DES3UE Datenbanksysteme WS 2024 Übung 3

Abgabetermin: siehe e-Learning, Abgabeform elektronisch

☐ DES3UEG1: Glock Name _____ Aufwand in h _____

☐ DES3UEG2: Werth Punkte _____ Kurzzeichen Tutorin _____

Hinweise und Richtlinien:

- Übungsausarbeitungen müssen den im elearning angegebenen Formatierungsrichtlinien entsprechen – Nichtbeachtung dieser Formatierungsrichtlinien führt zu Punkteabzug.
- Verwenden Sie EXECUTE SYS.KILL_MY_SESSIONS() um hängen gebliebene Sessions/Abfragen zu beenden.
- Zusätzlich zu den allgemeinen Formatierungsrichtlinien sind für diese Übungsausarbeitung folgende zusätzlichen Richtlinien zu beachten:
  - Treffen Sie, falls notwendig, sinnvolle Annahmen und dokumentieren Sie diese nachvollziehbar in ihrer Lösung!
  - Recherchieren Sie eventuell unbekannte Elemente nach Bedarf.

Ziel dieser Übung ist die Formulierung von hierarchischen Abfragen in SQL, die Manipulation von Zeilen und Spalten mit den Befehlen PIVOT und UNPIVOT und der Einsatz von analytischen Abfragen, sowie (materialisierte) Sichten.

Teilweise sind die Ergebnisse auszugsweise dargestellt. Geben Sie in Ihrer Lösung jeweils Screenshot ab, die größere Ausschnitte umfassen als in den Ergebnis-Auszügen dargestellt sind.

1. PIVOT und UNPIVOT - Sakila (3,5 Punkte – 1+1+1,5)

1. Erstellen Sie basierend auf dem angegebenen SQL-Statement eine Abfrage die für jeden Film aus dem Jahr 1993 die Sprache und die Original-Sprache enthält. Die Ergebnistabelle soll als Spalten den Film-Namen, die Sprache und die Art der Sprache (L, OL) enthalten und nach Film-Titel sortiert sein. *(44 Zeilen)*

Ergebnis-Auszug:

	TITEL	ART	SPRACHE
1	ATTRACTION NEWTON	L	German
2	ATTRACTION NEWTON	OL	Mandarin
3	AUTUMN CROW	L	Mandarin
4	AUTUMN CROW	OL	German
5	BRAVEHEART HUMAN	L	English
6	BRAVEHEART HUMAN	OL	German
7	BUTTERFLY CHOCOLAT	OL	French
8	BUTTERFLY CHOCOLAT	L	Italian
9	CELEBRITY HORN	L	Italian
10	CELEBRITY HORN	OL	German
11	CHARIOTS CONSPIRACY	OL	Mandarin

Folgender Code-Ausschnitt steht Ihnen bereits zur Verfügung:

```
SELECT f.title AS Titel, l1.name AS L, l2.name AS OL
FROM film f
    INNER JOIN language l1 USING(language_id)
    INNER JOIN language l2 ON (f.original_language_id = l2.language_id)
WHERE f.release_year = 1993;
```

2. Erstellen Sie basierend auf dem angegebenen SQL-Statement eine Pivot-Tabelle für die Kategorien „Family“, „Children“ und „Animation“ (Spalten), welche die Durchschnittslänge der Filme pro Sprache (Zeilen) angibt. *(6 Zeilen, 4 Spalten)*

Ergebnis-Auszug:

NAME	FAMILY_LAENGE	CHILDREN_LAENGE	ANIMATION_LAENGE
Japanese	116,764706	120	113,7
Italian	115,916667	97,1818182	93,6666667
French	102,714286	120,0625	99,1

Folgender Code-Ausschnitt steht Ihnen bereits zur Verfügung:

```
SELECT c.name AS category, l.name, length AS length
FROM film
    INNER JOIN film_category USING (film_id)
    INNER JOIN category c USING (category_id)
    INNER JOIN language l USING (language_id)
WHERE c.name IN ('Family', 'Children', 'Animation');
```

3. Sie sollen für jede Kategorie Anzahl der Filme (Inventar!) pro Filiale und insgesamt in der jeweiligen Kategorie berechnen. Erstellen Sie hierfür eine Kreuztabelle, die für Filialen und Kategorien die Anzahl der Filme (Inventar!) berechnet, geben Sie auch die Gesamtsummen pro Filiale und pro Kategorie aus. Ergänzen Sie für die Erstellung der Kreuztabelle den gegebenen Code-Ausschnitt und verwenden Sie diese als Sub-Select eines Pivot-Befehls.

Folgender Code-Ausschnitt steht Ihnen bereits zur Verfügung:

```
...
SELECT COUNT(*) AS anzahl,
    CASE WHEN GROUPING(name_id) THEN 'Gesamt' ELSE name END AS Kategorie,
    CASE WHEN WHEN GROUPING(store_id) THEN 'Gesamt' ELSE to_char(store_id) END AS Filiale
FROM inventory
    INNER JOIN film_category USING (film_id)
    INNER JOIN category USING (category_id)
GROUP BY /* ergänzen */ (store_id, name)
...
```

Ergebnis-Auszug:

KATEGORIE	1	2	3	4	5	6	GESAMT
Action	48	47	46	72	55	44	312
Games	46	42	53	49	37	49	276
Drama	43	44	57	53	55	48	300
Comedy	43	47	45	41	50	43	269

2. Hierarchische Abfragen – Human Resources (4 Punkte – 2+1+1)

1. Shelley Higgins verlässt das Unternehmen. Ihr Nachfolger wünscht Berichte über die Angestellten, die Higgins direkt unterstellt sind. Erstellen Sie eine SQL-Anweisung, um die Angestelltennummer, den Nachnamen, das Einstellungsdatum und das Gehalt anzuzeigen, wobei auf Folgendes eingeschränkt werden soll:
   a. Die Angestellten, die Higgins direkt unterstellt sind

   b. Die **gesamte** Organisationsstruktur unter Higgins (Angestelltennummer: 205)

2. Erstellen Sie eine hierarchische Abfrage, um die Angestelltennummer, die Managernummer und den Nachnamen für alle Angestellten unter De Haan anzuzeigen, die sich genau zwei Ebenen unterhalb dieses Angestellten (De Haan, Angestelltennummer: 102) befinden. Zeigen Sie zudem die Ebene des Angestellten an. *(2 Zeilen)*

3. Der CEO benötigt einen hierarchischen Bericht über alle Angestellten. Er teilt Ihnen die folgenden Anforderungen mit: Erstellen Sie einen hierarchischen Bericht, der den Nachnamen, die Angestelltennummer, die Managernummer und die Pseudospalte `LEVEL` des Angestellten anzeigt. Für jede Zeile der Tabelle `EMPLOYEES` soll eine Baumstruktur ausgegeben werden, die den Angestellten, seinen Manager, dessen Manager usw. zeigt. Verwenden Sie Einrückungen (LPAD) für die Spalte `LAST_NAME`.

Beispieldarstellung (Ergebnis-Auszug):

LAST_NAME	EMPLOYEE_ID	MANAGER_ID	LEVEL
King	100		1
Kochhar	101	100	1
__King	100		2
De Haan	102	100	1
__King	100		2
Hunold	103	102	1
__De Haan	102	100	2
____King	100		3

3. Hierarchische Abfragen – Sakila-Datenbank (2,5 Punkte – 0,5 + 2)

WICHTIGE HINWEISE:

- Beachten Sie, dass für diese beiden Abfragen nur **Filme mit einer ID <= 13** berücksichtigt werden.
- Für die Hierarchie soll die Bekanntschaftskette **nicht** innerhalb eines Filmes durchlaufen werden (zusätzliche Bedingung bei CONNECT BY).
- Verwenden Sie EXECUTE SYS.KILL_MY_SESSIONS() um hängen gebliebene Sessions/Abfragen zu beenden.

Für Schauspielerinnen wird ein soziales Netzwerk aufgebaut. Initial wird davon ausgegangen, dass sich Schauspielerinnen, die gemeinsam in einem Film spielen, bereits kennen. Das System soll nun eine Liste von **neuen** Kontaktvorschlägen generieren, die darauf beruht, jemanden „über (genau) einen gemeinsamen Bekannten“ zu kennen.

a) Verwenden Sie den unten angegebenen Codeausschnitt mit dem Entwurf einer View *partners*, die Ihnen Schauspielerinnen und Schauspieler-Kollegen ausgibt (wenn sie in irgendeinem Film miteinander gespielt haben).

Ergänzen Sie diese, sodass Sie darauf, dass keine Beziehung der SchauspielerInnen auf sich selbst entsteht. Die View soll beide Actor-IDs und die Film-ID enthalten. Damit das Ergebnis überschaubar bleibt, sollen Sie hierfür nur die ersten 13 Filme (film_id <= 13) der Datenbank heranziehen. *(2 Punkte, 438 Zeilen)*

b) Erstellen Sie aufbauend auf der obigen View eine nach IDs sortierte Vorschlagsliste für Schauspieler „JULIANNE DENCH“. Verwenden Sie hierfür den gegebenen Codeausschnitt. *(4 Punkte, 10 Zeilen)*

Ergebnis-Auszug:

VORSCHLAG
29
35
37

```
CREATE OR REPLACE VIEW partners AS
  SELECT fa1.actor_id, fa2.actor_id AS partner_id, fa1.film_id
  FROM film_actor fa1 INNER JOIN film_actor fa2 ON ergänzen
  WHERE --ergänzen AND --ergänzen
  ORDER BY fa1.actor_id;

SELECT * FROM partners;

SELECT DISTINCT partner_id AS Vorschlag
```

```
FROM partners
WHERE level = 2
START WITH
--ergänzen

ORDER BY Vorschlag;
```

4. Analytische Abfragen - Sakila (6 Punkte – 2+1,5+2,5)

Anmerkung: Verwenden Sie für die Lösung der folgenden Aufgaben analytische Funktionen!

1. Geben Sie zu jedem Film ID, Titel, Länge und die zugehörige Kategorie-Bezeichnung aus. Führen Sie außerdem pro Film an, wie viele Filme in der jeweiligen Film-Kategorie 5 Minuten kürzer oder länger als der Film selbst sind (Wie viele Empfehlungen für "ähnliche" Filme gibt es?).
   Hinweis: Zur Selbstkontrolle sortieren Sie die Liste nach Kategorie und Länge. Für "Bride Intrigue" (Action) gibt es 11 weitere Filme, deren Laufzeit 5 Minuten kürzer oder länger ist.

   Ergebnis-Auszug:

	FILM_ID	TITLE	NAME	LENGTH	SUGGESTIONS
1	869	SUSPECTS QUILLS	Action	47	4
2	292	EXCITEMENT EVE	Action	51	6
3	111	CADDYSHACK JEDI	Action	52	6
4	542	LUST LOCK	Action	52	6
5	794	SIDE ARK	Action	52	6
6	697	PRIMARY GLASS	Action	53	8
7	97	BRIDE INTRIGUE	Action	56	11

2. Ergänzen Sie den angegebenen Ausschnitt, sodass Sie zu jeder Filmkategorie drei Filme mit Kategorienamen, Filmtitel und Erscheinungsjahr ausgeben. Wählen Sie jene Filme, die neueren Datums sind. Verwenden Sie wie angegeben ROW_NUMBER().

```
SELECT name AS category, title, release_year
FROM (
  SELECT name, title, release_year,
         ROW_NUMBER() --ergänzen AS num
  FROM film
     INNER JOIN film_category USING (film_id)
     INNER JOIN category USING (category_id)
) WHERE --ergänzen;
```

3. Ermitteln Sie welche Kunden schon einmal mehr als 180 Tage zwischen zwei Verleihvorgängen verstreichen haben lassen. Verwenden Sie dafür den angegebenen Codeausschnitt.
   a) Erstellen Sie zuerst eine analytische Abfrage, die für jeden Kunden das aktuelle Verleihdatum und das nächste gegenüberstellt.
   b) Berechnen Sie aufbauend auf a) die Anzahl der Tage zwischen den Verleihvorgängen.

   Geben Sie für die geforderten Einträge den Vor- und Nachnamen des Kunden, sowie die Anzahl der verstrichenen Tage aus.

Achtung: Die Auswertung ist nur möglich, wenn der Kunde bereits mehr als einen Film ausgeborgt hat (beachten Sie NULL-Werte).

Ergebnis-Auszug:

FIRST_NAME	LAST_NAME	TAGE
KATHLEEN	ADAMS	238
KATHERINE	RIVERA	185
EMILY	DIAZ	190
ALICIA	MILLS	190

Codeausschnitt:

```
SELECT first_name, last_name, next_rental_date - rental_date AS tage
```

```
FROM (

--ergänzen

) INNER JOIN customer USING (customer_id)

WHERE next_rental_date - rental_date >= 180;
```

5. LISTAGG – Sakila (4 Punkte - 2 + 2)

1. Geben Sie für alle Filme die 1991 erschienen sind, die Länge (‚length') und den Filmtitel (‚film') aus, sortieren Sie nach Länge absteigend.

   Zusätzlich soll für jeden Film eine Liste aller Schauspieler (‚actors'), die in dem Film mitspielen, ausgegeben werden (siehe Abbildung). In der Actors-Liste sollen die Namen nach Nachname und Vorname sortiert sein. Die Namen sollen so ausgegeben werden, dass jeweils der erste Buchstabe des Vornamens, '. ', und der Nachname angezeigt werden. Die einzelnen Schauspieler sind durch Komma ',' voneinander zu trennen. Geben Sie auch Filme aus, in denen keine Schauspieler mitspielen und verwenden Sie dafür den Text ‚no actors'.

Hinweis:

Verwenden Sie eine einfache Gruppierung und entsprechende Joins; damit auch WITHIN GROUP ohne Verwendung der OVER-Klausel! (DISTINCT mit OVER PARTITION BY wäre möglich, ABER: DISTINCT sollte man sehr sparsam einsetzen, meist bläst man sich die Datenmenge versehentlich auf, das passiert hier durch die Joins).

	LENGTH	FILM	ACTORS
1	184	CRYSTAL BREAKING	L. BERGMAN, P. CRONYN, R. KILMER, J. NEESON, F. WOOD
2	179	FLIGHT LIES	no actors
3	172	IDAHO LOVE	M. BOLGER, P. GOLDBERG, R. JOHANSSON, T. MCKELLEN, H. WILLIS, F. WOOD
4	171	JERICHO MULAN	V. BASINGER, W. HACKMAN, E. MARX, L. MONROE
5	167	ESCAPE METROPOLIS	N. HOPKINS, J. LOLLOBRIGIDA, E. MANSFIELD, F. TOMEI
6	164	VIRGINIAN PLUTO	J. BAILEY, K. BERRY, P. CRONYN, A. JOHANSSON, S. PECK, T. TEMPLE, G. WIL
7	163	QUEEN LUKE	R. DEAN, E. GOODING, S. PECK, J. SILVERSTONE, M. TANDY, R. WINSLET
8	161	OLEANDER CLUE	Z. CAGE, G. CHAPLIN, R. CLOSE, E. GOODING, W. JACKMAN, O. KILMER, S. KIL

2. Geben Sie zu jedem Kunden (Vorname Nachname), die dem Store mit der ID 4 zugeordnet sind, eine Liste der Filme aus, die sich der Kunde seit 2010 ausgeborgt hat. Zusätzlich zum Titel des Films geben Sie das Erscheinungsjahr in Klammer an. Sortieren Sie die Film-Liste so, dass die jüngsten Filme zuerst aufscheinen (*Beispielauszug siehe Abbildung*).

	CUSTOMER	FILMS
1	JENNIFER DAVIS	MASKED BUBBLE (2008), GOLD RIVER (2006), MONSTER SPARTACUS (2004), SLE
2	JIMMIE EGGLESTON	CHANCE RESURRECTION (2007), SHOCK CABIN (2007), SAINTS BRIDE (2006), U
3	PHYLLIS FOSTER	JEOPARDY ENCINO (2008), DRIVING POLISH (2006), STORY SIDE (2006), DECE
4	JACK FOUST	CROOKED FROGMEN (2008), REAR TRADING (2002), DRIVER ANNIE (1997), ZORR
5	BRANDY GRAVES	DRIVING POLISH (2006), SLEUTH ORIENT (2004), CONFESSIONS MAGUIRE (2003
6	COREY HAUSER	SUSPECTS QUILLS (2007), DRIVING POLISH (2006), MILLION ACE (2003), ARA
7	JILL HAWKINS	NUTS TIES (2008), MALKOVICH PET (2007), STAGECOACH ARMAGEDDON (1999),

6. Materialisierte Sichten (Sakila-Datenbank) (4 Punkte – 1+1+2)

1. Formulieren Sie eine Anfrage, die den Umsatz der Verkäufer (staff_id = 1 bzw. 2) pro Filmkategorie vergleicht und geben Sie auch das Verhältnis der beiden Umsätze (pro Kategorie) aus. „Speichern" Sie diese Abfrage als **virtuelle Sicht** „UE03_06a".

   Verwenden Sie hierfür den gegebenen Subquery-Block „revenues", der Ihnen für jeden Angestellten (staff_id) den Umsatz pro Filmkategorie (Name der Filmkategorie) berechnet.

   Ergebnis-Auszug:

	KATEGORIE	UMSATZ1	UMSATZ2	VERHÄLTNIS
1	Animation	3128,76	3073,29	1,02
2	Comedy	2649,26	2715,84	0,98
3	Family	3373,11	2988,13	1,13

```
CREATE
--ergänzen
AS
WITH revenues AS (
    SELECT SUM(amount) AS Umsatz, c.name AS Kategorie,
           r.staff_id As staff
    FROM category c
        INNER JOIN film_category USING (category_id)
        INNER JOIN inventory i USING (film_id)
        INNER JOIN rental r USING (inventory_id)
        INNER JOIN payment USING (rental_id)
    GROUP BY c.name, r.staff_id)
SELECT
--ergänzen
;

SELECT *
FROM UE03_06a;
```

2. Erzeugen Sie aus der Sicht in UE03_06a eine manuell zu aktualisierende **materialisierte Sicht** UE03_06b mit kompletter Neuberechnung (Re-Materialisierung).

```
CREATE
--ergänzen
AS
SELECT * FROM UE03_06a;
```

3. Verändern Sie den Aktualisierungszeitpunkt der erstellten materialisierten Sicht UE03_06b in der Weise, dass sie automatisch jeden Tag um 23:30 aktualisiert wird. Speichern Sie die geänderte materialisierte Sicht unter dem Namen UE03_06c. Löschen Sie die materialisierte Sicht UE03_06c anschließend wieder.