

Datenbanken 1 Praktikum 5

Aufgabe 1 – Abfragen in Oracle

Formulieren Sie SQL-Abfragen zu der Aufgabe 1 und führen Sie diese auf der Datenbank aus.

- a) Zeigen Sie die Vornamen und Nachnamen aller weiblichen Kunden, dessen Vorname mit einem „A“ beginnt.

```
select vorname, nachname
```

```
from kunde where anrede='Frau' and Vorname Like 'A%';
```

VORNAME	NACHNAME
1 Amber	Love

- b) Zeigen Sie die Artikel die „Saft“ in ihrem Artikelnamen haben.

```
SELECT artikelname  
FROM Artikel  
WHERE UPPER ( Artikelname ) LIKE '%SAFT%';
```

BEZEICHNUNG
1 Multisaft
2 Kirschsaft

- c) Zeigen Sie die Nachnamen aller Kunden aus Maieskuel an, die älter als 35 Jahre sind.

```
SELECT Nachname  
FROM Kunde  
WHERE FLOOR ( ( SYSDATE - Geburtsdatum ) / 365 ) > 35 ;
```

NACHNAME
1 Mueller
2 Duck
3 Wille
4 Humbug
5 Kratzbaum
6 S. S. Assin
7 Metzger
8 Drach
9 Döhmer
10 Meier
11 Ratte
12 Mark

- d) Zeigen Sie die Nachnamen Kunden absteigend sortiert an, die nicht in dem Ort „Maieskuel“ wohnen und jünger als 35 Jahre sind.

```
SELECT Nachname FROM Kunde  
WHERE FLOOR ( ( SYSDATE - Geburtsdatum ) / 365 ) < 35  
MINUS
```

Datenbanken 1 Praktikum 5

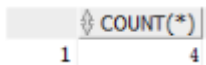
```
SELECT Nachname FROM Kunde
WHERE UPPER (ort) = 'MAIESKUDEL'
ORDER BY Nachname DESC;
```

Aufgabe 2 – SQL-Anfragen formulieren

Formulieren Sie folgende Anfragen an die Datenbank:

- a) Lassen Sie sich die Anzahl aller Kunden anzeigen, deren Nachname mit M beginnt.

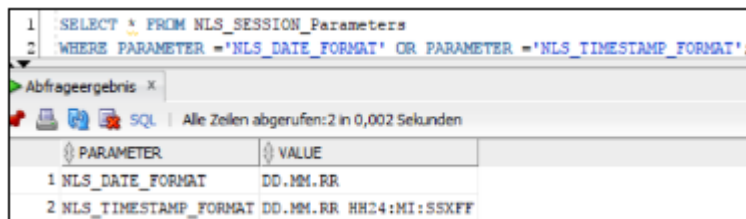
```
SELECT COUNT(*) FROM Kunde WHERE Nachname LIKE 'M%';
```



COUNT(*)
4

- b) Lassen Sie sich das eingestellte Format für Datumswerte und Zeitstempel anzeigen. Diese sind in der Tabelle NLS_SESSION_PARAMETERS hinterlegt. Recherchieren Sie, wie diese Formate geändert werden können.

```
SELECT * FROM NLS_SESSION_Parameters
WHERE PARAMETER='NLS_DATE_FORMAT'
OR PARAMETER='NLS_TIMESTAMP_FORMAT';
```



PARAMETER	VALUE
1 NLS_DATE_FORMAT	DD.MM.RR
2 NLS_TIMESTAMP_FORMAT	DD.MM.RR HH24:MI:SSXFF

Die Formate können mit dem Befehl ALTER SESSION geändert werden.

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD.MM.YYYY';
```

```
ALTER SESSION
SET NLS_TIMESTAMP_FORMAT = 'DD.MM.YYYY HH24:MI:SSXFF';
```

Das DB-Objekt SESSION beinhaltet die Parameter der Datenbankverbindung des jeweiligen Benutzers.

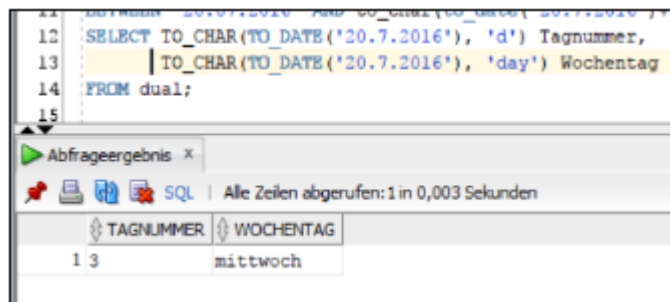
- c) Wie viele Bestellungen wurden zwischen dem 20.07.2016 und dem darauffolgenden Sonntag aufgegeben? Ermitteln Sie das Ergebnis durch eine einzelne SQL-Anfrage.
Tipp: Nutzen Sie die Oracle-Dokumentation, um passende Datumsformate zu finden.

Datenbanken 1 Praktikum 5

In der Oracle-Dokumentation finden Sie Erläuterungen zu den Datumsformaten unter:
<https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/Format-Models.html#GUID-DFB23985-2943-4C6A-96DF-DF0F664CED96>
(Stand 04.11.2022)

Der auf den 20.7.2016 ist ein Mittwoch (3. Wochentag) folgende Sonntag ist der 7. Wochentag.

```
SELECT TO_CHAR(TO_DATE('20.07.2016'), 'd') Tagnummer,  
       TO_CHAR(TO_DATE('20.07.2016'), 'day') Wochentag FROM dual;
```



Mit diesen Angaben kann das

```
SELECT COUNT(*) FROM Bestellung WHERE Bestelldatum BETWEEN '20.07.2016'  
AND TO_CHAR(TO_DATE('20.07.2016') + (7 - TO_CHAR(TO_DATE('20.07.2016'), 'd')));
```

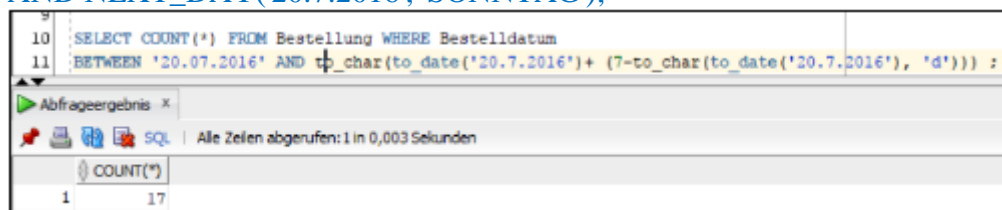
Zur Erklärung:

`TO_CHAR(TO_DATE('20.07.2016'), 'd')`

Gibt den Wochentag vom 20.7.2016 numerisch wieder. Der 20.7.2016 war ein Mittwoch, der 3. Tag einer Woche, daher gibt die Funktion den Wert 3 zurück. Subtrahiert man diese 3 von 7 (Anzahl aller Wochentage) erhält man die Anzahl der Tage zum nächsten Sonntag, in diesem Fall würde dieser Wert 4 betragen. Diese 4 Tage addieren wir anschließend zu dem aktuellen Datum und erhalten damit den Datumswert.

Alternativ:

```
SELECT COUNT(*) FROM Bestellung WHERE Bestelldatum  
BETWEEN '20.07.2016'  
AND NEXT_DAY('20.07.2016', 'SONNTAG');
```



- d) Welchen Preis besitzt der günstigste Artikel? Kontrollieren Sie Ihre Lösung anhand des Datenbestands der Tabelle Artikel.

Lösung mit Oracle 12c:

Datenbanken 1 Praktikum 5

```
SELECT * FROM Artikel  
ORDER BY Preis ASC  
FETCH FIRST 1 ROWS ONLY;
```

PREIS	BEZEICHNUNG
2,5	Pils

Lösung mit Oracle 11xe:

```
SELECT MIN(Preis) FROM Artikel
```

MIN(PREIS)
1 2.5

Um auch die Artikelbezeichnung mit anzugeben, muss eine Gruppierung erfolgen.

```
SELECT Bezeichnung, MIN(Preis) FROM Artikel  
GROUP BY Bezeichnung  
ORDER BY Min(Preis);
```

Allerdings werden dann alle gruppierten Artikel angezeigt. Um nur den günstigsten Artikel anzuzeigen, muss eine Verschachtelung erfolgen (vgl. Vorlesung Datenbankabfragen (Verschachtelte Anfragen)).