# PostgreSQL Retake Exam - 7 April 2025

# Space Colonization

*Humanity has colonized the known universe. However, the Council races are starting a new expansion phase. Their goal is to establish new homeworlds in a recently discovered Galaxy. Plenty of citizens are being sent aboard space transportation vessels to achieve it. The Council has tasked you with developing a Colonization Management System to track the colonists' journeys to the space stations.*

## Section 0: Database Overview

You are given an Entity/Relationship Diagram of the **Space** **Colonization** System. This diagram illustrates the connections among different entities within this system, offering a visual depiction of the database structure:

A screenshot of a computer

AI-generated content may be incorrect.

The **Space** **Colonization System** must store information about **planets**, **spaceships**, **colonists**, **stations**, **journeys**, and **assignments**.

You need to create a database named space\_colonization with the following **tables**:

* planets - stores information about **planets**.
* spaceships - stores information about **spaceships**.
* colonists - stores information about **colonists**.
* stations - stores information about space **stations**.
  + Each space station is related to a **planet**.
* journeys - stores information about space **journeys**.
  + Each journey is related to a space **station** and a **spaceship**.
* assignments - stores information about colonists' **assignments**.
  + Each assignment links a colonist to a journey.

## Section 1: Data Definition Language (DDL) - 30 Pts

Make sure you implement the **database** **tables** correctly.

**Important Note:** When working with dates, please adhere strictly to the specified data types in the model tables. For example, if a column is defined as type '**DATE**,' ensure you utilize the '**DATE**' data type. Similarly, if a column is designated as '**TIMESTAMP**,' use the '**TIMESTAMP**' data type. Failure to use the correct data type may result in your submission being rejected by the Judge system.

### Table Design

You have been tasked to create the tables in the database by following the specified models.

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

#### planets

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**,from **1** to **2,147,483,647** | **Primary Key**, Unique table identification, Auto-incremented |
| name | A **string** containing a maximum of **60 characters** | **NULL** is **NOT** permitted,  **UNIQUE** values |

#### spaceships

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**,from **1** to **2,147,483,647** | **Primary Key**, Unique table identification, Auto-incremented |
| name | A **string** containing a maximum of **60 characters** | **NULL** is **NOT** permitted, **UNIQUE** values |
| manufacturer | A **string** containing a maximum of **50 characters** | **NULL** is permitted |
| speed | **Integer**,from **0** to **2,147,483,647** | **NULL** is **NOT** permitted, **DEFAULT 1000** |

#### colonists

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**,from **1** to **2,147,483,647** | **Primary Key**, Unique table identification, Auto-incremented |
| first\_name | A **string** containing a maximum of **40 characters** | **NULL** is **NOT** permitted |
| last\_name | A **string** containing a maximum of **40 characters** | **NULL** is **NOT** permitted |
| identity | A **string** with a **fixed** length of **10 characters** | **NULL** is **NOT** permitted, **UNIQUE** values |
| birth\_date | A **DATE** of birth | **NULL** is **NOT** permitted |

#### stations

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**,from **1** to **2,147,483,647** | **Primary Key**, Unique table identification, Auto-incremented |
| name | A **string** containing a maximum of **60 characters** | **NULL** is **NOT** permitted |
| max\_capacity | **Integer**,from **0** to **2,147,483,647** | **NULL** is **NOT** permitted. The value must be **greater than or equal to 0** (**>= 0**) |
| planet\_id | **Integer**, from **1** to **2,147,483,647** | Relationship with table planets. Cascade operations. **NULL** is **NOT** permitted. |

#### journeys

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**,from **1** to **2,147,483,647** | **Primary Key**, Unique table identification, Auto-incremented |
| start\_time | A **TIMESTAMP** | **NULL** ispermitted. |
| end\_time | A **TIMESTAMP** | **NULL** ispermitted. |
| purpose | A **string** containing a maximum of **8 characters** | **NULL** is **NOT** permitted. Should contain **only** one of the following purposes: **'Civil'** or **'Military'** |
| station\_id | **Integer**, from **1** to **2,147,483,647** | Relationship with table stations. Cascade operations. **NULL** is **NOT** permitted. |
| spaceship\_id | **Integer**, from **1** to **2,147,483,647** | Relationship with table spaceships. Cascade operations. **NULL** is **NOT** permitted. |

#### assignments

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| colonist\_id | **Integer**,from **1** to **2,147,483,647** | Relationship with table colonists. Cascade operations. **NULL** is **NOT** permitted |
| journey\_id | **Integer**,from **1** to **2,147,483,647** | Relationship with table journeys. Cascade operations. **NULL** is **NOT** permitted |
| - | **-** | **Composite Primary Key** on both columns (**colonist\_id** and **journey\_id**). Unique table identification |
| task | An **unlimited** **text** | **NULL** is **NOT** permitted. **DEFAULT 'Engineering activities'** |

## Section 2: Data Manipulation Language (DML) - 10 Pts

Before starting, ensure you import the **'dataset.sql'** file. Successful insertion of data is contingent upon the proper creation of the database structure.

This section entails executing various data manipulation tasks:

### Insert

You must **insert** records with specific data into the **spaceships** table using data **derived** from the **colonists** table.

**Insert** the following **data** into the **spaceships** table based on the **colonists** **born** **after** **'2001-01-01'**:

• **name** - concatenate the **reversed** value for **last\_name** and the **identity** value of the corresponding colonist.

• **manufacturer** - use the **first two letters** from the **first\_name** (**capitalized**), concatenated with the **month** from the **birth\_date**.

• **speed** – use the **year** from the **birth\_date**, increased by **100**.

Do **not hard-code** values! Derive them dynamically from the **colonists** table using SQL queries.

Ensure you only insert data from the **colonists** **born** **after** **'2001-01-01'**.

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **manufacturer** | **speed** |
| … | … | … | … |
| 14 | Adder | Fivebridge | 7000 |
| 15 | CS Hannibal | Oyoba | 9000 |
| 16 | nawoC3675463850 | CL3 | 2102 |
| 17 | illeccU0309591317 | BI6 | 2101 |
| 18 | yelswoM1893106179 | RO9 | 2101 |
| … | … | … | … |
| 34 | ittuppiliF0934906106 | CH12 | 2108 |
| 35 | seveirhS8673439787 | FA4 | 2102 |

### Update

**Update** the **purpose** columnin the **journeys** table based on its **current state**:

* If the **purpose** is currently **'Military'**, change it to **'Civil'**.
* If the **purpose** is currently set to **'Civil'**, change it to **'Military'**.

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **purpose** | **station\_id** | **spaceship\_id** |
| 1 | Military | 25 | 10 |
| 2 | Military | 30 | 11 |
| 3 | Civil | 9 | 1 |
| 4 | Military | 23 | 9 |
| 5 | Civil | 19 | 15 |
| … | … | … | … |

### Delete

As you may recall, data was inserted and updated during our initial work. Now, you need to **remove** certain records from the database.

**Delete** all **colonists** who have **not** been **assigned** to **any journeys**.

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **first\_name** | **last\_name** |
| … | … | … |
| 93 | Kirstin | Steade |
| 94 | Chantalle | Filipputti |
| 95 | Far | Shrieves |

## Section 3: Querying - 40 Pts

**Important Note**: Now, we'll conduct some data extraction tasks. Please ensure that the database is cleared of any manipulations from the previous operations in the Data Manipulation Language (DML) section. **Insert** the provided **dataset** **again** to maintain consistency with the examples in this section.

### Colonists

**Extract** information about **colonists** whose **identity** contains the **numbers '00'** and **last name ends** with letter **'s'**.

**Sort** the results by **last\_name** in **descending** order and then by **identity ascending**.

#### Required Columns

* id
* last\_name
* identity

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **last\_name** | **identyty** |
| 37 | Woods | 7888873006 |
| 17 | Gatus | 3407200129 |
| 49 | Gatus | 5642134000 |

### Journeys by Date

**Extract** information about **journeys** (**id**, **start\_time**, **station\_id**).

**Filter** journeys **started** after **10-02-2025** and with **purpose** set to **'Civil'**.

Format the **start time** as **'DD-MM-YY HH24:MI'**.

**Order** the results by **start\_time ascending**, then by **station\_id ascending**,and finally by **id descending**.

Display the **first 3** results.

#### Required Columns

* id
* start\_time
* station\_id

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **start\_time** | **station\_id** |
| 11 | 13-02-25 20:20 | 3 |
| 13 | 19-02-25 16:40 | 10 |
| 9 | 22-02-25 08:37 | 18 |

### Managers

**Identify** all **colonists** **tasked** with **'Management'** on their **assignments** and **display** the required data.

Write a query that returns the colonists' **full** **names** (concatenate the **first\_name** and **last\_name**, separated by **space**), **identity**, and **role**.

Construct the **role** value by concatenating the string **'Manager/'** andthe **journey\_id**.

Order the results by **role** **descending**, then by **identity ascending**.

#### Required Columns

* full\_name (first\_name + ' ' + last\_name)
* identity
* role (a resulting string, according to the description above)

#### Example

|  |  |  |
| --- | --- | --- |
| **full\_name** | identity | role |
| Kennith Teasell | 0361925158 | Manager/8 |
| Chantalle Filipputti | 0934906106 | Manager/8 |
| Reid Kiera | 8944188416 | Manager/8 |
| Sheena Oleszkiewicz | 0516669745 | Manager/7 |
| … | … | … |

### Planets by Stations Count

Write a query to **extract data** about **planets** by their **station count**. The query should include the following columns:

**planet\_name -** concatenate the string **'Planet: '** with the value of the planet's **name**.

**station\_count** - the **total** **number** of **stations** associated with the **planet**.

**avg\_capacity** - the **average** value for **stations' maximum capacities** related to each planet. **Truncate** the resulting value to the **second decimal place**.

Sort the results by **station\_count** in **descending** order, then by **avg\_capacity** in **descending** order and finally by **planet\_name** **ascending**.

#### Required Columns

* planet\_name
* station\_count
* avg\_capacity

#### Example

|  |  |  |
| --- | --- | --- |
| planet\_name | station\_count | avg\_capacity |
| Planet: Kascarth | 4 | 100.00 |
| Planet: Jeayama | 3 | 100.00 |
| Planet: Pleceliv | 3 | 100.00 |
| Planet: Teutera | 3 | 76.66 |
| … | … | … |
| Planet: WR8 | 1 | 100.00 |
| Planet: Aswuenerth | 0 | [null] |
| Planet: Chumeter | 0 | [null] |
| … | … | … |

### Top 5 Spaceships

Write a query to **extract** the **top 5** **spaceships** with the **highest count** of associated **journeys**.

For each spaceship, **display** the following information:

The **spaceship's name**, the **journeys count**, and the **total number** of **colonists** associated with these journeys.

Display the **first 5** results.

**Sort** the results by **journeys count** in **descending** order, then by the **colonists count** in **descending** order, and finally by the **spaceship's name** **ascending**.

#### Required Columns

* spaceship\_name
* journeys\_count
* colonists\_count

#### Example

|  |  |  |
| --- | --- | --- |
| **spaceship\_name** | journeys\_count | colonists\_count |
| Fade | 3 | 20 |
| Katherina | 1 | 10 |
| SC Serpent | 1 | 10 |
| LWSS Romulus | 1 | 9 |
| LWSS Dark Phoenix | 1 | 8 |

## Section 4: Programmability - 20 Pts

Now it's time to showcase your database skills with some dynamic scripting. Write a series of functions and procedures to demonstrate your versatility.

### Spaceship Journeys Count

Create a **user-defined function** named **udf\_spaceship\_journeys\_count(spaceship\_name VARCHAR(60))** that **receives** a **spaceship** **name** and **returns** the **total number** of associated **journeys**.

* If no associated journeys are found for the given spaceship name, the function should return **zero** (**0**).

Submit **only** your **user-defined function** to the Judge system.

#### Example

|  |
| --- |
| **Test Query** |
| **SELECT** udf\_spaceship\_journeys\_count('Nonexistent') **AS** journey\_count; |
| **Result** |
| **journey\_count** |
| 0 |
|  |
| **Test Query** |
| SELECT udf\_spaceship\_journeys\_count('Fade') AS journey\_count; |
| Result |
| journey\_count |
| 3 |

### Update Max Capacity

Create a stored procedure **udp\_update\_max\_capacity(planet\_name VARCHAR(60), increment INT)** with the following parameters:

* planet\_name - VARCHAR(60)
* increment - INTEGER

The procedure **udp\_update\_max\_capacity(planet\_name VARCHAR(60), increment INT)** receives a **planet name** and an **increment** as **input** and **modifies** the **stations' maximum capacities** **associated** with the given **planet**.

* Update the **maximum capacity** by **increasing** its value with the given **increment** argument.
* The **increment** will always be a **positive integer greater than 0**. You are not supposed to validate this condition.

Submit **only** your **stored procedure** to the Judge system.

#### Example

|  |
| --- |
| **Test Query** |
| CALL udp\_update\_max\_capacity('Teutera', 50); |

|  |  |  |
| --- | --- | --- |
| **Initial State for Stations** | | |
| **name** | **max\_capacity** | **planet\_id** |
| … | … | … |
| Juno Station | 100 | 3 |
| Tartarus | 200 | 3 |
| Torus Base | 30 | 5 |
| Themis | 100 | 5 |
| Zeus Colony | 100 | 5 |
| Illume | 200 | 7 |
| Borealis Colony | 100 | 7 |
| … | … | … |

|  |  |  |
| --- | --- | --- |
| **Resulting State for Stations** | | |
| **name** | **max\_capacity** | **planet\_id** |
| … | … | … |
| Juno Station | 100 | 3 |
| Tartarus | 200 | 3 |
| Torus Base | 80 | 5 |
| Themis | 150 | 5 |
| Zeus Colony | 150 | 5 |
| Illume | 200 | 7 |
| Borealis Colony | 100 | 7 |
| … | … | … |