

28.11.2023

Übungsblatt 7

Diskussionsteil (im PS zu lösen; keine Abgabe nötig)

- a) ☐ ★ Gegeben sei ein Relationenmodell mit Relationenschema Verkauf (ID, KundeID, ArtikelID, Datum, Menge, Einzelpreis). Schreiben Sie eine SQL-Abfrage, die ermittelt, wie viele unterschiedliche Kund*innen im Jänner 2020 zumindest einen Artikel gekauft haben.
- b) ☐ ★ Übersetzen Sie die SQL-Abfrage der vorherigen Unteraufgabe in einen zu ihr äquivalenten Relationenalgebra-Ausdruck.
- c) ☐ ★★ Übersetzen Sie folgende SQL-Abfrage, die auf einen korrelierte Subquery zurückgreift, in einen zu ihr äquivalenten Relationenalgebra-Ausdruck.
- ```
1 SELECT StudentName
2 FROM Student
3 WHERE EXISTS (
4 SELECT 1
5 FROM attends
6 WHERE Student.StudentID = attends.StudentID
7 AND attends.grade = 2)
```
- d) ☐ ★★ Diskutieren Sie die folgenden beiden Fragen mit Ihren Kolleg\*innen:
- Sind der LEFT OUTER JOIN und der RIGHT OUTER JOIN – unter Vernachlässigung der Reihenfolge der Attribute in der Ergebnistabelle – im Allgemeinen kommutativ, d. h. gilt beispielsweise  $A \bowtie B = B \bowtie A$ ?
  - Sind nicht-korrelierte Subqueries immer performanter als korrelierte Subqueries?

## Hausaufgabenteil (Zuhause zu lösen; Abgabe nötig)

Wie bereits im vorherigen Übungsblatt zum Thema SQL, wird dieselbe Beispieldatenbank (Pagila) benötigt. Falls die Datenbank bei Ihnen nicht eingerichtet ist, erstellen Sie über Ihren SQL-Client eine neue Datenbank. Importieren Sie das Schema `pagila-schema.sqlOLAT` und die Daten `pagila-insert-data.sqlOLAT`.

### Hinweis



Achten Sie bitte darauf, dass Ihre Lösungen auf PostgreSQL 15 lauffähig sein müssen. Ihre Lösungen werden automatisch auf Korrektheit überprüft.

## Aufgabe 1 (Gruppierung und Aggregation)

[3 Punkte]

Diese Aufgabe befasst sich mit Abfragen, die Gruppierungen und Aggregationen beinhalten.

### Hinweis



Geben Sie für jede Unteraufgabe eine SQL-Datei, die die Abfrage (Query) und eine TXT-Datei, die das Resultat beinhaltet, mit den angegebenen Dateinamen ab.

- a) 1 Punkt Ermitteln Sie für jeden Film (Tabelle `film`), der länger als 180 ist und bereits einmal ausgeliehen wurde, wie viel dieser über den Verleih (Tabelle `rental`) insgesamt eingespielt hat. Es sind nur Filme relevant, die weniger als 170 eingespielt haben.

Reihenfolge und Bezeichnung der Ergebnisspalten:

- `title` (Name des Films)
- `length` (Länge des Films)
- `total_payment` (Summe der Zahlungen)

Sortieren Sie das Resultat **absteigend** nach `total_payment`.



### Hinweis



Sie benötigen zusätzlich die Tabellen `inventory` und `payment`. Machen Sie sich mit allen Tabellen vertraut und versuchen die Beziehungen zwischen den Tabellen zu verstehen.

### Abgabe



-  `exercise1/a.sql`
-  `exercise1/a_result.txt`

- b) 1 Punkt Ermitteln Sie wie oft jede\*r Schauspieler\*in (Tabelle `actor`) in einem Film (Tabelle `film`) mitgespielt hat.



Reihenfolge und Bezeichnung der Ergebnisspalten:

- `first_name` (Vorname Schauspieler\*in)
- `last_name` (Nachname Schauspieler\*in)
- `movie_count` (Anzahl der Filme)

Sortieren Sie das Resultat **absteigend** nach `movie_count` und zusätzlich **alphabetisch absteigend** nach `last_name`.

#### Abgabe



 `exercise1/b.sql`  
 `exercise1/b_result.txt`

- c) 1 Punkt Ermitteln Sie für jede Kategorie (Tabelle `category`) die durchschnittliche Mietdauer der Filme (Tabelle `film`).



Reihenfolge und Bezeichnung der Ergebnisspalten:

- `category_name` (Name der Kategorie)
- `avg_rental_duration` (Durchschnittliche Mietdauer der Filme)

Sortieren Sie das Resultat **aufsteigend** nach `avg_rental_duration`.

#### Abgabe



 `exercise1/c.sql`  
 `exercise1/c_result.txt`

## Aufgabe 2 (Subqueries)

[3 Punkte]

Diese Aufgabe befasst sich mit Abfragen, die Subqueries beinhalten.

#### Hinweis



Geben Sie für jede Unteraufgabe eine SQL-Datei, die die Abfrage (Query) und eine TXT-Datei, die das Resultat beinhaltet, mit den angegebenen Dateinamen ab.

- a) 1 Punkt Ermitteln Sie unter Verwendung einer Subquery, welche Schauspieler (Tabelle `actor`) im Film (Tabelle `film`) mit dem Titel (Spalte `title`) *LUKE MUMMY* mitgespielt haben. Verwenden Sie dafür eine Subquery — etwa mittels eines `IN`-Operators in der `WHERE`-Klausel. Die Information, welche\*r Schauspieler\*in in welchem Film mitgespielt hat, finden Sie in der Tabelle `film_actor`.



Reihenfolge und Bezeichnung der Ergebnisspalten:

- `first_name` (Vorname Schauspieler\*in)
- `last_name` (Nachname Schauspieler\*in)

Sortieren Sie das Resultat **alphabetisch aufsteigend** nach `last_name`.

#### Abgabe



 `exercise2/a.sql`  
 `exercise2/a_result.txt`

- b) **1 Punkt** Ermitteln Sie für jeden Film (Tabelle `film`), wie viel dieser über den Verleih (Tabelle `rental`) insgesamt eingespielt hat. Auch jene Filme die nie ausgeliehen wurden, müssen im Ergebnis enthalten sein (hier muss `total_payment` explizit ein **NULL** Eintrag sein, also nicht 0).

#### Hinweis



Die Lösung ist nicht dieselbe wie bei Aufgabe 1a, die eine ähnliche Aufgabenstellung hat.

Reihenfolge und Bezeichnung der Ergebnisspalten:

- `title` (Name des Films)
- `total_payment` (Summe der Zahlungen)

Sortieren Sie das Resultat **aufsteigend** nach `total_payment` und **alphabetisch aufsteigend** nach `title`.

#### Abgabe



`exercise2/b.sql`

`exercise2/b_result.txt`

- c) **1 Punkt** Ermitteln Sie für jede\*n Mitarbeiter\*in (Tabelle `staff`), wie viel die Einnahmen pro Kund\*in durchschnittlich betrugen. Entnehmen Sie anschließend den höchsten Wert und geben Sie diesen als `highest_avg_payment_from_customer` an.

#### Hinweis



Diese Aufgabe muss mit einer korrelierten Subquery gelöst werden.

Reihenfolge und Bezeichnung der Ergebnisspalten:

- `first_name` (Vorname Mitarbeiter\*in)
- `last_name` (Nachname Mitarbeiter\*in)
- `highest_avg_payment_from_customer` (Höchstwert der durchschnittlichen Zahlungen der Kund\*innen pro Mitarbeiter\*in)

Sortieren Sie das Resultat **aufsteigend** nach `highest_avg_payment_from_customer` und **alphabetisch aufsteigend** nach `last_name`.

#### Hinweis



Am Ende sollte für jeden Mitarbeiter genau eine Zeile im Ergebnis enthalten sein.

#### Abgabe



`exercise2/c.sql`

`exercise2/c_result.txt`

## Aufgabe 3 (Mengenoperationen)

[2 Punkte]

Bei dieser Aufgabe muss eine Abfrage mithilfe des Mengenoperators `UNION ALL` geschrieben werden.

### Hinweis



Geben Sie für diese Aufgabe eine SQL-Datei, die die Abfrage (Query) und eine TXT-Datei, die das Resultat beinhaltet, mit den angegebenen Dateinamen ab.

Ermitteln Sie das Ergebnis der folgenden Abfragen und bilden anschließend die Vereinigung, mittels dem `UNION ALL`-Operator, der beiden Abfragen:

- Für die erste Abfrage müssen Sie (ähnlich wie in Aufgabe 1a) für jeden Film, die Summe der Zahlungen ermitteln. Geben Sie zusätzlich an, wie viel beim Verleih im Durchschnitt für den jeweiligen Film gezahlt wurde.
- Für die zweite Abfrage müssen Sie das gleiche Prinzip auf Kategorien anwenden, um herauszufinden wie viel jede einzelne Kategorie insgesamt eingespielt hat und wie viel durchschnittlich gezahlt worden ist. Fügen Sie bei den Einträgen der Kategorie die Spalte `title` mit dem Inhalt *Category Pricings* ein.

Beispielsweise eine Abfrage in folgender Form ist gefragt:

```
1 SELECT /* snip - calculate sum and avg for each film */
2 UNION ALL
3 SELECT /* snip - calculate sum and avg for each category */
```

Reihenfolge und Bezeichnung der Ergebnisspalten:

- `title` (Filmtitel bzw. bei Kategorien *Category Pricings*)
- `category_name` (Name der Kategorie)
- `total_earnings` (Summe der Zahlungen)
- `average_payment` (Durchschnittliche Zahlung)

Achten Sie darauf, dass die Ergebnisse **absteigend** nach `total_earnings`, **alphabetisch absteigend** nach `title` und `category_name` sortiert sein sollen.

Die Ausgabe sollte also etwa wie folgt aussehen (Beispiel):

| title             | category_name | total_earnings | average_payment |
|-------------------|---------------|----------------|-----------------|
| Category Pricings | Sports        | 5959.61        | 8.76            |
| Category Pricings | Sci-Fi        | 5189.42        | 7.63            |
| ...               | ...           | ...            | ...             |
| VIDEOTAPE ARSENIC | Games         | 131.27         | 6.56            |
| DOGMA FAMILY      | Animation     | 116.83         | 5.84            |
| ...               | ...           | ...            | ...             |

### Abgabe



3.sql

3\_result.txt

## Aufgabe 4 (Report Entleihungen)

[2 Punkte]

In dieser Aufgabe soll ein kleiner Bericht mittels SQL erstellt werden.

### Hinweis



Geben Sie für diese Aufgabe eine SQL-Datei, die die Abfrage (Query) und eine TXT-Datei, die das Resultat beinhaltet, mit den angegebenen Dateinamen ab.

Stellen Sie sich folgendes Szenario vor: Ihr Chef möchte, um Werbemaßnahmen gezielter zu steuern, wissen, welche Kategorie von Filmen im August 2005 an welchem Wochentag wie oft entliehen wurden. Die Ausgabe sollte nach Kategorie **alphabetisch** sortiert sein. Neben der Kategorie sollen Spalten für alle Wochentage und eine Gesamtspalte ausgegeben werden.

Ein Ergebnis für die Abfrage sieht beispielsweise wie folgt aus:

| category_name | mon | tue | wed | thu | fri | sat | sun | total |
|---------------|-----|-----|-----|-----|-----|-----|-----|-------|
| Action        | 15  | 27  | 53  | 61  | 55  | 89  | 73  | 373   |

Reihenfolge und Bezeichnung der Ergebnisspalten:

- category\_name (Name der Kategorie)
- mon (Montag)
- tue (Dienstag)
- wed (Mittwoch)
- thu (Donnerstag)
- fri (Freitag)
- sat (Samstag)
- sun (Sonntag)
- total (Summe der Entleihungen für den Zeitraum)

### Hinweis



Sehen Sie sich die FILTER-Klausel für Aggregatfunktionen<sup>a</sup> an. Weiters stellt Ihnen PostgreSQL<sup>b</sup> Funktionen zum extrahieren des Datums zur Verfügung.

<sup>a</sup><https://www.postgresql.org/docs/13/sql-expressions.html#SYNTAX-AGGREGATES>

<sup>b</sup><https://www.postgresql.org/docs/13/functions-datetime.html#FUNCTIONS-DATETIME-EXTRACT>

### Abgabe



4.sql

4\_result.txt

**Wichtig:** Laden Sie bitte Ihre Lösung in OLAT hoch und geben Sie mittels der Ankreuzliste auch unbedingt an, welche Aufgaben Sie gelöst haben. Die Deadline dafür läuft am Vortag des Proseminars um 16:00 ab.