Übungen Advanced SQL

Klaus-Georg Deck

2025-07-24

Einleitung

In diesem Skript findest du typische SQL-Übungen mit Beispielcode und den erwarteten Ergebnistabellen. Ziel ist es, fortgeschrittene SQL-Techniken wie Outer Join, Group-by, Aggregationen und einfache Subselects zu erlernen und vertiefen.

Voraussetzungen

Du verfügst über einen Zugang zu einem Datenbank-Server (Oracle oder Postgres) mit den Tabellen des Bike-Verleih-Szenarios und hast Grundkenntnisse in SQL, etwa indem Du die vorherigen Übungen erfolgreich absolviert hast.

Aufgabe 1: Verknüpfung von Tabellen

a) Einfache OUTER-Join Anweisung

Für diesen Übungsteil benötigst Du die Tabellen `EMPLO` und `SHOP`. Gib eine SQL-Anweisung an, die den Namen der Mitarbeitenden zusammen mit dem Namen des Shops, dem sie zugeordnet sind, anzeigt. Es sollen alle Mitarbeitenden angezeigt werden, auch jene, die keinem Shop angehören. In diesem Fall soll der Wert des Shop-Namens leer sein. Sortiere die Datensätze nach `ID` aus `EMPLO` absteigend.

Die Resultatmenge hat folgende Gestalt:

ID	ENAME	SNAME
16055	Leon	
16051	Bea	
16050	Felix	Vaduz CyclePoint
16049	Anna	Bern Rollt
…	…	…

b) Vergleich zwischen OUTER- und INNER-Join

Ersetzst Du in der Lösung der vorherigen Teilaufgabe `LEFT OUTER JOIN` durch `JOIN`, dann sind die Resultate unterschiedlich. Zeige dies, indem Du in beiden Fällen die Anzahl der Datensätze ermittelst. Was kann man grundsätzlich über die Anzahl der Datensätze in solchen Fällen aussagen? Welche Datensätze kommen in welchem Resultat nicht vor?

Verwende anschliessend als Verknüpfung `FULL OUTER JOIN`. Wie viele Datensätze erhält man dann?

Verwende einen Teil der Lösung der vorherigen Teilaufgabe:

```
SELECT E.ID, E.NAME ENAME, S.NAME SNAME
FROM EMPLO E
LEFT OUTER JOIN SHOP S ON E.SHOP = S.ID
```

c) Join: INNER, LEFT, RIGHT, FULL und OUTER

Begründe, warum ein `LEFT INNER`- oder `RIGHT INNER`-Join keinen Sinn macht und warum bei `LEFT OUTER`-, `RIGHT OUTER`- und `FULL OUTER`-Join das Schlüsselwort `OUTER` weggelassen werden kann. Erstelle eine Übersichtstabelle, in der die jeweilige Lang- und Kurzform der verschiedenen Joins aufgelistet sind.

Aufgabe 2: Verknüpfungen und `EXISTS`

Für diese Aufgabe benötigst Du die Tabellen `CUSTOMER`, `RENTAL` und `BIKE`, die Informationen über Bike-Ausleihen von Kunden enthalten.

a) Einfache Verknüpfung

Welche Kunden haben am 2. Januar 2025 oder früher Bikes ausgeliehen? Gib die Ausleih-ID, den Namen des Kunden und das Ausleihdatum, sortiert nach Ausleih-ID an. Das erwartete Resultat:

R_ID	NAME	DATUM
60023	Emma	2025-01-01
60046	Lena	2025-01-02
60078	Francesco	2025-01-01

b) Kunden mit und ohne Ausleihen

Welche Kunden haben bisher (irgendwann einmal) Bikes ausgeliehen?

Welche Kunden haben bisher noch keine Bikes ausgeliehen?

Gib jeweils die Kunden-ID und den Namen der Person an, aufsteigend sortiert nach Kunden-ID.

Tipp:
Verwende für den ersten Teil die vorherige Teilaufgabe. Beachte, dass Kunden nicht mehrfach genannt werden sollen. Für den zweiten Teil kann man einen `LEFT JOIN` verwenden und diejenigen Datensätze aus dem Resultat herausfiltern, für die die `ON`-Bedingung nicht zutrifft. (Alternativ kann man auch einen Subselect mit der SQL-Klausel `EXISTS` resp. `NOT EXISTS` verwenden.)

c) Anzahl der Kunden

Wie viele Kunden gibt es insgesamt, wie viele haben ein Bike des Modells ‘Pathway’ ausgeliehen und wie viele haben kein solches Modell ausgeliehen.

Aufgabe 3: Gruppierung und Aggregatsfunktionen

a) Average: Gehalt und Tage im Unternehmen

Gib eine SQL-Anweisung an, die das Durchschnitts-Gehalt und die Durchschnitts-Zeit im Unternehmen aller Mitarbeitenden ausgibt. Das Gehalt soll gerundet auf ganze Werte, die Zeit im

Unternehmen gerundet als ganze Tage ausgegeben werden.
Zusätzlich soll angegeben werden, wie viele Mitarbeitenden es insgesamt gibt und wie viele jeweils in die Gehalts- resp. Zeit-Berechnung einfliessen. Beachte, dass fehlende Werte (`NULL`) bei den meisten Aggregatsfunktionen ignoriert werden.

Das Ergebnis sieht wie folgt aus:

CT_ALL	AVG_SAL	CT_SAL	AVG_DAYS	CT_DAYS
24	5'457	23	1'119	21

b) Average: Gehalt und Tage im Unternehmen pro Shop

Ermittle die gleichen Kennzahlen wie in der vorherigen Teilaufgabe, jedoch nicht für das gesamte Unternehmen, sondern pro Shop, dem die Mitarbeitenden zugeordnet sind. Gib dazu zusätzlich die Shop-ID aus und sortiere die Ausgabe nach der Shop-ID.
Das Resultat hat die folgende Form:

SHOP_ID	CT_ALL	AVG_SAL	CT_SAL	AVG_DAYS	CT_DAYS
10	4	5'157	4	1'300	3
12	4	5'425	4	935	4
...	...	...	...	...	...

c) Average: Mehr Gehalt als der Durchschnitt

Gebe eine SQL-Anweisung an, die ermittelt, wer mehr als der Durchschnitt verdient. Von den Mitarbeitenden soll die `ID`, der Name, das Salär, die Shop-ID ausgegeben werden, sortiert nach `ID`.
Gebe in einer weiteren SQL-Anweisung zusätzlich in jedem Datensatz das Durchschnittsgehalt (gerundet) aus.

d) Average: Mehr Gehalt als der Durchschnitt pro Shop

Formuliere SQL-Anweisungen, mit denen analog zur vorherigen Teilaufgabe alle Mitarbeitenden ausgegeben werden, die *mindestens so viel* wie der Durchschnitt der Mitarbeitenden des gleichen Shops erhalten. Die auszugebenden Attribute sind die gleichen, nur dass sich das Durchschnittsgehalt jetzt auf den Shop bezieht.

Aufgabe 4: Mehrere GROUP-BY Attribute

Diese Aufgabe befasst sich mit den Bike-Ausleihen aus der Tabelle `RENTAL`. Beachte, dass Bikes mehrmals ausgeliehen werden können. Beantworte mittels SQL-Anweisungen die folgenden Fragen:

- Wie viele Ausleihen gab es pro Jahr und Monat und wie viele Bikes wurden pro Jahr und Monat ausgeliehen?
- Wie viele Ausleihen gab es pro Jahr, Monat und Shop, und wie viele Bikes wurden pro Jahr, Monat und Shop ausgeliehen?

Das Ergebnis soll jeweils die beteiligten Gruppierungsattribute sowie die Anzahl Bikes enthalten. Die Ausgabereihenfolge erfolgt nach den Gruppierungsattributen.

Das erwartete Resultat sieht wie folgt aus:

RENTAL_YEAR	RENTAL_MONTH	CT_RENTALS	CT_BIKES
2025	1	40	27
2025	2	31	25
...	...	...	...

RENTAL_YEAR	RENTAL_MONTH	SHOP	CT_RENTALS	CT_BIKES
2025	1	10	2	2
2025	1	12	13	8
...	...	...	...	...

Tipps: Mit `EXTRACT( YEAR FROM RENTAL_DATE )` kann man das Jahr aus einem Datumswert ermitteln, analog auch den Monat.
Der Shop einer Ausleihe lässt sich ermitteln, indem man beachtet, welchem Shop das ausgeliehene Bike zugeordnet ist.

Aufgabe 5: GROUP-BY und HAVING

Welche Bikes wurden vor dem 4. März 2025 mindestens 3 mal ausgeliehen? Formuliere ein SQL-Anweisung, mit der die Bike-ID, Modell und die Häufigkeit der Ausleihe für dieses Bike ausgegeben werden. Die Reihenfolge soll nach Bike-ID erfolgen.

Aufgabe 6: Subselects

Beantworte via SQL-Anweisungen die folgenden Fragen:

- An welchem Tag wurden erstmals Bikes ausgeliehen? Gib von allen Ausleihen, die an diesem Tag getätigt wurden, die ID, die Bike-ID und das Datum aus.
- Welche Kunden haben die meisten Ausleihen getätigt? Gib alle diese Kunden (Customer-ID) aus, zusammen mit der Anzahl der Ausleihen.
- Welche Kunden haben die meisten Bikes ausgeliehen? Gib alle diese Kunden (Customer-ID) aus, zusammen mit der Anzahl entliehener Bikes.