МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Functions и Triggers

Practice 1. Определение функции максимальной продажи

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Создать функцию, чтобы вычислить наибольшую сумму продажи за один вызов, определяет наибольшую сумму продажи в базе данных. На данном этапе отдел маркетинга делает много запросов по анализу данных, и необходимо разработать более эффективный механизм их обработки, так как в настоящее время они занимают слишком много времени.

Practice 1. Определение функции максимальной продажи

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Шаги для выполнения запроса PostgreSQL:

1. Откройте PostgreSQL и подключитесь к базе данных sqlda

psql sqlda

После успешного подключения вам будет представлен интерфейс к базе данных PostgreSQL.

2. Создайте функцию с именем max_sale, которая не принимает никаких входных аргументов, но возвращает числовое значение, называемое big_sale.
3. Объявите переменную big_sale и запустите функцию.
4. Вставьте максимальную сумму продажи в переменную big_sale.
5. Верните значение big_sale.
6. Завершите функцию оператором LANGUAGE.
7. Вызовите функцию, чтобы узнать, какая самая большая сумма продажи в базе данных?

```
  max
--------
 115000
(1 row)
```

Practice 2. Создание функций с аргументами

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Создать функцию с аргументами и вычислить результат.

В этом упражнении создать функцию, которая вычисляет среднюю сумму продаж для транзакционных продаж в течение определенного диапазона дат. Каждая дата должна быть предоставлена функции в виде текстовой строки.

Practice 2. Создание функций с аргументами

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Шаги для выполнения запроса PostgreSQL:

1. Создайте определение функции для функции с именем avg_sales_window, которая возвращает числовое значение и принимает два значения DATE для указания даты начала и окончания в формате **ГГГГ-ММ-ДД**.
2. Объявите возвращаемую переменную как числовой тип данных и запустите функцию.
3. Выберите среднюю сумму продаж в качестве возвращаемой переменной, если дата операции продажи находится в пределах указанной даты.
4. Возвратите функциональную переменную, завершите функцию и укажите оператор LANGUAGE.
5. Используйте функцию, чтобы определить средние значения продаж между 2013-04-12 и 2014-04-12.

Practice 2. Создание функций с аргументами **решение**

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Шаги для выполнения запроса PostgreSQL:

Создайте определение функции для функции с именем avg_sales_window, которая возвращает числовое значение и принимает значение DATE для указания даты в форме **ГГГГ-мм-дд**.

```
CREATE FUNCTION avg_sales_window(from_date DATE, to_date DATE) RETURNS numeric AS
$sales_avg$
DECLARE sales_avg numeric;
BEGIN
SELECT AVG(sales_amount)
FROM sales INTO sales_avg
WHERE sales_transaction_date > from_date AND sales_transaction_date < to_date;
RETURN sales_avg;
END;
$sales_avg$
LANGUAGE PLPGSQL;
```

Practice 2. Создание функций с аргументами **решение**

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Шаги для выполнения запроса PostgreSQL:

Используйте функцию для определения средней стоимости продаж **с** 12 **апреля** 2013 **г**.

```
SELECT avg_sales_window('2013-04-12', '2014-04-12');
```

```
avg_sales_window|
----------------+
477.686246311006|
```

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Triggers

Practice 3. Создание триггера для отслеживания среднего количества покупок

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Создать триггер для отслеживания обновляемых данных.

Допустим, вы работаете специалистом по данным в Monkey Islands, лучшем дистрибьюторе сомнительных и малоизвестных товаров. Бизнес рассматривает возможность попробовать несколько разных стратегий, чтобы увеличить количество товаров в каждой продаже. Чтобы упростить анализ, вы решили добавить простой триггер, который для каждого нового заказа вычисляет среднее количество по всем заказам и помещает результат в новую таблицу вместе с соответствующим order_id.

Practice 3. Создание триггера для отслеживания среднего количества покупок

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Шаги для выполнения запроса PostgreSQL:

1. Подключитесь к базе данных smalljoins.
2. Создайте новую таблицу с именем avg_qty_log, состоящую из целочисленного поля order_id и числового поля avg_qty.
3. Создайте функцию с именем avg_qty, которая не принимает никаких аргументов, но возвращает триггер. Функция вычисляет среднее значение для всех объемов заказа (order_info.qty) и вставляет среднее значение вместе с самым последним order_id в avg_qty.
4. Создайте триггер с именем avg_trigger, который вызывает функцию avg_qty **ПОСЛЕ** того, как каждая строка вставляется в таблицу order_info.
5. Вставьте несколько новых строк в таблицу order_info с количеством 6, 7 и 8.
6. Посмотрите записи в avg_qty_log. Увеличивается ли средний объем каждого заказа?

```
 order_id |      avg_qty
----------+--------------------
     1625 | 4.7500000000000000
     1626 | 5.0000000000000000
     1627 | 5.3000000000000000
(3 rows)
```

Practice 3. Создание триггера для отслеживания среднего количества покупок **РЕШЕНИЕ**

Шаги для выполнения запроса PostgreSQL:

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Подключиться к базе данных smalljoins

1. Создайте новую таблицу с именем avg_qty_log, состоящую из целочисленного поля order_id и числового поля avg_qty.

```
CREATE TABLE avg_qty_log (order_id integer, avg_qty numeric);
```

Practice 3. Создание триггера для отслеживания среднего количества покупок **РЕШЕНИЕ**

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Шаги для выполнения запроса PostgreSQL:

2. Создайте функцию с именем avg_qty, которая не принимает никаких аргументов, но возвращает триггер. Функция вычисляет среднее значение для всех объемов заказа (order_info.qty) и вставляет среднее значение вместе с самым последним идентификатором order_id в avg_qty.

```
CREATE FUNCTION avg_qty() RETURNS TRIGGER AS $_avg$
DECLARE _avg numeric;
BEGIN
      SELECT AVG(qty) INTO _avg FROM order_info;
      INSERT INTO avg_qty_log (order_id, avg_qty) VALUES (NEW.order_id, _avg);
      RETURN NEW;
END; $_avg$
LANGUAGE PLPGSQL;
```

Practice 3. Создание триггера для отслеживания среднего количества покупок **РЕШЕНИЕ**

Шаги для выполнения запроса PostgreSQL:

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

3. Создайте триггер с именем avg_trigger, который вызывает функцию avg_qty ПОСЛЕ того, как каждая строка вставлена в таблицу order_info.

```
CREATE TRIGGER avg_trigger
AFTER INSERT ON order_info
FOR EACH ROW
EXECUTE PROCEDURE avg_qty();
```

Practice 3. Создание триггера для отслеживания среднего количества покупок **РЕШЕНИЕ**

Шаги для выполнения запроса PostgreSQL:

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

4. Вставьте несколько новых строк в таблицу order_info с количеством 6, 7 и 8.

```
SELECT insert_order(3, 'GROG1', 6);
SELECT insert_order(4, 'GROG1', 7);
SELECT insert_order(1, 'GROG1', 8);
```

5. Посмотрите на записи в avg_qty_log, увеличивается ли среднее количество каждого заказа?

```
SELECT * FROM avg_qty_log;
```

```
order_id|avg_qty           |
--------+------------------+
    1625|4.7500000000000000|
    1626|5.0000000000000000|
    1627|5.2000000000000000|
    1628|5.4545454545454545|
```