DES3UE WS 2021 Übung 11+12

Mit Maximillian Bauer bearbeitet

Abgabetermin: 19.01.2022

☐ **DES3UEG1: Niklas** **Name** Angelos Angelis **Aufwand in h** 5
☒ **DES3UEG2: Niklas**
☐ **DES3UEG3: Traxler** **Punkte** _____ **Kurzzeichen Tutor** _____

Ziel dieser Übung ist die Modellierung und Erstellung einer kleinen **Data Warehouse Anwendung** und die Durchführung von Abfragen darauf.

Arbeiten Sie die Übung in **2er Gruppen** aus und geben Sie Ihre/n Partner/in an. Eine Abgabe für beide gemeinsam ist ausreichend.

Beschreibung Sachverhalt

Ihr Auftraggeber ist der Besitzer der Film-Verleihkette „**Sakila**“, die in mehreren Städten Stores zum Verleih von Filmen betreibt. Die Grundlage für den laufenden Betrieb bildet eine Datenbank, die alle Filme enthält, außerdem werden Kunden und Personal erfasst, sowie die Entlehnungen und Rückgaben (inkl. Bezahlung).

Ihr Auftraggeber möchte eine stabile Datenbasis erstellen, die es ihm ermöglicht, viele **analytische Abfragen** einfach durchzuführen, um den laufenden Betrieb zu beobachten und maßgebliche Entscheidungen zu treffen. Es soll möglich werden, **Kennzahlen des laufenden Geschäftsbetriebs** zu erfassen. Es ist ausreichend, wenn der Datenstand **monatsweise** aktualisiert wird, eine Auskunft auf einer täglichen Basis ist nicht notwendig.

Die Filmkette ist dafür bekannt, auf der ganzen Welt Filme in verschiedenen Sprachen zu verleihen und verschiedene Film-Genres zu besitzen. Vor allem für die Ersetzung von Filmen ist es maßgeblich wie viele Filme jeweils verliehen wurden. Für die Analyse der Geschäftsprozesse ist es jedoch nicht wichtig, Details zu kennen wie den Namen eines Films oder die mitspielenden SchauspielerInnen. Jedoch ist das Länge des Films relevant (short <= 60 Minuten, medium <= 120 Minuten, long > 120 Minuten) und die Stadt/das Land in dem der Film verliehen wurde. Ihr Auftraggeber ist auch an den Umsatzzahlen interessiert und möchte diese quartals- und monatsweise erfassen (Q1 = Jan, Feb, Mar; Q2 = Apr, Mai, Jun, …).

Die Film-Genres werden zu Analysezwecken innerhalb der Verleih-Kette in vier Klassen eingeteilt: Storyline (Animation, Sci-Fi, Sports), Narrative (Children, Comedy, Documentary, Drama, Family, Foreign, Travel), Mood (Action, Horror, Music) und Others (alle übrigen). Dies ist zwar für den laufenden Betrieb unerheblich, sollte jedoch bei der Auswertung zur Verfügung stehen. Beachten Sie dazu auch die bereits beschriebenen Business-Fälle (Abfragen) ganz unten.

1. Modellierung ADAPT (10 Punkte)

Modellieren Sie den oberhalb beschriebenen Sachverhalt (unter Berücksichtigung der beschriebenen Business-Fälle) in der Modelliersprache ADAPT.

2. Modellierung STAR-Schema (8 Punkte)

Erstellen sie aus der ADAPT Modellierung, die Sie im vorigen Schritt erstellt haben, ein STAR-Schema. Verwenden Sie dazu ein Werkzeug Ihrer Wahl.

3. Erstellung STAR-Schema (11 Punkte)

Erstellen Sie das modellierte STAR-Schema physisch in der Datenbank. Verwenden Sie dazu

- eine Sequenz (um die Primärschlüssel zu generieren),
- Tabellen für die Dimensionstabellen (Präfix dim) und
- eine Materialisierte Sicht für die Faktentabelle (Präfix fact), wobei das Update jeweils am Ersten des Monats erfolgen soll. Recherchieren Sie bei Bedarf die benötigte Datums-Syntax.

Befüllen Sie Ihre Dimensions-Tabellen, bevor Sie die Materialisierte Sicht erstellen: zB CREATE TABLE dimTable AS, DML-Statements, anonymer Block, gespeicherte Prozedur, EXTRACT (datetime). Rechnen Sie mit ganzen Tagen (d.h. Aufrunden).

4. Abfragen & Interpretation (14 Punkte)

Sie haben acht Fragestellungen von Ihrem Auftraggeber erhalten. Erstellen Sie **pro Frage EINE Abfrage**, mit der Sie den Sachverhalt erörtern können. **Zusätzlich** zur gewohnten Formatierung und Abgabe der Ergebnisse Ihrer Statements, beschreiben Sie das jeweilige Ergebnis der gestellten Frage **in textueller Form**.

Hinweise (je Frage in *kursiv*): Mit den angegebenen Hinweisen können Sie Ihre Lösung überprüfen. Die Hinweise sind dabei „Nebenprodukte“ eines Statements, das zur Analyse herangezogen werden kann.

Allg. Hinweis: Jahr, Monat und Quartal, Wochentag sollen nicht aus dem Datumsfeld neu ermittelt werden, diese Werte liegen bereits vorberechnet vor!

1. Stellen Sie fest, welche Film-Laufzeiten (short, medium, long) am öftesten ausgeliehen werden! *(3 Zeilen)*

2. Welches Genre ist hier am lukrativsten, gemessen an den durchschnittlichen Einnahmen pro Verleihvorgang? *Ermitteln Sie Platz 1 (Platz 2 ist Travel).*

3. Analysieren Sie, ob in einem bestimmten Quartal mehr Umsatz erzielt wurde. *Ermitteln Sie Platz 1 (Platz 4 ist Quartal 4).*

4. Stellen Sie gegenüber, wie lange Filme durchschnittlich ausgeliehen werden, analysieren Sie den Sachverhalt monatsweise und quartalsweise. Können Sie einen Zusammenhang feststellen? In welchem Monat werden Filme am längsten ausgeliehen, in welchem Quartal? *(sehr kleiner Unterschied, 17 Zeilen)*

5. Analysieren Sie, ob ein Zusammenhang besteht zwischen Spieldauer (Film-Laufzeit) und Land anhand der Anzahl an ausgeliehenen Filmen. Stellen Sie auch die Einnahmen dafür gegenüber. Welche Filmlängen (short, medium, long) würden Sie in welchen Ländern ausmustern (welche erzielen die wenigsten Einnahmen)? *(20 Zeilen, Australien hat die wenigsten Einnahmen – alleine betrachtet)*

6. Sie möchten ermitteln, ob es bestimmte Wochentage gibt, an denen häufiger Filme verliehen werden. Unterscheiden sich diese in verschiedenen Ländern? *(die wenigsten Verleihvorgänge gibt es an Montagen in Australien)*

7. Erstellen Sie für die Film-Klasse Narrative eine (kumulierte) Summe des Umsatzes (monatlich). Ermitteln Sie dazu zuerst die Gesamtsumme der Ausleihvorgänge jedes Monats und kumulieren

Sie diese Anzahl innerhalb des jeweiligen Jahres. *(im März 2014 ist der Umsatz 2289,47, kumuliert seit Jahresbeginn 7218,91)*

5. Pipelined Table Function (5 Punkte)

Erstellen Sie eine Pipelined Table Function `get_best_worst_genre`, die zwei optionale Parameter bietet:

- `p_city`
- `p_country`

Je nach gegebenen Parametern ermittelt die Funktion Umsatz und Genre aller Verleihvorgänge an den angegebenen Standort(en). Jeder Verleihvorgang wird einzeln bewertet, dh. keine Bildung von Durchschnitt oder Summe. Das umsatzstärkste Genre wird inkl. Umsatz und dem Hinweis ‚best' ausgegeben, ebenso wird das umsatzschwächste Genre ausgegeben mit dem Hinweis ‚worst'. Berücksichtigen Sie auch Ties, dh. wenn mehrere gleichstark/-schwach sind, sollen alle diese ausgegeben werden.

Erstellen Sie für die Pipelined Table Function die notwendigen Typen `genre_row` und `genre_tab`:

```
DROP TYPE genre_tab;
DROP TYPE genre_row;

CREATE TYPE genre_row AS OBJECT (
  genre     VARCHAR2(100),
  amount    NUMBER,
  info      VARCHAR2(50)
);
/

CREATE TYPE genre_tab IS TABLE OF genre_row;
/
```

Es gibt keine zusätzlichen Einschränkungen innerhalb Ihrer Funktion zur Ermittlung der Datensätze.

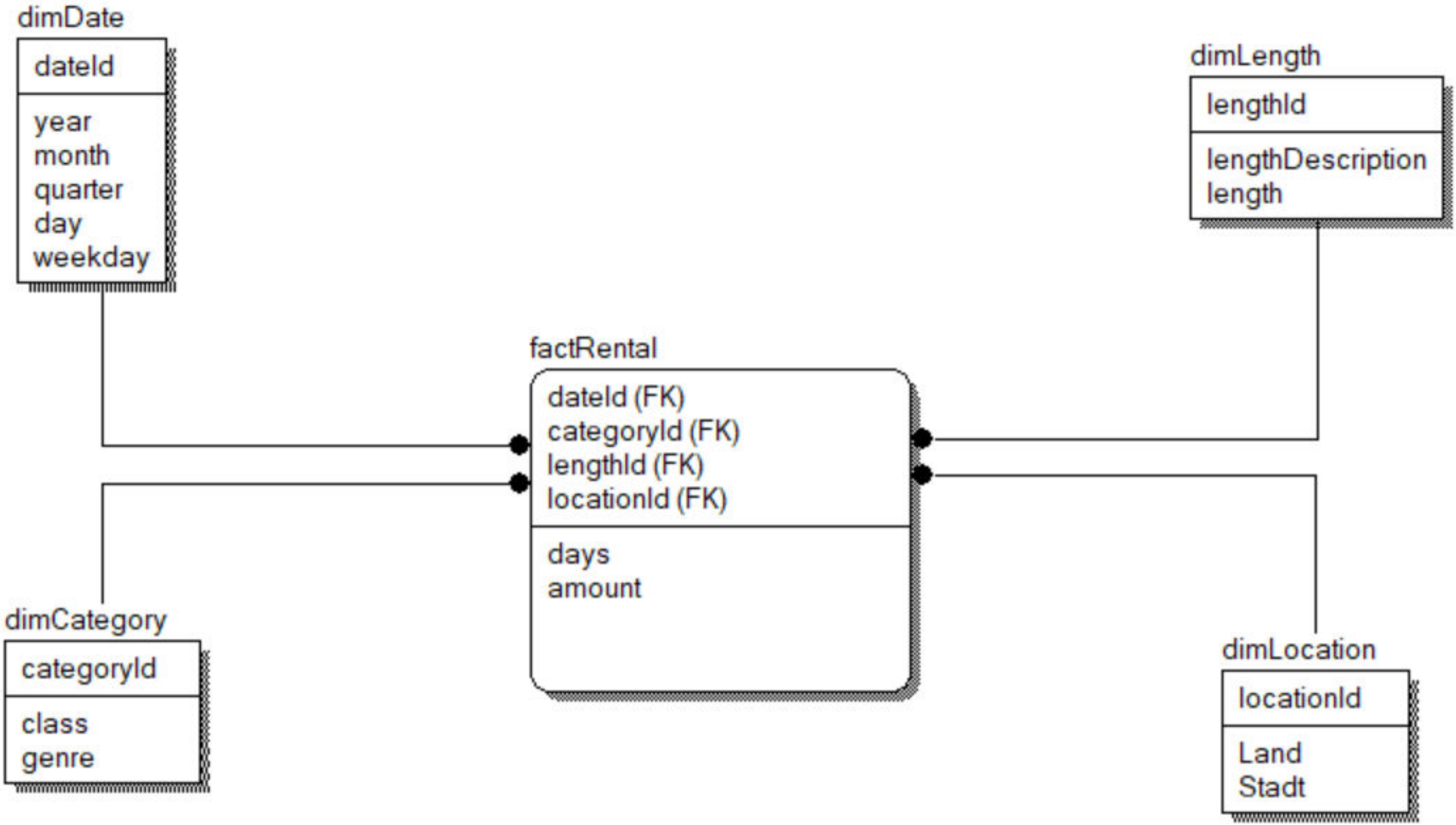
Prüfen Sie Ihre Funktion mit folgenden Aufrufen:

```
SELECT *
FROM TABLE(get_best_worst_genre());
-- 5 Zeilen

SELECT *
FROM TABLE(get_best_worst_genre(p_country => 'Japan'));
-- 2 Zeilen

SELECT *
FROM TABLE(get_best_worst_genre(p_city => 'Linz'));
-- 3 Zeilen

SELECT *
FROM TABLE(get_best_worst_genre(p_city => 'Vancouver',
                                p_country => 'Canada'));
-- 2 Zeilen
```


dimDate
dateId
year
month
quarter
day
weekday
dimLength
lengthId
lengthDescription
length
factRental
dateId (FK)
categoryId (FK)
lengthId (FK)
locationId (FK)
days
amount
dimCategory
categoryId
class
genre
dimLocation
locationId
Land
Stadt

Zeit
Hierarchy
Jahr
Quartal
Monat
Woche
Tag
Wochentag
Kennzahl
Umsatz
Verleihdauer
Verleih
Zeit
Länge
Filiale
Kategorie
Kennzahl
Länge
Short
Medium
Long
Kategorie
Storyline
Sci-Fi
Sport
Animation
Narrative
Children
Comedy
Documentary
Drama
Family
Foreign
Travel
Mood
Action
Horror
Music
Others
alle Uebrigen
Filiale
Hierarchy
Land
Stadt

```
3)
CREATE SEQUENCE identifiers
START WITH 1
INCREMENT BY 1
MAXVALUE 1000000000;

-- dimLocation: city, country, locationID, mapping-ID
CREATE TABLE dimLocation AS
    SELECT identifiers.nextval AS location_id, city, country,
originalAddressId
    FROM ( SELECT DISTINCT city, country, address_id AS originalAddressId
                FROM store
                    INNER JOIN address USING (address_id)
                    INNER JOIN city USING (city_id)
                    INNER JOIN country USING (country_id)
    );

ALTER TABLE dimLocation ADD CONSTRAINT dimLocation_pk PRIMARY KEY
(location_id);

SELECT * FROM dimLocation;

-- dimCategory:
CREATE TABLE dimCategory AS
    SELECT identifiers.nextval AS category_id,'classification' AS class,
genre, originalCategoryId
    FROM ( SELECT DISTINCT name AS genre, category_id AS originalCategoryId
                FROM category
    );

ALTER TABLE dimCategory ADD CONSTRAINT dimCategory_pk PRIMARY KEY
(category_id);
ALTER TABLE dimCategory MODIFY class VARCHAR2(25) NOT NULL;

SELECT * FROM dimCategory;

UPDATE dimCategory
SET class = 'Storyline'
WHERE genre IN ('Animation', 'Sci-Fi', 'Sports');

UPDATE dimCategory
SET class = 'Narrative'
WHERE genre IN ('Children', 'Comedy', 'Documentary', 'Drama', 'Family',
'Foreign', 'Travel');

UPDATE dimCategory
SET class = 'Mood'
WHERE genre IN ('Action', 'Horror', 'Music');

UPDATE dimCategory
SET class = 'Others'
WHERE genre NOT IN ('Animation', 'Sci-Fi', 'Sports', 'Children', 'Comedy',
'Documentary', 'Drama', 'Family', 'Foreign', 'Travel','Action', 'Horror',
'Music');

--dimDate:
CREATE TABLE dimDate
```

```
(
    date_id  NUMBER,
    ddate    DATE,
    year     NUMBER,
    month    NUMBER,
    quarter NUMBER,
    day      NUMBER,
    weekday VARCHAR(10)
);
ALTER TABLE dimDate ADD CONSTRAINT dimDate_pk PRIMARY KEY (date_id);

DECLARE
    startDate DATE;
    endDate DATE := SYSDATE;
BEGIN
    SELECT TRUNC(MIN(payment_date)) INTO startDate FROM payment;

    FOR n IN 0..(endDate - startDate) LOOP
        INSERT INTO dimDate (date_id, ddate, year, month, quarter, day,
weekday)
        VALUES(identifiers.nextval, startDate + n, EXTRACT(year FROM
startDate + n),
               EXTRACT(month FROM startDate + n), to_char(startDate + n,
'Q'),
               EXTRACT(day FROM startDate + n), to_char(startDate + n,
'Day'));
    END LOOP;
END;
/

SELECT * FROM dimDate;

--dimLength:
CREATE TABLE dimLength AS
    SELECT identifiers.nextval AS length_id, length,
        CASE WHEN length <= 60 THEN 'short'
             WHEN length > 60 AND length <= 120 THEN 'medium'
             WHEN length > 120 THEN 'long'
        END AS lengthDescription,
        film_id AS originalFilmId
        FROM film;

ALTER TABLE dimLength ADD CONSTRAINT dimLength_pk PRIMARY KEY (length_id);

DROP TABLE dimLength;

SELECT * FROM dimLength;

SELECT * FROM FILM;


-- factRental: Hue: dimLocation und Aktualisierungszeitpunkt

CREATE MATERIALIZED VIEW factRental
AS
SELECT CEIL(return_date - rental_date) days, amount, dimLocation.location_id,
dimDate.date_id, dimCategory.category_id, dimLength.length_id
```

```
FROM rental
    LEFT JOIN payment USING (rental_id)
    INNER JOIN inventory USING (inventory_id)
    INNER JOIN store USING (store_id)
    INNER JOIN film_category USING (film_id)
    INNER JOIN dimLocation ON (store.address_id =
dimLocation.originalAddressId)        -- Mapping
    INNER JOIN dimDate ON (TRUNC(payment_date) = dimDate.ddate)
-- Mapping
    INNER JOIN dimCategory ON (dimCategory.originalCategoryId =
film_category.category_id)
    INNER JOIN dimLength ON (dimLength.originalFilmId = film_id);

DROP MATERIALIZED VIEW factRental;


SELECT COUNT(*), SUM(amount), class
FROM factRental
    INNER JOIN dimCategory USING (category_id)
GROUP BY class;

SELECT * FROM factRental;
```

4)

```
--4
--4.1
SELECT lengthDescription, COUNT(lengthDescription) AS rentals
FROM factRental
    INNER JOIN dimLength USING (length_id)
GROUP BY lengthDescription
ORDER BY rentals DESC;

--Anhand der Ausgabe kann man sehen, dass Filme mit einer Länge >= 120
Minuten
--am öftesten ausgeliehen werden. Diese sind dicht gefolgt von Filmen mit
einer Länge
--zwischen 60 und 120 Minuten und das Schlusslicht bilden Filme mit einer
Länge von
--weniger als 60 Minuten.

--4.2
SELECT genre, ROUND(AVG(amount), 2)
FROM factRental
    INNER JOIN dimCategory USING (category_id)
GROUP BY genre
ORDER BY AVG(amount) DESC;

--An den Daten kann man sehen, dass 'Sci-Fi' Filme die lukrativsten sind und
--somit ist das Genre wahrscheinlich das beliebteste der Kunden. Platz zwei
--und drei belegen in diesem Fall das Genre 'Travel' und 'Documentary'.

--4.3
SELECT quarter, SUM(amount)
FROM factRental
    INNER JOIN dimDate USING(date_id)
```

```
GROUP BY quarter
ORDER BY SUM(amount) DESC;

--Am Ergebnis lässt sich feststellen, dass im ersten Quartal eines Jahres die
--die meisten Filme ausgeliehen werden. Folglich ist das erste Quartal das
lukrativste.

--4.4
SELECT month, quarter,
       AVG(days)
FROM factRental
    INNER JOIN dimDate USING (date_id)
GROUP BY GROUPING SETS((month), (quarter), ())
ORDER BY AVG(days) DESC;

--Die Ausleihdauer ist im ersten Monat eines Jahres die Längste und führt
--mit einer durchschnittlichen Anzahl von 5.5 Tagen.
--Folglich ist das erste Quatal eines Jahres, jenes mit der Längsten
durchschnittlichen
--Ausleihdauer.
--Überraschenderweise ist das vierte Quartal, obwohl dieses bei der Abfrage,
welches
--Quartal am lukrativsten ist, den vierten Platz belegt, in dieser Abfrage
auf dem zweiten Platz
--der Quartale.

--4.5
SELECT country, lengthDescription, COUNT(*), SUM(amount)
FROM factRental
    INNER JOIN dimLength USING (length_id)
    INNER JOIN dimLocation USING (location_id)
GROUP BY CUBE(country, lengthDescription)
ORDER BY SUM(amount);

--Österreich und Australien leihen am wenigsten Filme aus, wohingegen Japan
der
--Spitzenreiter ist. Australier leihen sich eher Filme, mit einer Länge
zwischen 60 und 120
--Minuten aus, als Filme, welche länger als 120 Minuten dauern.
--Da Filme mit einer Laufzeit von weniger als 60 Minuten, am wenigsten zum
Gewinn beitragen,
--wäre es durchaus eine Überlegung wert diese Filme aus dem Sortiment zu
nehmen.

--4.6
SELECT country, weekday, COUNT(*)
FROM factRental
    INNER JOIN dimLocation USING (location_id)
    INNER JOIN dimDate USING (date_id)
GROUP BY weekday, country
ORDER BY country, count(*);

--Australien bildet das Schlusslicht. Dort werden an einem Montag nur um die
353 Filme
--ausgeliehen.
--Überraschend ist, dass Österreich an einem Sonntag die wenigsten Filme
ausleiht.
```

```
--Möglicherweise weil der Sonntag in Österreich einen hohen Stellenwert hat.
--Japan hingegen sitzt an der Spitze der Ergebnisse, denn dort werden an
einem Sonntag bis zu
--818 Filme ausgeliehen. Das könnte auf die hohe Bevölkerungszahl
zuruckzuführen sein.

--4.7
SELECT month, year, sum_Month,
       SUM(sum_Month) OVER (PARTITION BY year
                            ORDER BY year, month ROWS BETWEEN
                            UNBOUNDED PRECEDING AND CURRENT ROW)
       AS cumulative_Sum
FROM (SELECT month, year, SUM(amount) AS sum_Month
      FROM factRental
            INNER JOIN dimCategory USING (category_id)
            INNER JOIN dimDate USING (date_id)
      WHERE class = 'Narrative'
      GROUP BY year, month)
GROUP BY month, year, sum_Month;

--Auffallend ist vor allem die Tatsache, dass im Jahr 2014 vom Januar bis
November
--mehr eingenommen wurde, als im Jahr 2015 im selben Zeitraum.
```

5)

```
CREATE OR REPLACE FUNCTION get_best_worst_genre (p_city IN VARCHAR2 := NULL,
p_country IN VARCHAR2 := NULL)
RETURN genre_tab PIPELINED AS
    min_amount NUMBER := 0;
    max_amount NUMBER := 0;
    info VARCHAR2(60) := NULL;

    CURSOR genre_cursor (min_amount NUMBER, max_amount NUMBER) IS
        SELECT genre, amount
            FROM factRental
                INNER JOIN dimCategory USING (category_id)
                INNER JOIN dimLocation USING (location_id)
            WHERE amount IN (min_amount, max_amount) AND p_city = city AND
p_country = country;

BEGIN
    SELECT MIN(amount), MAX(amount) INTO min_amount, max_amount
    FROM factRental
        INNER JOIN dimCategory USING (category_id)
        INNER JOIN dimLocation USING (location_id)
    WHERE  p_city = city AND p_country = country;

    FOR genre_rec IN genre_cursor(min_amount, max_amount) LOOP
        IF genre_rec.amount = min_amount THEN
            info := 'worst';
        ELSE
            info := 'best';
        END IF;
        info := info || ' in genre ' || genre_rec.genre;
        -- hier erfolgt die laufende Übergabe an das aufrufende Query
        PIPE ROW(genre_row(genre_rec.genre, genre_rec.amount, info));
    END LOOP;
```

```
  RETURN;        -- RETURN ist leer, da nichts mehr zurückgegeben wird
END;
/
```