PostgreSQL Exam Preparation III

Exam problems for the PostgreSQL course @ Software University.

Submit your solutions in the SoftUni Judge Contest.

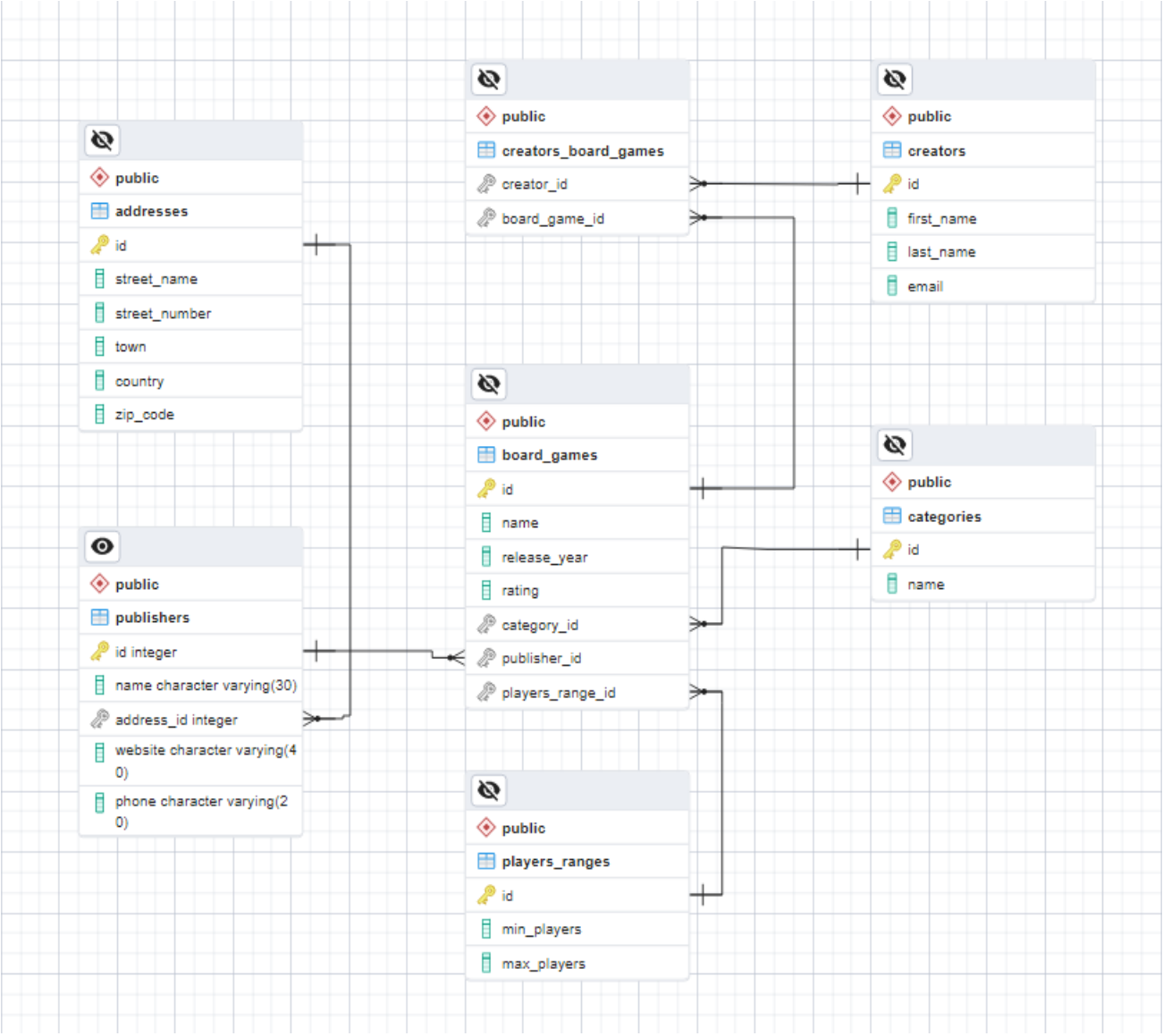
Board Games

Emily has a passion for collecting and playing all sorts of board games. Her game collection has grown significantly over the years, and she is constantly on the lookout for new and exciting games to add to it.

As her collection grew, Emily found it increasingly challenging to keep track of all the games she owned. She realized that she needed a way to manage her collection more efficiently and decided to create a database that would store all the relevant information about her games.

Section 1. Data Definition Language (DDL) - (30 pts)

The following E/R Diagram of the **Board Games** database was created by Emily.



Assist Emily in setting up a PostgreSQL database called **"board_games_db"** which should include seven tables:

- **"categories"** - hold data about the name of each board game category;
- **"addresses"** - store information regarding the locations of board game publishers;
- **"publishers"** - contain information about the publishers of the board games;
- **"players_ranges"** - hold information about the minimum and maximum player counts for each game;
- **"creators"** - store data about the creators of the board games;
- **"board_games"** - contain information about each individual board game;
- **"creators_board_games"** - serves as a mapping table between the creators and board games.

NOTE: It's important to keep in mind that foreign keys should adhere to the following naming convention:

fk_<referencing_table>_<referenced_table>

Your first assignment is to create the database tables based on the provided models. Follow the given specifications to create the tables. **Ensure that the constraints match the order of columns.**

categories

Column Name	Data Type	Constraints
id	**Integer** from **0** to **2,147,483,647**	Primary Key, Unique table identification, Auto-increment
name	**String** up to **50** symbols	**NULL** is **not** allowed

addresses

Column Name	Data Type	Constraints
id	**Integer** from **0** to **2,147,483,647**	Primary Key, Unique table identification, Auto-increment
street_name	**String** up to **100** symbols	**NULL** is **not** allowed
street_number	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, Must be a positive number
town	**String** up to **30** symbols	**NULL** is **not** allowed
country	**String** up to **50** symbols	**NULL** is **not** allowed
zip_code	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, Must be a positive number

publishers

Column Name	Data Type	Constraints
id	**Integer** from **0** to **2,147,483,647**	Primary Key, Unique table identification, Auto-increment
name	**String** up to **30** symbols	**NULL** is **not** allowed.

address_id	**Integer** from **0** to **2,147,483,647**	Relationship with table **addresses**, Cascade Operations, **NULL** is **not** allowed
website	**String** up to **40** symbols	**NULL** is permitted
phone	**String** up to **20** symbols	**NULL** is permitted

players_ranges

Column Name	Data Type	Constraints
id	**Integer** from **0** to **2,147,483,647**	Primary Key, Unique table identification, Auto-increment
min_players	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, Must be a positive number
max_players	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, Must be a positive number

creators

Column Name	Data Type	Constraints
id	**Integer** from **0** to **2,147,483,647**	Primary Key, Unique table identification, Auto-increment
first_name	**String** up to **30** symbols, Unicode	**NULL** is **not** allowed
last_name	**String** up to **30** symbols, Unicode	**NULL** is **not** allowed
email	**String** up to **30** symbols, Unicode	**NULL** is **not** allowed

board_games

Column Name	Data Type	Constraints
id	**Integer** from **0** to **2,147,483,647**	Primary Key, Unique table identification, Auto-increment
name	**String** up to **30** symbols	**NULL** is **not** allowed
release_year	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, Must be a positive number
rating	**Numeric** number with two-digit precision	**NULL** is **not** allowed
category_id	**Integer** from **0** to **2,147,483,647**	Relationship with table **categories**, Cascade Operations, **NULL** is **not** allowed
publisher_id	**Integer** from **0** to **2,147,483,647**	Relationship with table **publishers**, Cascade Operations, **NULL** is **not** allowed
players_range_id	**Integer** from **0** to **2,147,483,647**	Relationship with table **players_ranges**, Cascade Operations, **NULL** is **not** allowed

creators_board_games

Column Name	Data Type	Constraints
`creator_id`	`Integer` from `0` to `2,147,483,647`	Relationship with table `creators`, Cascade Operations, `NULL` is **not** allowed
`board_game_id`	`Integer` from `0` to `2,147,483,647`	Relationship with table `board_games`, Cascade Operations, `NULL` is **not** allowed

1. Database Design

Submit only your **CREATE** statements for all tables to the Judge.

Section 2. Data Manipulation Language (DML) - (10 pts)

Prior to beginning, it is necessary to import "dataset.sql". If the structure has been properly created, the data should be inserted successfully.

This section requires performing various data manipulations:

2. Insert

Your task is to insert sample data into the database by writing a query that adds the provided records into their **respective tables**, with **all "id" values generated automatically**.

board_games

name	release_year	rating	category_id	publisher_id	players_range_id
Deep Blue	2019	5.67	1	15	7
Paris	2016	9.78	7	1	5
Catan: Starfarers	2021	9.87	7	13	6
Bleeding Kansas	2020	3.25	3	7	4
One Small Step	2019	5.75	5	9	2

publishers

name	address_id	website	phone
Agman Games	5	www.agmangames.com	+16546135542
Amethyst Games	7	www.amethystgames.com	+15558889992
BattleBooks	13	www.battlebooks.com	+12345678907

3. Update

The next assignment is to update the **"players_ranges"** table by increasing the **maximum player** count by **1** for board games that have a player range of **[2, 2]**. Furthermore, you need to change the **names of "board_games"** that were published in **2020 or later** by adding **' V2'** to the end of their original names.

Example

Before update

id	min_players	max_players
1	2	2
2	2	3
3	2	4

After update

id	min_players	max_players
1	2	3
2	2	3
3	2	4

Before update

id	name	release_year	rating	category_id	publisher_id	players_range_id
1	Beyond the Sun	2021	8.19	6	1	1
2	Sumatra	2021	7.08	4	2	2
…	…	…	…	…	…	…
11	Glasgow	2018	7.37	6	11	1
…	…	…	…	…	…	…
49	Bleeding Kansas	2020	3.23	3	7	4
50	One Small Step	2019	5.75	5	9	2

After update

id	name	release_year	rating	category_id	publisher_id	players_range_id
1	Beyond the Sun V2	2021	8.19	6	1	1
2	Sumatra V2	2021	7.08	4	2	2
…	…	…	…	…	…	…
11	Glasgow	2018	7.37	6	11	1
…	…	…	…	…	…	…
49	Bleeding Kansas V2	2020	3.23	3	7	4
50	One Small Step	2019	5.75	5	9	2

4. Delete

In the **"addresses"** table, remove all countries that have a **"town"** starting with the letter **'L'**. Take into consideration that there might be **conflicts with foreign key constraints**.

Example

Before delete

id	name	release_year	rating	category_id	publisher_id	players_range_id
1	Beyond the Sun V2	2021	8.19	6	1	1
2	Sumatra V2	2021	7.08	4	2	2
…	…	…	…	…	…	…
15	Alma Mater	2018	7.68	5	15	5
16	Santa Monica	2018	7.54	4	1	6
…	…	…	…	…	…	…
46	Deep Blue	2019	5.67	1	15	7
47	Paris	2016	9.87	7	1	5
…	…	…	…	…	…	…
50	One Small Step	2019	5.75	5	9	2

id	name	address_id	website	phone
1	Fantasy Flight Games	5	www.fantasyflightgames.com	+18553828880
2	Z-Man Games	9	www.zmangames.com	+12165461654
…	…	…	…	…
16	Agman Games	5	www.agmangames.com	+16546135542
17	Amethyst Games	7	www.amethystgames.com	+15558889992
18	BattleBooks	13	www.battlebooks.com	+12345678907

id	street_name	street_number	town	country	zip_code
…	…	…	…	…	…
4	High Street	8	Boston	USA	68732
5	Chapman Ave	15	Los Angeles	USA	35746
6	Zaokopowa	534	Warsaw	Poland	10000
…	…	…	…	…	…

After delete

id	name	release_year	rating	category_id	publisher_id	players_range_id
2	Sumatra V2	2021	7.08	4	2	2
…	…	…	…	…	…	…
15	Alma Mater	2018	7.68	5	15	5
…	…	…	…	…	…	…
46	Deep Blue	2019	5.67	1	15	7
…	…	…	…	…	…	…
50	One Small Step	2019	5.75	5	9	2

id	name	address_id	website	phone
2	Z-Man Games	9	www.zmangames.com	+12165461654
…	…	…	…	…

17	Amethyst Games	7	www.amethystgames.com	+15558889992
18	BattleBooks	13	www.battlebooks.com	+12345678907

id	street_name	street_number	town	country	zip_code
…	…	…	…	…	…
4	High Street	8	Boston	USA	68732
6	Zaokopowa	534	Warsaw	Poland	10000
…	…	…	…	…	…

Section 3. Querying - (40 pts)

Emily currently needs to extract data, and it's important to note that the sample results in this section are based on a new database. To ensure maximum consistency with the examples provided in this section, it's strongly advised that the database modified by the previous DML problems be cleared and the given "dataset.sql" be re-inserted.

5. Board Games by Release Year

The purpose of this task is to retrieve a list of all board games and sort them in **ascending order** by their **"release_year"**. If there are multiple games released in the same year, they should be sorted in **descending order** based on their **"name"**. The result set should include only the **"name"** and **"rating"** columns.

Example

name	rating
Battle Line: Medieval	7.73
The Castles of Tuscany	7.39
Santa Monica	7.54
…	…
GOLD	7.01
Betrayal at Mystery Mansion	6.89

6. Board Games by Category

Your task is to write a SQL query that selects all board games with **"Strategy Games"** or **"Wargames"** categories and orders them by their **"release_year"** in **descending order**. The query should also include the following columns in the output:

- **id**
- **name**

- **release_year**
- **category_name**

Example

id	name	release_year	category_name
6	Polis	2022	Wargames
7	Pan Am	2022	Strategy Games
19	Kemet: Blood and Sand	2021	Strategy Games
…	…	…	…
28	Undaunted: North Africa	2020	Wargames
…	…	…	…
11	Glasgow	2018	Strategy Games
17	Battle Line: Medieval	2017	Strategy Games

7. Creators without Board Games

Write a PostgreSQL query to retrieve all **"creators"** who **don't have any board games** associated with them. The result should be ordered in **ascending order** based on the **creator's name**. The query should return the following columns:

- **id**
- **creator_name** (the first and last name of the creator concatenated with a space)
- **email**

Example

id	creator_name	email
5	Corey Konieczka	corey@konieczka.com
7	Jamey Stegmaier	jamey@stegmaier.com

8. First 5 Board Games

Get the initial **5** board games with a **"rating"** higher than **7.00** that either contain the letter **'a'** in the board game **"name"** OR have a **"rating"** greater than **7.50**, and have a player count range between **2** and **5**. Arrange the outcome set by the **board game "name"** in **ascending order**, and by **"rating"** in **descending order** in the case of multiple games having the same name. The necessary columns are:

- **name**
- **rating**
- **category_name**

Example

name	rating	category_name
Abandon All Artichokes	7.12	Family Games
Alma Mater	7.68	Strategy Games
Ankh: Gods of Egypt	7.20	Strategy Games
Azul: Summer Pavilion	7.83	Abstract Games
Battle Line: Medieval	7.73	Strategy Games

9. Creators with Emails

Retrieve the **full name**, **email**, and **highest-rated** board game for **creators** whose email ends in **".com"**. Sort the result set in **ascending order** by the creator's **full name**. Required columns:

- **full_name**
- **email**
- **rating**

Example

full_name	email	rating
Alexander Pfister	alexander@pfister.com	8.58
Bruno Cathala	bruno@cathala.com	8.58
Emerson Matsuuchi	emerson@matsuuchi.com	8.60

10. Creators by Rating

Write an SQL query to select the **last name**, **average rating** (rounded up to the next biggest integer), and **publisher's name** for all creators who have created a board game. Only show results for creators whose games are published by **"Stonemaier Games"**. Sort the results by **"average_rating"** in **descending order**.

Example

last_name	average_rating	publisher_name
Leacock	9	Stonemaier Games
Matsuuchi	9	Stonemaier Games
Cathala	8	Stonemaier Games
Pfister	8	Stonemaier Games
Rosenberg	8	Stonemaier Games

Section 4. Programmability - (20 pts)

11. Creator of Board Games

Write an SQL query to create a user-defined function named **`fn_creator_with_board_games()`** that takes the **first name of a board game creator** as a **`VARCHAR(30)`** input. The function should return the **total number of board games** created by the input creator.

For this task, please only submit your **user-defined function** in the Judge system.

Example

Query	Output
`SELECT fn_creator_with_board_games('Bruno')`	13
`SELECT fn_creator_with_board_games('Alexander')`	19

12. Search for Board Games

As part of your task, you need to create a stored procedure called **`usp_search_by_category()`**. This procedure will have a parameter called **`"category"`**, which can have a **maximum length of 50 characters**. The purpose of this procedure is to retrieve detailed information about all board games belonging to the specified category. The information to be displayed includes the game's **`"name"`**, **`"release_year"`**, **`"rating"`**, **`"category_name"`**, publisher's **`"name"`**, **`"min_players"`**, and **`"max_players"`**. To indicate the player counts, append the string **`" people"`** at the end. The results should be sorted in **ascending order** based on the **`"publisher_name"`**. If a publisher has multiple games, then the results should be sorted in **descending order** based on the **`"release_year"`**.

*** Please be aware that to view the procedure's results in a tabular format and conduct efficient testing within the Judge System, it's crucial to establish a table named **`"search_results"`**. This table will serve as a container for the data generated by your stored procedure. Before creating the procedure, itself, execute the subsequent SQL query to create the **`"search_results"`** table:

```
CREATE TABLE search_results (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    release_year INT,
    rating FLOAT,
    category_name VARCHAR(50),
    publisher_name VARCHAR(50),
    min_players VARCHAR(50),
    max_players VARCHAR(50)
);
```

In this task, please ensure that you only submit your **stored procedure and the SQL query to create the table** in the Judge system.

Example

Query

`CALL usp_search_by_category('Wargames')`
 	
`SELECT * FROM search_results;`

Output				
name	Verdun 1916: Steel Inferno	Brief Border Wars	Undaunted: North Africa	Polis
release_year	2020	2020	2020	2022
rating	8.60	7.54	8.09	8.58
category_name	Wargames	Wargames	Wargames	Wargames
publisher_name	Gamewright	Lookout Games	Stronghold Games	Zczech Games Edition
min_players	4 people	3 people	4 people	3 people
max_players	5 people	3 people	4 people	4 people