Abgabetermin: FR, 21.11.2014 (15:00)

Name _____

☐ **DES3UEG1**

☐ **DES3UEG2** **Punkte** _____ **Kurzzeichen Tutor** _____

PL/SQL

Die folgenden Aufgaben beziehen sich auf Beispieltabellen, die mit einem vorgegebenen SQL-Skript gemeinsam mit einigen Datensätzen (Insert-Statements) zu erstellen sind. Kommentieren sie ihre Programme ausreichend und führen sie alle relevanten Überprüfungen (z.B. der Eingabeparameter) und Ausnahmebehandlungen durch. Geben Sie dabei entsprechende Fehlermeldungen über das PL/SQL-Paket `dbms_output` aus. Beachten sie, dass nach jeder Prozedur/Funktion bzw. anonymen Block in der nächsten Zeile ein Slash (/) stehen muss, um eine Kompilierung durchzuführen. Testen sie ihre Prozeduren/Funktionen ausführlich.

Aufgabe 1: Erstellen neuer Bestellungen

Erstellen Sie 2 Prozeduren mit denen Bestellungen und Bestellpositionen erfasst werden können.

a) Erstellen Sie zunächst eine Prozedur namens ADD_ORDER, um eine neue Bestellung in die Tabelle ORD aufzunehmen. Die Prozedur soll zwei Parameter aufweisen. Der erste Parameter enthält eine Kundennummer und der zweite Parameter enthält einen Kommissionsplan (Buchstabe zur Kennzeichnung des Kommissionsplans), dessen Standardwert Null ist. Die Bestellnummer ist mit Hilfe der ORDID-Sequenz zu generieren. Verwenden Sie das aktuelle Datum (SYSDATE) als Datum der Bestellung. Die Spalten SHIPDATE und TOTAL werden nicht gesetzt.

   Fügen Sie neue Bestellungen ein (gültige Kundennummer z.B. 108). Überprüfen Sie, ob Zeilen hinzugefügt wurden und merken Sie sich die neuen ORDIDs, um im nächsten Schritt Bestellpositionen zur Bestellung hinzuzufügen.

   Hinweis zur Sequenz: Eine Sequenz dient zur Generierung von Nummern.Verwendung:

   1) Erstellen einer Sequenz (Achtung: die für dieses Beispiel zu verwendende Sequenz existiert bereits): `CREATE SEQUENCE my_seq;`
   2) Generierung und Zugriff auf die nächst höhere Sequenz-Nummer: `my_seq.NEXTVAL`
   3) Zugriff auf die aktuelle Sequenz-Nummer: `my_seq.CURRVAL`

b) Erstellen Sie nun eine Prozedur namens ADD_ITEM, um einer vorhandenen Bestellung eine neue Position, d.h. der Tabelle ITEM eine neue Zeile, hinzuzufügen. Geben Sie Bestellnummer, Produktnummer, Preis und Menge (QTY) für das Produkt mit Hilfe von vier Parametern an. Generieren Sie die Positionsnummer, indem Sie die höchste Positionsnummer für die jeweilige Bestellung um 1 erhöhen. Existiert die Bestellnummer in der Tabelle ITEM nicht, soll die Positionsnummer standardmäßig 1 sein. Generieren Sie die Positionssumme, indem Sie den aktuellen Preis mit der Menge multiplizieren.

Führen Sie eine Fehlerbehandlung für die folgenden Fälle ein:

- Versuch, eine nicht existierende Produktnummer zu bestellen.
- Versuch, einer nicht existierenden Bestellung eine Bestellposition hinzuzufügen.

Führen Sie die Prozedur unter anderem mit folgenden Daten aus (Ihre Bestellnummer soll einer Ihrer zuvor generierten Bestellungen entsprechen und unterscheidet sich möglicherweise vom Beispielwert 622.):

```
SQL> EXEC add_item(622, 100860, 35, 2)
SQL> EXEC add_item(622, 100870, 2.5, 4)
SQL> EXEC add_item(100, 100860, 35, 4)
SQL> EXEC add_item(622, 100, 10, 10)
```

Aufgabe 2: Anzahl der bestellten Produkte

Erstellen Sie eine gespeicherte Funktion namens `Get_Product_Count`, welche die Anzahl der *unterschiedlichen* von einem Kunden bestellten Produkte abruft und zurück gibt. Die Funktion soll als Parameter eine Kundennummer aufweisen. Berücksichtigen sie dabei, dass eine ungültige Kundennummer eingegeben oder ein Kunde keine Produkte bestellt haben kann. Zum Testen können sie u.a. die folgenden Daten verwenden:

```
SQL> EXEC DBMS_OUTPUT.PUT_LINE(get_product_count(107))
SQL> EXEC DBMS_OUTPUT.PUT_LINE(get_product_count(108))
SQL> EXEC DBMS_OUTPUT.PUT_LINE(get_product_count(10))
```

Aufgabe 3: Überprüfen des Kreditlimits

Erstellen Sie eine Prozedur die überwacht, ob Kunden ihren Kreditrahmen überschritten haben. Fügen Sie der Tabelle CUSTOMER mit dem folgenden Befehl eine Spalte hinzu:

```
SQL> ALTER TABLE customer
  2  ADD (creditlimit_indicate  VARCHAR2(3) DEFAULT 'NO'
  3       CONSTRAINT customer_creditlimit_ck CHECK
  4       (creditlimit_indicate IN ('YES', 'NO')));
```

Schreiben Sie eine Prozedur namens CHECK_CREDITLIMIT, welche das Kreditlimit aller Kunden der Tabelle CUSTOMER gegen den Gesamtbetrag (TOTAL) der Bestellungen des jeweiligen Kunden in der Tabelle ORD prüft und die Spalte CREDITLIMIT_INDICATE in der Tabelle CUSTOMER aktualisiert, indem diese Spalte auf YES gesetzt wird, falls dieser Kunde sein Kreditlimit überschritten hat, bzw. auf NO, falls dies nicht der Fall ist. Testen Sie die Prozedur. Geben Sie die entsprechenden Spalten der Tabelle CUSTOMER vor und nach den Prozedur-Aufrufen aus.

Organisatorische Hinweise

Die Lösungen sind von den Studierenden allein auszuarbeiten, Teamarbeit ist nicht erlaubt.

Die Lösungen sind zum Abgabetermin im E-Learning-System online abzugeben. Die Übungen werden von folgenden Tutoren betreut:

- DES3UEG1: Daniel Glaser S1210307059@students.fh-hagenberg.at
- DES3UEG2: Melanie Mayrhofer S1010307074@students.fh-hagenberg.at