DES3UE WS 2018 Übung 2

Abgabetermin: 17.10.2018, 13:30 Uhr
Abgabeform elektronisch

☐ **DES3UEG1: Niklas** **Name** Niklas Vest **Aufwand in h** 4
☐ **DES3UEG2: Niklas**
☒ **DES3UEG3: Traxler** **Punkte** _____ **Kurzzeichen Tutor** _____

Hinweise und Richtlinien:

Die Übungsausarbeitungen müssen den im eLearning angegebenen Formatierungsrichtlinien entsprechen. Nichtbeachtung dieser Formatierungsrichtlinien führt zu Punkteabzug.

Ziel dieser Übung ist es, die Basiselemente der Anfragesprache SQL mit einem neuen Datenbank-Schema zu wiederholen. Dabei wird das Schema der „Sakila-Datenbank“ für die folgenden Übungen eingeführt.

1. Grundlagen **(6 Punkte,** 1-4: je 0,5, 5-8: 1 Pkt.**)**

1. Erstellen Sie eine Liste aller Schauspieler, Verketten Sie Vor- und Nachname (getrennt durch ein Leerzeichen) und nennen Sie die neue Spalte "Name". Sortieren Sie das Ergebnis nach dem Nachnamen.
2. Geben Sie eine Liste mit Titel und Länge all jener Filme aus, die kürzer als 50 Minuten dauern.
3. Geben Sie eine Liste mit Filmtitel aus, deren Namen an vierter Stelle ein 'A' enthält, geben Sie die Titel so aus, dass jeweils der erste Buchstabe eines Wortes mit einem Großbuchstaben beginnt (zB Atlantis Cause).
4. Geben Sie den Titel jener Filme aus, bei denen eine Original-Sprache eingetragen ist.
5. Geben Sie die Anzahl der verliehenen Filme zwischen 1.1.2015 und 31.12.2015 aus (Start des Verleihvorgangs, *rental_date*).
6. Geben Sie alle Inventar-Ids aus, die noch nie verliehen wurden.
7. Erstellen Sie eine Liste mit Kunden (Vor- und Nachname), die in Newcastle, Linz und London wohnen.
8. Geben Sie für den Kunden mit der ID 420 alle Verleihvorgänge mit dem Start des Verleihvorgangs und dem bezahlten Betrag aus. Geben Sie das Datum im Format ‚Mittwoch, 10. Okt. 2018‘ aus, verwenden Sie dazu die Funktion *to_char* und recherchieren Sie bei Bedarf in der Oracle-Dokumentation.

2. Gruppierungen und Unterabfragen **(12 Punkte,** je 1,5 Pkt**)**

1. Geben Sie den durchschnittlichen Leihpreis pro Filmkategorie aus (sollte ein Film zu mehreren Kategorien gehören, zählt er zu allen). Runden Sie auf zwei Nachkommastellen.
2. Geben Sie die Titel aller Filme aus, die länger dauern als der Film mit der ID 50 und deren Ersetzungskosten größer sind als der Film mit der ID 101.
3. Geben Sie die Titel aller Filme aus, die kürzer als 60 Minuten dauern und in den gleichen Kategorien spielen als die Filme mit den IDs 10, 20 oder 30.

4. Erstellen Sie eine Liste aus Schauspieler (Vor- und Nachname) und Anzahl der Filme, in denen sie mitspielen. Die Liste soll nur jene Schauspieler enthalten, die in mehr als 35 Filmen mitspielen.
5. Erstellen Sie eine Liste mit den Titeln und der Länge jener Filme, die länger als der Durchschnitt sind.
6. Ermitteln Sie die Namen jener Filmkategorien zu denen weniger als 60 Filme gehören.
7. Ermitteln Sie alle Filme, die die längsten in ihrem Erscheinungsjahr sind und geben Sie Titel, Dauer (*length*) sowie Erscheinungsjahr aus.
8. Geben Sie die neun zuletzt verliehenen Filme und das Verleihdatum im Format YYYY-MM-DD aus.

3. Insert, Update und Delete (6 Punkte)

1. Erstellen Sie eine neue Tabelle "new_film", diese soll den gleichen Aufbau wie die Tabelle "film" haben, jedoch nur die neusten Filme (jene Filme, die das höchste Erscheinungsjahr in der Datenbank aufweisen) enthalten. *(1 Punkt)*
2. Fügen Sie den Film "Jason Bourne" mit ID = 1001 in Englisch (*language_id* = 1) mit 5 Tagen Verleihdauer (zum Preis von 1,79) mit Ersetzungskosten von 16,99 in die Tabelle ein. *(1 Punkt)*
3. Erhöhen Sie den Leihpreis der Filme in der Tabelle „new_film“ um 15%, wenn der Leihpreis kleiner als 2 ist. (*0,5 Punkt)*
4. Erstellen Sie eine View, die alle Filme der Tabelle "new_film" enthält, deren Leihpreis maximal 2 beträgt, vergeben Sie die Check-Option. Die View soll nur den Filmtitel, die Beschreibung, den Leihpreis und die Länge enthalten. *(1 Punkt)*
5. Welche Auswirkungen haben COMMIT und ROLLBACK an dieser Stelle (nachdem Sie die View erstellt haben).
6. Können Sie die Datensätze Ihrer neuen View verändern? Wenn ja, führen Sie die Erhöhung der Filme (10%) ein weiteres Mal durch, diesmal auf Ihre View. Wenn nein, warum nicht? (*0,5 Punkt)*
7. Löschen Sie alle Einträge aus der Tabelle new_film, deren Leihpreis über 1.79 liegt. (*0,5 Punkt)*
8. Können Sie den Leihpreis der Filme in der View nun erhöhen? Erklären Sie dieses Verhalten. (*0,5 Punkt)*
9. Löschen Sie die Tabelle "new_film" und Ihre erstellte View wieder. (*0,5 Punkt)*

Ausarbeitung Übung 02

Grundlagen

```
-- 1.1
SELECT first_name || ' ' || last_name AS "Name"
FROM actor
ORDER BY last_name ASC;
```

	NAME
1	DEBBIE AKROYD
2	CHRISTIAN AKROYD
3	KIRSTEN AKROYD
4	KIM ALLEN
5	MERYL ALLEN
6	CUBA ALLEN
7	ANGELINA ASTAIRE
8	RUSSELL BACALL
9	JESSICA BAILEY
10	AUDREY BAILEY
11	HARRISON BALE
12	RENEE BALL
13	JULIA BARRYMORE
14	VIVIEN BASINGER
15	MICHAEL BENING

```
-- 1.2
SELECT
       title,
       length
FROM film
WHERE length < 50;
```

	TITLE	LENGTH
1	ACE GOLDFINGER	48
2	ALIEN CENTER	46
3	HURRICANE AFFAIR	49
4	IRON MOON	46
5	KWAI HOMEWARD	46
6	LABYRINTH LEAGUE	46
7	GROSSE WONDERFUL	49
8	HALLOWEEN NUTS	47
9	HANOVER GALAXY	47
10	HAWK CHILL	47
11	HEAVEN FREEDOM	48
12	HEAVENLY GUN	49
13	HOOK CHARIOTS	49
14	DIVORCE SHINING	47
15	DOORS PRESIDENT	49

```
-- 1.3
SELECT INITCAP(title) AS "Name"
FROM film
WHERE title LIKE '___A%';
```

	NAME
1	Coma Head
2	Comancheros Enemy
3	Creatures Shakespeare
4	Affair Prejudice
5	Breakfast Goldfinger
6	Breaking Home
7	Armageddon Lost
8	Atlantis Cause
9	Attacks Hate
10	Bedazzled Married
11	Behavior Runaway
12	Casablanca Super
13	Cheaper Clyde
14	Escape Metropolis
15	Fatal Haunted

```sql
-- 1.4
SELECT title
FROM film
WHERE original_language_id IS NOT NULL;
```

	TITLE
1	CLOSER BANG
2	CLUB GRAFFITI
3	CLUE GRAIL
4	CLUELESS BUCKET
5	CLYDE THEORY
6	COLDBLOODED DARLING
7	COLOR PHILADELPHIA
8	COMA HEAD
9	COMANCHEROS ENEMY
10	COMFORTS RUSH
11	COMMAND DARLING
12	COMMANDMENTS EXPRESS
13	CONEHEADS SMOOCHY
14	CONFESSIONS MAGUIRE
15	CONFIDENTIAL INTERVIEW

```sql
-- 1.5
SELECT COUNT(*) AS "Nr. of rented films"
FROM rental
WHERE rental_date < TO_DATE('31.12.2015', 'DD.MM.YYYY') AND
      rental_date > TO_DATE('01.01.2015', 'DD.MM.YYYY');
```

	"Nr. of rented films"
1	7037

```sql
-- 1.6
SELECT inventory_id
FROM inventory
MINUS
SELECT inventory_id
FROM rental;
```

	INVENTORY_ID
1	5

```sql
-- 1.7
SELECT
      first_name,
      last_name
FROM customer
      INNER JOIN address USING (address_id)
      INNER JOIN city USING (city_id)
WHERE city IN ('Newcastle', 'Linz', 'London');
```

	FIRST_NAME	LAST_NAME
1	MATTIE	HOFFMAN
2	EDWIN	BURK
3	JILL	HAWKINS
4	CECIL	VINES

```sql
-- 1.8
SELECT
      to_char(rental_date, 'Day, DD.Mon.YYYY') AS rental_date,
      amount
FROM payment p
      INNER JOIN rental r USING (rental_id)
```

```
WHERE p.customer_id = 420;
```

	RENTAL_DATE	AMOUNT
1	Tuesday , 09.Sep.2014	0.43
2	Saturday , 11.Jul.2015	2.98
3	Thursday , 18.Jun.2015	10.76
4	Wednesday, 15.Jul.2015	0.39
5	Saturday , 25.Apr.2015	7.45
6	Friday , 24.Apr.2015	6.36
7	Saturday , 24.May.2014	5.97
8	Wednesday, 01.Jul.2015	1.56
9	Wednesday, 22.Apr.2015	0.95
10	Wednesday, 19.Aug.2015	6.27
11	Friday , 28.Feb.2014	8.12
12	Saturday , 26.Sep.2015	8.97
13	Thursday , 19.Jun.2014	0.89
14	Thursday , 05.Dec.2013	10.71
15	Saturday , 26.Sep.2015	15.54

Gruppierungen und Unterabfragen

```
-- 2.1
SELECT
       ROUND(AVG(rental_rate), 2),
       category_id
FROM film
       INNER JOIN film_category USING (film_id)
GROUP BY category_id;
```

	ROUND(AVG(RENTAL_RATE),2)	CATEGORY_ID
1	1.66	1
2	1.72	6
3	1.64	11
4	1.48	13
5	1.57	2
6	1.83	14
7	1.71	5
8	1.44	4
9	1.71	8
10	1.48	7
11	1.6	3
12	1.62	9
13	1.59	15
14	1.53	12
15	1.87	16

```
-- 2.2
SELECT title
FROM film
WHERE length > (SELECT length
                FROM film
                WHERE film_id = 50) AND
       replacement_cost > (SELECT replacement_cost
                           FROM film
                           WHERE film_id = 101);
```

	TITLE
1	CONSPIRACY SPIRIT
2	GANGS PRIDE
3	WIFE TURN
4	SOLDIERS EVOLUTION
5	SWEET BROTHERHOOD

```
-- 2.3
SELECT title
FROM film
       INNER JOIN film_category USING (film_id)
WHERE length < 60 AND
       category_id IN (SELECT category_id
                       FROM film
                              INNER JOIN film_category USING (film_id)
                       WHERE film_id IN (10, 20, 30));
```

	TITLE
1	SENSE GREEK
2	MOSQUITO ARMAGEDDON
3	DIVORCE SHINING
4	CRANES RESERVOIR
5	MINORITY KISS
6	LEGEND JEDI
7	HEAVENLY GUN
8	HANOVER GALAXY
9	GROSSE WONDERFUL
10	GO PURPLE
11	SIMON NORTH
12	COMMANDMENTS EXPRESS
13	AIRPORT POLLOCK
14	ACE GOLDFINGER

```sql
-- 2.4
SELECT
       first_name,
       last_name, Count(*) AS "Nr. of films"
FROM actor
       INNER JOIN film_actor USING (actor_id)
WHERE actor_id IN (SELECT actor_id
                   FROM film_actor
                   GROUP BY actor_id
                   HAVING COUNT(*) > 35)
GROUP BY actor_id, first_name, last_name;
```

	FIRST_NAME	LAST_NAME	Nr. of films
1	WALTER	TORN	41
2	GINA	DEGENERES	42
3	MATTHEW	CARREY	39
4	SANDRA	KILMER	37
5	SCARLETT	DAMON	36
6	MARY	KEITEL	40

```sql
-- 2.5
SELECT
       title,
       length
FROM film
WHERE length > (SELECT AVG(length)
                FROM film);
```

	TITLE	LENGTH
1	CLYDE THEORY	139
2	COLOR PHILADELPHIA	149
3	COMMAND DARLING	120
4	CONFIDENTIAL INTERVIEW	180
5	CONFUSED CANDLES	122
6	CONNECTICUT TRAMP	172
7	CONQUERER NUTS	173
8	CONSPIRACY SPIRIT	184
9	CONTACT ANONYMOUS	166
10	CONTROL ANTHEM	185
11	COWBOY DOOM	146
12	CRAZY HOME	136
13	CREATURES SHAKESPEARE	139
14	CREEPERS KANE	172
15	CROOKED FROGMEN	143

```sql
-- 2.6
SELECT UNIQUE name
FROM film_category
       INNER JOIN category USING (category_id)
WHERE category_id IN (SELECT category_id
                      FROM film_category
                      GROUP BY category_id
                      HAVING COUNT(film_id) < 60);
```

	NAME
1	Comedy
2	Travel
3	Horror
4	Classics
5	Music

```
-- 2.7
SELECT
       title,
       length,
       release_year
FROM film f1
WHERE length >= ALL (SELECT length
                     FROM film f2
                     WHERE f2.release_year = f1.release_year);
```

	TITLE	LENGTH	RELEASE_YEAR
1	CONSPIRACY SPIRIT	184	2003
2	CONTROL ANTHEM	185	1992
3	ALLEY EVOLUTION	180	1989
4	CATCH AMISTAD	183	1997
5	CAUSE DATE	179	2005
6	CHICAGO NORTH	185	2001
7	FRONTIER CABIN	183	2008
8	GANGS PRIDE	185	2007
9	KING EVOLUTION	184	2006
10	LAWLESS VISION	181	2004
11	HOME PITY	185	2002
12	HOTEL HAPPINESS	181	2000
13	CRYSTAL BREAKING	184	1991
14	DARN FORRESTER	185	1984
15	MOONWALKER FOOL	184	1999

```
-- 2.8
SELECT
       film_id,
       to_date(rental_date, 'YYYY-MM-DD') AS rental_date
FROM (SELECT *
      FROM rental
             INNER JOIN inventory USING (inventory_id)
             INNER JOIN film USING (film_id)
      ORDER BY rental_date DESC)
FETCH FIRST 9 ROWS ONLY;
```

	FILM_ID	RENTAL_DATE
1	282	0004-11-15 00:00:00
2	938	0004-11-15 00:00:00
3	43	0004-11-15 00:00:00
4	369	0004-11-15 00:00:00
5	946	0004-11-15 00:00:00
6	995	0004-11-15 00:00:00
7	27	0004-11-15 00:00:00
8	818	0004-11-15 00:00:00
9	873	0004-11-15 00:00:00

Insert, Update und Delete

```
-- 3.1
CREATE TABLE new_film
  AS (SELECT *
      FROM film
      WHERE release_year >= (SELECT MAX(release_year) FROM film));

-- 3.2
INSERT INTO new_film
    (film_id, title, language_id, rental_duration, rental_rate, replacement_cost, release_year,
last_update)
VALUES
```

```
sql> INSERT INTO new_film
        (film_id, title, language_id, rental_duration, rental_rate, replacement_cost, release_year, last_update)
    VALUES
        (1001, 'Jason Bourne', 1, 5, 1.75, 16.99, 2016, (SELECT sysdate FROM dual))
[2018-10-16 17:41:35] 1 row affected in 22 ms
```

```
dual));

-- 3.3
UPDATE new_film
SET rental_rate = rental_rate * 1.15
```

```
sql> UPDATE new_film
    SET rental_rate = rental_rate * 1.15
    WHERE rental_rate < 2
[2018-10-16 18:25:24] 26 rows affected in 25 ms
```

```
-- 3.4
CREATE OR REPLACE VIEW cheap_film
  AS SELECT
        title,
        description,
        rental_rate,
        length
  FROM new_film
  WHERE rental_rate <= 2
```

```
sql> CREATE OR REPLACE VIEW cheap_film
     AS SELECT
           title,
           description,
           rental_rate,
           length
     FROM new_film
     WHERE rental_rate ≤ 2
     WITH CHECK OPTION
[2018-10-16 18:25:54] completed in 57 ms
```

```
-- 3.5: COMMIT and ROLLBACK does not affect the session since
--  data-dictionary manipulations (3.4) are followed by an automatic commit

-- 3.6
UPDATE cheap_film
SET rental_rate = rental_rate * 1.15
WHERE rental_rate < 2;
```

```
sql> UPDATE cheap_film
    SET rental_rate = rental_rate * 1.15
    WHERE rental_rate < 2
[2018-10-16 18:28:12] [44000][1402] ORA-01402: view WITH CHECK OPTION where-clause violation
-- Does not work since some films' rental_rate would
-- exceed the upper boundary (2)

-- 3.7
```

```
DELETE
FROM new_film
WHERE rental_rate > 1.79;
```

```
sql> DELETE
     FROM new_film
     WHERE rental_rate > 1.79
[2018-10-16 18:28:44] 20 rows affected in 22 ms
```

```
-- 3.8 (1.79 * 1.15) > 2, Since those entries are deleted in 3.7, all other values (x * 1.15)
```

```
sql> UPDATE cheap_film
     SET rental_rate = rental_rate * 1.15
     WHERE rental_rate < 2
[2018-10-16 18:29:36] 18 rows affected in 23 ms
```

```
-- 3.9
DROP TABLE new_film;
sql> DROP TABLE new_film
[2018-10-16 21:53:13] completed in 160 ms
DROP VIEW cheap_film;
  sql> DROP VIEW cheap_film
  [2018-10-16 18:30:48] completed in 41 ms
```