

<input type="checkbox"/> DES3UEG1: Niklas	Name	Angelos Angelis	Aufwand in h	5
<input checked="" type="checkbox"/> DES3UEG2: Niklas				
<input type="checkbox"/> DES3UEG3: Traxler	Punkte		Kurzzeichen Tutor	

In dieser Übung sollen materialisierte Sichten und Abfragen mit dem LISTAGG-Operator anhand der Sakila Datenbank vertieft werden. Weiters werden Data Dictionary und künstliche Schlüssel anhand theoretischer und praktischer Beispiele erarbeitet.

1. LISTAGG – Sakila

(7 Punkte - 2 + 2 + 3)

- Geben Sie für alle Filme das Erscheinungsjahr („year“) und den Filmtitel („film“) aus, sortiert nach Jahr und Titel. 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.

	RELEASE_YEAR	Film	actors
1	1983	BORN SPINAL	M. ALLEN, R. JOHANSSON, K. PALTROW, K. PALTROW, R. REYN
2	1983	BOWFINGER GABLES	K. BERRY, C. HUNT, M. MCCONAUGHEY, W. WILSON, M. ZELLWE
3	1983	BUNCH MINDS	K. BERRY, C. BRIDGES, L. BULLOCK, J. CAGE, E. GOODING,
4	1983	CHITTY LOCK	V. BOLGER, S. DAVIS, N. DEGENERES, R. DUKAKIS, A. GARLA
5	1983	CIDER DESIRE	J. CHASE, F. DAY-LEWIS, J. DEGENERES, J. MCQUEEN, P. PI
6	1983	CLOSER BANG	J. DEGENERES, J. FAWCETT, E. GUINNESS, G. MALDEN, K. PAL
7	1983	DIVIDE MONSTER	C. AKROYD, H. BERRY, A. DREYFUSS, S. KILMER, G. MCKELLE
8	1983	DRAGON SQUAD	A. CRONYN, S. DAVIS, S. DEPP, E. GUINNESS, G. HOPKINS, J
9	1983	DRUMS DYNAMITE	V. BERGEN, J. CRUISE, L. DEE, M. HOPKINS, W. HURT, J. S

- Geben Sie zu jedem Kunden (Vorname Nachname) eine Liste der Filme aus, die sich der Kunde innerhalb der letzten sieben Jahre ausborgt 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.

CUSTOMER	FILMS
1 RAFAEL ABNEY	CHOCOLAT HARRY (2006), CONEHEADS SMOOCHY (2005), GOODFELLAS SALUTE (2002), POCUS PULP (1995)
2 NATHANIEL ADAM	TIGHTS DAWN (2006), GATHERING CALENDAR (2005), ORIENT CLOSER (2005), ROCKY WAR (2005), HANGI
3 KATHLEEN ADAMS	SPY MILE (2008), SWARM GOLD (2003), SUNDANCE INVASION (2002), ALONE TRIP (1998), INSIDER ARI
4 DIANA ALEXANDER	REBEL AIRPORT (2008), SHOW LORD (2006), JASON TRAP (2003), GENTLEMEN STAGE (2001), ROXANNE R
5 GORDON ALLARD	SILVERADO GOLDFINGER (2008), BINGO TALENTED (2006), DETAILS PACKER (2005), ALADDIN CALENDAR

- Der Store in Linz (store_id = 3) möchte die Interessen seiner Kunden näher bestimmen. Besonders interessant sind die Filmkategorien, die einzelne Kunden zuletzt ausgeliehen haben. Geben Sie Vor- und Nachname des Kunden als „customer“ und eine **Liste** der Kategorien (getrennt durch „.“) aller Filme, die sich der jeweilige Kunde ausborgt hat, als „recent_interests“ aus. Beschränken Sie sich auf Kunden, die im Store in der Stadt Linz registriert sind und verwenden Sie für das Ergebnis nur die drei zuletzt geliehenen Kategorien.

Sonderfälle: Achten Sie darauf, dass dies auch mehrere Kategorien sein könnten, wenn mehrere Filme zum gleichen Zeitpunkt geliehen wurden. Außerdem sollen Kategorien nicht mehrfach in der Liste erscheinen.

Eine Vereinfachung können Sie unter anderem durch eine entsprechende Vorverarbeitung der Datenmenge (analytische Funktion) erreichen z.B. durch die Verwendung von WITH. Recherchieren Sie bei Bedarf.

Fügen Sie ausreichend Testfälle in die Abgabe ein, um auch Sonderfälle entsprechend zu belegen.

CUSTOMER	RECENT_INTERESTS
1 GARCIA CAROL	Documentary, Classics, Animation
2 ALLEN SHIRLEY	Foreign, Children
3 HILL ANNA	Comedy, Drama
4 ADAMS KATHLEEN	Music, Children, Action
5 WATSON THERESA	Animation, Sports, Comedy
6 SANDERS TAMMY	Travel, Action, Drama
7 ALEXANDER DIANA	Games, Sports
8 DIAZ EMILY	Comedy, Animation, Family
9 ELLIS GRACE	Classics, Children, Games

2. Materialisierte Sichten (Sakila-Datenbank)

(6 Punkte – 3+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** „UE05_02a“.

Tipp: Erstellen Sie hierfür zuerst einen Subquery-Block „revenues“, der Ihnen für jeden Angestellten (staff_id) den Umsatz pro Filmkategorie (Name der Filmkategorie) berechnet.

2. Erzeugen Sie aus der virtuellen Sicht UE05_02a eine manuell zu aktualisierende materialisierte Sicht UE05_02b mit kompletter Neuberechnung (Re-Materialisierung).
3. Verändern Sie den Aktualisierungszeitpunkt der erstellten materialisierten Sicht UE05_02b in der Weise, dass sie automatisch jeden Tag um 23:30 aktualisiert wird. Speichern Sie die geänderte materialisierte Sicht unter dem Namen UE05_02c. Löschen Sie die materialisierte Sicht UE05_02c anschließend wieder.

3. Data Dictionary

(5 Punkte – 1 + 1 + 2 + 1)

Das Data Dictionary von Oracle speichert alle Informationen, die zur Verwaltung der Objekte (z.B. Tabellen, Sichten, Indizes, Prozeduren, Funktionen, Trigger) in der Datenbank benötigt werden. Obwohl dies üblicherweise in den Zuständigkeitsbereich des Datenbankadministrators fällt, stellt das Data Dictionary auch für Entwickler und Datenbankanwender eine wertvolle Informationsquelle dar. Sie lernen in den folgenden Aufgaben ausgewählte Bereiche des Data Dictionary aus der Perspektive des Datenbankanwenders kennen.

1. Erstellen Sie ein Skript für eine angegebene Tabelle, das die Spaltennamen, die Datentypen und die Länge der Datentypen sowie eine Information darüber liefert, ob Nullwerte zulässig sind. Fordern Sie den Benutzer auf, den Tabellennamen einzugeben (&-Operator). Geben Sie auch die Spalten DATA_PRECISION und DATA_SCALE aus und weisen Sie ihnen geeignete Aliasnamen zu.
2. Fügen Sie der Tabelle STORE einen Kommentar (SQL: COMMENT ON <Tablename> IS '<Comment>') hinzu. Fragen Sie anschließend die View USER_TAB_COMMENTS ab, um zu prüfen, ob der Kommentar hinzugefügt wurde.
3. Erstellen Sie ein Skript, das den Spaltennamen (COLUMN_NAME), den Constraint-Namen (CONSTRAINT_NAME), den Constraint-Typ (CONSTRAINT_TYPE), das Suchkriterium (SEARCH_CONDITION) und den Status (STATUS) für eine angegebene Tabelle liefert. Sie müssen die Tabellen USER_CONSTRAINTS und USER_CONS_COLUMNS verknüpfen, um alle diese Informationen zu erhalten. Fordern Sie den Benutzer auf, den Tabellennamen einzugeben.
4. Bereiten Sie ein Skript vor, das Ihnen am Ende des Semesters ermöglicht, alle ihre angelegten Objekte zu entfernen. Erstellen Sie dafür eine Abfrage auf die Tabelle user_objects und generieren Sie die DROP-Statements entsprechend.

Achtung: Führen Sie die generierten Statements nicht aus, ansonsten entfernen Sie auch die noch benötigten Tabellen des HR- und Sakila-Schemas!

4. Künstliche Schlüssel

(3 Punkte)

Eine große Firma entnimmt regelmäßig Wasserproben (Brunnen, See, Fließgewässer) an verschiedenen Orten. Die Proben werden von drei Personen bzw. Teams entnommen, die dazu eine spezielle Ausrüstung und Software verwenden. Da die Probennahme auch an entlegenen Orten erfolgen kann, verwendet die Software eine interne Datenbank, die regelmäßig mit der Zentrale synchronisiert wird. Jeder Probe soll bereits bei der Entnahme eine eindeutige Nummer zur Identifikation zugeordnet werden.

Legen Sie dafür eine Tabelle „Probe“ (Proben-ID, Zeitpunkt der Probenname, Probentyp, Kommentar) an. Erstellen Sie weitere Datenbank-Objekte, um Identifikationsnummern pro Team erzeugen zu können. Gehen Sie bei der Vergabe der Identifikation so vor, dass sie zukünftig sofort ablesen können, welches Team die Probe entnommen hat.

Zeigen Sie beispielhaft, wie die drei Personen/Teams Datensätze einfügen können. Fragen Sie diese Daten auch ab.

5. Regelmäßige Aufgaben (Teil 1)

(3 Punkte)

Sie erhalten die Aufgabe, die Aktivität in der Datenbank zu analysieren. Als ersten Anlaufpunkt möchten Sie die Anzahl der offenen Sessions überwachen.

1. Erstellen Sie eine Abfrage, welche die Anzahl der momentan offenen Benutzer-Sessions ermittelt. Analysieren Sie dazu die Tabelle `v$session`.
2. Erstellen Sie eine Tabelle `monitor_sessions` für die Protokollierung der Abfrage über die Anzahl der Sessions. Fügen Sie zur Anzahl auch noch einen Zeitstempel hinzu und sehen Sie auch eine Spalte für das Einfügen eines Primärschlüssels vor. Legen Sie dafür eine eigene Sequenz an und verwenden Sie diese.
3. Erstellen Sie ein Insert-Skript, das Sie aufrufen können und die Anzahl der offenen Benutzer-Sessions in Ihre Logging-Tabelle einfügt. Rufen Sie das Skript zu verschiedenen Zeitpunkten auf.
4. Fragen Sie Ihre Datensätze ab. Setzen Sie dazu eine beliebige statistische Kennzahl ein.

Ausblick: Sie lernen in der nächsten Übung eine Möglichkeit kennen, wie Sie diese Tabelle regelmäßig und automatisch befüllen können.

AUFGABE 1)

-- 1. Geben Sie für alle Filme das Erscheinungsjahr (,year') und den Filmtitel (,film') aus, sortiert
 -- nach Jahr und Titel. 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.

```
SELECT F.RELEASE_YEAR, F.TITLE,
       LISTAGG(SUBSTR(A2.FIRST_NAME, 1, 1) || '. ' || A2.LAST_NAME, ';' ) WITHIN
GROUP ( ORDER BY A2.LAST_NAME, A2.FIRST_NAME) AS Actors
FROM FILM F
JOIN FILM_ACTOR FA on F.FILM_ID = FA.FILM_ID
JOIN ACTOR A2 on FA.ACTOR_ID = A2.ACTOR_ID
GROUP BY F.RELEASE_YEAR, F.TITLE;
```

	RELEASE_YEAR	TITLE	ACTORS
1	2000	OPUS ICE	D. CRAWFORD; J. FAWCETT; H. GARLAND; S. WILLIAMS
2	2000	TAXI KICK	S. DEPP; B. FAWCETT; M. HOPPER; W. JOLIE; A. NOLTE; K. PA...
3	2000	TRAP GUYS	V. BERGEN; W. HACKMAN; W. HOFFMAN; A. NOLTE; K. PESCI; B...
4	2000	LION UNCUT	P. PINKETT; B. WALKEN; U. WOOD
5	2000	SPEED SUIT	D. CRAWFORD; C. DEPP; M. HAWKE; S. KILMER; B. NICHOLSON; ...
495	1995	SAVANNAH TOWN	J. BAILEY; P. GOLDBERG; M. KILMER; S. KILMER; G. MOSTEL; ...
496	1995	SIERRA DIVIDE	R. BACALL; G. DEGENERES; M. KEITEL; J. MOSTEL; W. TORN
497	1995	ARGONAUTS TOWN	J. BARRYMORE; K. GARLAND; G. PENN; D. STREEP; G. WILLIS
498	1995	ELEMENT FREDDY	M. GIBSON; A. HUDSON; K. JOVOVICH; B. WALKEN
499	1995	WEDDING APOLLO	H. BALE; L. BERGMAN; V. BOLGER; E. CHASE; C. GABLE; H. GA...
500	1995	AIRPORT POLLOCK	S. DAVIS; L. DEE; F. KILMER; G. WILLIS

-- 2. Geben Sie zu jedem Kunden (Vorname Nachname) eine Liste der Filme aus,
 die sich der Kunde
 -- innerhalb der letzten sieben Jahre 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.

```
SELECT C.FIRST_NAME, C.LAST_NAME, LISTAGG(F.TITLE || '(' || F.RELEASE_YEAR ||
')', ';' ) WITHIN GROUP ( ORDER BY F.RELEASE_YEAR) AS Rented_Films
FROM CUSTOMER C
JOIN RENTAL R on C.CUSTOMER_ID = R.CUSTOMER_ID
JOIN INVENTORY I on R.INVENTORY_ID = I.INVENTORY_ID
JOIN FILM F on I.FILM_ID = F.FILM_ID
WHERE RENTAL_DATE = trunc(SYSDATE - interval '7' year, 'YEAR')
GROUP BY C.FIRST_NAME, C.LAST_NAME;
```

	FIRST_NAME	LAST_NAME	RENTED_FILMS
1	JOY	GEORGE	GRIT CLOCKWORK(1997)
2	DAVE	GARDINER	HOUSE DYNAMITE(2004)
3	CASEY	MENA	GRADUATE LORD(2008)
4	CLARA	SHAW	SABRINA MIDNIGHT(2003)
5	FRANK	WAGGONER	NIGHTMARE CHILL(2003)

20	DEBORAH	WALKER	CHISUM BEHAVIOR(2001)
21	KENNETH	GOODEN	HOTEL HAPPINESS(2000)
22	THEODORE	CULP	EMPIRE MALKOVICH(2001)
23	CHRISTINE	ROBERTS	JET NEIGHBORS(1986)

```
-- 3. Der Store in Linz (store_id = 3) möchte die Interessen seiner Kunden
nher bestimmen.
-- Besonders interessant sind die Filmkategorien, die einzelne Kunden zuletzt
ausgeliehen haben.
-- Geben Sie Vor- und Nachname des Kunden als „customer“ und eine Liste der
Kategorien
-- (getrennt durch ',') aller Filme, die sich der jeweilige Kunde ausgeborgt
hat, als „recent_interests“
-- aus. Beschrnken Sie sich auf Kunden, die im Store in der Stadt Linz
registriert sind und
-- verwenden Sie fr das Ergebnis nur die drei zuletzt geliehenen Kategorien.
-- Sonderflle: Achten Sie darauf, dass dies auch mehrere Kategorien sein
knnten, wenn mehrere
-- Filme zum gleichen Zeitpunkt geliehen wurden. Auerdem sollen Kategorien
nicht mehrfach in
-- der Liste erscheinen.
-- Eine Vereinfachung knnen Sie unter anderem durch eine entsprechende
Vorverarbeitung der
-- Datenmenge (analytische Funktion) erreichen z.B. durch die Verwendung von
WITH.
-- Recherchieren Sie bei Bedarf.
-- Fgen Sie ausreichend Testflle in die Abgabe ein, um auch Sonderflle
entsprechend zu belegen.
SELECT LAST_NAME, LISTAGG(DISTINCT Interests, ',') AS RECENT_INTERESTS
FROM (
    SELECT C2.LAST_NAME,
           RENTAL_DATE,
           row_number() over (PARTITION BY C2.LAST_NAME ORDER BY
RENTAL_DATE DESC) AS Row_num,
           LISTAGG(DISTINCT C3.NAME, ',') WITHIN GROUP ( ORDER BY
R.RENTAL_DATE) AS Interests
    FROM CUSTOMER C2
        JOIN RENTAL R on C2.CUSTOMER_ID = R.CUSTOMER_ID
        JOIN INVENTORY I on R.INVENTORY_ID = I.INVENTORY_ID
        JOIN FILM F on I.FILM_ID = F.FILM_ID
        JOIN FILM_CATEGORY FC on F.FILM_ID = FC.FILM_ID
        JOIN CATEGORY C3 on FC.CATEGORY_ID = C3.CATEGORY_ID
        JOIN STORE S on C2.STORE_ID = S.STORE_ID
    WHERE S.STORE_ID = 3
    GROUP BY C2.LAST_NAME, RENTAL_DATE
    ORDER BY LAST_NAME, RENTAL_DATE DESC
)
WHERE Row_num < 4
GROUP BY LAST_NAME;
```

	LAST_NAME	RECENT_INTERESTS
1	ADAMS	Action,Children,Music
2	ALEXANDER	Games,Sports
3	ALLEN	Children,Foreign
4	ARSENAULT	Comedy,Drama,Foreign
5	ASHER	Children,Classics,Documentary
39	SWAFFORD	Classics,Family,Sports
40	TUBBS	Children,Drama
41	WATKINS	Action,Comedy,Drama
42	WATSON	Animation,Comedy,Sports
43	WAUGH	Animation,Comedy,Family
44	WELCH	Animation,Children

Aufgabe 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 „UE05_02a“.

-- Tipp: Erstellen Sie hierfür zuerst einen Subquery-Block „revenues“, der Ihnen für jeden Angestellten (staff_id) den Umsatz pro Filmkategorie (Name der Filmkategorie) berechnet.

CREATE VIEW UE05_02a **AS**

```

SELECT S1.STAFF_ID, FC.category_id, sum(AMOUNT) as revenue
FROM payment P
INNER JOIN STAFF S1 ON (S1.staff_id = P.staff_id)
INNER JOIN STORE S2 ON (S1.store_id = S2.store_id)
INNER JOIN RENTAL R ON (P.rental_id = R.rental_id)
INNER JOIN INVENTORY i ON (R.inventory_id = i.inventory_id)
INNER JOIN FILM_CATEGORY FC ON (i.film_id = FC.film_id)
WHERE S1.STAFF_ID = 1 OR S1.STAFF_ID = 2
GROUP BY S1.STAFF_ID, FC.category_id
ORDER BY category_id;

```

SELECT * **FROM** UE05_02A;

	STAFF_ID	CATEGORY_ID	REVENUE
1	1	1	3028.58
2	2	1	3166.74
3	1	2	2955.02
4	2	2	3247.03
5	1	3	2468.93
6	2	3	2679.6

28	2	14	3597.07
29	1	15	3214.04
30	2	15	3245.12
31	1	16	2766.35
32	2	16	2387

-- 2. Erzeugen Sie aus der virtuellen Sicht UE05_02a eine manuell zu aktualisierende materialisierte
 -- Sicht UE05_02b mit kompletter Neuberechnung (Re-Materialisierung).

CREATE MATERIALIZED VIEW UE05_02b **AS**

SELECT * FROM UE05_02a;

-- 3. Verändern Sie den Aktualisierungszeitpunkt der erstellten materialisierten Sicht UE05_02b in der
 -- Weise, dass sie automatisch jeden Tag um 23:30 aktualisiert wird.
 Speichern Sie die geänderte

-- materialisierte Sicht unter dem Namen UE05_02c. Löschen Sie die materialisierte Sicht

-- UE05_02c anschließend wieder.

CREATE MATERIALIZED VIEW UE05_02c **AS**

SELECT * FROM UE05_02a;

ALTER MATERIALIZED VIEW UE05_02c **REFRESH**

START WITH SYSDATE

NEXT TRUNC(SYSDATE) + 23.50/24;

drop materialized view UE05_02c;

Aufgabe 3)

-- 1. Erstellen Sie ein Skript für eine angegebene Tabelle, das die Spaltennamen, die Datentypen und
 -- die Länge der Datentypen sowie eine Information darüber liefert, ob Nullwerte zulässig sind.

-- Fordern Sie den Benutzer auf, den Tabellennamen einzugeben (&-Operator). Geben Sie auch die

-- Spalten DATA_PRECISION und DATA_SCALE aus und weisen Sie ihnen geeignete Aliasnamen zu.

SELECT COLUMN_NAME, DATA_TYPE, DATA_LENGTH, NULLABLE, DATA_PRECISION AS dp, DATA_SCALE AS Ds

FROM USER_TAB_COLS

WHERE TABLE_NAME = UPPER(&table);

-- 2. Fügen Sie der Tabelle STORE einen Kommentar (SQL: COMMENT ON <Tablename> IS

-- '<Comment>') hinzu. Fragen Sie anschließend die View USER_TAB_COMMENTS ab, um zu

-- prüfen, ob der Kommentar hinzugefügt wurde.

COMMENT ON TABLE STORE **IS** 'THIS IS THE STORE TABLE YEYE';

SELECT *

FROM USER_TAB_COMMENTS

WHERE TABLE_NAME = 'STORE';

	TABLE_NAME	TABLE_TYPE	COMMENTS	ORIGIN_CON_ID
1	STORE	TABLE	THIS IS THE STORE TABLE YE...	0

```
-- 3. Erstellen Sie ein Skript, das den Spaltennamen (COLUMN_NAME), den
Constraint-Namen
-- (CONSTRAINT_NAME), den Constraint-Typ (CONSTRAINT_TYPE), das Suchkriterium
-- (SEARCH_CONDITION) und den Status (STATUS) für eine angegebene Tabelle
liefert. Sie müssen
-- die Tabellen USER_CONSTRAINTS und USER_CONS_COLUMNS verknüpfen, um alle
diese
-- Informationen zu erhalten. Fordern Sie den Benutzer auf, den Tabellennamen
einzugeben.
```

```
SELECT COLUMN_NAME, CONSTRAINT_NAME, c.CONSTRAINT_TYPE,
c.SEARCH_CONDITION, c.STATUS
FROM USER_CONSTRAINTS c
JOIN USER_CONS_COLUMNS USING (CONSTRAINT_NAME)
WHERE c.TABLE_NAME = UPPER('&tableName');
```

```
-- 4. Bereiten Sie ein Skript vor, das Ihnen am Ende des Semesters
ermöglicht, alle ihre angelegten
-- Objekte zu entfernen. Erstellen Sie dafür eine Abfrage auf die Tabelle
user_objects und
-- generieren Sie die DROP-Statements entsprechend.
-- Achtung: Führen Sie die generierten Statements nicht aus, ansonsten
entfernen Sie auch die noch
-- benötigten Tabellen des HR- und Sakila-Schemas!
select 'drop table ', table_name, 'cascade constraints;' from user_tables;
```

Aufgabe 4)

```
-- Eine große Firma entnimmt regelmäßig Wasserproben (Brunnen, See,
Fließgewässer) an
-- verschiedenen Orten. Die Proben werden von drei Personen bzw. Teams
entnommen, die dazu eine
-- spezielle Ausrüstung und Software verwenden. Da die Probennahme auch an
entlegenen Orten
-- erfolgen kann, verwendet die Software eine interne Datenbank, die
regelmäßig mit der Zentrale
-- synchronisiert wird. Jeder Probe soll bereits bei der Entnahme eine
eindeutige Nummer zur
-- Identifikation zugeordnet werden.
-- Legen Sie dafür eine Tabelle „Probe“ (Proben-ID, Zeitpunkt der Probenname,
Probentyp,
-- Kommentar) an. Erstellen Sie weitere Datenbank-Objekte, um
Identifikationsnummern pro Team
-- erzeugen zu können. Gehen Sie bei der Vergabe der Identifikation so vor,
dass sie zukünftig sofort
-- ablesen können, welches Team die Probe entnommen hat.
-- Zeigen Sie beispielhaft, wie die drei Personen/Teams Datensätze einfügen
können. Fragen Sie diese
-- Daten auch ab.
```

```
CREATE TABLE Probe (
    Probe_id NUMBER PRIMARY KEY,
    Probe_Zeitpunkt DATE,
    ProbeTyp VARCHAR(20),
    Kommentar VARCHAR(50),
    Identifikationsnummer NUMBER
```



```
CONSTRAINT TEAM_IDENTIFIKATIONNUMMER_FK REFERENCES
TEAM(Identifikationsnummer)
);

CREATE TABLE TEAM
(
    Identifikationsnummer INT GENERATED BY DEFAULT AS IDENTITY
    ( START WITH 200 INCREMENT BY 1
    MINVALUE 200 MAXVALUE 100000) PRIMARY KEY ,
    Name Varchar2(20)
);

CREATE SEQUENCE Probe_sq1
START WITH 1000
INCREMENT BY 1
MAXVALUE 2000
CACHE 10;

CREATE SEQUENCE Probe_sq2
START WITH 2001
INCREMENT BY 1
MAXVALUE 3000
CACHE 10;

CREATE SEQUENCE Probe_sq3
START WITH 3001
INCREMENT BY 1
MAXVALUE 4000
CACHE 10;

INSERT INTO TEAM(Identifikationsnummer,Name)
VALUES(DEFAULT,'TEAM1');
INSERT INTO TEAM(Identifikationsnummer,Name)
VALUES(DEFAULT,'TEAM2');

SELECT * FROM TEAM;

INSERT INTO Probe
VALUES (Probe_sq1.nextval, SYSDATE, 'SEE', 'SEE PROBE 1',200);

INSERT INTO Probe
VALUES (Probe_sq1.nextval, SYSDATE, 'SEE', 'SEE PROBE 2',200);

INSERT INTO Probe
VALUES (Probe_sq2.nextval, SYSDATE, 'Fließgewässer', 'Fließgewässer Probe
1',200);

INSERT INTO Probe
VALUES (Probe_sq2.nextval, SYSDATE, 'Fließgewässer', 'Fließgewässer Probe
2',201);

INSERT INTO Probe
VALUES (Probe_sq2.nextval, SYSDATE, 'Fließgewässer', 'Fließgewässer Probe
3',201);

INSERT INTO Probe
VALUES (Probe_sq3.nextval, SYSDATE, 'Brunnen', 'Brunnen Probe 1',201);
```

```
INSERT INTO Probe
VALUES (Probe_sq3.nextval, SYSDATE, 'Brunnen', 'Brunnen Probe 2',201);
```

```
INSERT INTO Probe
VALUES (Probe_sq3.nextval, SYSDATE, 'Brunnen', 'Brunnen Probe 3',201);
```

```
SELECT *
FROM Probe;
```

```
drop table PROBE cascade constraints;
drop sequence PROBE_SQ1;
drop sequence PROBE_SQ2;
drop sequence PROBE_SQ3;
```

	PROBE_ID	PROBE_ZEITPUNKT	PROBETYP	KOMMENTAR	IDENTIFIKATIONSNUMMER
1	1001	2021-11-09 23:40:11	SEE	SEE PROBE 1	200
2	1002	2021-11-09 23:40:16	SEE	SEE PROBE 2	200
3	2001	2021-11-09 23:40:19	Fließgewässer	Fließgewässer Probe 1	200
4	2002	2021-11-09 23:40:21	Fließgewässer	Fließgewässer Probe 2	201
5	2003	2021-11-09 23:40:27	Fließgewässer	Fließgewässer Probe 3	201
6	3001	2021-11-09 23:40:29	Brunnen	Brunnen Probe 1	201
7	3002	2021-11-09 23:40:31	Brunnen	Brunnen Probe 2	201
8	3003	2021-11-09 23:40:33	Brunnen	Brunnen Probe 3	201

Aufgabe5)

```
SELECT COUNT(SID)
FROM V$SESSION
WHERE SCHEMANAME <> 'SYS';
```

	"COUNT(SID)"
1	5

```
-- 2. Erstellen Sie eine Tabelle monitor_sessions für die Protokollierung der
Abfrage über die
-- Anzahl der Sessions. Fügen Sie zur Anzahl auch noch einen Zeitstempel
hinzu und sehen Sie
-- auch eine Spalte für das Einfügen eines Primärschlüssels vor. Legen Sie
dafür eine eigene
-- Sequenz an und verwenden Sie diese.
```




```
CREATE TABLE monitor_sessions (
    MS_ID INT GENERATED BY DEFAULT AS IDENTITY
    ( START WITH 200 INCREMENT BY 1
    MINVALUE 200 MAXVALUE 100000) PRIMARY KEY,
    COUNT NUMBER,
    LAST_CHECK DATE
);
```

```
-- 3. Erstellen Sie ein Insert-Skript, das Sie aufrufen können und die Anzahl
der offenen BenutzerSessions in Ihre
-- Logging-Tabelle einfügt. Rufen Sie das Skript zu verschiedenen Zeitpunkten
auf.
```

```
INSERT INTO monitor_sessions(MS_ID,COUNT, LAST_CHECK)
VALUES (
    DEFAULT,
```

```
(SELECT COUNT(SID)
FROM V$SESSION
WHERE SCHEMANAME <> 'SYS'),
SYSDATE
);
```

```
SELECT *
FROM monitor_sessions;
```

	 MS_ID	 COUNT	 LAST_CHECK
1	240	5	2021-11-09 00:06:48