Aufgabe 1 – Gespeicherte Funktionen

a) Erstellen Sie eine gespeicherte Funktion, welche die Zwischensumme der Artikel im Warenkorb eines Kunden zurückliefert.

```
CREATE OR REPLACE FUNCTION Zwischensumme (knr IN NUMBER)
  RETURN NUMBER
IS
  rueckgabe NUMBER;

BEGIN

SELECT SUM(Preis*Anzahl) INTO rueckgabe
FROM Warenkorb NATURAL JOIN Artikel
WHERE Kundennummer = knr;

RETURN rueckgabe;
END;

Testfall:
SELECT Kundennummer, Zwischensumme(Kundennummer)
FROM Kunde
WHERE Zwischensumme(Kundennummer) IS NOT NULL;
```

b) Erstellen Sie eine gespeicherte Funktion, welche den gesamten Lagerbestand eines Artikels an allen Standorten anhand der Artikelnummer zurückliefert.

```
CREATE OR REPLACE FUNCTION Lagerbestandabfrage (artnr IN NUMBER)
  RETURN NUMBER
IS
  rueckgabe NUMBER;

BEGIN

SELECT SUM(Lagerbestand) INTO rueckgabe
FROM Lager
WHERE ANummer = artnr;

RETURN rueckgabe;
END;

Testfall:
SELECT DISTINCT ANummer, Lagerbestandabfrage(ANummer)  FROM lager;
SELECT DISTINCT Lagerbestandabfrage(4812) from Lager;
```

Aufgabe 2 – Gespeicherte Prozeduren

Bestellt ein Kunde Waren, so wird der Inhalt des Warenkorbs in eine Bestellung überführt und der Warenkorb geleert. Eine Bestellung besitzt ein Bestelldatum, einen Bestellstatus(bestellt, bestaetigt, geliefert). Initial soll der Versandstatus auf „bestellt“ gesetzt werden.
Zu jeder Bestellposition muss der jeweilige Artikelpreis und die bestellte Anzahl mit abgespeichert werden, da für die Rechnungserstellung der Artikelpreis zum Zeitpunkt der Bestellung maßgebend ist.

a) Ergänzen Sie das Oracle-Datenbankschema in geeigneter Weise. Die Ids für Bestellnummer und Bestellposition sollen automatisch durch das DBMS vergeben werden.

Oracle 11g:

```
DROP TABLE Bestellung cascade constraints;
DROP TABLE Bestellposition cascade constraints;
DROP SEQUENCE Best_seq;
DROP SEQUENCE Pos_seq;

CREATE TABLE Bestellung(
Bestellnummer INTEGER,
Bestelldatum DATE,
Kundennummer INTEGER,
wunschtermin TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
Bestellstatus VARCHAR(10) CHECK (Bestellstatus IN ('bestellt', 'bestaetigt', 'geliefert')),
PRIMARY KEY (Bestellnummer),
FOREIGN KEY (Kundennummer) REFERENCES Kunden (Kundennummer)
);

CREATE TABLE Bestellposition(
Bestellnummer INTEGER,
Pos INTEGER,
Artikelnummer INTEGER,
Anzahl INTEGER CHECK( Anzahl >=0),
Preis NUMERIC (7,2) CHECK (Preis >0),
PRIMARY KEY (Bestellnummer, Position),
FOREIGN KEY (Bestellnummer) REFERENCES Bestellung(Bestellnummer)
ON DELETE CASCADE,
FOREIGN KEY (Artikelnummer) REFERENCES Artikel(Artikelnummer)
 );
```

```
CREATE SEQUENCE Best_seq
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOCYCLE
CACHE 10;

CREATE SEQUENCE Pos_seq
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOCYCLE
CACHE 10;
```

Oracle 12c (**andere Version von SEQUENCE**):

```
DROP TABLE Bestellung cascade constraints;
DROP TABLE Bestellposition cascade constraints;

CREATE TABLE Bestellung(
Bestellnummer INTEGER GENERATED AS IDENTITY INCREMENT BY 1,
Bestelldatum DATE,
Kundennummer INTEGER,
Bestellstatus VARCHAR(9) CHECK (Bestellstatus IN ('bestellt', 'bestaetigt', 'geliefert')),
wunschtermin TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY (Bestellnummer),
FOREIGN KEY (Kundennummer) REFERENCES Kunden (Kundennummer)
);

CREATE TABLE Bestellposition(
Bestellnummer INTEGER,
Position INTEGER GENERATED AS IDENTITY INCREMENT BY 1,
Artikelnummer INTEGER,
Anzahl INTEGER CHECK( Anzahl >=0),
Preis NUMERIC (7,2) CHECK (Preis >0),
Reduktion NUMERIC(4,2) DEFAULT 0.0,
PRIMARY KEY (Bestellnummer, Position),
FOREIGN KEY (Bestellnummer) REFERENCES Bestellung(Bestellnummer)
ON DELETE CASCADE,
FOREIGN KEY (Artikelnummer) REFERENCES Artikel(Artikelnummer)
 );
```

b) Implementieren Sie den Bestellvorgang, bei dem eine Bestellung durch den Aufruf einer gespeicherten Prozedur (z.B. call bestellung(<kundennummer>)) ausgelöst wird. Zur besseren Lesbarkeit ist der Programmcode mit Kommentaren zu versehen.

```
CREATE OR REPLACE PROCEDURE bestellt (Knr IN INT)
AS
        AnzahlArtikel INTEGER :=0;
         anzahl INTEGER :=0;
         preis NUMERIC(5,2):=0;
         artikelnummer INTEGER :=0;
         n INTEGER :=0;
        keineArtikel EXCEPTION;
         CURSOR warenkorbcursor
         IS SELECT Artikelnummer, Anzahl, Preis
         FROM Warenkorb w Natural Join Artikel a
         WHERE Kundennummer=Knr;
BEGIN
-- Prüfung, ob der Warenkorb leer ist
         SELECT count(*)
         INTO AnzahlArtikel
         FROM Warenkorb
         WHERE Kundennummer=Knr;
         IF AnzahlArtikel=0 THEN
         RAISE keineArtikel;
         END IF;
 -- Einfügen der Bestellung
         INSERT INTO Bestellung (Bestellnummer, Bestelldatum, Kundennummer,
         Bestellstatus, Versandstatus)
         VALUES (Best_seq.nextVal,SYSDATE,Knr, 'offen','vorbereitet');
 -- Alternative ab Oracle 12c beim INSERT mit INTEGER GENERATED AS IDENTITY aus Aufgabe 2a):
 -- INSERT INTO Bestellung (Datum, Kundennummer,  Bestellstatus)
         VALUES (SYSDATE,Knr, ,'bestellt') returning into BNR;
-- Auslesen des Warenkorbs
         OPEN warenkorbcursor;
         FOR n IN 1..AnzahlArtikel LOOP
         FETCH warenkorbcursor INTO artikelnummer, anzahl, preis;
         INSERT INTO bestellposition (Bestellnummer, Pos,
        Artikelnummer, Anzahl,Preis)
         VALUES(Best_seq.currVal,n,artikelnummer,anzahl, preis);

-- Zusätzlich ein alternativer INSERT bei Oracle 12c:
-- INSERT INTO bestellposition (Bestellnummer, Position, Artikelnummer, Anzahl,Preis)
VALUES(BNR,n,artikelnummer,anzahl, preis);

 END LOOP;
 CLOSE warenkorbcursor;


 -- Entfernen aller Warenkorbeinträge zu dieser Bestellung
 DELETE FROM Warenkorb
 WHERE Kundennummer=Knr;
 commit;
```

```
EXCEPTION
WHEN keineArtikel
 THEN
 rollback;
 raise_application_error(-20500,'Fehler: keine Artikel ');
END;
```

Austesten: CALL bestellt(3);

c) Ergänzen Sie die gespeicherte Prozedur um die folgende Integritätsbedingung:
   - Eine Bestellung wird abgebrochen, wenn der Warenkorb des Kunden leer ist.

   Lösung bereits in a) enthalten

   Austesten: CALL bestellt(3);