

<input checked="" type="checkbox"/> DES3UEG1: Glock	Name <u>Elias Leonhardsberger</u>	Aufwand in h <u>6</u>
<input type="checkbox"/> 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;
```

- 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');
```

- 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)

- 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:
 - Die Angestellten, die Higgins direkt unterstellt sind
 - Die **gesamte** Organisationsstruktur unter Higgins (Angestelltennummer: 205)
- 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 fal.actor_id, fa2.actor_id AS partner_id, fal.film_id
FROM film_actor fal INNER JOIN film_actor fa2 ON ergänzen
WHERE --ergänzen AND --ergänzen
ORDER BY fal.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 ausgelohnt 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)

- 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 ‘,’ von einander 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

- 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 ausgeliehen 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)

- 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.

DES3UE Übung 3

Elias Leonhardsberger

2. Dezember 2024, Hagenberg

Inhaltsverzeichnis

1	PIVOT und UNPIVOT - Sakila	9
1.1	SQL Queries	9
1.2	Ergebnisse	10
2	Hierarchische Abfragen - Human Resources	11
2.1	SQL Queries	11
2.2	Ergebnisse	11
3	Hierarchische Abfragen - Sakila	13
3.1	SQL Queries	13
3.2	Ergebnisse	13
4	Analytische Abfragen - Sakila	15
4.1	SQL Queries	15
4.2	Ergebnisse	16
5	LISTAGG - Sakila	18
5.1	SQL Queries	18
5.2	Ergebnisse	19
6	Materialisierte Sichten - Sakila	20
6.1	SQL Queries	20
6.2	Ergebnisse	21

1 PIVOT und UNPIVOT - Sakila

1.1 SQL Queries

```
1  -- 1
2  SELECT *
3  FROM (SELECT f.title AS Titel, l1.name AS L, l2.name AS OL
4  FROM film f
5  INNER JOIN LANGUAGE l1
6    ON (f.language_id = l1.language_id)
7  INNER JOIN LANGUAGE l2
8    ON (f.original_language_id = l2.language_id)
9  WHERE f.release_year = 1993)
10 UNPIVOT (name FOR LANGUAGE IN (L, OL))
11 ORDER BY Titel;
12
13 -- 2
14 SELECT *
15 FROM (SELECT C.name AS category, l.name, LENGTH AS LENGTH
16 FROM film
17 INNER JOIN film_category
18   USING (film_id)
19 INNER JOIN category C
20   USING (category_id)
21 INNER JOIN LANGUAGE l
22   USING (language_id)
23 WHERE C.name IN ('Family', 'Children', 'Animation'))
24 PIVOT (
25   AVG(LENGTH) AS LENGTH
26   FOR category IN ('Family' AS Family, 'Children' AS Children,
27   → 'Animation' AS Animation)
28 );
29
30 -- 3
31 SELECT *
32 FROM (SELECT COUNT(*) AS Anzahl,
33   CASE WHEN GROUPING(name) = 1 THEN 'Gesamt' ELSE name END AS
34   → Kategorie,
35   CASE WHEN GROUPING(store_id) = 1 THEN 'Gesamt' ELSE TO_CHAR(store_id)
36   → END AS Filiale
37 FROM inventory
38 INNER JOIN film_category
39   USING (film_id)
40 INNER JOIN category
41   USING (category_id)
42 GROUP BY CUBE (store_id, name))
43 PIVOT (
44   SUM(Anzahl)
```

43

1.2 Ergebnisse

44 rows			
	TITEL	LANGUAGE	NAME
1	ATTRACTION NEWTON	OL	Mandarin
2	ATTRACTION NEWTON	L	German
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	OL	German
10	CELEBRITY HORN	L	Italian

Abbildung 1: Aufgabe 1 Ergebnis 1

[illegible]

Abbildung 2: Aufgabe 1 Ergebnis 2

	KATEGORIE ▾	FILIALE1 ▾	FILIALE2 ▾	FILIALE3 ▾	FILIALE4 ▾	FILIALE5 ▾	FILIALE6 ▾	GESAMT ▾
1	Family		47	47	52	55	51	310
2	Foreign		53	45	53	49	51	300
3	Children		43	45	49	37	47	269
4	Classics		40	53	42	41	47	270
5	Documentary		50	52	50	55	46	294
6	New		52	45	43	46	43	275
7	Sports		51	61	54	60	72	344
8	Gesamt		757	755	753	780	784	4581
9	Horror		43	40	40	43	37	248
10	Travel		45	30	37	29	54	235

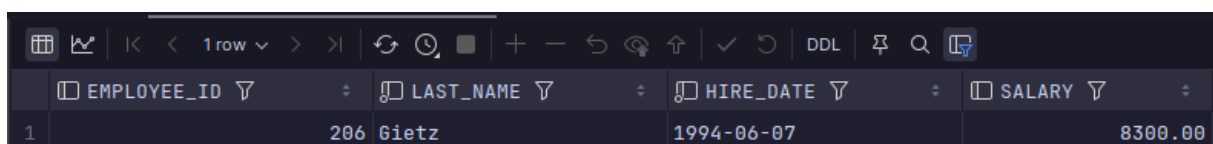
Abbildung 3: Aufgabe 1 Ergebnis 3

2 Hierarchische Abfragen - Human Resources

2.1 SQL Queries

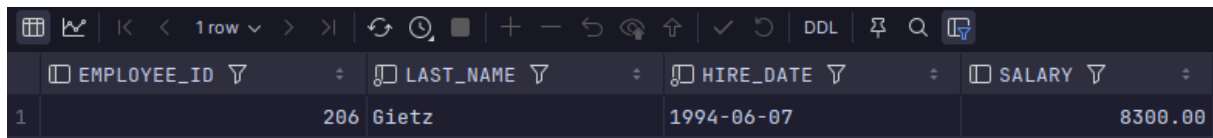
```
1  -- 1a
2  SELECT e.EMPLOYEE_ID, e.LAST_NAME, e.HIRE_DATE, e.SALARY
3  FROM EMPLOYEES m
4  INNER JOIN EMPLOYEES e
5       ON e.MANAGER_ID = m.EMPLOYEE_ID
6  WHERE m.EMPLOYEE_ID = 205;
7
8  -- 1b(unter => ohne 205)
9  SELECT EMPLOYEE_ID, LAST_NAME, HIRE_DATE, SALARY
10 FROM (SELECT EMPLOYEE_ID, LAST_NAME, HIRE_DATE, SALARY
11 FROM EMPLOYEES
12 START WITH EMPLOYEE_ID = 205
13 CONNECT BY NOCYCLE PRIOR EMPLOYEE_ID = MANAGER_ID)
14 WHERE EMPLOYEE_ID != 205;
15
16 -- 2
17 SELECT EMPLOYEE_ID, MANAGER_ID, LAST_NAME, LEVEL
18 FROM EMPLOYEES
19 WHERE LEVEL = 3
20 START WITH EMPLOYEE_ID = 102
21 CONNECT BY NOCYCLE PRIOR EMPLOYEE_ID = MANAGER_ID;
22
23 -- 3
24 SELECT LPAD(LAST_NAME, LENGTH(LAST_NAME) + LEVEL - 1, '_') AS LAST_NAME,
25        EMPLOYEE_ID,
26        MANAGER_ID,
27        LEVEL AS Hierarchy_Level
28 FROM EMPLOYEES
29 START WITH EMPLOYEE_ID = 100
30 CONNECT BY NOCYCLE PRIOR EMPLOYEE_ID = MANAGER_ID;
```

2.2 Ergebnisse



	EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
1	206	Gietz	1994-06-07	8300.00

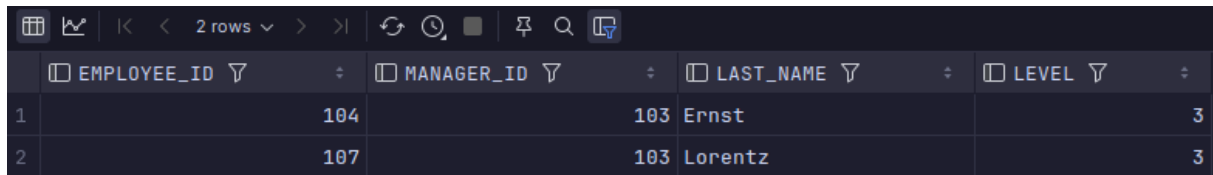
Abbildung 4: Aufgabe 2 Ergebnisse 1



The screenshot shows a database interface with a toolbar at the top. The query result is displayed in a table with the following columns: EMPLOYEE_ID, LAST_NAME, HIRE_DATE, and SALARY. The first row contains the data for employee 206, Gietz, hired on 1994-06-07, with a salary of 8300.00.

	EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
1	206	Gietz	1994-06-07	8300.00

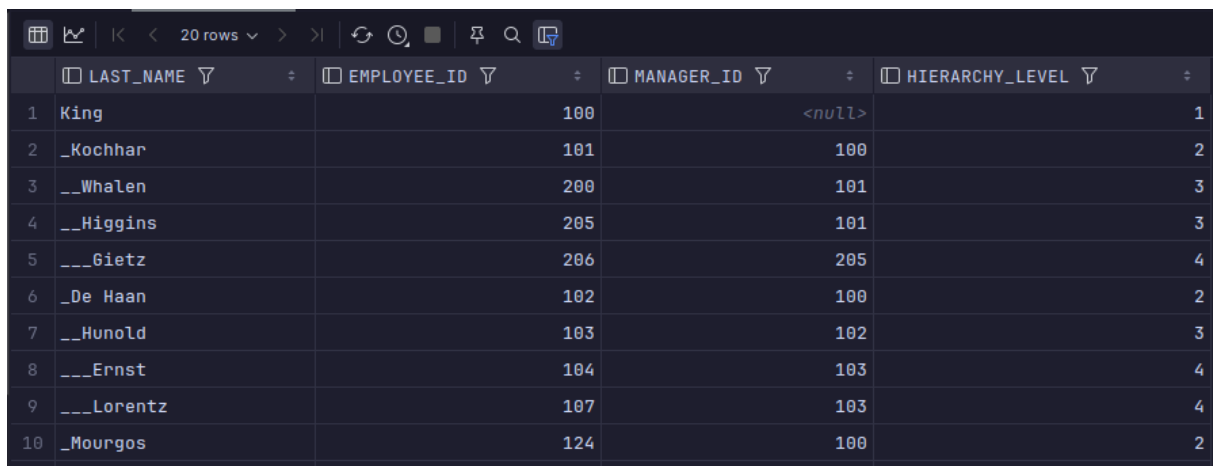
Abbildung 5: Aufgabe 2 Ergebnisse 2



The screenshot shows a database interface with a toolbar at the top. The query result is displayed in a table with the following columns: EMPLOYEE_ID, MANAGER_ID, LAST_NAME, and LEVEL. The first two rows show the hierarchy for employees 104 and 107, both managed by employee 103.

	EMPLOYEE_ID	MANAGER_ID	LAST_NAME	LEVEL
1	104	103	Ernst	3
2	107	103	Lorentz	3

Abbildung 6: Aufgabe 2 Ergebnisse 3



The screenshot shows a database interface with a toolbar at the top. The query result is displayed in a table with the following columns: LAST_NAME, EMPLOYEE_ID, MANAGER_ID, and HIERARCHY_LEVEL. The table lists 10 employees and their hierarchy levels, starting with King at level 1 and ending with Mourgos at level 2.

	LAST_NAME	EMPLOYEE_ID	MANAGER_ID	HIERARCHY_LEVEL
1	King	100	<null>	1
2	_Kochhar	101	100	2
3	__Whalen	200	101	3
4	__Higgins	205	101	3
5	___Gietz	206	205	4
6	_De Haan	102	100	2
7	__Hunold	103	102	3
8	___Ernst	104	103	4
9	___Lorentz	107	103	4
10	_Mourgos	124	100	2

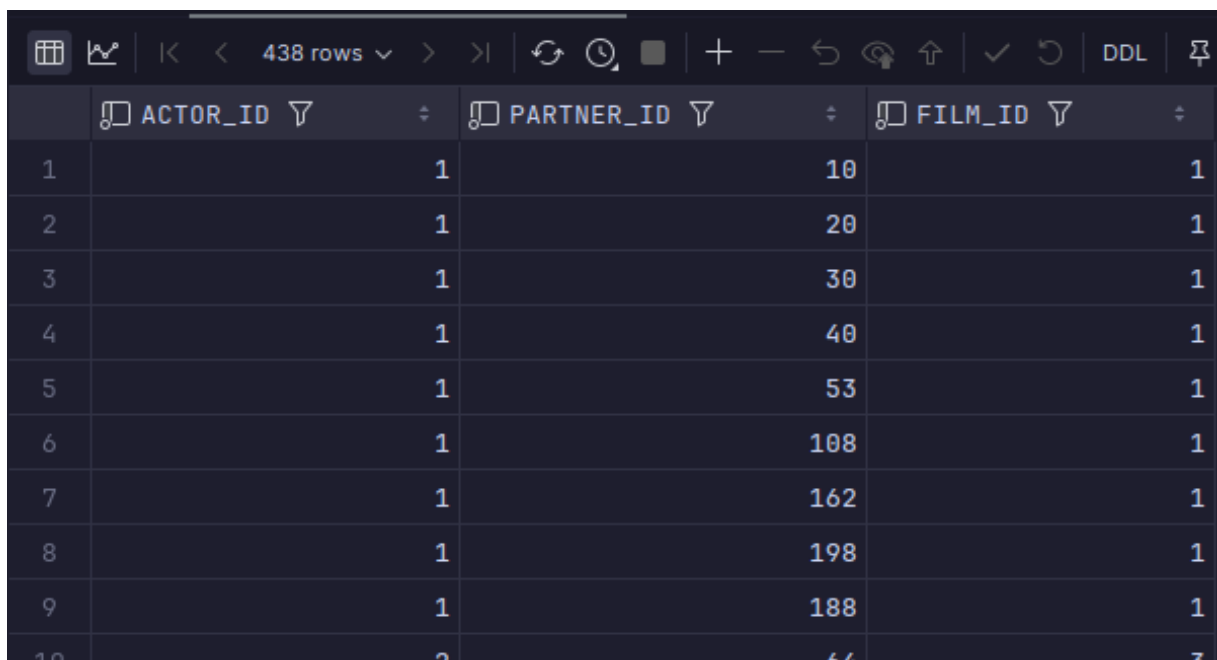
Abbildung 7: Aufgabe 2 Ergebnisse 4

3 Hierarchische Abfragen - Sakila

3.1 SQL Queries

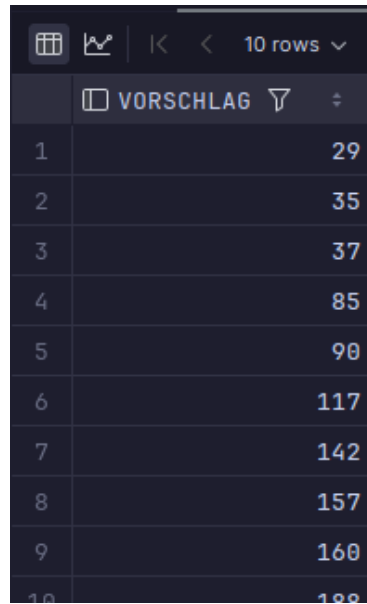
```
1  -- a
2  CREATE OR REPLACE VIEW partners AS
3  SELECT fa1.actor_id, fa2.actor_id AS partner_id, fa1.film_id
4  FROM film_actor fa1
5  INNER JOIN film_actor fa2
6      ON fa1.film_id = fa2.film_id AND fa1.actor_id != fa2.actor_id
7  WHERE fa1.FILM_ID <= 13
8  ORDER BY fa1.actor_id;
9
10 SELECT *
11 FROM partners;
12
13 SELECT DISTINCT partner_id AS Vorschlag
14 FROM partners
15 WHERE LEVEL = 2
16 START WITH actor_id = (SELECT actor_id FROM ACTOR WHERE FIRST_NAME =
    ↳ 'JULIANNE' AND LAST_NAME = 'DENCH')
17 CONNECT BY NOCYCLE PRIOR partner_id = actor_id
18           AND PRIOR film_id != film_id
19 ORDER BY Vorschlag;
```

3.2 Ergebnisse



	ACTOR_ID	PARTNER_ID	FILM_ID
1	1	10	1
2	1	20	1
3	1	30	1
4	1	40	1
5	1	53	1
6	1	108	1
7	1	162	1
8	1	198	1
9	1	188	1
10	2	66	3

Abbildung 8: Aufgabe 3 Ergebnisse 1



	VORSCHLAG
1	29
2	35
3	37
4	85
5	90
6	117
7	142
8	157
9	160
10	188

Abbildung 9: Aufgabe 3 Ergebnisse 2

4 Analytische Abfragen - Sakila

4.1 SQL Queries

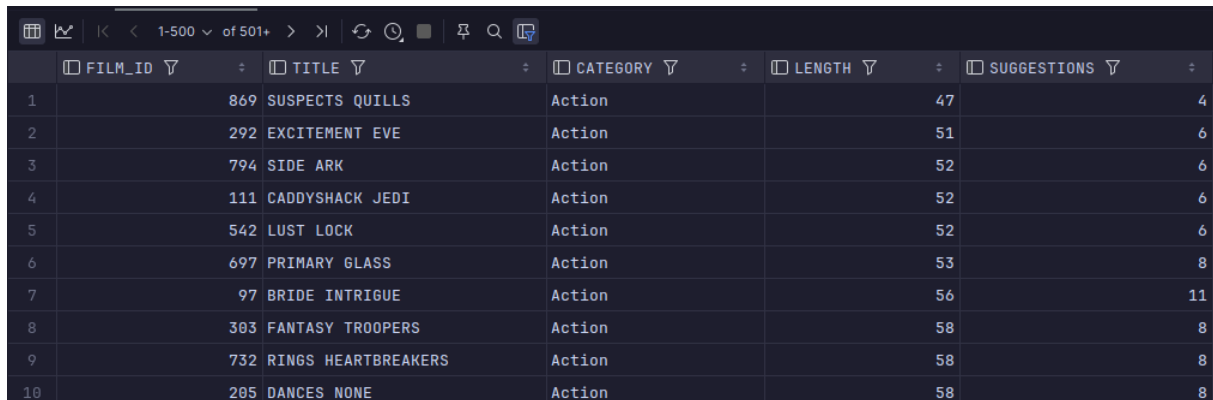
```
1  -- 1
2  -- -1 to exclude the current row
3  SELECT f.FILM_ID,
4         TITLE,
5         C.NAME AS CATEGORY,
6         LENGTH,
7         COUNT(*) OVER (PARTITION BY C.CATEGORY_ID ORDER BY LENGTH RANGE
8         ↪ BETWEEN 5 PRECEDING AND 5 FOLLOWING) -
9         1 AS Suggestions
10 FROM FILM f
11 INNER JOIN FILM_CATEGORY fc
12     ON f.FILM_ID = fc.FILM_ID
13 INNER JOIN CATEGORY C
14     ON fc.CATEGORY_ID = C.CATEGORY_ID;
15
16 -- 2
17 SELECT name AS category, title, release_year, num
18 FROM (SELECT name, title, release_year, ROW_NUMBER() OVER (PARTITION BY
19     ↪ name ORDER BY release_year DESC) AS num
20 FROM film
21 INNER JOIN film_category
22     USING (film_id)
23 INNER JOIN category
24     USING (category_id))
25 WHERE num < 4;
26
27 -- 3a
28 SELECT C.FIRST_NAME,
29        C.LAST_NAME,
30        r.RENTAL_DATE,
31        LAG(r.RENTAL_DATE, 1) OVER (PARTITION BY C.CUSTOMER_ID ORDER BY
32        ↪ r.rental_date ) AS previos_rental_date
33 FROM CUSTOMER C
34 INNER JOIN RENTAL r
35     ON C.CUSTOMER_ID = r.CUSTOMER_ID;
36
37 -- 3b
38 SELECT FIRST_NAME, LAST_NAME, DAYS
39 FROM (SELECT C.FIRST_NAME,
40        C.LAST_NAME,
41        r.RENTAL_DATE - LAG(r.RENTAL_DATE, 1) OVER (PARTITION BY
42        ↪ C.CUSTOMER_ID ORDER BY r.rental_date ) AS DAYS
43 FROM CUSTOMER C
44 INNER JOIN RENTAL r
```

```

41     ON C.CUSTOMER_ID = r.CUSTOMER_ID)
42 WHERE DAYS > 180;

```

4.2 Ergebnisse



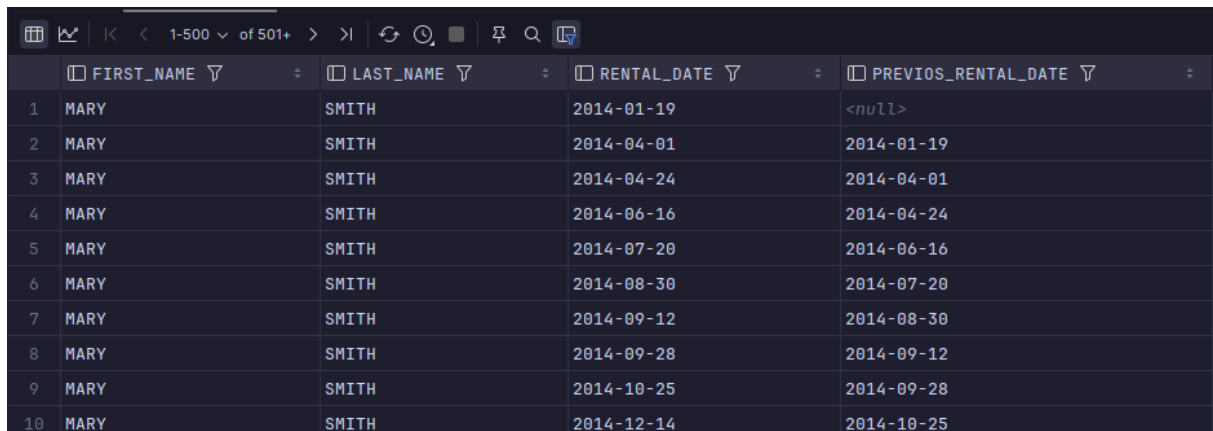
	FILM_ID	TITLE	CATEGORY	LENGTH	SUGGESTIONS
1	869	SUSPECTS QUILLS	Action	47	4
2	292	EXCITEMENT EVE	Action	51	6
3	794	SIDE ARK	Action	52	6
4	111	CADDYSHACK JEDI	Action	52	6
5	542	LUST LOCK	Action	52	6
6	697	PRIMARY GLASS	Action	53	8
7	97	BRIDE INTRIGUE	Action	56	11
8	303	FANTASY TROOPERS	Action	58	8
9	732	RINGS HEARTBREAKERS	Action	58	8
10	205	DANCES NONE	Action	58	8

Abbildung 10: Aufgabe 4 Ergebnisse 1



	CATEGORY	TITLE	RELEASE_YEAR	NUM
1	Action	CLUELESS BUCKET	2008	1
2	Action	LORD ARIZONA	2008	2
3	Action	ANTITRUST TOMATOES	2008	3
4	Animation	WAIT CIDER	2008	1
5	Animation	ANACONDA CONFESSIONS	2007	2
6	Animation	SUGAR WONKA	2007	3
7	Children	CROOKED FROGMEN	2008	1
8	Children	GRADUATE LORD	2008	2
9	Children	GORGEOUS BINGO	2007	3
10	Classics	DYNAMITE TARZAN	2008	1

Abbildung 11: Aufgabe 4 Ergebnisse 2



The screenshot shows a database interface with a table of rental records. The table has four columns: FIRST_NAME, LAST_NAME, RENTAL_DATE, and PREVIOUS_RENTAL_DATE. The data shows a sequence of rentals for Mary Smith from January 2014 to December 2014, with each rental date corresponding to the previous rental date.

	FIRST_NAME	LAST_NAME	RENTAL_DATE	PREVIOUS_RENTAL_DATE
1	MARY	SMITH	2014-01-19	<null>
2	MARY	SMITH	2014-04-01	2014-01-19
3	MARY	SMITH	2014-04-24	2014-04-01
4	MARY	SMITH	2014-06-16	2014-04-24
5	MARY	SMITH	2014-07-20	2014-06-16
6	MARY	SMITH	2014-08-30	2014-07-20
7	MARY	SMITH	2014-09-12	2014-08-30
8	MARY	SMITH	2014-09-28	2014-09-12
9	MARY	SMITH	2014-10-25	2014-09-28
10	MARY	SMITH	2014-12-14	2014-10-25

Abbildung 12: Aufgabe 4 Ergebnisse 3



The screenshot shows a database interface with a table of customer information. The table has three columns: FIRST_NAME, LAST_NAME, and DAYS. The data shows 10 rows of customer information, including names like Kathleen Adams, Katherine Rivera, Emily Diaz, Alicia Mills, Annette Olson, Lucy Wheeler, Brian Wyman, Jonathan Scarborough, Glen Talbert, and Lester Kraus, along with their respective rental days.

	FIRST_NAME	LAST_NAME	DAYS
1	KATHLEEN	ADAMS	238
2	KATHERINE	RIVERA	185
3	EMILY	DIAZ	190
4	ALICIA	MILLS	190
5	ANNETTE	OLSON	204
6	LUCY	WHEELER	193
7	BRIAN	WYMAN	191
8	JONATHAN	SCARBOROUGH	280
9	GLEN	TALBERT	213
10	LESTER	KRAUS	181

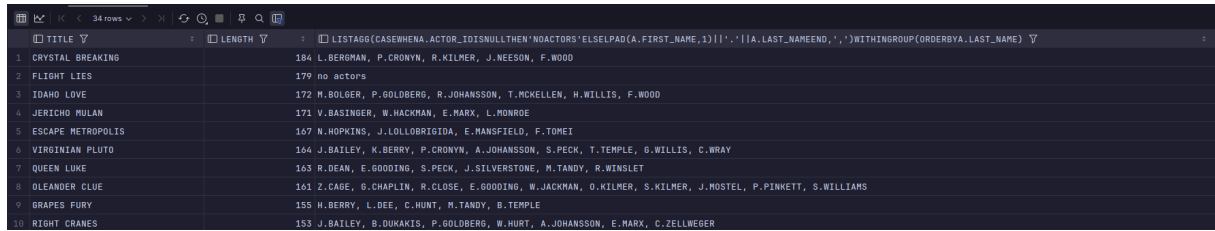
Abbildung 13: Aufgabe 4 Ergebnisse 4

5 LISTAGG - Sakila

5.1 SQL Queries

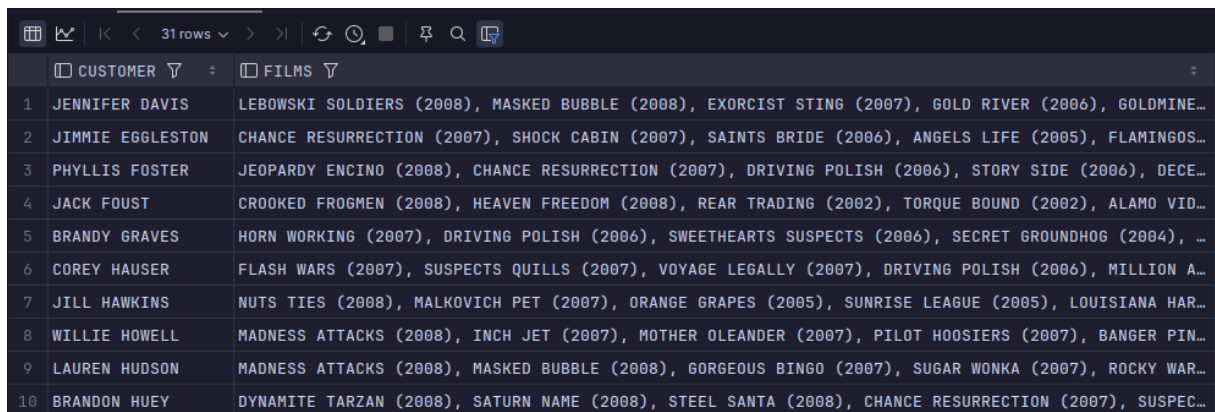
```
1  -- 1
2  SELECT f.TITLE,
3         f.LENGTH,
4         LISTAGG(
5             CASE
6                 WHEN a.ACTOR_ID IS NULL THEN
7                     'no actors'
8                 ELSE
9                     LPAD(a.FIRST_NAME, 1) || '.' || a.LAST_NAME END, ','
10                    ↪ ' '
11            WITHIN GROUP ( ORDER BY a.LAST_NAME )
12  FROM FILM f
13  LEFT JOIN FILM_ACTOR fa
14      ON f.FILM_ID = fa.FILM_ID
15  LEFT JOIN ACTOR a
16      ON fa.ACTOR_ID = a.ACTOR_ID
17  WHERE f.RELEASE_YEAR = 1991
18  GROUP BY f.TITLE, f.LENGTH
19  ORDER BY f.LENGTH DESC;
20
21 -- 2
22 SELECT C.FIRST_NAME || ' ' || C.LAST_NAME AS customer,
23        LISTAGG(f.TITLE || ' (' || f.RELEASE_YEAR || ')', ', ')
24        WITHIN GROUP ( ORDER BY f.RELEASE_YEAR DESC) AS films
25 FROM CUSTOMER C
26 INNER JOIN STORE s
27     ON C.STORE_ID = s.STORE_ID
28 INNER JOIN RENTAL r
29     ON C.CUSTOMER_ID = r.CUSTOMER_ID
30 INNER JOIN INVENTORY i
31     ON r.INVENTORY_ID = i.INVENTORY_ID
32 INNER JOIN FILM f
33     ON i.FILM_ID = f.FILM_ID
34 WHERE s.STORE_ID = 4
35 GROUP BY C.FIRST_NAME, C.LAST_NAME
36 ORDER BY LAST_NAME;
```

5.2 Ergebnisse



	TITLE	LENGTH	LISTAGG(CASEWHEN A.ACTOR_ID IS NULL THEN 'NO ACTORS' ELSE PAD(A.FIRST_NAME, 1) ' ' A.LAST_NAME END, ',') WITHIN GROUP (ORDER BY A.LAST_NAME)
1	CRYSTAL BREAKING	184	L.BERGMAN, P.CRONYN, R.KILMER, J.NEESON, F.WOOD
2	FLIGHT LIES	179	no actors
3	IDAHO LOVE	172	M.BOLGER, P.GOLDBERG, R.JOHANSSON, T.MCKELLEN, H.WILLIS, F.WOOD
4	JERICO MULAN	171	V.BASINDER, W.HACKMAN, E.MARK, L.MUNROE
5	ESCAPE METROPOLIS	167	N.HOPKINS, J.LOLLOBRIGIDA, E.MANSFIELD, F.TOMEI
6	VIRGINIAN PLUTO	164	J.BAILEY, K.BERRY, P.CRONYN, A.JOHANSSON, S.PECK, T.TEMPLE, G.WILLIS, C.WRAY
7	QUEEN LUKE	163	R.DEAN, E.GOODING, S.PECK, J.SILVERSTONE, M.TANDY, R.WINSLET
8	OLEANDER CLUE	161	Z.CABE, G.CHAPLIN, R.CLOSE, E.GOODING, W.JACKMAN, O.KILMER, S.KILMER, J.MOSTEL, P.PINKETT, S.WILLIAMS
9	GRAPES FURY	155	H.BERRY, L.DEE, C.HUNT, M.TANDY, B.TEMPLE
10	RIGHT CRANES	153	J.BAILEY, B.DIMAKIS, P.GOLDBERG, W.HURT, A.JOHANSSON, E.MARK, C.ZELLMER

Abbildung 14: Aufgabe 5 Ergebnisse 1



	CUSTOMER	FILMS
1	JENNIFER DAVIS	LEBOWSKI SOLDIERS (2008), MASKED BUBBLE (2008), EXORCIST STING (2007), GOLD RIVER (2006), GOLDMINE...
2	JIMMIE EGGLESTON	CHANCE RESURRECTION (2007), SHOCK CABIN (2007), SAINTS BRIDE (2006), ANGELS LIFE (2005), FLAMINGOS...
3	PHYLLIS FOSTER	JEOPARDY ENCINO (2008), CHANCE RESURRECTION (2007), DRIVING POLISH (2006), STORY SIDE (2006), DECE...
4	JACK FOUST	CROOKED FROGMEN (2008), HEAVEN FREEDOM (2008), REAR TRADING (2002), TORQUE BOUND (2002), ALAMO VID...
5	BRANDY GRAVES	HORN WORKING (2007), DRIVING POLISH (2006), SWEETHEARTS SUSPECTS (2006), SECRET GROUNDHOG (2004), ...
6	COREY HAUSER	FLASH WARS (2007), SUSPECTS QUILLS (2007), VOYAGE LEGALLY (2007), DRIVING POLISH (2006), MILLION A...
7	JILL HAWKINS	NUTS TIES (2008), MALKOVICH PET (2007), ORANGE GRAPES (2005), SUNRISE LEAGUE (2005), LOUISIANA HAR...
8	WILLIE HOWELL	MADNESS ATTACKS (2008), INCH JET (2007), MOTHER OLEANDER (2007), PILOT HOOSIERS (2007), BANGER PIN...
9	LAUREN HUDSON	MADNESS ATTACKS (2008), MASKED BUBBLE (2008), GORGEOUS BINGO (2007), SUGAR WONKA (2007), ROCKY WAR...
10	BRANDON HUEY	DYNAMITE TARZAN (2008), SATURN NAME (2008), STEEL SANTA (2008), CHANCE RESURRECTION (2007), SUSPEC...

Abbildung 15: Aufgabe 5 Ergebnisse 2

6 Materialisierte Sichten - Sakila

6.1 SQL Queries

```
1  -- a
2  CREATE OR REPLACE VIEW UE03_06a
3  AS
4  WITH revenues AS (SELECT SUM(amount) AS Umsatz,
5      C.name AS Kategorie,
6      r.staff_id
7  FROM category C
8  INNER JOIN film_category
9      USING (category_id)
10 INNER JOIN inventory i
11     USING (film_id)
12 INNER JOIN rental r
13     USING (inventory_id)
14 INNER JOIN payment
15     USING (rental_id)
16 GROUP BY C.name, r.staff_id)
17 SELECT Kategorie, "1_UMSATZ", "2_UMSATZ", ROUND("1_UMSATZ" / "2_UMSATZ",
18     ↪ 2) AS Verhältnis
19 FROM revenues r
20     PIVOT (
21         SUM(Umsatz) AS UMSATZ
22         FOR staff_id IN ('1' AS "1", '2' AS "2")
23     )
24 ORDER BY Kategorie;
25
26 SELECT *
27 FROM UE03_06a;
28
29 --b
30 CREATE MATERIALIZED VIEW UE03_06b
31 REFRESH COMPLETE ON DEMAND
32 AS
33 SELECT *
34 FROM UE03_06a;
35
36 SELECT *
37 FROM UE03_06b;
38
39 -- c
40 CREATE MATERIALIZED VIEW UE03_06c
41 REFRESH COMPLETE START WITH TRUNC(SYSDATE) + 23.5 / 24 NEXT SYSDATE +
42     ↪ 1
43 AS
44 SELECT *
```

```

43 FROM UE03_06b;
44
45 SELECT *
46 FROM UE03_06c;
47
48 -- cleanup
49 DROP VIEW UE03_06a;
50 DROP MATERIALIZED VIEW UE03_06b;
51 DROP MATERIALIZED VIEW UE03_06c;

```

6.2 Ergebnisse

	KATEGORIE	"1_UMSATZ"	"2_UMSATZ"	VERHÄLTNIS
1	Action	2963.02	3232.3	0.92
2	Animation	3128.76	3073.29	1.02
3	Children	2818.14	2330.39	1.21
4	Classics	2608.66	2159.89	1.21
5	Comedy	2649.26	2715.84	0.98
6	Documentary	3244.97	3111.05	1.04
7	Drama	2375.35	2499.58	0.95
8	Family	3373.11	2988.13	1.13
9	Foreign	2759.16	3167.68	0.87
10	Games	2712.76	2560.6	1.06

Abbildung 16: Aufgabe 6 Ergebnisse