;
```

Problem: Find Book Titles

- Write a query to find all book titles that start with "**The**"
  - Query table **books** from the **book_library** database

	title character varying (100)
1	The Mysterious Affair at Styles
2	The Big Four
3	The Murder at the Vicarage
4	The Mystery of the Blue Train
5	The Ring
6	The Alchemist
7	The Fifth Mountain
8	The Zahir

Solution: Find Book Titles

```
SELECT title FROM books
WHERE SUBSTRING(title, 1, 3) = 'The'
ORDER BY id;
```

	title character varying (100)
1	The Mysterious Affair at Styles
2	The Big Four
3	The Murder at the Vicarage
4	The Mystery of the Blue Train
5	The Ring
6	The Alchemist
7	The Fifth Mountain
8	The Zahir

Get Chars

- Get characters from the beginning or the end of a string

```
LEFT(string, count)
```

```
RIGHT(string, count)
```

- Example: short titles (first 10 letters)

```
SELECT id, year_of_release,
       LEFT(title, 10) AS short_title
FROM books;
```

Replacing Strings

- Replacing all occurrences of a string with another
  - Performs a case-sensitive matching

```
REPLACE(string, pattern, replacement)
```

```
REPLACE('SoftUni', 'Soft', 'Hard')
```

```
HardUni
```

- Example: **censor** the word 'Murder' from book titles

```
SELECT REPLACE(title, 'Murder', '*******')
AS title_censored
FROM books;
```

Problem: Replace Titles

- Write a query to find all book titles that start with "**The**" and replace the substring with "*******"
  - Query table **`books`** from the **`book_library`** database

	Title text
1	*** Mysterious Affair at Styles
2	*** Big Four
3	*** Murder at the Vicarage
4	*** Mystery of the Blue Train
5	*** Ring
6	*** Alchemist
7	*** Fifth Mountain
8	*** Zahir

Solution: Replace Titles

```
SELECT REPLACE(title, 'The','***')
     AS "Title"
FROM books
WHERE SUBSTRING(title, 1, 3) = 'The'
ORDER BY id;
```

	Title text
1	*** Mysterious Affair at Styles
2	*** Big Four
3	*** Murder at the Vicarage
4	*** Mystery of the Blue Train
5	*** Ring
6	*** Alchemist
7	*** Fifth Mountain
8	*** Zahir

Remove Unwanted Chars

- Remove spaces/chars from **both** sides of a string

```
SELECT TRIM(string)
```

```
SELECT TRIM(BOTH ' ' FROM '  Uni ' )
```

- Remove spaces/chars from the **left** side of a string

```
SELECT TRIM(LEADING FROM string)
```

- Remove spaces/chars from the **right** side of a string

```
SELECT TRIM(TRAILING FROM string)
```

Other String Functions

- Change letter casing

```
LOWER(string)
```

```
UPPER(string)
```

- Reverse order of all characters in a string

```
REVERSE(string)
```

- Repeat string

```
REPEAT(string, count)
```

Count Number of Chars

- Count the number of **characters** in a string

```
LENGTH(string)
```

```
CHAR_LENGTH(string)
```

- Count the number of **bits** in a string

```
BIT_LENGTH(string)
```

- More string functions at PostgreSQL documentation

Math Functions

Arithmetic Operators and Numeric Functions

Math Functions

- PostgreSQL supports **basic arithmetic operations**
- Problem: find the area of triangles by the given side and height

id [PK] integer	side numeric	height numeric
1	2	4
2	1	18
3	4.5	3
4	8	12
5	3	5

id [PK] integer	area numeric
1	4.0000000000000000
2	9.0000000000000000
3	6.7500000000000000
4	48.0000000000000000
5	7.5000000000000000

```
SELECT id,
       (side*height)/2 AS area
FROM triangles
ORDER BY id;
```

Arithmetic Operators

- Supported common arithmetic operators

Operator/ Function	Description
-	Subtraction
+	Addition
*	Multiplication
/	Division
%, MOD(a, b)	Modulo
^, POWER(a, b)	Exponentiation
|/, SQRT(a)	Square root
@, ABS(a)	Absolute value

Math Functions

- Get the value of **Pi** (15-digit precision)

```
SELECT PI(); #3.141592653589793
```

- Get the result from an integer division

```
SELECT DIV(11, 2); #5
```

FLOOR & CEILING

- **FLOOR** & **CEILING** – return the nearest integer

```
FLOOR(value)
```

```
SELECT FLOOR(33.68);     #33
```

```
CEILING(value)
```

```
SELECT CEILING(33.68);   #34
```

Math Functions

- **ROUND** – obtain desired precision

```
ROUND(value)
```

```
ROUND(value, precision)
```

Can be negative

```
SELECT ROUND(33.6888);     #34
```

```
SELECT ROUND(33.6888, 2); #33.69
```

```
SELECT ROUND(33.6888,-1); #30
```

Math Functions

- **TRUNC - truncate to n decimal places**

```
TRUNC(value)
```

```
SELECT TRUNC(12.588);      #12
```

```
TRUNC(value, precision)
```

```
SELECT TRUNC(12.588, 1);   #12.5
```

Problem: Format Costs

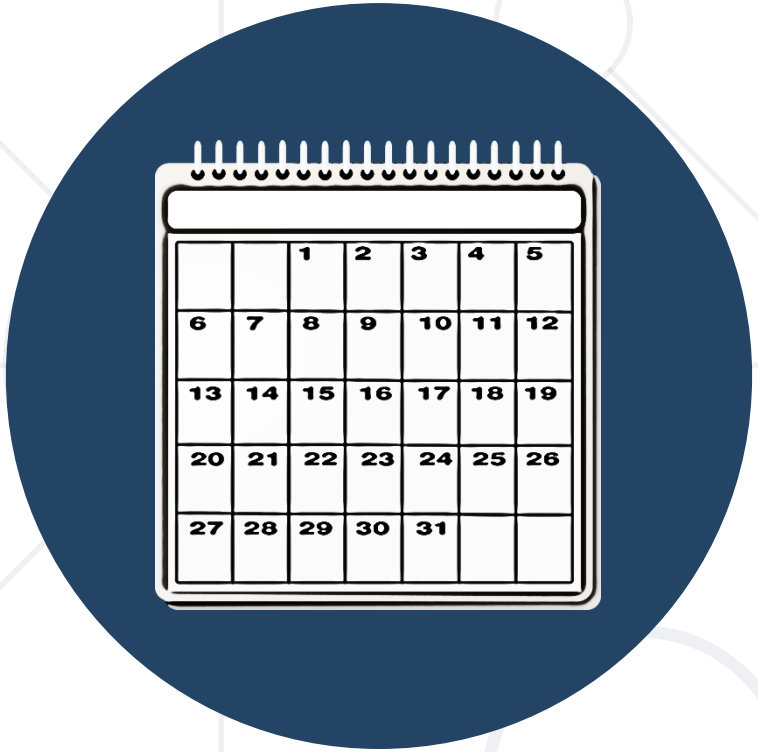
- Write a query to display each book's **`title`** and **`cost`**
  - Query table **`books`** from the **`book_library`** database
  - **Format** the cost to **3 digits** after the decimal point
  - Name the column **`modified_price`**

title character varying (100)	modified_price numeric
Unfinished Portrait	15.990
The Mysterious Affair at Styles	17.990
The Big Four	14.990
The Murder at the Vicarage	13.990
The Mystery of the Blue Train	12.990
Julius Caesar	11.990
Timon of Athens	13.990
As You Like It	18.990

Solution: Format Costs

```
SELECT title,
       TRUNC(cost, 3)
AS modified_price
FROM books
ORDER BY id;
```

title character varying (100)	modified_price numeric
Unfinished Portrait	15.990
The Mysterious Affair at Styles	17.990
The Big Four	14.990
The Murder at the Vicarage	13.990
The Mystery of the Blue Train	12.990
Julius Caesar	11.990
Timon of Athens	13.990
As You Like It	18.990



Date / Time Functions

Date Functions

- **EXTRACT** – extract a segment from a date as an integer
  - ***Part*** can be second, minute, hour, day, week, month, year

```
EXTRACT(part FROM date)
```

- **AGE** – find the difference between two dates

```
AGE(first_date, second_date)
```

Date Functions – Example: Life Span

- Write a query to calculate how long have authors lived
  - Use **AGE**
  - Query table **`authors`**

Full Name text	Life Span interval
Agatha Christie	85 years 3 mons 27 days
William Shakespeare	51 years 11 mons 27 days
Danielle Schuelein-Steel	[null]
Joanne Rowling	[null]
Lev Tolstoy	82 years 2 mons 11 days
Paulo Souza	[null]
Stephen King	[null]
John Tolkien	81 years 7 mons 30 days

Solution Example: Life Span

```
SELECT  CONCAT(first_name, ' ', last_name) AS "Full Name",
      AGE(died, born) AS "Life Span"
FROM authors;
```

Full Name text	Life Span interval
Agatha Christie	85 years 3 mons 27 days
William Shakespeare	51 years 11 mons 27 days
Danielle Schuelein-Steel	[null]
Joanne Rowling	[null]
Lev Tolstoy	82 years 2 mons 11 days
Paulo Souza	[null]
Stephen King	[null]
John Tolkien	81 years 7 mons 30 days

Problem & Solution: Year of Birth

- Write a query to show the author's year of birth
  - Use **EXTRACT**
  - Query table **authors**

```
SELECT first_name, last_name,
       EXTRACT(year FROM born)
       AS year
FROM authors;
```

first_name character varying (30)	last_name character varying (30)	year numeric
Agatha	Christie	1890
William	Shakespeare	1564
Danielle	Schuelein-Steel	1947
Joanne	Rowling	1965
Lev	Tolstoy	1828
Paulo	Souza	1947
Stephen	King	1947

Date/Time Functions

- **NOW** – obtains the current date and time, including time zone

```
SELECT NOW(); #2023-02-23 10:49:42.662178+02
```

- **CURRENT_DATE** and **CURRENT_TIME**

```
SELECT CURRENT_DATE;    #2023-02-23
```

```
SELECT CURRENT_TIME;    #10:55:17.495425+02:00
```

- **TO_CHAR** – formats the date value according to the format

```
SELECT TO_CHAR(NOW(), 'DD Month YYYY') AS "Date";
```

```
                    #23 February  2023
```

Problem: Format Dates of Birth

- Write a query to display each author's **last name**, **date of birth**
  - Query table **`authors`** from the **`book_library`** database
  - **Format** the **`born`** field, for example, **`15 (Mon) Sep 1890`**
  - Name the column **`Date of Birth`**

Last Name character varying (30)	Date of Birth text
Christie	15 (Mon) Sep 1890
Shakespeare	26 (Sun) Apr 1564
Schuelein-Steel	14 (Mon) Jul 1947
Rowling	31 (Sat) Jul 1965
Tolstoy	09 (Tue) Sep 1828
Souza	24 (Sun) Aug 1947
King	21 (Sun) Sep 1947

Solution: Format Dates of Birth

```
SELECT last_name AS "Last Name",
       TO_CHAR(born,
       'DD (Dy) Mon YYYY') AS
       "Date of Birth"
FROM authors;
```

Last Name character varying (30)	Date of Birth text
Christie	15 (Mon) Sep 1890
Shakespeare	26 (Sun) Apr 1564
Schuelein-Steel	14 (Mon) Jul 1947
Rowling	31 (Sat) Jul 1965
Tolstoy	09 (Tue) Sep 1828
Souza	24 (Sun) Aug 1947
King	21 (Sun) Sep 1947

Wildcards

Selecting Results by Partial Match

Wildcards

- Used to substitute any other character(s) in a string
  - '**%**' - represents zero, one, or multiple characters
  - '**_**' - represents a single character
  - Can be used in combinations
- Used with **LIKE** operator in a **WHERE** clause
  - Similar to **Regular Expressions**

Wildcards – Examples

- Find any values that start with "S"

```
WHERE last_name LIKE 'S%';
```

- Find any values that have "o" in the second position

```
WHERE middle_name LIKE '_o%';
```

- Find any values that start with "A" and end with "a"

```
WHERE first_name LIKE 'A%a';
```

Wildcard Escape Characters

- **ESCAPE** – specify a prefix to treat special characters as normal

```
SELECT id, last_name
FROM authors
WHERE last_name LIKE '%l!_%' ESCAPE '!';
```

Problem: Harry Potter Books

- Write a query to retrieve information about the titles of all **Harry Potter books**
  - Use **Wildcards**
  - Query **`book_library`** database, table **`books`**

title character varying (100)
Harry Potter and the Philosophers Stone
Harry Potter and the Chamber of Secrets
Harry Potter and the Prisoner of Azkaban
Harry Potter and the Goblet of Fire
Harry Potter and the Order of the Phoenix
Harry Potter and the Half-Blood Prince
Harry Potter and the Deathly Hallows
Harry Potter and the Deathly Hallows

Solution: Harry Potter Books

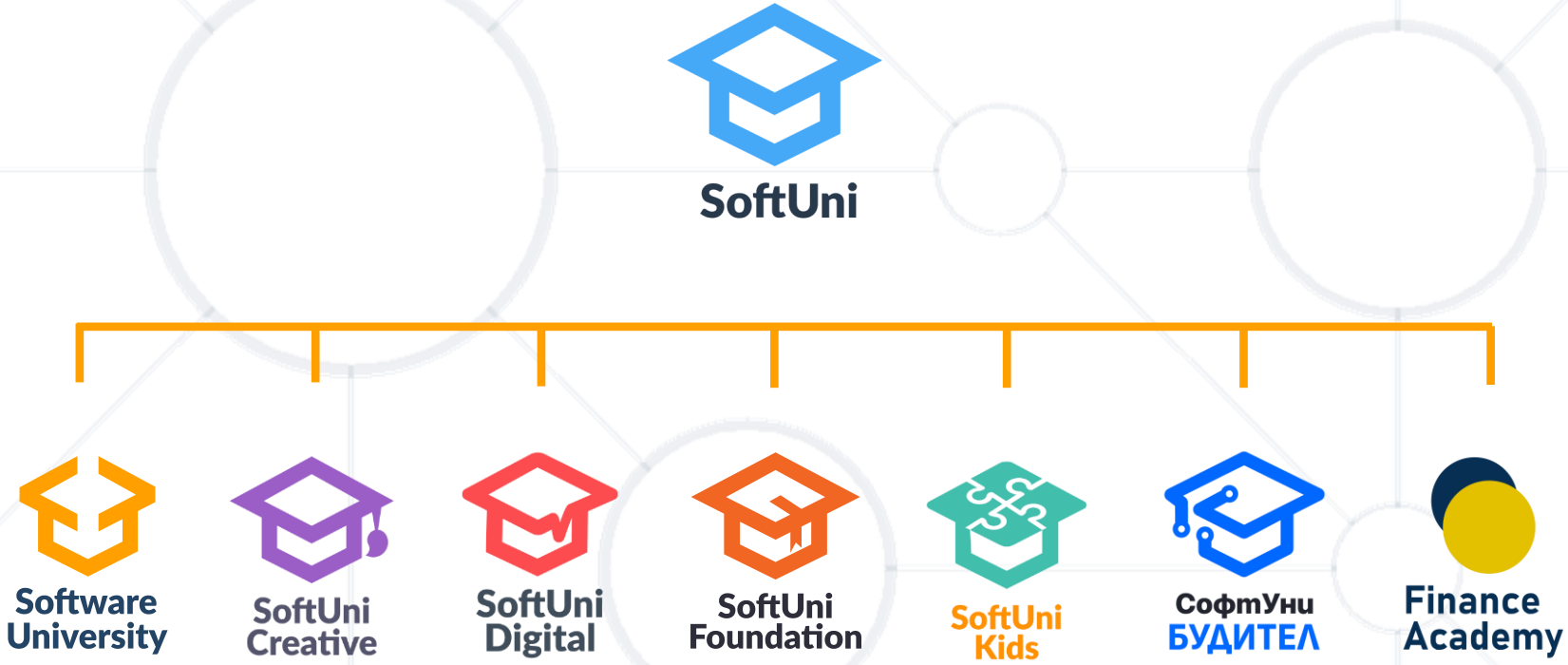
```
SELECT title FROM books
WHERE title LIKE '%Harry Potter%'
ORDER BY id;
```

title character varying (100)
Harry Potter and the Philosophers Stone
Harry Potter and the Chamber of Secrets
Harry Potter and the Prisoner of Azkaban
Harry Potter and the Goblet of Fire
Harry Potter and the Order of the Phoenix
Harry Potter and the Half-Blood Prince
Harry Potter and the Deathly Hallows
Harry Potter and the Deathly Hallows

Summary

- PostgreSQL provides various **built-in functions**
  - **String** functions
  - **Math** functions
  - **Date/Time** functions
- Using Wildcards, we can obtain results by **partial string matches**
  - WHERE clause and **LIKE** operator

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

SOFTWARE GROUP

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg