Aufgabenblatt 3
SQL

- Abgabetermin: **Sonntag, 15.06.2025, 23:59**
- Zur Prüfungszulassung muss ein Aufgabenblatt mit mind. 25% der Punkte bewertet werden und alle weiteren Aufgabenblätter mit mindestens 50% der Punkte.
- Die Aufgaben sollen in Zweiergruppen bearbeitet werden.
- Abgabe über Moodle:
  `https://moodle.hpi.de/course/view.php?id=906`
  - ausschließlich eine pdf-Datei *im A4-Format*
  - **eine neue Seite für jede Aufgabe**

Vorbemerkung und Hinweise zur Bearbeitung

In dieser Übung verwenden wir eine lokale PostgreSQL Installation, um Ergebnisse von SQL Anfragen direkt betrachten zu können.

Als Datensatz verwenden wir Daten aus der IMDB und Daten aus der Simulation eines Großhandels (genannt Sales), welche automatisch generiert wurden. Die Daten findet ihr in der Nextcloud[1] als PostgreSQL dump. Eine Anleitung zum Importieren der Daten folgt auf der nächsten Seite.
Hilfreiche/Weiterführende Links zu PostgreSQL:

- `https://www.postgresql.org/docs/current/index.html`
- `https://www.postgresql.org/docs/current/queries.html`
- `https://www.postgresql.org/docs/current/ddl.html`

Wenn Ihr Schwierigkeiten habt, eine komplexe Abfrage zu entwickeln oder durch lange Ausführungszeiten blockiert werdet, kann es sinnvoll sein, eine eigene Testdatenbank mit wenigen Testtupeln zu erstellen, um das Debuggen/Testen zu erleichtern. Alternativ könnt Ihr auch Schlüsselbedingungen/Indizes anlegen, die die Ausführungszeit beschleunigen. Überlegt, welche Schlüssel/Indizes sinnvoll sind und legt diese selbstständig an (`https://www.postgresql.org/docs/current/sql-altertable.html` und `https://www.postgresql.org/docs/current/sql-createindex.html`).

[1] `https://nextcloud.hpi.de/s/5yJPrg6sx5sBp52` Passwort: CZMpwg4K8F

Import der Daten

a) Ladet die Dateien herunter[2] und entpackt die ZIP-Datei.

b) Führt `\i <Pfad zu den Daten>/IMDB.pgdbdump` in der `psql` Shell aus. Wiederholt dies für den *Sales.pgdbdump*.

c) In der `psql` Shell könnt ihr testen, ob alles funktioniert hat, indem ihr überprüft, ob die in Abbildung 2 und 3 gezeigten Queries die gleichen Resultate liefern. Lediglich der Owner kann bei euch anders sein.

```
postgres=# \c imdb
You are now connected to database "imdb" as user "postgres".
imdb=# \d
            List of relations
 Schema |     Name     | Type  |  Owner
--------+--------------+-------+----------
 public | nbasics      | table | postgres
 public | tmovies      | table | postgres
 public | tprincipals  | table | postgres
 public | tratings     | table | postgres
(4 rows)

imdb=# SELECT COUNT(*) FROM tmovies;
 count
--------
 421474
(1 row)
```

Abbildung 1: Test Queries with Output (IMDB)

```
imdb=# \c Sales
You are now connected to database "Sales" as user "postgres".
Sales=# SELECT * FROM orders LIMIT 10;
 o_orderkey | o_custkey | o_orderstatus | o_totalprice | o_orderdate | o_orderpriority |     o_clerk     | o_shippriority |                    o_comment
------------+-----------+---------------+--------------+-------------+-----------------+-----------------+----------------+-------------------------------------------------
          1 |     36901 | O             |    173665.47 | 1996-01-02  | 5-LOW           | Clerk#000000951 |              0 | nstructions sleep furiously among
          2 |     78002 | O             |     46929.18 | 1996-12-01  | 1-URGENT        | Clerk#000000880 |              0 |  foxes. pending accounts at the pending, silent asymptot
          3 |    123314 | F             |    193846.25 | 1993-10-14  | 5-LOW           | Clerk#000000955 |              0 | sly final accounts boost. carefully regular ideas cajole carefully. depos
          4 |    136777 | O             |     32151.78 | 1995-10-11  | 5-LOW           | Clerk#000000124 |              0 | sits. slyly regular warthogs cajole. regular, regular theodolites acro
          5 |     44485 | F             |    144659.2  | 1994-07-30  | 5-LOW           | Clerk#000000925 |              0 | quickly. bold deposits sleep slyly. packages use slyly
          6 |     55624 | F             |     58749.59 | 1992-02-21  | 4-NOT SPECIFIED | Clerk#000000858 |              0 | ggle. special, final requests are against the furiously specia
          7 |     39136 | O             |    252004.18 | 1996-01-10  | 2-HIGH          | Clerk#000000470 |              0 | ly special requests
         32 |    130057 | O             |    208660.75 | 1995-07-16  | 2-HIGH          | Clerk#000000616 |              0 | ise blithely bold, regular requests. quickly unusual dep
         33 |     66958 | F             |    163243.98 | 1993-10-27  | 3-MEDIUM        | Clerk#000000409 |              0 | uriously. furiously final request
         34 |     61001 | O             |     58949.67 | 1998-07-21  | 3-MEDIUM        | Clerk#000000223 |              0 | ly final packages. fluffily final deposits wake blithely ideas. spe
(10 rows)

Sales=# SELECT COUNT(*) FROM orders;
  count
---------
 1500000
(1 row)
```

Abbildung 2: Test Queries with Output (Sales)

[2] `https://nextcloud.hpi.de/s/5yJPrg6sx5sBp52` Passwort: CZMpwg4K8F

Hinweise zum IMDb-Datensatz

- Es handelt sich hier um einen Auszug aus echten Daten, die hier beschrieben werden: `https://developer.imdb.com/non-commercial-datasets/`. *tmovies* ist eine Projektion und Auswahl von *tbasics* (title.basics.tsv.gz). Die Projektion/Selektion wurde so vorgenommen, dass wir nur Filme (und nicht andere Medienwerke wie Videospiele) einbeziehen und um einige unnötige Attribute zu entfernen. Die Semantik aller beibehaltenen Attribute bleibt unverändert. Die anderen drei Tabellen *nbasics* (name.basics.tsv.gz), *tprincipals* (title.principals.tsv.gz) und *tratings* (title.ratings.tsv.gz) enthalten nur Tupel, zu denen ein Eintrag in der Filmtabelle existiert. Es ist Teil der Übung, das Schema zu verstehen, aber mit der Dokumentation der IMDB, sollte dies nicht schwer sein. Wenn Ihr Fragen habt, stellt sie wie immer gern im Forum.
- Um den Datenimport zu vereinfachen, gibt es in der Datenbank keine expliziten Schlüssel oder Fremdschlüssel. Die Schlüssel und Fremdschlüssel sind die Attribute *nconst* und *tconst*.
- Wenn wir von einer Person sprechen, die in einem Film in einer bestimmten Rolle mitwirkt (z. B. als Schauspieler*in oder Drehbuchautor*in, etc.), meinen wir das Attribut *tprincipals.category* und nicht *tprincipals.job*.
- Einige Attribute (wie z.B. *tmovies.genres*) verwenden einen Array-Typ. PostgreSQLs Dokumentation zum Zugriff auf Arrays, findet Ihr hier: `https://www.postgresql.org/docs/current/arrays.html`

Aufgabe 1: Deutsch → SQL

Nenne für jede, der folgenden natürlichsprachlichen Fragen eine geeignete SQL-Anfrage und führe sie auf den Daten der IMDb aus. Gib sowohl die Anfrage als auch deren Ergebnis (maximal 10 Tupel und die Anzahl der Tupel) an.

a) Wieviele verschiedene Filmtitel gibt es? **2 P**

b) Gib die Namen aller Personen an, die in mindestens einem Film Producer waren (nach Producernamen sortiert). Jeder Producer soll dabei nur einmal ausgegeben werden. **3 P**

c) Gib die IDs aller verschiedenen Filmpaare aus, in denen mindestens eine gemeinsame Person mitgewirkt hat! Sortiere das Ergebnis nach der ID des zweiten Films. **3 P**

d) Gib für alle Schauspieler*innen des Films *Inception* (tconst=tt1375666), ihren Namen und die Anzahl an Filmen, an denen sie insgesamt mitgewirkt haben, an. Sortiert das Ergebnis nach dieser Anzahl absteigend. Denkt daran, dass Schauspieler und Schauspielerinnen unterschiedlich in der Datenbank gespeichert werden (`tprincipals.category = actor` oder `actress`). **3 P**

e) Gib die ID und den Namen aller Personen an, die an mindestens einem Film des Genres *Action* beteiligt waren und deren Name mit *T* beginnt. **3 P**

f) Gib die ID und den Namen aller Personen an, die *nur* an Filmen des Genres *Action* beteiligt waren und deren Name mit *T* beginnt. **4 P**

g) Gib die ID (tconst), den Titel, das Jahr und die Bewertung für alle Filme mit 3 oder mehr Produzenten an, in denen eine Figur namens `["Peter Pan"]` vorkommt. Sortiert das Ergebnis nach der Bewertung absteigend. **4 P**

Anmerkung: Die eckigen Klammern und Anführungszeichen sind Teil des Namensstrings, da `tprincipals.characters` eine String-Variable ist, die Json-Arrays speichert - willkommen in der realen Datenmodellierung :)

h) Formuliere *eine* Anfrage, die die Jahreszahl und die Anzahl der in diesem Jahr veröffentlichten Filme abfragt, für

- das höchste (späteste) vorkommende Jahr und
- das Jahr mit den meisten veröffentlichten Filmen

5 P

Aufgabe 2: Relationale Algebra → SQL

Formuliere die folgenden drei Anfragen der relationalen Algebra als SQL-Anfragen!

Verwendetes Schema:

- Stadt (StadtName, LandID, p1950, p2000, p2015)
  wobei `p1950`, `p2000` und `p2015` die Bevölkerungszahlen in diesen Jahren darstellen
- Land (LandID, Name, Kontinent, Hauptstadt, Bevölkerung)
- Geographie (LandID, Landfläche, Wasserfläche, Küstenlänge, urbar)
  wobei `urbar` die urbare Fläche des Landes beschreibt

a) $\pi_{\text{Name, Kontinent}}(\sigma_{\text{Bevölkerung}>200.000.000}(\text{Land}))$ **2 P**

b) $\pi_{\text{Name}}(\sigma_{(\text{Bevölkerung}<2*p1950)\vee(\text{Bevölkerung}<4*p2000)}(\sigma_{\text{StadtName=Hauptstadt}}(\text{Stadt} \bowtie \text{Land})))$ **3 P**

c) $\pi_{\text{Name}}(\text{Land} \bowtie \text{Geographie}) -$
$\pi_{\text{Name}}(\text{Land} \bowtie (\sigma_{G1.\text{urbar}<G2.\text{urbar}}(\rho_{G1}(\text{Geographie}) \times \pi_{\text{urbar}}(\rho_{G2}(\text{Geographie})))))$ **4 P**

Erklärungen zum Sales Datensatz

Für die folgenden Aufgaben betrachten wir den Sales-Datensatz. Dieser modelliert das Geschäft eines Großhandels. Das Schema der Tabellen ist in Abbildung 4 zu sehen. Es handelt sich dabei nicht um ein high-level E/R Diagramm, sondern um eine direkte Abbildung der Tabellen. Die Pfeile zeigen Schlüssel/Fremdschlüssel Beziehungen, aber der Pfeil zeigt hier in die Richtung der 1-zu-n Beziehung (anders als bei E/R Diagrammen! D.h. wie erwartet hat ein Land genau eine Region (Kontinent) aber eine Region mehrere Länder). Die Klammern hinter den Bezeichnungen der Entitätstypen spezifizieren einen Präfix, den jedes Attribut in den Tabellen besitzt (d.h. Partkey heißt in der Relation PART *P_PARTKEY* und in der Relation PARTSUPP *PS_PARTKEY*). Die Tabellen beinhalten folgende Daten:

- **PART** - Einzelteile, welche verkauft werden können. Schlüssel: `P_PARTKEY`
- **PARTSUPP** - Kombinationen aus Einzelteilen und Lieferanten für diese Teile. Schlüssel: `(PS_PARTKEY, PS_SUPPKEY)`. **Anmerkung:** `AVAILQTY` steht für available quantity also die Anzahl an verfügbaren Teilen bei diesem Lieferanten.
- **LINEITEM** - Verkäufe einer bestimmten Anzahl eines Einzelteils (Intuitiv: eine Zeile auf der Quittung). Schlüssel: `(L_ORDERKEY, L_LINENUMBER)`
- **ORDERS** - Bestellungen, Schlüssel: `O_ORDERKEY`
- **SUPPLIER** - Lieferanten, Schlüssel: `S_SUPPKEY`
- **CUSTOMER** - Kunden, Schlüssel: `C_CUSTKEY`
- **NATION** - Länder/Staaten, Schlüssel: `N_NATIONKEY`
- **Region** - Kontinente, Schlüssel: `R_REGIONKEY`

Um den Import zu beschleunigen, sind in den euch zur Verfügung gestellten Daten die Schlüssel/Fremdschlüssel nicht als explizit festgehalten. Die gestellten Anfragen haben unter Umständen sehr lange Ausführungszeiten. Diese können durch das Anlegen von Schlüsseln/Indizes verkürzt werden. Überlegt, welche Indizes sinnvoll sind und legt diese selbstständig an (`https://www.postgresql.org/docs/current/sql-altertable.html` und `https://www.postgresql.org/docs/current/sql-createindex.html`).

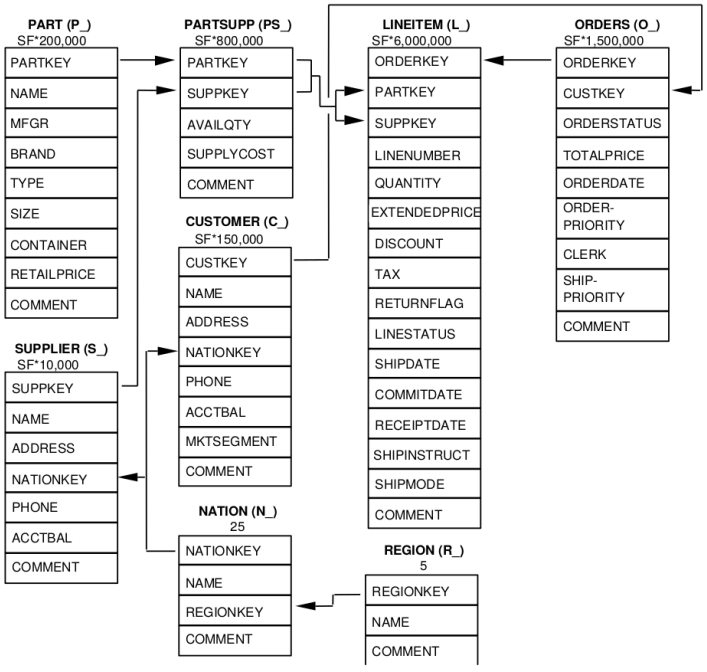

Abbildung 3: Sales Schema

Aufgabe 3: Sales-Datensatz: SQL → Deutsch

Gib natürlichsprachlich wieder, wonach folgende SQL-Anfragen suchen:

a)
```
SELECT n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
FROM customer, orders, lineitem, supplier, nation, region
WHERE
    c_custkey = o_custkey
    AND l_orderkey = o_orderkey
    AND l_suppkey = s_suppkey
    AND c_nationkey = s_nationkey
    AND s_nationkey = n_nationkey
    AND n_regionkey = r_regionkey
    AND r_name = 'EUROPE'
    AND o_orderdate >= date '1992-01-01'
    AND o_orderdate < date '1992-01-01' + interval '1' year
GROUP BY n_name
ORDER BY revenue DESC;
```

5 P

b)
```
SELECT supp_nation, cust_nation, l_year, sum(volume) as revenue
FROM (
    SELECT
        n1.n_name as supp_nation,
        n2.n_name as cust_nation,
        EXTRACT(YEAR FROM l_shipdate) as l_year,
        l_extendedprice * (1 - l_discount) as volume
    FROM supplier, lineitem, orders, customer, nation n1, nation n2
    WHERE
        s_suppkey = l_suppkey
        AND o_orderkey = l_orderkey
        AND c_custkey = o_custkey
        AND s_nationkey = n1.n_nationkey
        AND c_nationkey = n2.n_nationkey
        AND (
            (n1.n_name = 'GERMANY' AND n2.n_name = 'UNITED STATES')
            OR (n1.n_name = 'UNITED STATES' AND n2.n_name = 'GERMANY')
        )
        AND l_shipdate between date '1995-01-01' and date '1996-12-31'
    ) as shipping
GROUP BY supp_nation, cust_nation,
    l_year
ORDER BY supp_nation, cust_nation, l_year;
```

6 P

Aufgabe 4: Sales-Datensatz: Deutsch → SQL

Wir befassen uns weiterhin mit dem Datensatz aus der vorherigen Aufgabe. Nenne für jede der folgenden natürlichsprachlichen Fragen eine geeignete SQL-Anfrage und führe sie auf den Sales-Daten aus. Gib auf deiner Abgabe die Anfrage, die ersten 10 Zeilen des Ergebnisses sowie die Anzahl an Tupeln des Ergebnisses an.

a) Schreiben Sie eine Query, die für jedes Land L und jedes Jahr den Gesamt-Profit aus dem Verkauf aller in diesem Jahr bestellten Teile (Part), dessen Name den String *'chocolate'* enthält, auflistet, wobei die Teile von einem Lieferanten (Supplier) im Land L geliefert worden sein müssen. Der Profit eines Teils in einem Verkauf (Lineitem) ist definiert als: (l_extendedprice*(1-l_discount)) - (ps_supplycost * l_quantity). Die Ausgabe soll aufsteigend nach dem Land und innerhalb eines Landes absteigend nach dem Jahr sortiert werden. Zur Selbstkontrole sind hier die ersten 5 Zeilen des Ergebnisses angegeben:

```
ALGERIA,1998,28755250.6829
ALGERIA,1997,51619211.9021
ALGERIA,1996,50971117.0264
ALGERIA,1995,50178326.0903
ALGERIA,1994,50354843.0773
```

8 P

b) Gib für Deutschland ('GERMANY') für alle dort verfügbaren Teile (Part) den Schlüssel, den Namen, sowie die Gesamtanzahl der verfügbaren Teile aus (egal von welchem Supplier, solange er zu Deutschland gehört). Gib jedoch nur die Teile aus, deren Gesamtanzahl mehr als 0.001% der Anzahl aller überhaupt verfügbaren Teile in Deutschland ausmachen. Das Ergebnis soll nach der Gesamtanzahl absteigend sortiert werden. Zur Selbstkontrolle sind hier die ersten 5 Zeilen des Ergebnisses angegeben:

```
85606,dodger khaki honeydew lawn mint,26531
60932,peru goldenrod ghost magenta white,25274
80958,tomato white tan drab thistle,22290
139035,salmon navajo cornflower grey maroon,21778
164254,spring ghost orchid saddle beige,21116
```

8 P