Übungsblatt 5 - Erweitertes SQL

Aufgabe

Aufgabe 1 User Defined Function (UDF)

Arbeiten Sie mit dem Clubdata Datensatz (Übungsblatt 3).

Erstellen Sie eine UDF, die die Gesamtkosten für eine Buchung berechnet. Die Funktion hat die folgenden Inputparameter facid (INT), guest (Boolean) und Slots (integer).

Folgende Logik soll abgebildet werden:

- Wenn guest true, dann verwende den Gastpreis aus der Tabelle facilities
- Wenn guest false, dann verwende den Mitgliedspreis
- Berechnen Sie die Gesamtkosten der Buchung als slots * membercost/guestcost und geben ihn aus (numeric)

Testen Sie Ihre UDF mit den folgenden Beispielen:

SELECT calculate_booking_cost(1, True, 2) AS kosten; --Ergibt 50

SELECT calculate_booking_cost(1, False, 2) AS kosten; -- Ergibt 10

Aufgabe 2 Stored Procedure

Erstellen Sie eine Stored Procedure, die eine neue Buchung in der Tabelle bookings erstellt. Die Stored Procedure benötigt folgende Input-Parameter:

- facid

- memid

- starttime

- slots

Logik

- Prüfen Sie, ob das Mitglied (memid) existiert. Falls nicht, geben Sie eine Fehlermeldung aus (RAISE EXCEPTION)
- Überprüfen Sie, ob die Einrichtung (facid) zur gewünschten Zeit (starttime) verfügbar ist.
  - Falls verfügbar, fügen Sie die Buchung zur Tabelle bookings hinzu und geben eine Bestätigungsnachricht aus.
  - Falls die Einrichtung belegt ist, geben Sie eine Fehlermeldung aus.

Testfall : Führen Sie einmal CALL create_booking(1, 1, TIMESTAMP '2023-10-02 09:00:00', 1) aus =>= Erfolgreich

Dann erneut mit erwarteter Exception

Aufgabe 3 Trigger Funktion

Erstellen Sie einen Trigger, der sicherstellt, dass ein Mitglied maximal 3 Buchungen für einen bestimmten Tag durchführen kann

- Trigger-Bedingung: Ausgelöst vor einer INSERT-Operation in der Tabelle bookings.
- Prüfen Sie, ob das Mitglied bereits 3 Buchungen für den definierten Tag gemacht hat. Falls ja, verhindern Sie die Einfügung der Buchung und geben eine Fehlermeldung aus.

Zum Testen führen Sie aus:

```
INSERT INTO cd.bookings (bookid, facid, memid, starttime, slots)
VALUES (10001, 1, 1, '2024-11-12 10:00:00', 1);

INSERT INTO cd.bookings (bookid, facid, memid, starttime, slots)
VALUES (10002, 1, 1, '2024-11-12 10:00:00', 1);

INSERT INTO cd.bookings (bookid, facid, memid, starttime, slots)
VALUES (10003, 1, 1, '2024-11-12 10:00:00', 1);

INSERT INTO cd.bookings (bookid, facid, memid, starttime, slots)
VALUES (10004, 1, 1, '2024-11-12 10:00:00', 1);
```

Aufgabe 4 Window Function

Legen Sie zunächst folgende Beispieltabelle mit Beispieldaten an:

```
 CREATE TABLE orders (
order_id INT GENERATED BY DEFAULT AS IDENTITY,
order_date DATE NOT NULL,
customer_id INT NOT NULL,
order_total DECIMAL(10,2) NOT NULL,
PRIMARY KEY (order_id)
);



INSERT INTO orders (order_date, customer_id, order_total) VALUES
('2021-01-06', 1, 120.00),
('2021-01-07', 2, 180.00),
('2021-01-08', 3, 220.00),
('2021-01-09', 1, 130.00),
('2021-01-10', 2, 210.00),
('2021-01-11', 3, 160.00),
('2021-01-12', 1, 140.00),
('2021-01-13', 2, 190.00),
('2021-01-14', 3, 170.00),
('2021-01-15', 1, 150.00),
('2021-01-16', 2, 200.00),
('2021-01-17', 3, 180.00),
('2021-01-18', 1, 160.00),
('2021-01-19', 2, 210.00),
('2021-01-20', 3, 190.00),
('2021-01-21', 1, 170.00),
('2021-01-22', 2, 220.00),
('2021-01-23', 3, 200.00),
('2021-01-24', 1, 180.00),
('2021-01-25', 2, 230.00),
('2021-01-26', 3, 210.00),
('2021-01-27', 1, 190.00),
```

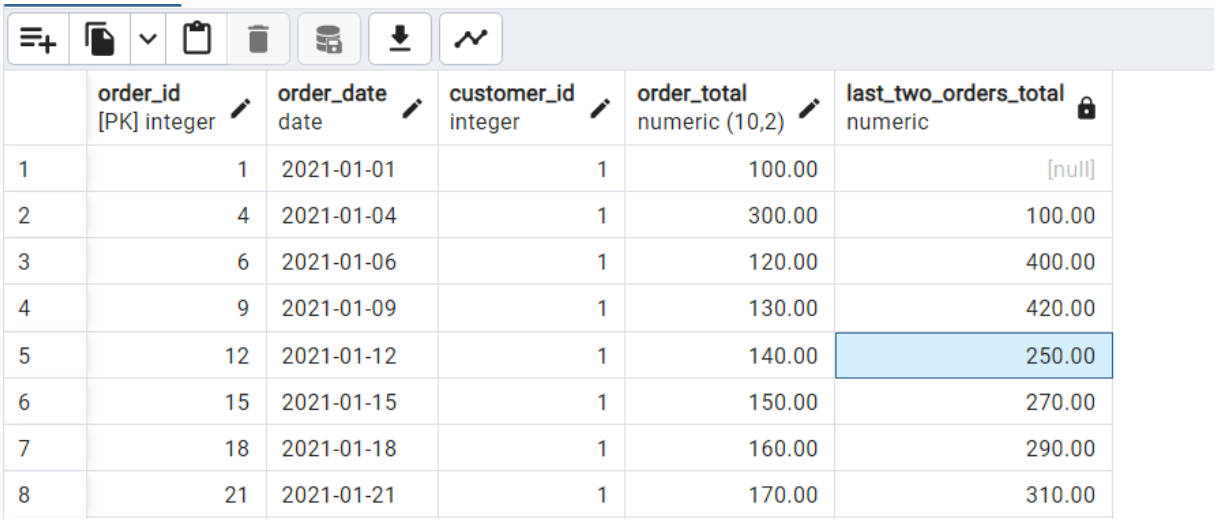
```
('2021-01-28', 2, 240.00),
('2021-01-29', 3, 220.00),
('2021-01-30', 1, 200.00),
('2021-01-31', 2, 250.00),
('2021-02-01', 3, 230.00),
('2021-02-02', 1, 210.00),
('2021-02-03', 2, 260.00),
('2021-02-04', 3, 240.00);
```

4.1 Geben Sie alle Daten der Tabelle "orders" mit der fortlaufenden Summe (order_total) des Umsatzes aus.

4.2 Geben Sie alle Daten der Tabelle "orders" mit einer Spalte, die die Summe des Umsatzes der letzten zwei Bestellungen enthält. Die Ergebnisse sollten nach dem Bestelldatum sortiert sein.

4.3 Erweitern Sie die Abfrage aus Aufgabe 4.2, um für jede Bestellung die Summe der Umsätze der letzten zwei Bestellungen desselben Kunden anzuzeigen.

Der Output soll so aussehen



	order_id [PK] integer	order_date date	customer_id integer	order_total numeric (10,2)	last_two_orders_total numeric
1	1	2021-01-01	1	100.00	[null]
2	4	2021-01-04	1	300.00	100.00
3	6	2021-01-06	1	120.00	400.00
4	9	2021-01-09	1	130.00	420.00
5	12	2021-01-12	1	140.00	250.00
6	15	2021-01-15	1	150.00	270.00
7	18	2021-01-18	1	160.00	290.00
8	21	2021-01-21	1	170.00	310.00