DES3UE WS 2021 Übung 4

Abgabetermin: 04.11.2021

☐ **DES3UEG1: Niklas** **Name** Angelos Angelis **Aufwand in h** 9
☒ **DES3UEG2: Niklas**
☐ **DES3UEG3: Traxler** **Punkte** _____ **Kurzzeichen Tutor** _____

Hinweise und Richtlinien:

- Übungsausarbeitungen müssen den im elearning angegebenen Formatierungsrichtlinien entsprechen – Nichtbeachtung dieser Formatierungsrichtlinien führt zu Punkteabzug.
- 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.

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. Hierarchische Abfragen – Human Resources (5 Punkte – 2+1+1+1)

1. Neena Kochhar verlässt das Unternehmen. Ihr Nachfolger wünscht Berichte über die Angestellten, die Kochhar 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 Kochhar direkt unterstellt sind
   b. Die **gesamte** Organisationsstruktur unter Kochhar (Angestelltennummer: 101)

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

4. Erstellen Sie einen Bericht, der alle Angestellten enthält (Vor- und Nachname) und bestimmt, von wie vielen Personen er/sie Vorgesetzte/r ist. Zählen Sie nicht nur die direkt unterstellten, sondern die **gesamte** Hierarchie darunter (zB Steven King ist der Vorgesetzte von 19 Personen). Sie benötigen für diese Aufgabe die Pseudo-Spalte *connect_by_root* - recherchieren Sie diese!

Ergebnis-Auszug:

LAST_NAME	FIRST_NAME	BOSS_OF
Ernst	Bruce	0
Lorentz	Diana	0
Gietz	William	0
Kochhar	Neena	3
Hartstein	Michael	1

2. Hierarchische Abfragen – Sakila-Datenbank (6 Punkte)

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.

1. Erstellen Sie eine View partners, die Ihnen Schauspielerinnen und Schauspieler-Kollegen ausgibt (wenn sie in irgendeinem Film miteinander gespielt haben). Achten 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)*

2. Erstellen Sie aufbauend auf der obigen View eine nach IDs sortierte Vorschlagsliste für Schauspielerin „JULIANNE DENCH“. *(4 Punkte, 10 Zeilen)*

Ergebnis-Auszug:

VORSCHLAG
29
35
37

3. PIVOT und UNPIVOT (6 Punkte – 2+2+2)

1. Erstellen Sie 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

2. Sie sollen für jede Kategorie Anzahl der Filme (Inventar!) pro Filiale und insgesamt in 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.

Code-Ausschnitt:

```
...
SELECT COUNT(*) AS anzahl,
       CASE WHEN /* ergänzen */ THEN 'Gesamt' ELSE name END AS Kategorie,
       CASE WHEN /* ergänzen */ 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

3. Erstellen Sie eine Anfrage die für jeden Film aus dem Jahr 1990 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. *(54 Zeilen)*

Ergebnis-Auszug:

TITEL	AR	SPRACHE
ALICE FANTASIA	OL	French
ALICE FANTASIA	L	Mandarin
ALIEN CENTER	L	French
ALIEN CENTER	OL	German
CASABLANCA SUPER	OL	German
CASABLANCA SUPER	L	Japanese

4. Analytische Abfragen (7 Punkte – 2+2+3)

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 10 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 15 weitere Filme, deren Laufzeit 10 Minuten kürzer oder länger ist.

   Ergebnis-Auszug:

FILM_ID	TITLE	NAME	LENGTH	SUGGESTIONS
869	SUSPECTS QUILLS	Action	47	6
292	EXCITEMENT EVE	Action	51	12
542	LUST LOCK	Action	52	12
794	SIDE ARK	Action	52	12

2. Geben Sie zu jeder Filmkategorie drei Filme aus, wählen Sie jene Filme, die neueren Datums sind. Geben Sie den Kategorienamen aus, den Filmtitel und das Erscheinungsjahr. Verwenden Sie ROW_NUMBER().
3. Ermitteln Sie welche Kunden schon einmal mehr als 180 Tage zwischen zwei Verleihvorgängen verstreichen haben lassen.
   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

Aufgabe1)

```
-- 1. Neena Kochhar verlässt das Unternehmen. Ihr Nachfolger wünscht Berichte über die
-- Angestellten, die Kochhar 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 Kochhar direkt unterstellt sind
-- b. Die gesamte Organisationsstruktur unter Kochhar (Angestelltennummer: 101)
SELECT E.EMPLOYEE_ID,E.LAST_NAME,E.HIRE_DATE,E.SALARY
FROM EMPLOYEES E
WHERE E.MANAGER_ID = 101;
```

	EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
1	200	Whalen	1987-09-17	4400.00
2	205	Higgins	1994-06-07	12000.00

```
SELECT E.EMPLOYEE_ID,E.LAST_NAME,E.HIRE_DATE,E.SALARY
FROM EMPLOYEES E
START WITH E.EMPLOYEE_ID = 101
CONNECT BY PRIOR E.EMPLOYEE_ID = E.MANAGER_ID;
```

	EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
1	101	Kochhar	1989-09-21	17000.00
2	200	Whalen	1987-09-17	4400.00
3	205	Higgins	1994-06-07	12000.00
4	206	Gietz	1994-06-07	8300.00

```
-- 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)
SELECT E.EMPLOYEE_ID,E.MANAGER_ID,E.LAST_NAME,LEVEL
FROM EMPLOYEES E
WHERE LEVEL = 3
START WITH E.EMPLOYEE_ID = 102
CONNECT BY PRIOR E.EMPLOYEE_ID = E.MANAGER_ID;
```

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

```
-- 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.
SELECT LPAD(E.LAST_NAME,LENGTH(E.LAST_NAME)+(LEVEL-1),'_') AS
LAST_NAME,E.EMPLOYEE_ID, E.MANAGER_ID, LEVEL
FROM EMPLOYEES E
START WITH E.EMPLOYEE_ID = 100
CONNECT BY PRIOR E.EMPLOYEE_ID = E.MANAGER_ID;
```

	LAST_NAME	EMPLOYEE_ID	MANAGER_ID	"LEVEL"
1	King	100	<null>	1
2	_Kochhar	101	100	2
3	__Whalen	200	101	3
4	__Higgins	205	101	3
17	__Taylor	176	149	3
18	__Grant	178	149	3
19	_Hartstein	201	100	2
20	__Fay	202	201	3

```
-- 4. Erstellen Sie einen Bericht, der alle Angestellten enthält (Vor- und Nachname) und bestimmt,
-- von wie vielen Personen er/sie Vorgesetzte/r ist. Zählen Sie nicht nur die direkt unterstellten,
-- sondern die gesamte Hierarchie darunter (zB Steven King ist der Vorgesetzte von 19 Personen).
-- Sie benötigen für diese Aufgabe die Pseudo-Spalte connect_by_root - recherchieren Sie diese!
SELECT E.FIRST_NAME,E.LAST_NAME,COUNT(connect_by_root MANAGER_ID)
FROM EMPLOYEES E
CONNECT BY PRIOR E.MANAGER_ID= E.EMPLOYEE_ID
GROUP BY E.FIRST_NAME, E.LAST_NAME;
```

	FIRST_NAME	LAST_NAME	BOSSOF
8	Alexander	Hunold	3
9	Steven	King	19

Aufgabe2)

```
-- 1. Erstellen Sie eine View partners, die Ihnen Schauspielerinnen und
Schauspieler-Kollegen ausgibt
-- (wenn sie in irgendeinem Film miteinander gespielt haben). Achten 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)

CREATE VIEW partners(Actor,partners,inFilm) AS
SELECT FA.ACTOR_ID,FA2.ACTOR_ID,FA.FILM_ID
FROM FILM_ACTOR FA
JOIN FILM_ACTOR FA2 on FA.FILM_ID = FA2.FILM_ID
WHERE FA.FILM_ID <= 13
AND FA.ACTOR_ID <> FA2.ACTOR_ID
CONNECT BY NOCYCLE PRIOR FA.FILM_ID = FA2.FILM_ID;

SELECT *
FROM partners
```

	ACTOR	PARTNERS	INFILM
1	198	188	1
2	162	188	1
3	108	188	1
4	10	188	1
5	1	188	1
435	176	91	13
436	77	91	13
437	114	91	13
438	94	91	13

```
-- 2. Erstellen Sie aufbauend auf der obigen View eine nach IDs sortierte
Vorschlagsliste für
-- Schauspielerin „JULIANNE DENCH“. (4 Punkte, 10 Zeilen)

SELECT partners AS Vorschlag
FROM partners
WHERE Actor = ( SELECT ACTOR_ID
                FROM ACTOR
                WHERE FIRST_NAME = 'JULIANNE'
                AND LAST_NAME = 'DENCH')
ORDER BY partners;
```

	VORSCHLAG
1	2
2	19
3	24
4	64

Aufgabe 3)

```
-- 1. Erstellen Sie eine Pivot-Tabelle für die Kategorien „Family“,
„Children“ und „Animation“
-- (Spalten), welche die Durchschnittslänge der Filme pro Sprache (Zeilen)
angibt. (6 Zeilen, 4
-- Spalten)
SELECT *
FROM    (SELECT CATEGORY.NAME AS CAT_NAME,LANGUAGE.NAME AS LANG_NAME,LENGTH
         FROM FILM_CATEGORY FC
         JOIN CATEGORY USING (CATEGORY_ID)
         JOIN FILM USING (FILM_ID)
         JOIN LANGUAGE USING (LANGUAGE_ID))
PIVOT(
     AVG(LENGTH)
     FOR CAT_NAME
     IN ('Family' AS Family,
     'Children' AS Children,
     'Animation' AS Animation)
     );
```

	LANG_NAME	FAMILY	CHILDREN	ANIMATION
1	Japanese	116.76470588235294117647058823529411764705	120	113.7
2	Italian	115.91666666666666666666666666666666666667	97.181818181818181818181818181818181818	93.666666666666666666666666666666666667
3	French	102.71428571428571428571428571428571428571	120.0625	99.1
4	Mandarin	108.08333333333333333333333333333333333333	129.8	105.666666666666666666666666666666666667
5	German	107	111	112.866666666666666666666666666666666667
6	English	145.875	80.3	128.92307692307692307692307692307692307077

```
-- 2. Sie sollen für jede Kategorie Anzahl der Filme (Inventar!) pro Filiale
und insgesamt in
-- 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.
SELECT COUNT(*) AS anzahl,
 CASE WHEN GROUPING(name) = 1 THEN 'Gesamt' ELSE name END AS Kategorie,
 CASE WHEN GROUPING(store_id) = 1 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 CUBE (store_id, name);
```

	ANZAHL	KATEGORIE	FILIALE
1	4581	Gesamt	Gesamt
2	275	New	Gesamt
3	300	Drama	Gesamt
4	276	Games	Gesamt
5	232	Music	Gesamt

115	51	Foreign	6
116	48	Children	6
117	47	Classics	6
118	56	Animation	6
119	41	Documentary	6

```
-- 3. Erstellen Sie eine Anfrage die für jeden Film aus dem Jahr 1990 die Sprache
-- und die OriginalSprache 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
WITH Films1990 AS(
    SELECT *
    FROM(SELECT TITLE, LANGUAGE_ID, ORIGINAL_LANGUAGE_ID
         FROM FILM
         WHERE RELEASE_YEAR = 1990)
    UNPIVOT(
    LANGUAGE_ID
    FOR ART
    IN (LANGUAGE_ID AS 'L',
    ORIGINAL_LANGUAGE_ID AS 'OL'))
)

SELECT TITLE, ART, NAME AS LANGUAGE
FROM Films1990
JOIN LANGUAGE USING (LANGUAGE_ID)
ORDER BY TITLE;
```

	TITLE	ART	LANGUAGE
1	ALICE FANTASIA	OL	French
2	ALICE FANTASIA	L	Mandarin
3	ALIEN CENTER	OL	German
4	ALIEN CENTER	L	French
5	CASABLANCA SUPER	OL	German
6	CASABLANCA SUPER	L	Japanese
53	WEREWOLF LOLA	OL	Mandarin
54	WEREWOLF LOLA	L	French
55	WON DARES	OL	English
56	WON DARES	L	Italian

Aufgabe 4)

```
-- 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 10 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 15 weitere Filme, deren Laufzeit 10 Minuten kürzer oder länger ist.
SELECT F.FILM_ID,F.TITLE,C2.NAME,F.LENGTH, COUNT(F.FILM_ID) OVER ( PARTITION BY C2.NAME ORDER BY F.LENGTH RANGE BETWEEN 10 PRECEDING AND 10 FOLLOWING)-1 AS TEST
FROM FILM F
JOIN FILM_CATEGORY FC on F.FILM_ID = FC.FILM_ID
JOIN CATEGORY C2 on C2.CATEGORY_ID = FC.CATEGORY_ID
ORDER BY C2.NAME;
```

	FILM_ID	TITLE	NAME	LENGTH	LONGERTHAN
1	869	SUSPECTS QUILLS	Action	47	6
2	292	EXCITEMENT EVE	Action	51	12
3	794	SIDE ARK	Action	52	12
4	542	LUST LOCK	Action	52	12
5	111	CADDYSHACK JEDI	Action	52	12
6	697	PRIMARY GLASS	Action	53	13
7	97	BRIDE INTRIGUE	Action	56	15
495	498	KILLER INNOCENT	Family	161	5
496	345	GABLES METROPOLIS	Family	161	5
497	822	SOUP WISDOM	Family	169	7
498	43	ATLANTIS CAUSE	Family	170	7
499	359	GLADIATOR WESTWARD	Family	173	6
500	832	SPLASH GUMP	Family	175	7

```
-- Geben Sie zu jeder Filmkategorie drei Filme aus, wählen Sie jene Filme, die neueren Datums
-- sind. Geben Sie den Kategorienamen aus, den Filmtitel und das Erscheinungsjahr. Verwenden
-- Sie ROW_NUMBER().
SELECT *
FROM
    (SELECT ROW_NUMBER() over (PARTITION BY C.NAME ORDER BY C.NAME DESC) AS row_num,
            C.NAME AS Kategorie,F.TITLE AS FILM
    FROM FILM F
    JOIN FILM_CATEGORY FC on F.FILM_ID = FC.FILM_ID
    JOIN CATEGORY C on C.CATEGORY_ID = FC.CATEGORY_ID)
```

```
WHERE row_num <= 3;
```

	ROW_NUM	KATEGORIE	FILM
1	1	Action	AMADEUS HOLY
2	2	Action	SKY MIRACLE
3	3	Action	SIDE ARK
4	1	Animation	ALTER VICTORY
5	2	Animation	WONKA SEA
6	3	Animation	WATCH TRACY
43	1	Sports	ALADDIN CALENDAR
44	2	Sports	SIERRA DIVIDE
45	3	Sports	VICTORY ACADEMY
46	1	Travel	BOILED DARES
47	2	Travel	BOONDOCK BALLROOM
48	3	Travel	BORN SPINAL

```
-- Ermitteln Sie welche Kunden schon einmal mehr als 180 Tage zwischen zwei Verleihvorgängen
-- verstreichen haben lassen.
-- 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).
SELECT FIRST_NAME, LAST_NAME, RENTAL_DATE AS active_rental_date,
       LEAD(RENTAL_DATE, 1) OVER (PARTITION BY CUSTOMER_ID ORDER BY RENTAL_DATE) AS next_rental_date
FROM CUSTOMER
     JOIN RENTAL USING(CUSTOMER_ID)
ORDER BY FIRST_NAME;
```

	FIRST_NAME	LAST_NAME	ACTIVE_RENTAL_DATE	NEXT_RENTAL_DATE
1	AARON	SELBY	2013-12-09	2013-12-25
2	AARON	SELBY	2015-11-03	<null>
3	AARON	SELBY	2013-12-28	2014-01-08
4	AARON	SELBY	2014-01-08	2014-01-25
5	AARON	SELBY	2014-01-25	2014-02-04
495	AMBER	DIXON	2015-07-17	2015-08-10
496	AMBER	DIXON	2015-05-15	2015-07-17
497	AMBER	DIXON	2015-04-29	2015-05-15
498	AMBER	DIXON	2015-04-27	2015-04-29
499	AMBER	DIXON	2015-02-25	2015-04-27
500	AMBER	DIXON	2014-12-29	2015-02-25

```
SELECT FIRST_NAME, LAST_NAME, (next_rental_date - active_rental_date) AS days
FROM (
SELECT FIRST_NAME, LAST_NAME, RENTAL_DATE AS active_rental_date,
       LEAD(RENTAL_DATE, 1) OVER (PARTITION BY CUSTOMER_ID ORDER BY
RENTAL_DATE) AS next_rental_date
FROM CUSTOMER
     JOIN RENTAL USING(CUSTOMER_ID) )
WHERE (next_rental_date - active_rental_date) > 180;
```

	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

9	GLEN	TALBERT	213
10	LESTER	KRAUS	181
11	RAFAEL	ABNEY	194
12	AUSTIN	CINTRON	226