Exercises: PostgreSQL Data Aggregation

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

Submit your solutions in the SoftUni Judge Contest.

Mr. Bodrog is a small and greedy goblin who serves as the supervisor of Gringotts, the biggest bank in the wizarding community. His most prized possession is a database that contains information about deposits made in the wizarding world, and his pastime is to extort money from others. Despite his eagerness to obtain your money, you do not possess any magical abilities. Instead, you have expertise in database management, which allows you to access this valuable data Mr. Bodrog insists that you provide him with daily reports, threatening to send a pack of starving werewolves after you if you fail to comply. To avoid these dangerous creatures, it is recommended to create a database named gringotts_db and open its query tool. Download the 04-Exercises-Data-Aggregationgringotts_db.sql file from the course instance, import it into the query tab of your database, and execute the queries provided in the file. Take some time to familiarize yourself with the tables in the gringotts db database, as they will be used in the exercises that follow.

1. COUNT of Records

After gaining access to this extremely valuable database, determine the exact number of records contained within

Submit your query for this task in the Judge system.

Example

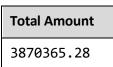
Count
162

2. Total Deposit Amount

Compose a SQL query that calculates the total sum of the deposit amount held at Gringotts Bank.

Submit your query for this task in the Judge system.

Example



3. AVG Magic Wand Size

In your role as the database manager, compute the average size of "magic_wand_size" that belongs to wizards and round the result to the third decimal place.

Submit your query for this task in the Judge system.

Average Magic Wand Size
21.037

















4. MIN Deposit Charge

To become acquainted with the database, determine the **smallest amount** of **"deposit_charge"**.

Submit your query for this task in the Judge system.

Example

Minimum Deposit Charge
1.00

5. MAX Age

Determine the **maximum "age"** among the wizards in the database.

Submit your query for this task in the Judge system.

Example

Maximum Age
73

6. GROUP BY Deposit Interest

Write a SQL query to order the "deposit_group" based on the total amount of "deposit_interest" in each group, and then sort the results in descending order by the total interest in each group.

Submit your query for this task in the Judge system.

Example

deposit_group	Deposit Interest
Troll Chest	685.35
Human Pride	676.97
Blue Phoenix	669.01
Venomous Tongue	574.64

7. LIMIT the Magic Wand Creator

Retrieve the "magic_wand_creator" with the smallest "magic_wand_size" from the "wizard_deposits" table. The query should group the results by "magic_wand_creator" and display the "Minimum Wand Size" for each creator. The results should be sorted in ascending order by the minimum wand size and limited to the top five smallest wand sizes.

Submit your query for this task in the Judge system.

magic_wand_creator	Minimum Wand Size
Mykew Gregorovitch	10











Ollivander family	10
Death	11
Jimmy Kiddell	12

8. Bank Profitability

Mr. Bodrog is interested in the profitability of the bank and wants to know the average interest rates of all "deposit_groups" rounded down to the nearest integer. The query should categorize the deposits based on whether they have **expired** or **not** and retrieve data only for deposits that have a **"deposit_start_date"** after '1985-01-01'. The results should be sorted in descending order by "deposit_group" and ascending order by the "is_deposit_expired" flag.

Submit your query for this task in the Judge system.

Example

deposit_group	is_deposit_expired	Deposit Interest
Venomous Tongue	0	16
Venomous Tongue	1	13
Troll Chest	0	21
Human Pride	1	13
Blue Phoenix	1	21

9. Notes with Dumbledore

Generate a SQL query to retrieve the "last_name" of each wizard and the number of "notes" they wrote that contains the word "Dumbledore" in the "wizard_deposits" table.

Submit your query for this task in the Judge system.

last_name	Notes with Dumbledore
Grindelwald	1
Brown	1
Lovegood	1
•••	
Creevey	2
Weasley	4















Dumbledore	1

10. Wizard View

Create a view in SQL named "view_wizard_deposits_with_expiration_date_before_1983_08_17" that fetches data from the "wizard_deposits" table. The view should display the full name of the wizard, concatenated from their "first_name" and "last_name", along with the "deposit_start_date", "deposit_expiration_date", and "deposit_amount". The view's results should be grouped by the "Wizard Name", "Start Date", "Expiration Date", and "Amount". Additionally, the view should only include deposits that have an expiration date before or on '1983-08-17', and should be ordered by the "Expiration Date" in ascending order.

Submit your query for this task in the Judge system.

Example

Wizard Name	Start Date	Expiration Date	Amount
Alicia Spinnet	1980-02-06	1980-03-04	6269.39
Anthony Goldstein	1980-05-11	1980-05-22	5264.16
Wilhelmina Grubbly-Plank	1980-08-19	1980-08-21	21263.21
Hermione Granger	1980-11-17	1981-01-13	20232.87
	•••		
Marvolo Gaunt	1981-04-12	1981-09-20	22895.49
Remus Lupin	1982-05-08	1982-06-04	17821.66
Hepzibah Smith	1983-05-25	1983-08-17	33665.13

11. Filter Max Deposit

Create a SQL query that retrieves the name of the "magic wand creator" and their maximum "deposit_amount" from the "wizard_deposits" table. The results should be grouped by the "magic wand creator" and filtered to only include those with a maximum "deposit amount" that falls outside the range of 20000 to 40000. Order the results by "Max Deposit Amount" in descending order, and limit the results to 3 records.

Submit your query for this task in the Judge system.

magic_wand_creator	Max Deposit Amount
Ollivander family	49964.03
Arturo Cephalopos	49767.47
Jimmy Kiddell	49041.09















12. Age Group

Create a SQL query that groups the wizards from the "wizard_deposits" table into age groups of '[0-10]', '[11-20]', '[21-30]', '[31-40]', '[41-50]', '[51-60]', and '[61+]'. The query should count the number of wizards in each "Age Group" and display the results in ascending order based on the "Age Group". Submit your query for this task in the Judge system.

Example

Age Group	count
[11-20]	21
[21-30]	32
[31-40]	28
[41-50]	24
[51-60]	26
[61+]	31

Congratulations on your effective management of the Gringotts database! Your expertise has earned you an invitation to become an analyst at **SoftUni**. To prepare for this role, you'll be working with a familiar database, which has been modified for these tasks. Start by creating a fresh database named

data_aggregation_softuni_management_db. Once done, retrieve the 04-Exercises-Data-**Aggregation-softuni management db.sql** file from the course instance, import it into your database's query tab, and execute the queries in the file. Take your time to familiarize yourself with the tables in the data_aggregation_softuni_management_db as they will be utilized in the upcoming exercises. This way, you'll be ready to tackle the tasks effectively and showcase your analytical skills.

13. SUM the Employees

Your first task as an analyst at SoftUni is to write an SQL query that calculates the total number of employees in each department. The "department_id" is stored in the "employees" table, and the following IDs are used to identify each department:

- 1 Engineering
- 2 Tool Design
- 3 Sales
- 4 Marketing
- 5 Purchasing
- 6 Research and Development
- 7 Production

Submit your query for this task in the Judge system.

Engineering	Tool Design	Sales	Marketing	Purchasing	Research and Development	Production
4	2	0	4	3	3	137













14. Update Employees' Data

You have been tasked with updating the salaries and job titles of employees based on their hire dates. Write a SQL query that updates the "salary" and "job_title" columns of the "employees" table according to the following rules:

- if the employee's "hire_date" is before '2015-01-16', their salary should be increased by 2500 and their job title should be **prefixed** with "Senior"
- if the employee's "hire_date" is before '2020-03-04, their salary should be increased by 1500 and their job title should be **prefixed** with "Mid-"
- otherwise, the employee's salary and job title should remain unchanged.

Submit your query for this task in the Judge system.

Example

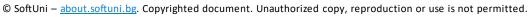
Before update

first_name	job_title	salary
Guy	Production Technician	12500.000
Kevin	Marketing Assistant	13500.000
Roberto	Engineering Manager	43300.000
Ruth	Production Technician	13500.000
•••		
Suroor	Production Technician	11000.000
Alex	Production Technician	10000.000
Hazem	Quality Assurance Manager	28800.000

After update

first_name	job_title	salary
Guy	Senior Production Technician	15000.000
Kevin	Senior Marketing Assistant	16000.000
Roberto	Senior Engineering Manager	45800.000
Ruth	Mid-Production Technician	15000.000
		•••
Suroor	Mid-Production Technician	12500.000
Alex	Production Technician	10000.000



















Hazem	Quality Assurance Manager	28800.000

15. Categorizes Salary

Write a SQL query that groups employees by their job titles and calculates the average salary for each group. The query should also add a column called "Category" that categorizes each "job_title" based on the following rules:

- if the average "salary" is greater than 45,800, the category should be "Good"
- if the average "salary" is between 27,500 and 45,800 (inclusive), the category should be "Medium"
- if the average salary for the job title is less than 27,500, the scale should be "Need Improvement"

Arrange the outcomes based on the "Category" column in ascending sequence. If there are several employees within the group, arrange them by their "job title" in alphabetical order.

Submit your query for this task in the Judge system.

Example

job_title	Category
Mid-Chief Financial Officer	Good
Senior Chief Executive Officer	Good
Senior Information Services Manager	Good
Mid-Accounts Manager	Medium
Mid-Application Specialist	Medium
Accountant	Need Improvement
Application Specialist	Need Improvement
Stocker	Need Improvement

16. WHERE Project Status

Create a SQL query that selects the "project_name" with the word '%Mountain%' in their name from the "projects" table. The project status should be determined based on the following criteria:

- if a project has NO "start_date" or "end_date", its status is "Ready for development"
- if a project has a "start date" but NO "end date", its status is "In Progress".
- otherwise, its status is "Done".

Submit your query for this task in the Judge system.

project_name	project_status
HL Mountain Frame	In Progress

















LL Mountain Frame	Done
Mountain-100	In Progress
Mountain	Done
Mountain	Ready for development
Mountain	Done
Women`s Mountain Shorts	Ready for development
Fender Set - Mountain	In Progress

17. HAVING Salary Level

Write a SQL query to retrieve the number of employees and salary level of each department from the "employees" table. The "salary_level" column should be determined based on the following rules:

- if the average "salary" of a department is above 50,000, the salary level is "Above average"
- if the average "salary" of a department is below or equal to 50,000, the salary level is "Below average"
- only departments with an average "salary" above 30,000 should be included in the result.

The resulting dataset should encompass the subsequent columns: "department_id", "num_employees" and "salary level". Arrange the output based on the "department id".

Submit your query for this task in the Judge system.

Example

department_id	num_employees	salary_level
1	4	Below average
6	3	Below average
11	10	Below average
16	2	Above average

18. * Nested CASE Conditions

To create a view ("view performance rating"), select the "first name", "last name", "job title", "salary", and "department id" from the "employees" table. Then, use the following conditions to generate the "performance_rating" column:

- if an employee's "salary" is greater than or equal to 25,000 and their "job_title" starts with 'Senior%', their "performance_rating" should be "High-performing Senior"
- if an employee's "salary" is greater than or equal to 25,000 and their "job title" does not start with "Senior", their "performance_rating" should be "High-performing Employee"
- if neither of the above criteria is met, the employee's "performance rating" should be "Averageperforming".

Submit your query for this task in the Judge system.















Example

first_name	last_name	job_title	salary	department_id	performance_rating
Guy	Gilbert	Senior Production Technician	15000.000	7	Average-performing
Kevin	Brown	Senior Marketing Assistant	16000.000	4	Average-performing
Roberto	Tamburello	Senior Engineering Manager	45800.000	1	High-performing Senior
Roberto	Tamburello	Senior Engineering Manager	45800.000	1	High-performing Senior
Reuben	D`sa	Mid- Production Supervisor	26500.000	7	High-performing Employee
Hazem	Abolrous	Quality Assurance Manager	28800.000	13	High-performing Employee

19. * Foreign Key

Create a table named "employees_projects" with columns for "id", "employee_id", and "project_id". The "employee_id" column should have a foreign key constraint that references the "id" column in the "employees" table, and the "project_id" column should have a foreign key constraint that references the "id" column in the "projects" table.

Submit your query for this task in the Judge system.

Example

id [PK] integer	employee_id integer	project_id intiger		

20. * JOIN Tables

Write a SQL query to join all columns from the "departments" table and the "employees" table where the "id" column in the "departments" table matches the "department_id" column in the "employees" table. The result set should include all columns from both tables.

Submit your query for this task in the Judge system.

















id	department_name	manager_id	id	first_name	last_name		department_id	
7	Production	148	1	Guy	Gilbert	:	7	
4	Marketing	46	2	Kevin	Brown		4	
1	Engineering	12	3	Roberto	Tamburello		1	
2	Tool Design	4	4	Rob	Walters		2	
2	Tool Design	4	5	Thierry	D`Hers		2	
		•••		•••		:		
7	Production	148	199	Stefen	Hesse		7	
13	Quality Assurance	274	200	Hazem	Abolrous		13	













