Übungen Grundlagen SQL

Klaus-Georg Deck

2025-07-16

Einleitung

In diesem Skript findest du typische SQL-Übungen mit Beispielcode und den erwarteten Ergebnistabellen. Ziel ist es, grundlegende SQL-Techniken wie Selektion, Bedingungen, Aggregation und Joins zu üben.

Voraussetzungen

Du verfügst über einen Zugang zu einem Datenbank-Server (Oracle oder Postgres), auf dem die Realisierung des Bike-Verleih-Szenarios erfolgreich installiert wurde. Eine Beschreibung, wie dies funktioniert, sowie eine Übersicht über diesen Anwendungsfall findest Du hier (Link/Referenz noch ergänzen).

Aufgabe 1: Überblick über die Tabellen `EMPLO` und `SHOP`

a) Anzahl der Datensätze

Ermittle die Anzahl der Datensätze dieser Tabellen in einer jeweils separaten SQL-Anweisung.

b) Sortieren und Limitieren

Zeige für beide Tabellen jeweils alle Datensätze an, sortiert nach dem Attribut `ID`.
Tipp: Mit `FETCH FIRST 20 ROWS ONLY` lässt sich die Anzahl beschränken.

c) Verweise

In jeder der beiden Tabellen gibt es ein Attribut, das auf das Attribut `ID` der anderen Tabelle verweist. Welche Attribute sind das jeweils und was ist vermutlich deren Bedeutung? (Für diese Aufgabe kann in beiden Tabellen das Attribut `ADDRESS` ignoriert werden.)

In der Tabelle `EMPLO` gibt es das Attribut `REPORTS_TO`. Worauf verweist dies und was ist vermutlich dessen Bedeutung?

Aufgabe 2: Selektion von Attributen und `ORDER BY`

a) Selektion von Attributen

Welche SQL-Anweisung liefert eine Liste aller Shops, wobei nur die Attribute `ID` und `NAME` angezeigt werden? Die Ausgabereihenfolge soll nach `ID` absteigend erfolgen.

b) ORDER BY mit Funktionen

Welche SQL-Anweisung liefert eine Liste aller Namen von Mitarbeitenden? Es soll nach der Länge des Namens, die längsten zuerst, und bei gleicher Länge zeichencodebasiert sortiert werden. Demnach kommt `Pauline` vor `Lea` vor `Leo`.
Wir sind nur an den Namen und nicht an der Häufigkeit ihres Vorkommens interessiert, Mehrfachvorkommen bitte ignorieren. Tipp: Die SQL-Funktion `LENGTH()` liefert die Anzahl der Zeichen einer Zeichenkette. Verwende diese in `ORDER BY`.

c) Limitieren

Ermittle die Namen und das Salär der 8 bestbezahlten Mitarbeitenden. Gehe dabei davon aus, dass es nicht mehrere Mitarbeitende gibt, die sich die 8. Position teilen.
Zusatzfrage: Wie könnte man das Resultat nach Salär aufsteigend sortiert ausgeben?

d) Berechnete Attribute

Ermittle eine Liste aller Mitarbeitenden, wobei jeweils `ID`, Name und das Jahresgehalt als `ANNUAL_SALARY` (entspricht 12 Monatsgehältern) in beliebiger Reihenfolge angezeigt werden.

Das Schlüsselwort `AS` für Spaltenaliase ist optional, erhöht jedoch die Lesbarkeit.

Aufgabe 3: Datentyp `DATE` in SQL

a) Anzahl Tage

Die Tabelle `EMPLO` beinhaltet mit `HIRE_DATE` das Einstellungsdatum von Mitarbeitenden. Erstelle eine SQL-Anweisung, mit der Mitarbeitende ausgegeben werden, die am 31.12.2025 weniger als 900 Tage im Unternehmen tätig waren. Die Ausgabe soll neben der `ID` den Namen, das Einstellungsdatum und die Anzahl der Tage im Unternehmen umfassen. Die Ausgabereihenfolge ist beliebig.
Anmerkungen:
Mit dem Ausdruck `DATE '2025-12-31'` erhält man den Wert für das gewünschte Datum. Die Differenz zwischen zwei Werten des Typs `DATE` liefert die Anzahl Tage, die zwischen diesen Kalenderdaten liegen.

b) Aktuelles Datum

Die Tabelle `RENTAL` enthält Informationen über Ausleihen von Velos, insbesondere das Ausleihdatum und Rückgabedatum. Ermittle via SQL die Ausleihen, bei denen das Velo noch nicht zurückgegeben wurde, und gebe deren Ausleihe-ID an sowie das Ausleihedatum und die Anzahl der Tage, die das Velo bis heute ausgeliehen war. Die Ausgabereihenfolge ist beliebig.
Anmerkungen:
Ausleihen, bei denen das Velo noch nicht zurückgegeben wurde, erkennt man daran, dass das Attribut `RETURN_DATE` leer (`NULL`) ist.
Um in SQL zur Laufzeit das aktuelle Datum zu ermitteln, kann man den Ausdruck `CURRENT_DATE` verwenden. Im Fall von Oracle wird `TRUNC(CURRENT_DATE)` empfohlen (Begründung siehe Lösung dieser und folgender Teilaufgabe).

c) Nullwertfunktion `COALESCE()`

Die Anweisung

```
SELECT AVG( RETURN_DATE - RENTAL_DATE ) FROM RENTAL
```

liefert die durchschnittliche Ausleihdauer in Tagen. Dabei werden noch nicht zurückgegebene Velos ignoriert. Erstelle eine modifizierte SQL-Anweisung, welche die noch nicht zurückgegebenen Velos berücksichtigt. Diese sollen in das Resultat einfliessen, indem man als Rückgabedatum das heutige Datum annimmt.
Tipp:
Verwende `CURRENT_DATE` zusammen mit der Funktion `COALESCE()`. Der Ausdruck `COALESCE(A, B)` liefert den Wert von `A`, sofern dieser nicht `NULL` ist, sonst `B`.

Aufgabe 4: Einfache Stringverarbeitung

Für die Mitarbeitenden sollen Namenskürzel eingeführt und möglichst automatisiert berechnet werden. Ein erster Versuch besteht darin, die ersten 3 Buchstaben (Kleinschreibung) des Namens zu verwenden. Die folgende Anweisung erzeugt eine Liste aller Mitarbeitenden mit `ID`, Name und Kürzel:

```
SELECT
   ID,
   NAME,
   SUBSTR( LOWER(NAME), 1, 3 ) AS KURZ
FROM EMPLO
ORDER BY NAME
```

Beachte die Verwendung der Funktionen `LOWER()` und `SUBSTR()`. Wir gehen hier davon aus, dass jeder Name mindestens 3 Zeichen enthält.

Wie man am Ergebnis erkennt, gibt es jedoch Duplikate.

ID	NAME	KURZ
16011	Anna	ann
16043	Anna	ann
16010	Ben	ben
…	…	…

Ergänze die SQL-Anweisung, so dass die folgenden Varianten für Namenskürzel als Attribute `KURZ_V1` und `KURZ_V2` ausgegeben werden:

- `KURZ_V1`: `KURZ` ergänzt um die letzten beiden Ziffern von `ID`
- `KURZ_V2`: `KURZ` ergänzt um das Einstellungsdatum in der Form `YYYYMMDD`

Die SQL-Anweisung soll also folgendes Resultat erzeugen:

ID	NAME	KURZ	KURZ_V1	KURZ_V2
16011	Anna	ann	ann11	ann20250101
16043	Anna	ann	ann43	ann20220101
16010	Ben	ben	ben10	ben20220101
…	…	…	…	…

Was ist von einer solchen Vergabe von Namenskürzel zu halten? Gibt es Alternativen?

Tipp:

Strings lassen sich mit der Funktion `CONCAT()` oder dem Operator `||` verketten. Zur Umwandlung des Datums in die gewünschte Form verwende die Funktion `TO_CHAR(HIRE_DATE, 'YYYYMMDD')` und für die Extraktion der letzten beiden Ziffern aus der `ID` kann die Funktion `MOD(ID, 100)` (Rest bei Division durch 100) eingesetzt werden, wobei das Resultat noch via `TO_CHAR()` mit Formatwert `FM00` zu formatieren ist. (Hier gibt es auch alternative Lösungswege.)

Aufgabe 5: `WHERE`-Klausel

a) Gleichheit - Ungleichheit

Erstelle eine `SELECT`-Anweisung, die alle Mitarbeitenden mit `ID`, Name und Einstellungsdatum ausgibt, die `Tim` heissen, sowei eine Anweisung, die Mitarbeitenden ausgibt, die *nicht* `Tim` heissen.

Erstelle eine weitere `SELECT`-Anweisung, die alle Informationen über diejenigen Mitarbeitenden ausgibt, deren Salär über 6'000 liegt.

In allen Fällen ist die Ausgabereihenfolge unwichtig.

Anstelle `NAME <> 'Tim'` kann man auch `NAME != 'Tim'` verwenden.

b) BETWEEN

Erstelle eine Anweisung, die alle Informationen über diejenigen Mitarbeitenden ausgibt, deren Salär zwischen 6'000 und 6'500 liegt. Dabei sollen Mitarbeitende, deren Salär direkt an der Grenze (6'000 resp. 6'500) liegt, ebenfalls berücksichtigt werden. Verwende dabei das Schlüsselwort `BETWEEN`.

Erstelle eine Anweisung ohne `BETWEEN`, die das gleiche Resultat liefert. Verwende Vergleiche und den Booleschen Operator `AND`.

In beiden Fällen sollen die Datensätze nach Salär aufsteigend sortiert werden.

Aufgabe 6: `NULL` - Das unbekannte Nichts

a) `NULL` in Vergleichen

Das Schlüsselwort `NULL` steht in SQL-Datenbankumfeld für *leer*, *unbekannt*, *kein Wert* etc. Dies tritt dann auf, wenn etwa bei optionalen Angaben zu Geschlecht, Telefonnummer oder Anzahl der Kinder nichts angegeben wird. So macht es einen Unterschied, ob jemand bei *Anzahl Kinder* den Zahlenwert `0` angibt oder keine Angabe macht. Im ersten Fall steht fest, dass die Person keine Kinder hat; im zweiten Fall ist unklar, ob sie Kinder hat – und wenn ja, wie viele.

In der Tabelle `EMPLO` ist das Attribut `REPORTS_TO` optional, d.h. es kann Datensätze geben, bei denen dieses Attribut leer ist.

Die folgende Anweisung zeigt, dass es genau einen solchen gibt:

```
SELECT ID, NAME, JOB, REPORTS_TO FROM EMPLO WHERE REPORTS_TO IS NULL
```

ID	NAME	JOB	REPORTS_TO
16051	Nole	CEO	

Hierbei handelt es sich um den oder die CEO des Bike-Verleih, was erklärt, warum `REPORTS_TO` leer ist. Denn bekanntermassen haben CEOs keine direkten Vorgesetzten.

Gib mit einer SQL-Anweisung die Anzahl der Mitarbeitenden mit direkten Vorgesetzten (als Attribut `W_MGR`) und in einer weiteren Anweisung die Anzahl der Mitarbeitenden ohne direkten Vorgesetzten (als Attribut `WO_MGR`) aus.

Versuche in einer dritten SQL-Anweisung, die Gesamtzahl der Mitarbeitenden, die Zahl der Mitarbeitenden mit sowie die Zahl der Mitarbeitenden ohne direkten Vorgesetzten auszugeben. Tipp: Verwende auch `COUNT( REPORTS_TO )`.

Aufgabe 7: JOIN - Verbinden von Tabellen

a) INNER JOIN - Verschiedene Varianten

Gib für jeden Mitarbeitenden die `ID`, den Namen als `ENAME` sowie den Namen des Shops als `SNAME` aus, dem er oder sie zugeordnet ist. Die Reihenfolge der Ausgabe soll nach der `ID` der Mitarbeitenden aufsteigend erfolgen.

Erstelle zwei SQL-Anweisungen in unterschiedlicher JOIN-Syntax, die dieses Ergebnis liefern.

b) JOIN - Attribute mit und ohne Tabellen-Spezifikation

Wir betrachten hier die Tabellen `EMPLO` und `ADDRESS`, die über `EMPLO.ADDRESS` und `ADDRESS.ID` inhaltlich verbunden sind.

Welche Mitarbeitenden wohnen im Kanton (`STATE`) Wallis (`VS`), aber nicht in der Stadt (`CITY`) `Sion`? Gib die Namen der Mitarbeitenden und ihren Wohnort (`CITY`) aus, sortiert nach Mitarbeitendenname.

Erstelle eine erste Anweisung, bei der jedes Attribut den voll qualifizierten Namen enthält, wie z.B. `EMPLO.NAME`.

In der zweiten Anweisung soll auf die Angabe von Tabellennamen nach Möglichkeit verzichtet werden. In welchen Fällen ist dies möglich, in welchen nicht?

Folgendes Resultat wird erwartet:

NAME	CITY
Corinne	Visp
Melanie	Visp
Pascal	Visp
Silvan	Zermatt

Aufgabe 8: JOIN - Komplexeres Beispiel

Im Bike-Verleih-Szenario können Kunden verschiedene Bikes ausleihen. Die beteiligten Tabellen sind `CUSTOMER`, `RENTAL` und `BIKE`. Wir interessieren uns für Kunden, die vor dem 15. Januar 2025 (mindestens einmal) ein Bike mit Rahmengrösse XL geliehen haben, das kein E-Bike ist.

a) JOIN mehrerer Tabellen

Formuliere eine SQL-Anweisung, mit der die `ID` der Ausleihen, `ID` der Kunden, Name der Kunden und Modell des ausgeliehenen Bikes ausgegeben werden kann, die der genannten Bedingung entsprechen. Die Ausgabe soll nach `ID` der Ausleihen sortiert sein. Das Resultat hat die folgende Form (es wird nur die erste Datenzeile angezeigt):

RID	KUNDE_ID	KUNDE_NAME	MODELL
60004	50151	Tom	CityLite
...	...	...	...

b) Eindeutigkeit mit COUNT(DISTINCT)

Verwende das Resultat der vorherigen Teilaufgabe und formuliere eine SQL-Anweisung, mit der die Anzahl der Ausleihen (als `ANZ_RT`), die Anzahl der Kunden (`ANZ_C`), die Anzahl der unterschiedlichen Namen von Kunden (`ANZ_NAME`) und die Anzahl der unterschiedlichen Bike-Modelle (`ANZ_MODELL`) ausgegeben werden. Erwartet wird das folgende Resultat:

ANZ_RT	ANZ_C	ANZ_NAME	ANZ_MODELL
8	7	6	7

Tipp: Verwende Ausdrücke wie `COUNT(DISTINCT R.ID)`, um die Anzahl unterschiedlicher Werte zu ermitteln.