

<input type="checkbox"/> DES3UEG1: Niklas	Name	Angelos Angelis	Aufwand in h	4
<input checked="" type="checkbox"/> DES3UEG2: Niklas				
<input type="checkbox"/> 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. Erstellen Sie eine Liste aller Schauspieler. Verketteten Sie Vor- und Nachname (getrennt durch ein Leerzeichen) und nennen Sie die neue Spalte "Name". Sortieren Sie das Ergebnis nach dem Nachnamen. (0,5 Punkte)
2. Geben Sie eine Liste mit Titel und Länge all jener Filme aus, die länger als 180 Minuten dauern. (0,5 Punkte)
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). (0,5 Punkte)
4. Geben Sie den Titel jener Filme aus, bei denen eine Originalsprache eingetragen ist. (0,5 Punkte)
5. Geben Sie die Anzahl der verliehenen Filme zwischen 1.1.2015 und 31.12.2015 aus (Start des Verleihvorgangs, *rental_date*). (1 Punkt)
6. Geben Sie alle Inventar-Ids aus, die noch nie verliehen wurden. (1 Punkt)
7. Erstellen Sie eine Liste mit Kunden (Vor- und Nachname), die in Newcastle, Linz und London wohnen. (1 Punkt)
8. Geben Sie für den Kunden mit der ID 240 alle Verleihvorgänge mit dem Start des Verleihvorgangs und dem bezahlten Betrag aus. Geben Sie das Datum im **Format** 'Mittwoch, 14. Okt. 2020' aus, verwenden Sie dazu die Funktion *to_char* und recherchieren Sie bei Bedarf in der Oracle-Dokumentation. (1 Punkt)

2. Gruppierungen und Unterabfragen**(12 Punkte)**

1. 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. (1,5 Punkte)
2. 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. (1,5 Punkte)
3. Erstellen Sie eine Liste aus Schauspielern (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. Zählen Sie nur jene Filme, die mindestens 60 Minuten dauern. (1 Punkt)

4. Erstellen Sie eine Liste mit den Titeln und der Länge jener Filme, die länger als der Durchschnitt sind. (1 Punkt)
5. Ermitteln Sie die Namen jener Filmkategorien zu denen weniger als 60 Filme gehören. (1 Punkt)
6. Ermitteln Sie alle Filme, die die längsten in ihrem Erscheinungsjahr sind und geben Sie Titel, Dauer (*length*) sowie Erscheinungsjahr aus. (1 Punkt)
7. Geben Sie den durchschnittlich bezahlten Preis (Tabelle *payment*) pro Filmkategorie aus (sollte ein Film zu mehreren Kategorien gehören, zählt er zu allen). Runden Sie auf zwei Nachkommastellen. (1 Punkt)
8. Geben Sie die neun zuletzt verliehenen Filme und das Verleihdatum im Format 'YYYY-MM-DD' aus. (1,5 Punkte)
9. Geben Sie alle Schauspieler (mind. Vor-/Nachname) aus, die in mehr als 15 **verschiedenen** Film-Kategorien spielen. Achten Sie dabei darauf, dass manche Schauspieler unter Umständen den gleichen Namen tragen. (1 Punkt)
10. Welcher Film wurde pro Film-Kategorie als erstes ausgeliehen (*rental_date*)? Geben Sie Film-Titel, Kategorie-Name und Verleihdatum aus. Sortieren Sie nach dem Kategorie-Namen. (1,5 Punkte)

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 neuesten 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 Punkte)
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). (0,5 Punkte)
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 Punkte)
7. Löschen Sie alle Einträge aus der Tabelle new_film, deren Leihpreis über 1.79 liegt. (0,5 Punkte)
8. Können Sie den Leihpreis der Filme in der View nun erhöhen? Erklären Sie dieses Verhalten. (0,5 Punkte)
9. Löschen Sie die Tabelle "new_film" und Ihre erstellte View wieder. (0,5 Punkte)

Aufgabe 1)

--1. Erstellen Sie eine Liste aller Schauspieler. Verketteten Sie Vor- und Nachname (getrennt durch
 --ein Leerzeichen) und nennen Sie die neue Spalte "Name". Sortieren Sie das
 Ergebnis nach dem
 --Nachnamen. (0,5 Punkte)

```
SELECT (FIRST_NAME || ' ' || LAST_NAME) AS Name
FROM ACTOR;
```

	NAME		NAME
1	PENELOPE GUINESS	196	BELA WALKEN
2	NICK WAHLBERG	197	REESE WEST
3	ED CHASE	198	MARY KEITEL
		199	JULIA FAWCETT
		200	THORA TEMPLE

--2. Geben Sie eine Liste mit Titel und Länge all jener Filme aus, die länger als 180 Minuten dauern. (0,5 Punkte)

```
SELECT TITLE, LENGTH
FROM FILM
WHERE LENGTH > 180;
```

TITLE	LENGTH	TITLE	LENGTH
1 BAKED CLEOPATRA	182	34 SMOOCHY CONTROL	184
2 ANALYZE HOOSIERS	181	35 SOLDIERS EVOLUTION	185
3 CATCH AMISTAD	183	36 SONS INTERVIEW	184
4 CHICAGO NORTH	185	37 SORORITY QUEEN	184
		38 STAR OPERATION	181
		39 SWEET BROTHERHOOD	185

--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). (0,5 Punkte)

```
SELECT INITCAP(TITLE)
FROM FILM
WHERE TITLE LIKE '___A%';
```

"INITCAP(TITLE)"	"INITCAP(TITLE)"
1 Atlantis Cause	63 Straight Hours
2 Attacks Hate	64 Strangelove Desire
3 Bedazzled Married	65 Stranger Strangers
4 Behavior Runaway	66 Strangers Graffiti
	67 Sugar Wonka

--4. Geben Sie den Titel jener Filme aus, bei denen eine Originalsprache eingetragen ist. (0,5

--Punkte)

```
SELECT TITLE
FROM FILM
WHERE ORIGINAL LANGUAGE ID IS NOT NULL ;
```

TITLE	TITLE
1 ARMY FLINTSTONES	496 MYSTIC TRUMAN
2 ARSENIC INDEPENDENCE	497 NATIONAL STORY
3 ARTIST COLDBLOODED	498 NECKLACE OUTBREAK
4 ATLANTIS CAUSE	499 NEIGHBORS CHARADE
5 ATTACKS HATE	500 NEMO CAMPUS

--5. Geben Sie die Anzahl der verliehenen Filme zwischen 1.1.2015 und 31.12.2015 aus (Start des Verleihvorgangs, rental_date). (1 Punkt)

```
alter session set nls_date_format = 'dd.mm.yyyy';
SELECT COUNT(*)
FROM RENTAL
WHERE RENTAL_DATE BETWEEN '01.01.2015' AND '31.12.2015';
```

"COUNT(*)"
1 7050

--6. Geben Sie alle Inventar-Ids aus, die noch nie verliehen wurden. (1 Punkt)

```
SELECT I.INVENTORY_ID
FROM INVENTORY I
LEFT JOIN RENTAL R on I.INVENTORY_ID = R.INVENTORY_ID
WHERE RENTAL_ID IS NULL ;
```

INVENTORY_ID
1 5

--7. Erstellen Sie eine Liste mit Kunden (Vor- und Nachname), die in Newcastle, Linz und London wohnen. (1 Punkt)

```
SELECT FIRST_NAME, LAST_NAME
FROM CUSTOMER C
JOIN ADDRESS A on C.ADDRESS_ID = A.ADDRESS_ID
JOIN CITY CI on A.CITY_ID = CI.CITY_ID
WHERE REGEXP LIKE(CI.CITY, 'Newcatsle|Linz|London');
```

	FIRST_NAME	LAST_NAME
1	JILL	HAWKINS
2	MATTIE	HOFFMAN
3	CECIL	VINES

--8. Geben Sie für den Kunden mit der ID 240 alle Verleihvorgänge mit dem Start des Verleihvorgangs und dem bezahlten Betrag aus. Geben Sie das Datum im Format 'Mittwoch, 14. Okt.

--2020' aus, verwenden Sie dazu die Funktion to_char und recherchieren Sie bei Bedarf in der

--Oracle-Dokumentation. (1 Punkt)

```
SELECT R.RENTAL_ID, TO_CHAR( R.RENTAL_DATE, 'DAY, dd. Mon. YYYY' ), P.AMOUNT
FROM CUSTOMER C
JOIN RENTAL R on C.CUSTOMER_ID = R.CUSTOMER_ID
JOIN PAYMENT P on C.CUSTOMER_ID = P.CUSTOMER_ID
WHERE C.CUSTOMER_ID = 240
```

RENTAL_ID	TO_CHAR(R.RENTAL_DATE, 'DAY, DD. MON. YYYY')	AMOUNT
1	643 MONTAG , 17. Feb. 2014	8.01
2	246 DIENSTAG , 31. Dez. 2013	8.01
3	460 DIENSTAG , 27. Mai. 2014	8.01
4	2196 SAMSTAG , 21. Mrz. 2015	8.01
5	2264 FREITAG , 18. Apr. 2014	8.01

RENTAL_ID	TO_CHAR(R.RENTAL_DATE, 'DAY, DD. MON. YYYY')	AMOUNT
496	246 DIENSTAG , 31. Dez. 2013	19.47
497	460 DIENSTAG , 27. Mai. 2014	19.47
498	2196 SAMSTAG , 21. Mrz. 2015	19.47
499	2264 FREITAG , 18. Apr. 2014	19.47
500	2872 SONNTAG , 18. Mai. 2014	19.47

Aufgabe 2)

--1. 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. (1,5 Punkte)

```
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. 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. (1,5 Punkte)

```
SELECT F.TITLE
FROM FILM F
JOIN FILM_CATEGORY FC on F.FILM_ID = FC.FILM_ID
WHERE F.LENGTH < 60
AND CATEGORY_ID IN (SELECT CATEGORY_ID
                   FROM FILM_CATEGORY
                   WHERE FILM_ID IN (10,20,30));
```

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

--3. Erstellen Sie eine Liste aus Schauspielern (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.

--Zählen Sie nur jene Filme, die mindestens 60 Minuten dauern. (1 Punkt)

```
SELECT A.FIRST_NAME, A.LAST_NAME, COUNT(*)
FROM ACTOR A
JOIN FILM_ACTOR FA on A.ACTOR_ID = FA.ACTOR_ID
JOIN FILM F on FA.FILM_ID = F.FILM_ID
WHERE f.LENGTH > 60
GROUP BY A.FIRST_NAME, A.LAST_NAME
HAVING COUNT(FA.ACTOR_ID) >35;
```

	FIRST_NAME	LAST_NAME	"COUNT(*)"
1	MATTHEW	CARREY	36
2	MARY	KEITEL	39
3	WALTER	TORN	38
4	SUSAN	DAVIS	48

--4. Erstellen Sie eine Liste mit den Titeln und der Länge jener Filme, die länger als der Durchschnitt sind. (1 Punkt)

```
SELECT TITLE, LENGTH
FROM FILM
WHERE LENGTH > (SELECT AVG(LENGTH)
                FROM FILM);
```

	TITLE	LENGTH		TITLE	LENGTH
1	ARMY FLINTSTONES	148	485	SWEET BROTHERHOOD	185
2	ARSENIC INDEPENDENCE	137	486	TADPOLE PARK	155
3	ARTIST COLDBLOODED	170	487	TALENTED HOMICIDE	173
4	ATLANTIS CAUSE	170	488	TELEGRAPH VOYAGE	148
5	BABY HALL	153	489	TELEMARK HEARTBREAKERS	152

--5. Ermitteln Sie die Namen jener Filmkategorien zu denen weniger als 60 Filme gehören. (1 Punkt)

```
SELECT NAME
FROM CATEGORY
WHERE CATEGORY_ID IN (SELECT CATEGORY_ID
                     FROM FILM_CATEGORY
                     GROUP BY CATEGORY_ID
                     HAVING COUNT(*) < 60);
```

	NAME
1	Classics
2	Comedy
3	Horror
4	Music
5	Travel

--6. Ermitteln Sie alle Filme, die die längsten in ihrem Erscheinungsjahr sind und geben Sie Titel, --Dauer (length) sowie Erscheinungsjahr aus. (1 Punkt)

```
SELECT F.TITLE, F.LENGTH, F.RELEASE_YEAR
FROM FILM F
WHERE F.LENGTH = (SELECT MAX(F2.LENGTH)
                 FROM FILM F2
                 WHERE F.RELEASE_YEAR = F2.RELEASE_YEAR);
```

	TITLE	LENGTH	RELEASE_YEAR		TITLE	LENGTH	RELEASE_YEAR
1	ALLEY EVOLUTION	180	1989	26	SONS INTERVIEW	184	1994
2	CATCH AMISTAD	183	1997	27	SORORITY QUEEN	184	1985
3	CAUSE DATE	179	2005	28	SPLASH GUMP	175	1995
4	CHICAGO NORTH	185	2001	29	STAR OPERATION	181	1988
5	CONSPIRACY SPIRIT	184	2003	30	SWEET BROTHERHOOD	185	1992

--7. Geben Sie den durchschnittlich bezahlten Preis (Tabelle payment) pro Filmkategorie aus (sollte --ein Film zu mehreren Kategorien gehören, zählt er zu allen). Runden Sie auf zwei Nachkommastellen. (1 Punkt)

```
SELECT FC.CATEGORY_ID, ROUND(AVG(P.AMOUNT),2)
FROM PAYMENT P
      JOIN RENTAL R on R.RENTAL_ID = P.RENTAL_ID
      JOIN INVENTORY I on R.INVENTORY_ID = I.INVENTORY_ID
      JOIN FILM_CATEGORY FC on I.FILM_ID = FC.FILM_ID
GROUP BY FC.CATEGORY_ID;
```

	CATEGORY_ID	ROUND(AVG(P.AMOUNT),2)
1	6	7.76
2	14	8.7
3	7	6.12
4	1	7.15
5	15	6.91
6	8	7.65
7	12	6.85
8	11	7.24
9	2	6.69
10	5	7.44
11	10	7.1
12	4	6.64
13	3	7
14	9	7.66
15	13	6.21
16	16	8.21

--8. Geben Sie die neun zuletzt verliehenen Filme und das Verleihdatum im Format 'YYYY-MMDD' aus. (1,5 Punkte)

```
SELECT I.FILM_ID, TO_CHAR(R.RENTAL_DATE, 'YYYY-MMDD')
FROM INVENTORY I
      JOIN RENTAL R on I.INVENTORY_ID = R.INVENTORY_ID
WHERE ROWNUM <= 9
ORDER BY RENTAL_DATE DESC;
```

	FILM_ID	TO_CHAR(R.RENTAL_DATE, 'YYYY-MMDD')
1	333	2015-0628
2	535	2015-0516
3	870	2015-0111
4	613	2014-1229
5	80	2014-1011
6	450	2014-0826
7	373	2014-0526
8	565	2014-0125
9	510	2014-0124

--9. Geben Sie alle Schauspieler (mind. Vor-/Nachname) aus, die in mehr als 15 verschiedenen
--Film-Kategorien spielen. Achten Sie dabei darauf, dass manche Schauspieler
unter Umständen
--den gleichen Namen tragen. (1 Punkt)

```
SELECT A.ACTOR_ID, A.FIRST_NAME, A.LAST_NAME
FROM ACTOR A
      JOIN FILM_ACTOR FA on A.ACTOR_ID = FA.ACTOR_ID
      JOIN FILM_CATEGORY FC on FA.FILM_ID = FC.FILM_ID
GROUP BY A.ACTOR_ID, A.FIRST_NAME, A.LAST_NAME
HAVING COUNT(FC.CATEGORY_ID) > 15;
```

	⚙️ ACTOR_ID	👤 FIRST_NAME	👤 LAST_NAME		⚙️ ACTOR_ID	👤 FIRST_NAME	👤 LAST_NAME
1	22	ELVIS	MARX	190	78	GROUCHO	SINATRA
2	30	SANDRA	PECK	191	79	MAE	HOFFMAN
3	7	GRACE	MOSTEL	192	81	SCARLETT	DAMON
4	8	MATTHEW	JOHANSSON	193	179	ED	GUINNESS
5	9	JOE	SWANK	194	200	THORA	TEMPLE
				195	137	MORGAN	WILLIAMS
				196	141	CATE	HARRIS
				197	145	KIM	ALLEN

--10. Welcher Film wurde pro Film-Kategorie als erstes ausgeliehen
(rental_date)? Geben Sie FilmTitel, Kategorie-Name
-- und Verleihdatum aus. Sortieren Sie nach dem Kategorie-Namen. (1,5Punkte)

```
SELECT F.TITLE, C.NAME, R.RENTAL_DATE
FROM FILM F
      JOIN FILM_CATEGORY FC on F.FILM_ID = FC.FILM_ID
      JOIN CATEGORY C on C.CATEGORY_ID = FC.CATEGORY_ID
      JOIN INVENTORY I on FC.FILM_ID = I.FILM_ID
      JOIN RENTAL R on I.INVENTORY_ID = R.INVENTORY_ID
WHERE R.RENTAL_DATE = (SELECT MIN(RENTAL_DATE)
                        FROM RENTAL R2
                        JOIN INVENTORY I2 on R2.INVENTORY_ID =
I2.INVENTORY_ID
                        JOIN FILM_CATEGORY FC2 on I2.FILM_ID =
FC2.FILM_ID
                        WHERE FC.CATEGORY_ID = FC2.CATEGORY_ID)
ORDER BY C.NAME
```

	📺 TITLE	👤 NAME	📅 RENTAL_DATE		📺 TITLE	👤 NAME	📅 RENTAL_DATE
1	MINDS TRUMAN	Action	2013-12-05	16	MASKED BUBBLE	Music	2013-12-05
2	PRIMARY GLASS	Action	2013-12-05	17	ANGELS LIFE	New	2013-12-05
3	DARN FORRESTER	Action	2013-12-05	18	REDEMPTION COMFORTS	New	2013-12-05
4	POTTER CONNECTICUT	Animation	2013-12-05	19	HARRIED GO	Sci-Fi	2013-12-05
5	FARGO GANDHI	Children	2013-12-08	20	SHAKESPEARE SADDLE	Sports	2013-12-05
				21	DISCIPLE MOTHER	Travel	2013-12-10
				22	SWEET BROTHERHOOD	Travel	2013-12-10
				23	KICK SAVANNAH	Travel	2013-12-10

Aufgabe 3)

--1. Erstellen Sie eine neue Tabelle "new_film", diese soll den gleichen Aufbau wie die Tabelle
--"film" haben, jedoch nur die neuesten Filme (jene Filme, die das höchste Erscheinungsjahr in
--der Datenbank aufweisen) enthalten. (1 Punkt)

```
CREATE TABLE new_film (  
    new_film_id INTEGER NOT NULL,  
    title VARCHAR2(255) NOT NULL,  
    description VARCHAR2(255),  
    release_year INTEGER,  
    language_id INTEGER NOT NULL,  
    original_language_id INTEGER,  
    rental_duration INTEGER NOT NULL,  
    rental_rate DECIMAL(4,2) NOT NULL,  
    length INTEGER DEFAULT NULL,  
    replacement_cost DECIMAL(5,2) NOT NULL,  
    rating VARCHAR2(20),  
    special_features VARCHAR2(255),  
    last_update TIMESTAMP NOT NULL,  
    CONSTRAINT new_film_pk PRIMARY KEY (new_film_id),  
    CONSTRAINT fk_new_film_language FOREIGN KEY (language_id) REFERENCES  
language (language_id) DEFERRABLE,  
    CONSTRAINT fk_new_film_language_original FOREIGN KEY (original_language_id)  
REFERENCES language (language_id) DEFERRABLE  
);
```

```
INSERT INTO new_film  
SELECT *  
FROM FILM  
WHERE RELEASE_YEAR = (SELECT MAX(RELEASE_YEAR)  
FROM FILM);
```

--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)

```
INSERT INTO FILM (FILM_ID, TITLE, LANGUAGE_ID, ORIGINAL_LANGUAGE_ID,  
RENTAL_DURATION, RENTAL_RATE, REPLACEMENT_COST, LAST_UPDATE)  
VALUES (1001, 'Jason Bourne', 1, 1, 5, 1.79, 16.99, CURRENT_TIMESTAMP);
```

--3. Erhöhen Sie den Leihpreis der Filme in der Tabelle „new_film“ um 15%,
wenn der Leihpreis
--kleiner als 2 ist. (0,5 Punkte)

```
UPDATE new_film  
SET rental_rate = rental_rate*1.15  
WHERE rental_rate < 2;
```

--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)

```
CREATE VIEW FILM_LESSTHAN_2 AS  
SELECT  
    TITLE, DESCRIPTION, RENTAL_RATE, LENGTH
```

```
FROM  
    NEW_FILM  
WHERE  
    RENTAL_RATE < 2 WITH CHECK OPTION;
```

--5. Welche Auswirkungen haben COMMIT und ROLLBACK an dieser Stelle (nachdem Sie die View erstellt haben). (0,5 Punkte)

Ein COMMIT speichert die Datenänderungen in die Datenbank, sodass der Zustand der vorherigen Daten überschrieben wird. Ab diesem Zeitpunkt können alle Benutzer die Ergebnisse sehen.

Ein ROLLBACK hingegen verwirft alle noch nicht gespeicherten Änderungen, wodurch der vorherige Zustand mit Hilfe vom UNDO-Tablespace hergestellt wird. Zudem werden die betroffenen gesperrten Zeilen wieder freigegeben.

--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

--Punkte)

Es geht nicht weil die Check Option dies verhindert. Nämlich würde dann die Erhöhung die rental_rates mancher Filme erhöhen, diese sollen dann aber durch der with check Option in der View nicht angezeigt

--7. Löschen Sie alle Einträge aus der Tabelle new_film, deren Leihpreis über 1.79 liegt. (0,5

--Punkte)

```
DELETE FROM NEW_FILM WHERE RENTAL_RATE > 1.79;
```

--8. Können Sie den Leihpreis der Filme in der View nun erhöhen? Erklären Sie dieses Verhalten.

--(0,5 Punkte)

--9. Löschen Sie die Tabelle "new_film" und Ihre erstellte View wieder. (0,5 Punkte)

```
drop view FILM_LESSTHAN_2;  
drop table NEW_FILM
```