# PostgreSQL Regular Exam - 8 February 2025

# Online Electronics Store

## *A client operating a large online store specializing in electronic devices has reached out for your expertise in managing their database. Your task is to design and implement an efficient database structure, optimize the existing system for better performance, and conduct an analysis to guide the client in crafting a robust future management strategy for the shop.*

## Section 0: Database Overview

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

A computer screen shot of a computer

Description automatically generated

The **Online Electronics Store** must contain information about **brands**, **classifications**, **customers**, **items**, **orders**, and **reviews**.

Your task is to set up a database named electronics\_store with the following **tables**:

* brands - contains information about the **brands**.
* classifications - contains information about the **item types** (classification).
* customers - contains information about the **customers**.
* items - contains information about the **items**.
  + Each item has a brand and a **classification**.
* orders - contains information about the **orders**.
  + Each order has a customer.
* reviews - contains information about the **reviews**.
  + Each review is related to an item and a customer.
* orders\_items - a **many-to-many** **mapping** table between the **orders** and the **items**.

## 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.

#### brands

|  |  |  |
| --- | --- | --- |
| **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 **50 characters** | **NULL** is **NOT** permitted,  **UNIQUE** values |

#### classifications

|  |  |  |
| --- | --- | --- |
| **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 **30 characters** | **NULL** is **NOT** permitted, **UNIQUE** values |

#### customers

|  |  |  |
| --- | --- | --- |
| **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 **30 characters** | **NULL** is **NOT** permitted |
| last\_name | A **string** containing a maximum of **30 characters** | **NULL** is **NOT** permitted |
| address | A **string** containing a maximum of **150 characters** | **NULL** is **NOT** permitted |
| phone | A **string** containing a maximum of **30 characters** | **NULL** is **NOT** permitted, **UNIQUE** values |
| loyalty\_card | A **Boolean** (TRUE or FALSE) | **NULL** is **NOT** permitted, **DEFAULT FALSE** |

#### items

|  |  |  |
| --- | --- | --- |
| **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 **50 characters** | **NULL** is **NOT** permitted |
| quantity | **Integer**,from **0** to **2,147,483,647** | **NULL** is **NOT** permitted. The value must be **greater than or equal to 0** (**>= 0**) |
| price | **DECIMAL**, up to **12 digits**, **2** of which are after the **decimal point** | **NULL** is **NOT** permitted. The value must be **greater than 0.00** (**> 0.00**) |
| description | An unlimited **text** | **NULL** is permitted |
| brand\_id | **Integer**, from **1** to **2,147,483,647**. | Relationship with table brands. Cascade operations. **NULL** is **NOT** permitted. |
| classification \_id | **Integer**, from **1** to **2,147,483,647**. | Relationship with table classifications. Cascade operations. **NULL** is **NOT** permitted. |

#### orders

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**,from **1** to **2,147,483,647** | **Primary Key**, Unique table identification, Auto-incremented |
| created\_at | A **TIMESTAMP** of the order | **NULL** is **NOT** permitted. **DEFAULT NOW()** |
| customer\_id | **Integer**,from **1** to **2,147,483,647** | Relationship with table customers. Cascade operations. **NULL** is **NOT** permitted |

#### reviews

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| customer\_id | **Integer**,from **1** to **2,147,483,647** | Relationship with table customers. Cascade operations. **NULL** is **NOT** permitted |
| item\_id | **Integer**,from **1** to **2,147,483,647** | Relationship with table items. Cascade operations. **NULL** is **NOT** permitted |
| - | **-** | **Composite Primary Key** on both columns (**customer\_id** and **item\_id**). Unique table identification |
| created\_at | A **TIMESTAMP** of the review | **NULL** is **NOT** permitted. **DEFAULT NOW()** |
| rating | A **DECIMAL**, up to **3** digits, **1** of which is after the decimal point. | **NULL** is **NOT** permitted. **DEFAULT** **0.0**.The value must be **less than or equal to 10.0** (**<= 10.0**) |

#### orders\_items

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| order\_id | **Integer**,from **1** to **2,147,483,647** | Relationship with table orders. Cascade operations. **NULL** is **NOT** permitted |
| item\_id | **Integer**,from **1** to **2,147,483,647** | Relationship with table items. Cascade operations. **NULL** is **NOT** permitted |
| - | **-** | **Composite Primary Key** on both columns (**order\_id** and **item\_id**). Unique table identification |
| quantity | **Integer**,from **0** to **2,147,483,647** | **NULL** is **NOT** permitted. The value must be **greater than or equal to 0** (**>= 0**) |

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

Before starting, ensure to 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 **items** table using data **derived** from the **first** **ten** (10) **rows** of the **reviews table**, ordered by **item\_id** in **ascending** order.

**Insert** the following **data** into the **items** table:

• **name** - concatenate the string '**Item**' and the **created\_at** value of the corresponding review.

• **quantity** - use the **customer\_id** from the review.

• **price** – calculated as the **rating** multiplied by **5** (five).

• **description** - set it to **NULL**.

• **brand\_id** - set it to the **item\_id** from the review.

• **classification\_id** - set it to the **smallest item\_id** from the reviews table (ordered by **item\_id**).

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

Ensure you only insert data from the **first ten** (10) **reviews** based on their **item\_id**.

#### Example

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **id** | **name** | **quantity** | **price** | **description** | **brand\_id** | **classification\_id** |
| … | … | … | … | … | … | … |
| 62 | Zathin++ | 87 | 1131.95 | It has three water levels: L/16L M/23L H/30L, and you can choose a suitable level according to the amount of clothes you need to wash | 7 | 7 |
| 63 | Cardguard SuperClean | 82 | 1063.16 | Do not hesitate to buy one | 7 | 7 |
| 64 | Item2019-02-11 21:33:00 | 9 | 11.00 | [null] | 1 | 1 |
| … | … | … | … | … | … | … |
| 70 | Item2018-11-12 22:24:00 | 8 | 24.50 | [null] | 1 | 1 |
| 71 | Item2018-12-13 10:21:00 | 14 | 16.00 | [null] | 2 | 1 |
| 72 | Item2018-07-27 23:49:00 | 24 | 23.50 | [null] | 3 | 1 |
| 73 | Item2020-12-12 18:34:00 | 3 | 22.50 | [null] | 4 | 1 |

### Update

**Update** the **rating** columnin the **reviews** table based on the **item\_id** and **customer\_id**:

* If the **item\_id** and **customer\_id** values are equal, set the **rating** to **10.0**
* If the **customer\_id** is **greater than** the **item\_id**,set the **rating** to **5.5**
* Otherwise, **do not change** the **rating**.

#### Example

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **item\_id** | **rating** |
| … | … | … |
| 4 | 6 | 2.7 |
| 5 | 6 | 3.0 |
| 6 | 6 | 10.0 |
| 7 | 7 | 10.0 |
| 8 | 1 | 5.5 |
| 9 | 1 | 5.5 |
| … | … | … |

### Delete

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

**Delete** all **customers** who have **no orders** placed.

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **first\_name** | **last\_name** |
| … | … | … |
| 2 | Scott | Anderson |
| 4 | Ronald | Lewis |
| 5 | Brian | Williams |
| 6 | Lauren | Johnson |
| 8 | Karen | Harris |
| … | … | … |

## 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.

### Customers

**Extract** information about **customers** with **loyalty cards** whose **last name** contains the **letter 'm'** or **'M'**.

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

#### Required Columns

* id
* last\_name
* loyalty\_card

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **last\_name** | **loyalty\_card** |
| 11 | Pomroy | true |
| 1 | Martin | true |

### Orders by Date

**Extract** information about **orders** (**id**, **created\_at**, **customer\_id**).

**Filter** **orders** **placed** after **01-01-2025** that have **customer\_id between 15** and **30**.

Format the **creation date** as **'DD-MM-YYYY'**.

**Order** the results by **creation date** **ascending** (formatted one), then by **customer\_id** **descending**,and finally by **order id ascending**.

Display the **first 5** results.

#### Required Columns

* id
* created\_at
* customer\_id

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **created\_at** | **customer\_id** |
| 11 | 02-01-2025 | 28 |
| 40 | 05-01-2025 | 27 |
| 43 | 05-01-2025 | 27 |
| 44 | 06-01-2025 | 16 |
| 14 | 11-01-2025 | 29 |

### Items on Sale

Some **items** need to be put **on sale**. You must **identify** them and **display** the following data.

Write a query that returns the items' **name**, **promotion**, **description**,and **quantity** of all **items** **not included** in any **orders**.

Construct the **promotion** value using the **item's brand name** (**upper case**),a **slash** (/),andthe **classification name** (**lower case**).

Display the **description** by **prepending** the **'On sale**: **'** string at the **beginning** (also applicable for NULL values).

Order the results by **quantity** in **descending** order then by **name ascending**.

#### Required Columns

* name
* promotion (a resulting string, according to the description above)
* description ('On sale: ' + current value)
* quantity

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | promotion | description | quantity |
| Daltfresh | FUJITSU/air conditioners and heaters | On sale: With 10,000 BTUs this unit will save you energy and money while cooling a room. | 106 |
| IronSDLight | ROWENTA/small domestic appliances | On sale: Patented Microsteam400 stainless steel soleplate technology creates perfect steam coverage through its 400 active micro-steam holes. | 88 |
| Galaxy Z Fold3 5G | SAMSUNG/phones and tablets | On sale: | 68 |
| Solarbreeze | DELL/laptops | On sale: With its small size, it can fit virtually anywhere that it is needed. | 55 |
| … | … | … | … |

### Customers without Reviews

Write a query to **identify customers** who have **placed at least one order** but have **not left any reviews**. The query should include the following columns:

**id** as **customer\_id**,

**full\_name** - concatenate the customer's **first\_name** and **last\_name**, separated by a **space**,

**total\_orders** - the total number of orders placed by the customer,

**loyalty\_status** - return **'Loyal Customer'** if the customer **has a loyalty card**, otherwise **'Regular Customer'**.

Sort the results by **total\_orders** in **descending** order, then by **customer\_id** in **ascending** order.

#### Required Columns

* id as customer\_id
* full\_name
* total\_orders
* loyalty\_status

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| customer\_id | full\_name | total\_orders | loyalty\_status |
| 31 | Risa Goldsbrough | 4 | Regular Customer |
| 28 | Selene Mateo | 3 | Regular Customer |
| 26 | Cherilyn Bradane | 2 | Regular Customer |
| … | … | … | … |
| 32 | Katlin Everitt | 1 | Regular Customer |
| 33 | Lira Stickler | 1 | Loyal Customer |

### Top-rated Items

Write a query to extract the **top 3** **items** with the **highest average ratings** from the **reviews**. For each item, **display** the following information:

The **item's name**, its **average rating** (**rounded to 2 decimal places**), the **total number** of **reviews** for the item, the **item's** **brand name**, the **item's classification name**

**Include** **only items** that have **at least 3 reviews**. Display the **first 3** results.

**Sort** the results by the **average rating** in **descending** order and then by the **item name** in **ascending** order.

#### Required Columns

* item\_name
* average\_rating
* total\_reviews
* brand\_name
* classification\_name

#### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **item\_name** | average\_rating | total\_reviews | brand\_name | classification\_name |
| XPS 17 | 4.47 | 3 | Dell | Laptops |
| CLD washer | 4.33 | 3 | Beko | Home appliances |
| ACER Cromebook CB311 | 3.81 | 7 | Acer | Laptops |

## 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.

### Items by Classification

Create a **user-defined function** named **udf\_classification\_items\_count(classification\_name VARCHAR(30))** that **receives** a **classification** **name** and **returns** the **total number** of associated **items** in a **string** format**:**

* "**Found {count} items.**"
* If no items are found for the given classification name, the function should return "**No items found.**"

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

#### Example

|  |
| --- |
| **Test Query** |
| **SELECT** udf\_classification\_items\_count('Nonexistent') **AS** message\_text; |
| **Result** |
| **message\_text** |
| No items found. |
|  |
| **Test Query** |
| SELECT udf\_classification\_items\_count('Laptops') AS message\_text; |
| Result |
| message\_text |
| Found 9 items. |

### Update Loyalty Status

Create a stored procedure **udp\_update\_loyalty\_status(min\_orders INT)** with the following parameters:

* min\_orders - INTEGER

The procedure **udp\_update\_loyalty\_status(min\_orders INT)** receives a **minimum count** of ordersas **input** and **modifies** the **loyalty card status** for **all customers** who have placed **at least** the **same minimum number** of **orders**.

* **Set** the **loyalty card status** for **those customers** to **TRUE**.
* **Leave** the **loyalty card status** of all **other** customers **unchanged**.
* The **min\_orders** value 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\_loyalty\_status(4); |

|  |  |  |  |
| --- | --- | --- | --- |
| **Initial State of Customers** | | | |
| **id** | **first\_name** | **last\_name** | **loyalty\_card** |
| … | … | … | … |
| 30 | Maridel | Toller | false |
| 31 | Risa | Goldsbrough | false |
| 32 | Katlin | Everitt | false |
| 33 | Lira | Stickler | true |
| 34 | Barbie | Gandey | false |
| 35 | Osborne | Sunderland | false |
| … | … | … | … |

|  |  |  |  |
| --- | --- | --- | --- |
| **Resulting State** | | | |
| **id** | **first\_name** | **last\_name** | **loyalty\_card** |
| … | … | … | … |
| 30 | Maridel | Toller | false |
| 31 | Risa | Goldsbrough | true |
| 32 | Katlin | Everitt | false |
| 33 | Lira | Stickler | true |
| 34 | Barbie | Gandey | false |
| 35 | Osborne | Sunderland | false |
| … | … | … | … |