SQL in der Praxis am Beispiel der Deutschen Marine

Projektszenario

Sie sind im S1-Bereich der Einsatzflottille 2 eingesetzt und verwalten die Personaldatenbank der Marine. Es sollen Abrechnungen für Wachdienste auf den Fregatten (Klassen `F123`, `F124`, `F125`) sowie dem Einsatzgruppenversorger der `EGV 702` erstellt werden. Ein Soldat erhält für einen Wachdienst einen Stundenlohn (basierend auf dem Dienstgrad) und ggf. einen Risikozuschlag (z.B. bei Gefechtsübungen).

Aufgabe 1: Besatzungsliste und Berechtigungen

Ausgangslage: Der Kommandant der `'FGS Brandenburg'` benötigt eine kompakte Liste seiner Mannschaft, aber nur von Soldaten mit dem Dienstgrad `'Hauptgefreiter'`. Zusätzlich soll diese Abfrage und der Zugriff darauf für die Zukunft fest eingerichtet werden.

Aufgabe: Erstellen Sie eine **VIEW**. Nutzen Sie **CONCAT** (oder `||`), um Dienstgrad und Name zusammenzufügen. Filtern Sie mittels **WHERE** nach Schiff und Dienstgrad. Richten Sie anschließend die **Security** ein: Erstellen Sie ein Schema, Rollen und Benutzer und vergeben Sie die Rechte mittels **GRANT**.

Ziel:

1. Erstellen der View `v_besatzungliste_hg`.
2. Spalten: `"Dienstgrad (kurz) Name"` als `Soldat` und `"Schiffsname (Geschwadername)"` als `Einheit`.
3. Rolle `r_fuehrung_vollzugriff` (für u_kommandant & u_io) und `r_fuehrung_lesen` (für u_wachtmeister).

```sql
DROP VIEW IF EXISTS v_besatzungliste_hg;
DROP ROLE IF EXISTS r_fuehrung_lesen;
DROP ROLE IF EXISTS r_fuehrung_vollzugriff;
DROP ROLE IF EXISTS u_wachtmeister;
DROP ROLE IF EXISTS u_kommandant;
DROP ROLE IF EXISTS u_io;

CREATE ROLE r_fuehrung_lesen;
CREATE ROLE r_fuehrung_vollzugriff;
CREATE USER u_wachtmeister WITH PASSWORD 'password';
CREATE USER u_kommandant WITH PASSWORD 'deralte';
CREATE USER u_io WITH PASSWORD 'chef';

GRANT r_fuehrung_lesen TO u_wachtmeister, u_refue;
GRANT r_fuehrung_vollzugriff TO u_kommandant, u_io;

CREATE OR REPLACE VIEW v_besatzungliste_hg AS
SELECT concat(d.dg_kurz, ' ', s.name)             AS "Soldat",
       concat(sch.schiffsname, ' (', g.name, ')') AS "Einheit"
FROM soldat s
         INNER JOIN dienstgrad d ON s.dienstgradid = d.dienstgradid
         INNER JOIN schiff sch ON s.stammschiffid = sch.schiffid
         INNER JOIN geschwader g ON s.geschwaderid = g.geschwaderid
WHERE sch.schiffsname = 'FGS Brandenburg'
  AND d.dg_lang = 'Hauptgefreiter';

GRANT SELECT ON v_besatzungliste_hg TO r_fuehrung_lesen;
GRANT ALL PRIVILEGES ON v_besatzungliste_hg TO r_fuehrung_vollzugriff;

SELECT *
FROM v_besatzungliste_hg;
```

Erwartetes Ergebnis:

Soldat	Einheit
HptGefr Schulz	FGS Brandenburg (2. Fregattengeschwader)
HptGefr Becker	FGS Brandenburg (2. Fregattengeschwader)

Aufgabe 2: Kostenanalyse nach Klassen

Ausgangslage: Der Rechnungsführer muss die Wachen abrechnen und benötigt die Gesamtkosten aller Wachdienste, aufgeschlüsselt nach den Schiffstypen.

Aufgabe: Berechnen Sie die **SUMME** der Kosten mit der *Formel:* `RisikoZuschlag + (Stunden * Stundensatz)`.

Ziel: 0002 **Hinweis zur Berechnung:**

```sql
EXTRACT(EPOCH FROM (EndeZeit - StartZeit)) / 3600 -- Ergibt die Stunden
```

```sql
SELECT sch.klasse,
       sum(w.risikozuschlag + ((extract(EPOCH FROM (w.endezeit - w.startzeit)) / 3600) *
d.stundensatz)) AS gesamtkosten
FROM wachdienst w
         INNER JOIN schiff sch ON w.schiffid = sch.schiffid
         INNER JOIN soldat s ON w.soldatid = s.persnr
         INNER JOIN dienstgrad d ON s.dienstgradid = d.dienstgradid
GROUP BY sch.klasse;
```

Erwartetes Ergebnis:

klasse	gesamtkosten
702	1298.1
F123	461.1
F124	757.8
F125	656.8

Aufgabe 3: Spezifische Belastungsprüfung

Ausgangslage: Es gibt Beschwerden über zu hohe Belastung der Hauptgefreiten auf den Fregatten der Klasse F124. Dies soll überprüft werden.

Aufgabe: Filtern Sie mittels **WHERE** nach der Klasse `'F124'` **UND** dem Dienstgrad `'Hauptgefreiter'`. Summieren (**SUM**) Sie anschließend die Stunden.

Ziel: Zeigen Sie die `Schiffsklasse` und die `Summe der Stunden` an.

```sql
SELECT sum(extract(EPOCH FROM (endezeit - startzeit)) / 3600)::NUMERIC(50, 1) AS
stunden_hg_auf_f124,
       count(extract(EPOCH FROM (endezeit - startzeit)) / 3600) AS anzahl_soldat,
       klasse
FROM wachdienst
         INNER JOIN soldat ON soldat.persnr = wachdienst.soldatid
         INNER JOIN dienstgrad on soldat.dienstgradid = dienstgrad.dienstgradid
         INNER JOIN schiff ON soldat.stammschiffid = schiff.schiffid
WHERE dg_kurz = 'HptGefr'
  AND klasse = 'F124'
GROUP BY klasse;
```

Erwartetes Ergebnis:

klasse	stunden_hg_auf_f124	anzahl_soldat
F124	8.0	1

Aufgabe 4: Identifikation von Dauerläufern

Ausgangslage: Um Übermüdung zu vermeiden, sollen Soldaten identifiziert werden, die insgesamt sehr viel Wache geschoben haben.

Aufgabe: Berechnen Sie die Summe der Stunden der Soldaten. Filtern Sie das **Ergebnis der Berechnung**, um nur Soldaten mit mehr als `5 Stunden` anzuzeigen.

Ziel: Zeigen Sie `Name` des Soldaten verbunden mit `Dienstgrad (kurz)` und die `Gesamtstunden` an. Sortieren Sie nach `Gesamtstunden` absteigend.

```
SELECT concat(d.dg_kurz, ' ', s.name)                              AS "Soldat",
       sum(extract(EPOCH FROM (w.endezeit - w.startzeit)) / 3600) AS gesamtstunden
FROM wachdienst w
         INNER JOIN soldat s ON w.soldatid = s.persnr
         INNER JOIN dienstgrad d ON s.dienstgradid = d.dienstgradid
GROUP BY s.name, d.dg_kurz
HAVING sum(extract(EPOCH FROM (w.endezeit - w.startzeit)) / 3600) > 5
ORDER BY gesamtstunden;
```

Erwartetes Ergebnis:

Soldat	gesamtstunden
Btsm Christ	10.0
HptGefr Ebert	10.0
Matr Bauer	9.0
Korp Arnold	8.0
Gefr Reimann	8.0
HptGefr Roth	8.0
HptBtsm Schreiber	6.0

Aufgabe 5: Übersicht Gefahrenzulagen

Ausgangslage: Für den Rechnungsführer soll eine permanente Übersicht aller Wachen erstellt werden, für die ein Risikozuschlag gezahlt wird.

Aufgabe: Erstellen Sie eine **VIEW** namens `V_Gefahrenzulage`. Filtern Sie mittels **WHERE**, sodass nur Wachen mit `Risikozuschlag > 0` enthalten sind. Sortieren Sie mittels **ORDER BY** aufsteigend nach dem Zuschlag. Da es sich hierbei um Besoldungsangelegenheiten handelt, soll die View in einem separaten Schema abgelegt werden. Richten Sie anschließend die **Security** ein: Erstellen Sie ein Schema, Rollen und Benutzer und vergeben Sie die Rechte mittels **GRANT**.

Ziel:

1. Erstellen der View `V_Gefahrenzulage`.
2. Spalten: `Name`, `PersNr`, `Schiffsname`, `Risikozuschlag`.
3. Schema: `besoldung`.

4. Rolle `r_refue_vollzugriff` (für u_refue) und `r_refue_lesen` (für u_svo & u_refue_uffz).

```
DROP VIEW IF EXISTS besoldung.v_gefahrenzulage;
DROP SCHEMA IF EXISTS besoldung;
CREATE SCHEMA besoldung;

DROP ROLE IF EXISTS r_refue_lesen;
DROP ROLE IF EXISTS r_refue_vollzugriff;
DROP ROLE IF EXISTS u_svo;
DROP ROLE IF EXISTS u_refue;
DROP ROLE IF EXISTS u_refue_uffz;

CREATE ROLE r_refue_lesen;
CREATE ROLE r_refue_vollzugriff;
CREATE USER u_svo WITH PASSWORD 'password';
CREATE USER u_refue WITH PASSWORD 'password';
CREATE USER u_refue_uffz WITH PASSWORD 'password';

GRANT r_refue_lesen TO u_svo, u_refue_uffz;
GRANT r_refue_vollzugriff TO u_refue;
GRANT USAGE ON SCHEMA besoldung TO r_refue_lesen; -- Berechtigung zum Zugriff aufs Schema,
GRANT USAGE ON SCHEMA besoldung TO r_refue_vollzugriff; -- Berechtigung zum Zugriff aufs
Schema,
-- REVOKE ALL ON SCHEMA besoldung FROM PUBLIC; -- alle anderen ausschließen
-- GRANT ALL PRIVILEGES ON SCHEMA besoldung TO r_fuehrung_vollzugriff; -- bekommt USAGE
und CREATE Rechte aufs Schema
-- GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA besoldung TO r_fuehrung_vollzugriff; --
bekommt Zugriff auf alle Tabellen

CREATE OR REPLACE VIEW besoldung.v_gefahrenzulage AS
SELECT s.name, s.persnr, sch.schiffsname, w.risikozuschlag
FROM wachdienst w
         INNER JOIN soldat s ON w.soldatid = s.persnr
         INNER JOIN schiff sch ON w.schiffid = sch.schiffid
WHERE w.risikozuschlag > 0
ORDER BY w.risikozuschlag ASC;

GRANT ALL PRIVILEGES ON besoldung.v_kosten_wache TO r_refue_vollzugriff;
GRANT SELECT ON besoldung.v_kosten_wache TO r_refue_lesen;

SELECT *
FROM besoldung.v_gefahrenzulage;
```

Erwartetes Ergebnis:

name	persnr	schiffsname	risikozuschlag
Bauer	10012	FGS Brandenburg	10.0
Bauer	10012	FGS Brandenburg	10.0
Ebert	20206	FGS Bonn	10.0
Vogel	20109	FGS Frankfurt am Main	15.0
Witt	20110	FGS Frankfurt am Main	15.0
Schreiber	10124	FGS Hessen	15.0
Noll	20101	FGS Frankfurt am Main	20.0
Kuntz	10166	FGS Nordrhein-Westfalen	20.0
Reuter	10168	FGS Nordrhein-Westfalen	20.0
Wolf	10022	FGS Bayern	25.0
Graf	10107	FGS Hamburg	30.0
Reimann	10211	FGS Rheinland-Pfalz	40.0
Brand	20002	FGS Berlin	45.0
Beck	10089	FGS Sachsen	50.0
Roth	10087	FGS Sachsen	50.0
Decker	20004	FGS Berlin	60.0
Vogel	10081	FGS Sachsen	100.0

Ohne Doppelung von Bauer

```sql
SELECT s.name, s.persnr, s.name, sum(w.risikozuschlag) AS gesamtsumme
FROM wachdienst w
         INNER JOIN soldat s ON w.soldatid = s.persnr
WHERE w.risikozuschlag > 0
GROUP BY s.name;
```

Erwartetes Ergebnis:

name	persnr	schiffsname	risikozuschlag
Bauer	10012	FGS Brandenburg	20.0
Ebert	20206	FGS Bonn	10.0
Vogel	20109	FGS Frankfurt am Main	15.0
Witt	20110	FGS Frankfurt am Main	15.0
Schreiber	10124	FGS Hessen	15.0
Noll	20101	FGS Frankfurt am Main	20.0
Kuntz	10166	FGS Nordrhein-Westfalen	20.0
Reuter	10168	FGS Nordrhein-Westfalen	20.0
Wolf	10022	FGS Bayern	25.0
Graf	10107	FGS Hamburg	30.0
Reimann	10211	FGS Rheinland-Pfalz	40.0
Brand	20002	FGS Berlin	45.0
Beck	10089	FGS Sachsen	50.0
Roth	10087	FGS Sachsen	50.0
Decker	20004	FGS Berlin	60.0
Vogel	10081	FGS Sachsen	100.0

Aufgabe 6: Der fleißigste Soldat

Ausgangslage: Der Kommandeur der Einsatzflottille 2 möchte den Soldaten auszeichnen, der (über alle Schiffe hinweg) die meisten Wachen geleistet hat.

Aufgabe: Zählen (**COUNT**) Sie die Wachdienste pro Soldat. Sortieren Sie das Ergebnis absteigend (**DESC**) und begrenzen Sie die Ausgabe auf den ersten Eintrag.

Ziel: Zeigen Sie den `Namen` und die `Anzahl` der Wachen an (nur Platz 1).

```sql
SELECT concat(d.dg_kurz, ' ', s.name) AS "Soldat", count(w.wachid) AS anzahl
FROM soldat s
         INNER JOIN dienstgrad d ON s.dienstgradid = d.dienstgradid
         INNER JOIN wachdienst w ON s.persnr = w.soldatid
GROUP BY s.name, d.dg_kurz
ORDER BY anzahl DESC
LIMIT 1;
```

Erwartetes Ergebnis:

name	anzahl
Matr Bauer	2

Aufgabe 7: Soldaten ohne Wache

Ausgangslage: Es fällt auf, dass manche Soldaten in der Personalliste stehen, aber noch gar nicht im Wachplan auftauchen. Diese "Karteileichen" sollen gefunden werden.

Aufgabe: Verbinden Sie Soldat und Wachdienst und prüfen Sie wo keine Verknüpfung zum Wachdienst besteht. Sortieren Sie alphabetisch nach Namen.

Ziel: Zeigen Sie `Dienstgrad (kurz)`, `Namen` und `Einheit` an.

```
SELECT concat(d.dg_kurz, ' ', s.name) AS "Soldat", sch.schiffsname AS "Einheit"
FROM soldat s
         LEFT JOIN wachdienst w ON s.persnr = w.soldatid
         INNER JOIN dienstgrad d ON s.dienstgradid = d.dienstgradid
         INNER JOIN schiff sch ON s.stammschiffid = sch.schiffid
WHERE w.wachid IS NULL
ORDER BY s.name ASC;
```

Erwartetes Ergebnis:

Soldat	Einheit
KptLt Acker	FGS Bonn
OBtsm Adler	FGS Rheinland-Pfalz
FKpt Albrecht	FGS Hamburg
FKpt Arndt	FGS Berlin
Gefr Arnold	FGS Hamburg
OBtsm Baier	FGS Bonn
Gefr Baumann	FGS Sachsen
HptGefr Becker	FGS Brandenburg
OBtsm Berger	FGS Sachsen
OBtsm Beyer	FGS Nordrhein-Westfalen
...	...

Aufgabe 8: Zeitraumanalyse (Sturm)

Ausgangslage: Wegen eines Sturms in der ersten Novemberwoche sollen alle Wachen geprüft werden, die in diesem Zeitraum begonnen haben.

Aufgabe: Filtern Sie das `Startdatum` mittels **BETWEEN** (oder `>=` und `<=`) für den Zeitraum `01.11.2023` bis `07.11.2023`. Sortieren Sie chronologisch (**ORDER BY**).

Ziel: Zeigen Sie `Schiffsname`, `Vorname`, `Name` und `Startzeit` an.

```sql
SELECT sch.schiffsname, s.vorname, s.name, w.startzeit
FROM wachdienst w
         INNER JOIN schiff sch ON w.schiffid = sch.schiffid
         INNER JOIN soldat s ON w.soldatid = s.persnr
WHERE w.startzeit BETWEEN '2023-11-01 00:00:00' AND '2023-11-07 23:59:59'
ORDER BY w.startzeit ASC;
```

Erwartetes Ergebnis:

schiffsname	name	startzeit
FGS Brandenburg	Jan	Schulz
FGS Brandenburg	Nils	Bauer
FGS Bayern	Christian	Wolf
FGS Brandenburg	Nils	Bauer
FGS Bayern	Max	Maier
FGS Sachsen	Oliver	Roth
FGS Sachsen	Torsten	Vogel
FGS Sachsen	Jonas	Beck
FGS Hamburg	Moritz	Graf
FGS Hessen	Mario	Schreiber

Aufgabe 9: Geschwader-Vergleich

Ausgangslage: Der Admiral der Einsatzflottille 2 möchte die Arbeitsbelastung zwischen den Geschwadern vergleichen (2. FG vs. 4. FG vs. Tross).

Aufgabe: Verknüpfen Sie die Tabellen über mehrere Ebenen (**JOIN** über 4 Tabellen: Wache -> Schiff -> Geschwader). Summieren Sie die Stunden auf und zeigen diese nach Geschwader an.

Ziel: Liste mit `Name des Geschwaders` und `Summe der Wachstunden`.

```sql
SELECT g.name                                                    AS geschwader,
       sum(extract(EPOCH FROM (w.endezeit - w.startzeit)) / 3600) AS gesamtstunden
FROM wachdienst w
         INNER JOIN schiff s ON w.schiffid = s.schiffid
         INNER JOIN geschwader g ON s.geschwaderid = g.geschwaderid
GROUP BY g.name
ORDER BY gesamtstunden DESC;
```

Erwartetes Ergebnis:

geschwader	gesamtstunden
Trossgeschwader	51.0
2. Fregattengeschwader	45.0
4. Fregattengeschwader	28.0

Aufgabe 10: Überdurchschnittliche Wachen

Ausgangslage: Der Flottenarzt möchte wissen, welche einzelnen Wachdienste länger als der Durchschnitt aller jemals geleisteten Wachen waren, um extreme Belastungsspitzen zu erkennen.

Aufgabe: Nutzen Sie eine **Sub-Query** (Unterabfrage) in der **WHERE**-Klausel. Berechnen Sie in der Sub-Query den Durchschnitt (**AVG**) aller Wachen und vergleichen Sie die einzelne Dauer damit (`>`).

Ziel: Zeigen Sie `WachID` , `Name` und `Dauer (Std)` an.

```
SELECT w.wachid,
       s.name,
       (extract(EPOCH FROM (w.endezeit - w.startzeit)) / 3600) AS dauer_std
FROM wachdienst w
         JOIN soldat s ON w.soldatid = s.persnr
WHERE (extract(EPOCH FROM (w.endezeit - w.startzeit)) / 3600) >
      (SELECT avg(extract(EPOCH FROM (endezeit - startzeit)) / 3600)
       FROM wachdienst);
```

Erwartetes Ergebnis:

wachid	name	dauer_std
3	Matr Bauer	5.0
7	HptGefr Roth	8.0
10	HptBtsm Schreiber	6.0
11	Korp Arnold	8.0
16	Gefr Reimann	8.0
107	Btsm Christ	10.0
109	HG Ebert	10.0

Aufgabe 11: Generierung Taktischer IDs

Ausgangslage: Für den verschlüsselten Funkverkehr soll eine „Taktische ID“ für jeden Soldaten generiert werden.

Aufgabe: Nutzen Sie String-Funktionen, um die ersten 3 Buchstaben des Namens großzuschreiben, und verbinden Sie diese mit der Personalnummer.

Ziel: Anzeige von `Name` , `PersNr` und der generierten `taktische_id` . *Format:* `NAM-12345` *Beispiel: Aus 'Schmidt' mit Pers-Nr 10001 wird 'SCH-10001'*

```sql
SELECT name,
       persnr,
       upper(substring(name, 1, 3)) || '-' || persnr AS taktische_id
FROM soldat;
```

Erwartetes Ergebnis:

name	persnr	taktische_id
Just	20211	JUS-20211
Iltis	20210	ILT-20210
Horn	20209	HOR-20209
Gross	20208	GRO-20208
Funk	20207	FUN-20207
Ebert	20206	EBE-20206

Aufgabe 12: Die "teuersten" Dienstgrade

Ausgangslage: Es soll analysiert werden, welche Dienstgrade überdurchschnittlich gut bezahlt werden (höherer Stundensatz als der Durchschnitt aller Sätze).

Aufgabe: Nutzen Sie eine **Sub-Query** in der **WHERE**-Klausel, um den Durchschnitt (**AVG**) der Stundensätze zu ermitteln und zu vergleichen. Sortieren Sie absteigend.

Ziel: Anzeige von `Dienstgrad (lang)` und `Stundensatz`.

```sql
SELECT dg_lang,
       stundensatz
FROM dienstgrad
WHERE stundensatz > (SELECT avg(stundensatz)
                     FROM dienstgrad)
ORDER BY stundensatz DESC;
```

Erwartetes Ergebnis:

dg_lang	stundensatz
Admiral	115.0
Vizeadmiral	98.0
Konteradmiral	85.0
Flottillenadmiral	75.0
Kapitaen zur See	60.0
Fregattenkapitaen	52.0
Korvettenkapitaen	45.0
Stabskapitaenleutnant	42.5
Kapitaenleutnant	39.0

Aufgabe 13: Statistik-Report

Ausgangslage: Der Stab der EF2 wünscht eine kompakte statistische Übersicht über die gezahlten Risikozuschläge, um Ausreißer nach oben oder unten zu bewerten.

Aufgabe: Nutzen Sie Aggregatfunktionen (**MIN**, **MAX**, **AVG**) in einer einzigen Abfrage. Runden Sie den Durchschnitt mit **ROUND** auf 2 Nachkommastellen. *Wichtig:* Betrachten Sie nur Wachen mit Zuschlag `> 0`.

Ziel: Eine Zeile mit `Minimum`, `Maximum` und `Durchschnitt`.

```
SELECT min(risikozuschlag)           AS minimum,
       max(risikozuschlag)           AS maximum,
       round(avg(risikozuschlag), 2) AS durchschnitt
FROM wachdienst
WHERE risikozuschlag > 0;
```

Erwartetes Ergebnis:

minimum	maximum	durchschnitt
10.00	100.00	31.47