

**كلية الهندسة المعلوماتية**

**السنة الخامسة - اختصاص هندسة البرمجيات**

**جامعة دمشق**

=

**وظيفة قواعد المعطيات المتقدمة**

إشراف:

م. عبد البديع مراد

م. أبو الخير الصوص



إعداد الطلاب:

❐

❐

عمار رفاعيه

أيهم الرفاعي

❐

❐

نور الدين الطيلوني

محمد مؤيد يونس

**الفهرس**

[القسم الأول: أوراكل 3](#_Toc153974613)

[الطلب الأول: **تعريف Tablespace باسم homeworkts حجمه 400MB مؤلف من أربع ملفات معطيات Datafile.** 3](#_Toc153974614)

[الطلب الثاني: **تعريف Profile باسم homeworkpf يحدد من خلاله السماح لمستخدم واحد الاتصال بقاعدة المعطيات بشرط أن تكون مدة الاتصال الفعال ساعة ومدة الاتصال الغير فعالة عشرة دقائق وضرورة تغيير كلمة السر كل سبعة أيام وحجم الذاكرة 50 كيلو بايت.** 4](#_Toc153974615)

[الطلب الثالث: **تعريف حساب User جديد في قاعدة المعطيات باسم homeworku مرتبط بـ Tablespace باسم homeworkts ومنحه. Profile homeworkpf** 5](#_Toc153974616)

[الطلب الرابع: **إعطاء كافة الصالحيات لهذا المستخدم من خلال Role واحدة، وأن يكون له صالحيات قراءة من جدول departments وتعديل وحذف بيانات جدول employees من حساب HR.** 6](#_Toc153974617)

[الطلب الخامس: **إجراء نسخة احتياطيه للحساب homeworku دون أخذ بيانات الجداول، وعرض ملف LOG.** 7](#_Toc153974618)

[انشاء قاعد المعطيات (تأجير الأفلام) 8](#_Toc153974619)

[**Language** 8](#_Toc153974620)

[**Category** 8](#_Toc153974621)

[**Actor** 8](#_Toc153974622)

[**Film** 8](#_Toc153974623)

[**Film\_Actor** 9](#_Toc153974624)

[**Film\_Category** 9](#_Toc153974625)

[**City** 9](#_Toc153974626)

[**Country** 9](#_Toc153974627)

[**Customer** 10](#_Toc153974628)

[**Address** 10](#_Toc153974629)

[**Inventory** 10](#_Toc153974630)

[**Store** 10](#_Toc153974631)

[**Rental** 11](#_Toc153974632)

[**Staff** 11](#_Toc153974633)

[**Payment** 11](#_Toc153974634)

[**Database Diagram** 12](file:///C:\Users\ahmad\OneDrive\Desktop\Database\Database.docx#_Toc153974635)

[القسم الثاني: قواعد المعطيات الفعالة وPL\_SQL 13](#_Toc153974636)

[الطلب الأول: **إضافة الأفلام تتم فقط يوم الخميس بين الساعة السادسة والثامنة صباحا، ويمنع حذف أي فلم.** 13](#_Toc153974637)

[الطلب الثاني: **قيمة الايجار في جدول الدفعات يجب أن تكون مساوية لقيمة تكلفة الايجار من جدول الافلام كافة أيام الاسبوع، الا في حال كان يوم الايجار سبت أو أحد تضاف قيمة %15 على تكلفة الايجار.** 15](#_Toc153974638)

[الطلب الثالث: **منع تأجير نفس الفيلم لنفس الزبون أكثر من مرة بالشهر.** 16](#_Toc153974639)

[الطلب الرابع: **بناء سجل متابعة لعمليات الإضافة والتعديل لجدول الحجوزات في جدول رديف (يجب بناء الجدول) يسجل قيم جميع الحقول قبل وبعد تنفيذ العملية مع نوع وتاريخ وزمن العملية.** 18](#_Toc153974640)

[الطلب الخامس: **تســــجيل زمــــن دخــــول وخـــروج أي مســــتخدم لقاعــــدة المعطيــــات فــــي جــــدول مناســــب ضـــمن حســــاب homeworku يستثنى من ذلك حسابي System ,Sys.** 20](#_Toc153974641)

[القسم الثالث: قواعد المعطيات متعددة الابعاد 21](#_Toc153974642)

[**Customer\_Dim** 22](#_Toc153974643)

[**Film\_Dim** 22](#_Toc153974644)

[**Rental\_Fact** 23](#_Toc153974645)

[**Time\_Dim** 23](#_Toc153974646)

[الطلب الأول: **بناء كافة الابعاد مع مراعاة عملية التعديل التاريخية لبعد الزبون، وتحديد جميع الهرميات المناسبة لكل بعد.** 23](#_Toc153974647)

[الطلب الثاني: **تجزئة جدول Fact بالطريقة التي تراها مناسبة وتعليل ذلك، هل من الممكن أن تكون هذه التجزئة مركبة وضح ذلك.** 23](#_Toc153974648)

[الطلب الثالث: **بناء VIEW MATERIALIZED يتضمن اسم الفيلم واسم الزبون وفترة الحجز، تتحدث بيانات هذا المنظور بشكل تراكمي.** 23](#_Toc153974649)

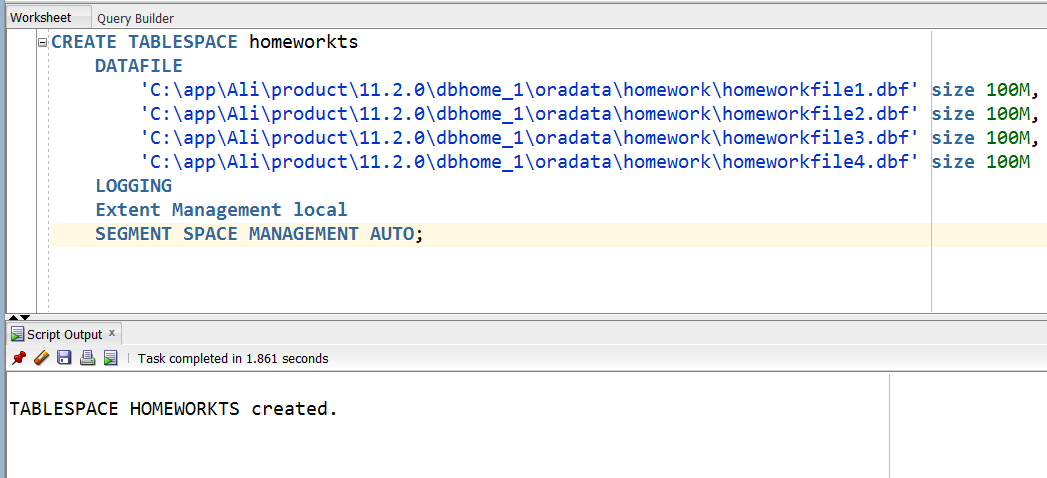
[الطلب الرابع: **عرض قائمة بقيم حجوزات الأفلام على مستوى الشهر والبلد والمدينة بكافة الاحتمالات الممكنة.** 23](#_Toc153974650)

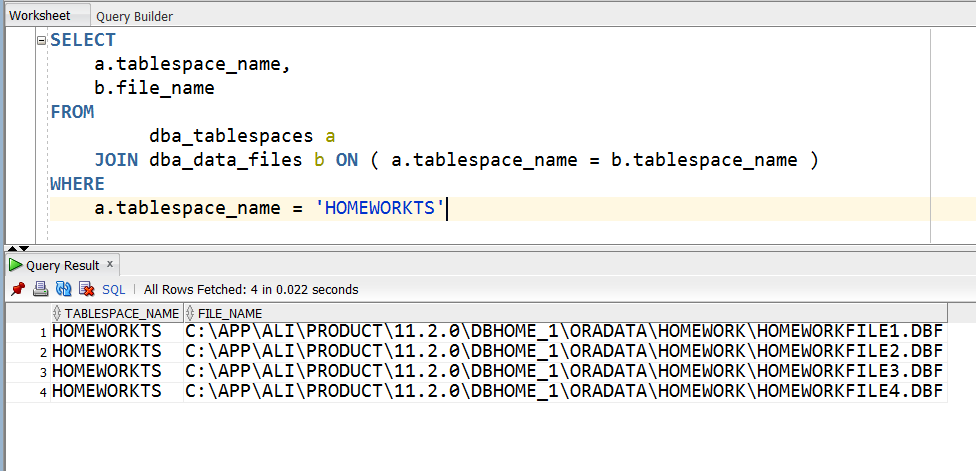
[الطلب الخامس: **أوجد ترتيب كل سنة وكل فيلم ضمن هذه السنة حسب قيمة التأجير.** 24](#_Toc153974651)

[الطلب السادس: **يريد مدير الشركة أن يعرف ما هي العالقة ما بين مدينة الزبون وفئة الأفلام ولغة الفيلم لكي يستخدم هذه المعلومة في الإعلانات الموجهة، اقترح طريقة مناسبة لمعرفة هذه العالقة ثم نفذها باستخدام weka.** 24](#_Toc153974652)

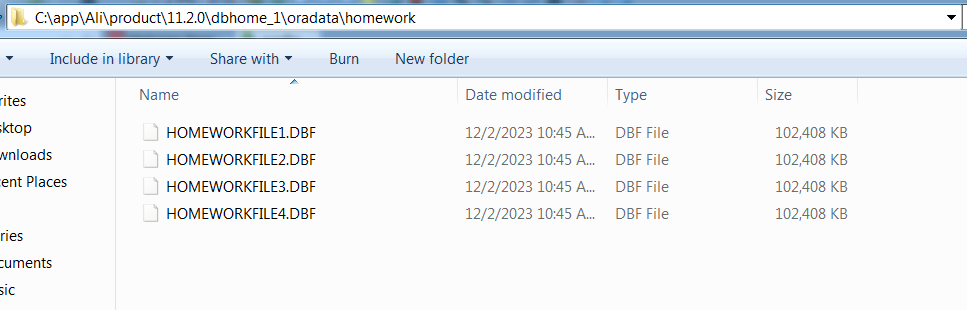
# القسم الأول: **أوراكل**

## الطلب الأول: **تعريف Tablespace باسم homeworkts حجمه 400MB مؤلف من أربع ملفات معطيات Datafile.**

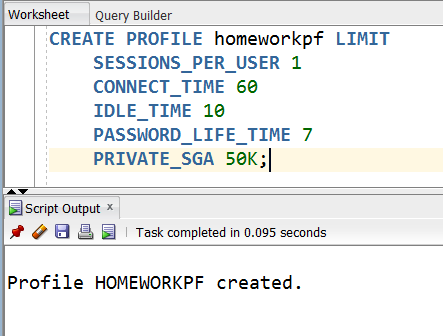




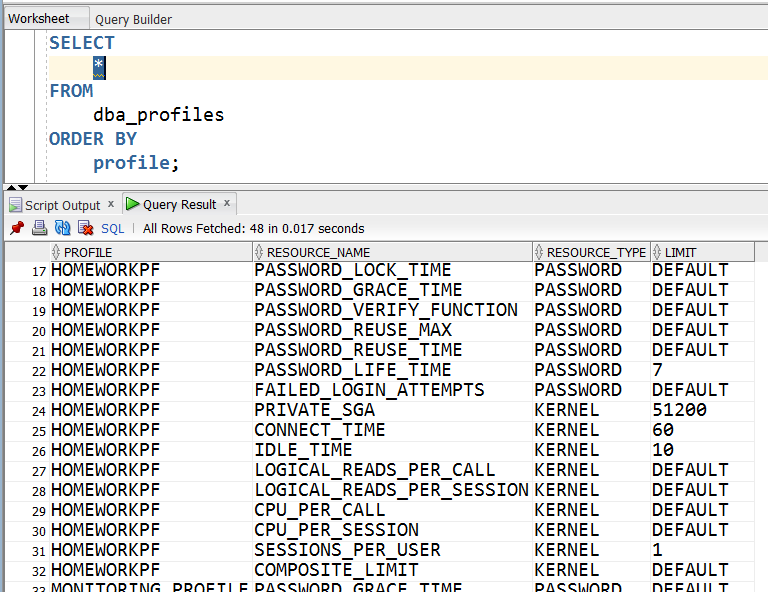
وبعد انشاء الـ TABLESPACE نتأكد من وجوده ومن الملفات الـ Datafiles المرتبطة به عن طريق الاستعلام التالي:

ونجد أيضاً الملفات في المسار المذكور:

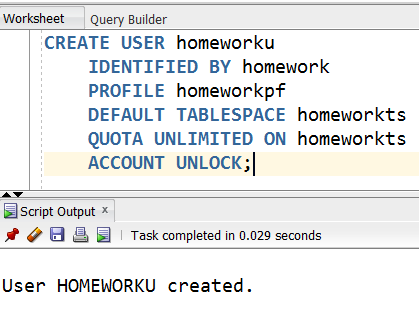
## الطلب الثاني: **تعريف Profile باسم homeworkpf يحدد من خلاله السماح لمستخدم واحد الاتصال بقاعدة المعطيات بشرط أن تكون مدة الاتصال الفعال ساعة ومدة الاتصال الغير فعالة عشرة دقائق وضرورة تغيير كلمة السر كل سبعة أيام وحجم الذاكرة 50 كيلو بايت.**

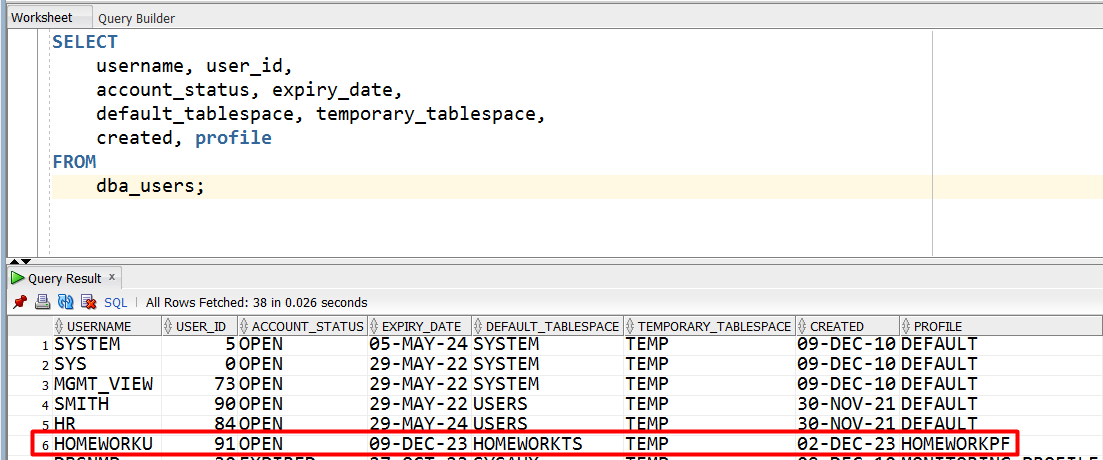


ونتأكد من انشاء الـ Profile والخصائص المسندة له بالاستعلام التالي:

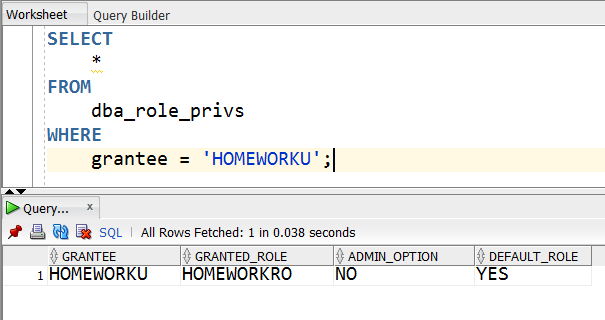


## الطلب الثالث: **تعريف حساب User جديد في قاعدة المعطيات باسم homeworku مرتبط بـ Tablespace باسم homeworkts ومنحه. Profile homeworkpf**

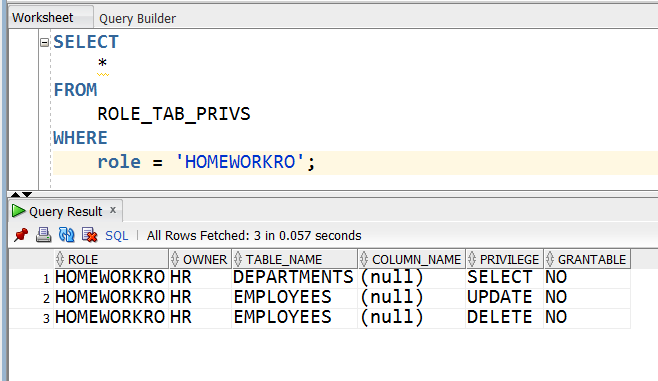
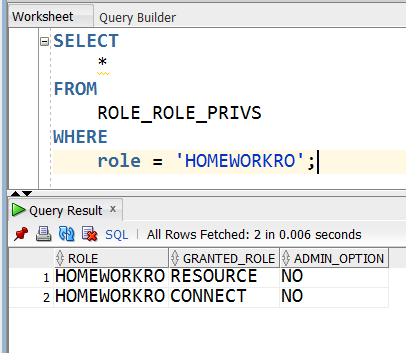


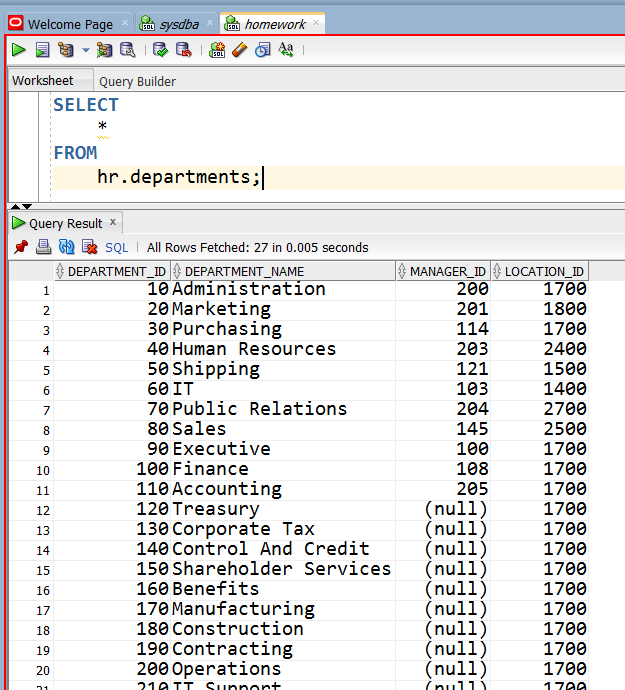
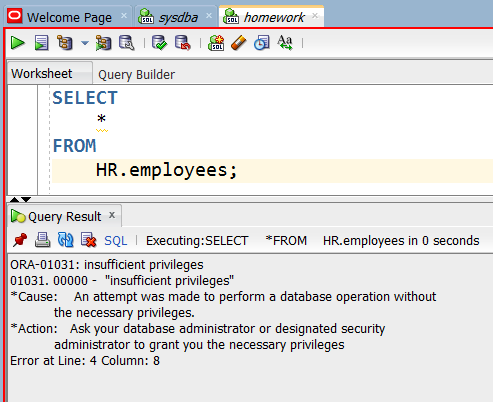
نتأكد من انشاء الـ User والخصائص المسندة له بالاستعلام التالي:

## الطلب الرابع: **إعطاء كافة الصالحيات لهذا المستخدم من خلال Role واحدة، وأن يكون له صالحيات قراءة من جدول departments وتعديل وحذف بيانات جدول employees من حساب HR.**



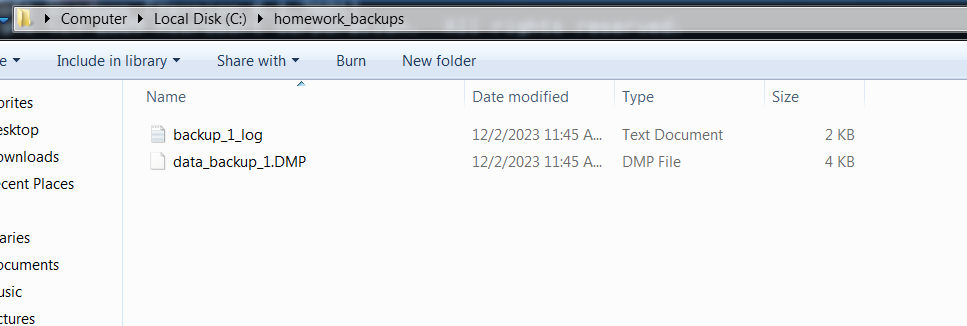
قمنا بإنشاء Role باسم homeworkro وقمنا بإعطائها كافة الصلاحيات اللازمة والمطلوبة في نص السؤال وفي السطر الأخير قمنا بإسناد هذا الـ Role للمستخدم الذي انشأناه سابقا ونتأكد من الاسناد بالاستعلام التالي:

ولرؤية الصلاحيات التي اسندناها للـ Role نقوم بالاستعلامات التالية

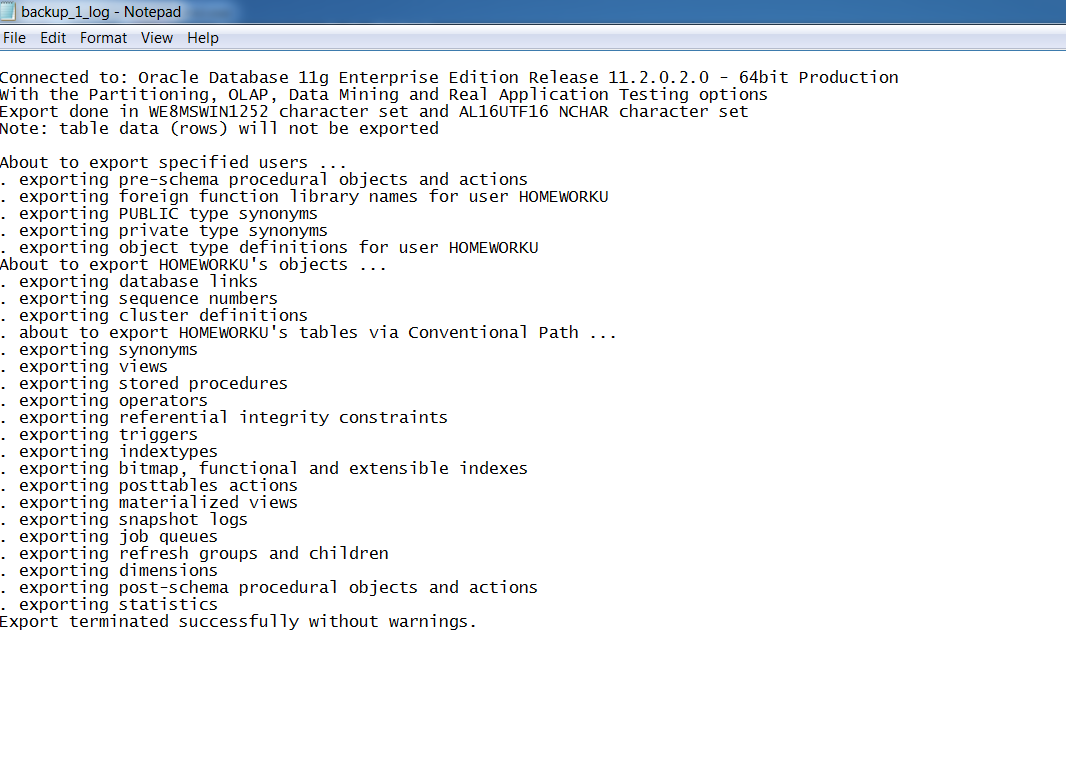
وإذا قمنا بتجربة تنفيذ الاستعلامات على جدولي employees and departments من حساب hr. وضمن الـ User (homeworku) سينجح من اجل الأقسام ويفشل من اجل الموظفين كالتالي:

## *New Text Document.txt - Notepadالطلب الخامس: **إجراء نسخة احتياطيه للحساب homeworku دون أخذ بيانات الجداول، وعرض ملف LOG.**

نقوم بكتابة التعليمة التالية في الـ CMD ضمن مجلد الـ BIN الموجود في مجلد قاعدة المعطيات لـ Oracle:



فيتم انشاء النسخة الاحتياطية ضمن المجلد المحدد:

والـ Log file كالتالي:

# **انشاء قاعد المعطيات (تأجير الأفلام)**

|  |  |
| --- | --- |
| **Language** | **Category** |
| CREATE TABLE language (      id          NUMBER(10) NOT NULL,      name        VARCHAR2(20) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT language\_pk PRIMARY KEY (id)  ); | CREATE TABLE category (      id          NUMBER(10) NOT NULL,      name        VARCHAR2(25) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT category\_pk PRIMARY KEY (id)  ); |

|  |  |
| --- | --- |
| **Actor** | **Film** |
| CREATE TABLE actor (      id          NUMBER(10) NOT NULL,      first\_name  VARCHAR2(255) NOT NULL,      last\_name   VARCHAR2(255) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT actor\_pk PRIMARY KEY ( id )  ); | CREATE TABLE film (      id               NUMBER(10) NOT NULL,      title            VARCHAR2(255) NOT NULL,      description      VARCHAR2(255),      release\_year     NUMBER(4),      rental\_duration  NUMBER(10),      rental\_rate      NUMBER(19, 0) NOT NULL,      lentgth          NUMBER(2),      replacement\_cost NUMBER(19, 0),      rating           NUMBER(10),      last\_update      TIMESTAMP,      special\_features VARCHAR2(255),      full\_text        VARCHAR2(255),      languageid       NUMBER(10),      CONSTRAINT film\_pk PRIMARY KEY ( id ),      CONSTRAINT film\_language\_fk FOREIGN KEY ( languageid )          REFERENCES language ( id )              ON DELETE CASCADE  ); |

|  |  |
| --- | --- |
| **Film\_Actor** | **Film\_Category** |
| CREATE TABLE film\_actor (      filmid      NUMBER(10) NOT NULL,      actorid     NUMBER(10),      last\_update TIMESTAMP,      CONSTRAINT film\_actor\_uk UNIQUE (filmid, actorid),      CONSTRAINT film\_film\_actor\_fk FOREIGN KEY (filmid)          REFERENCES film ( id )              ON DELETE CASCADE,      CONSTRAINT film\_actor\_fk FOREIGN KEY ( actorid )          REFERENCES actor ( id )              ON DELETE CASCADE); | CREATE TABLE film\_category (      filmid      NUMBER(10) NOT NULL,      categoryid  NUMBER(10) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT film\_category\_uk UNIQUE (filmid, categoryid),      CONSTRAINT film\_fk FOREIGN KEY (filmid)          REFERENCES film ( id )              ON DELETE CASCADE,      CONSTRAINT film\_category\_fk FOREIGN KEY (categoryid)          REFERENCES category ( id )              ON DELETE CASCADE  ); |

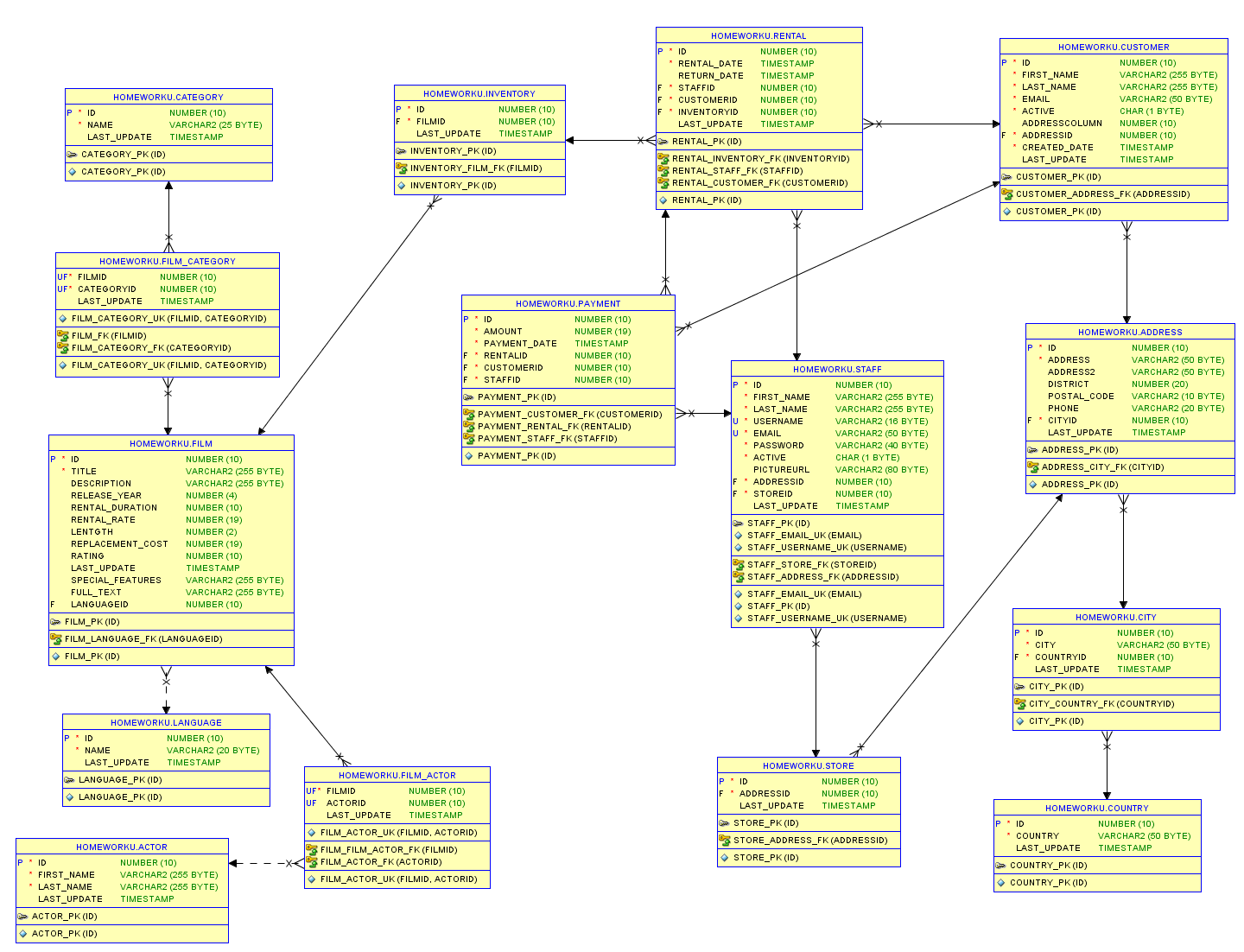
|  |  |
| --- | --- |
| **City** | **Country** |
| CREATE TABLE city (      id          NUMBER(10) NOT NULL,      city        VARCHAR2(50) NOT NULL,      countryid   NUMBER(10) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT city\_pk PRIMARY KEY ( id ),      CONSTRAINT city\_country\_fk FOREIGN KEY (countryid)          REFERENCES country ( id )              ON DELETE CASCADE  ); | CREATE TABLE country (      id          NUMBER(10) NOT NULL,      country     VARCHAR2(50) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT country\_pk PRIMARY KEY (id)  ); |

|  |  |
| --- | --- |
| **Customer** | **Address** |
| CREATE TABLE customer (      id            NUMBER(10) NOT NULL,      first\_name    VARCHAR2(255) NOT NULL,      last\_name     VARCHAR2(255) NOT NULL,      email         VARCHAR2(50) NOT NULL,      active        CHAR(1) NOT NULL,      addresscolumn NUMBER(10),      addressid     NUMBER(10) NOT NULL,      created\_date  TIMESTAMP NOT NULL,      last\_update   TIMESTAMP,      CONSTRAINT customer\_pk PRIMARY KEY ( id ),      CONSTRAINT customer\_address\_fk FOREIGN KEY ( addressid )          REFERENCES address ( id )              ON DELETE CASCADE  ); | CREATE TABLE address (      id          NUMBER(10) NOT NULL,      address     VARCHAR2(50) NOT NULL,      address2    VARCHAR2(50),      district    NUMBER(20),      postal\_code VARCHAR2(10),      phone       VARCHAR2(20),      cityid      NUMBER(10) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT address\_pk PRIMARY KEY ( id ),      CONSTRAINT address\_city\_fk FOREIGN KEY ( cityid )          REFERENCES city ( id )              ON DELETE CASCADE  ); |

|  |  |
| --- | --- |
| **Inventory** | **Store** |
| CREATE TABLE inventory (      id          NUMBER(10) NOT NULL,      filmid   NUMBER(10) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT inventory\_pk PRIMARY KEY (id),      CONSTRAINT inventory\_film\_fk FOREIGN KEY ( filmid )          REFERENCES film ( id )              ON DELETE CASCADE); | CREATE TABLE store (      id          NUMBER(10) NOT NULL,      addressid   NUMBER(10) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT store\_pk PRIMARY KEY ( id ),      CONSTRAINT store\_address\_fk FOREIGN KEY ( addressid )          REFERENCES address ( id )              ON DELETE CASCADE  ); |

|  |  |
| --- | --- |
| **Rental** | **Staff** |
| CREATE TABLE rental (      id          NUMBER(10) NOT NULL,      rental\_date TIMESTAMP ) NOT NULL,      return\_date TIMESTAMP,      staffid     NUMBER(10),      customerid  NUMBER(10),      inventoryid NUMBER(10),      last\_update TIMESTAMP,      CONSTRAINT rental\_pk PRIMARY KEY ( id ),      CONSTRAINT rental\_staff\_fk FOREIGN KEY ( staffid )          REFERENCES staff ( id )              ON DELETE CASCADE,      CONSTRAINT rental\_customer\_fk FOREIGN KEY ( customerid )          REFERENCES customer ( id )              ON DELETE CASCADE,      CONSTRAINT rental\_inventory\_fk FOREIGN KEY ( inventoryid )          REFERENCES inventory ( id )              ON DELETE CASCADE  ); | CREATE TABLE staff (      id          NUMBER(10) NOT NULL,      first\_name  VARCHAR2(255) NOT NULL,      last\_name   VARCHAR2(255) NOT NULL,      username    VARCHAR2(16) NOT NULL,      email       VARCHAR2(50) NOT NULL,      password    VARCHAR2(40) NOT NULL,      active      CHAR(1) NOT NULL,      pictureurl  VARCHAR2(80),      addressid   NUMBER(10) NOT NULL,      storeid     NUMBER(10) NOT NULL,      last\_update TIMESTAMP,      CONSTRAINT staff\_pk PRIMARY KEY ( id ),      CONSTRAINT staff\_username\_uk UNIQUE (username),      CONSTRAINT staff\_email\_uk UNIQUE (email),      CONSTRAINT staff\_address\_fk FOREIGN KEY (addressid)          REFERENCES address ( id )              ON DELETE CASCADE,      CONSTRAINT staff\_store\_fk FOREIGN KEY (storeid)          REFERENCES store ( id )              ON DELETE CASCADE  ); |

|  |
| --- |
| **Payment** |
| CREATE TABLE payment (      id           NUMBER(10) NOT NULL,      amount       NUMBER(19, 0) NOT NULL,      payment\_date TIMESTAMP NOT NULL,      rentalid     NUMBER(10) NOT NULL,      customerid   NUMBER(10) NOT NULL,      staffid      NUMBER(10) NOT NULL,      CONSTRAINT payment\_pk PRIMARY KEY ( id ),      CONSTRAINT payment\_rental\_fk FOREIGN KEY ( rentalid )          REFERENCES rental ( id )              ON DELETE CASCADE,      CONSTRAINT payment\_customer\_fk FOREIGN KEY ( customerid )          REFERENCES customer ( id )              ON DELETE CASCADE,      CONSTRAINT payment\_staff\_fk FOREIGN KEY ( staffid )          REFERENCES staff ( id )              ON DELETE CASCADE  ); |



## **Database Diagram**

# القسم الثاني: **قواعد المعطيات الفعالة وPL\_SQL**

## الطلب الأول: **إضافة الأفلام تتم فقط يوم الخميس بين الساعة السادسة والثامنة صباحا، ويمنع حذف أي فلم.**

CREATE OR REPLACE TRIGGER film\_insert\_delete\_trigger BEFORE

    INSERT OR DELETE ON film

BEGIN

    IF

(to\_char(sysdate, 'dy') != 'thu' AND inserting)

    OR

to\_char(sysdate, 'dy') = 'thu'

AND

(

        (to\_char(sysdate, 'HH24') NOT BETWEEN 6 AND 8

AND inserting)

        OR

(to\_char(sysdate, 'HH24') BETWEEN 6 AND 8 AND deleting)

      )

    THEN

        IF inserting THEN

        raise\_application\_error(-20500, 'Inserting only allowed

On Thursday between 6 AM and 8 AM');

        ELSE

        raise\_application\_error(-20501, 'Deleting not allowed On

Thursday between 6 AM and 8 AM');

        END IF;

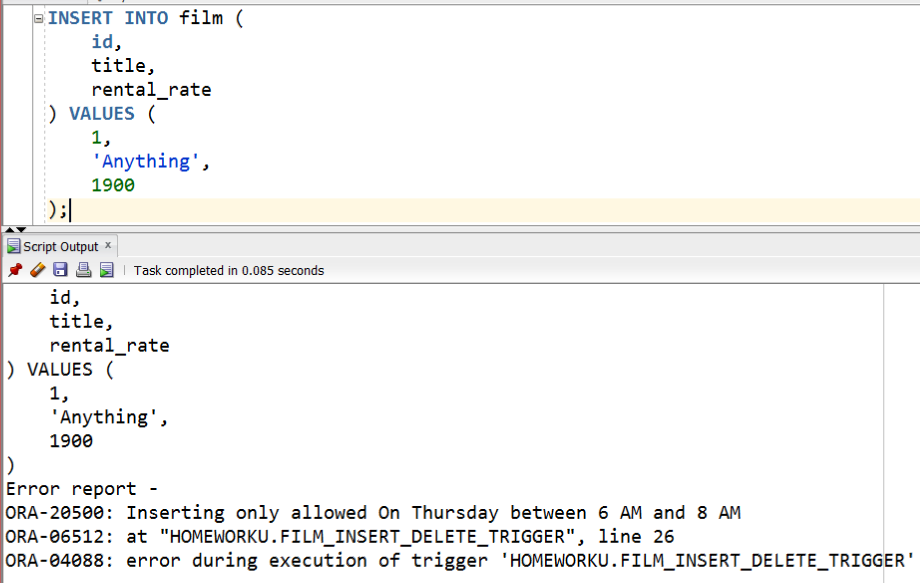
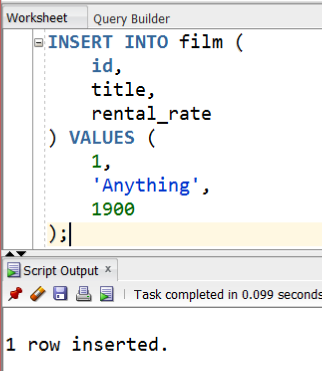
    END IF;

END;

**شرح الشروط الخاصة بالقادح السابق:**

1. إذا لم يكن اليوم هو الخميس وكانت العملية التي نقوم بها هي إضافة فيتم إعطاء خطأ.
2. إذا كان اليوم هو الخميس وكانت العملية هي إضافة فلم ولم يكن الوقت بين السادسة والثامنة صباحا فيتم رفع خطأ **أو** كان اليوم هو الخميس وكانت العملية هي حذف وكان الوقت بين السادسة والثامنة صباحا فيتم رفع خطأ وما تبقى من أوقات وايام فتصح فيها عملية الحذف.

**نلاحظ اننا قمنا بعمل STATEMENT LEVEL TRIGGER وليس ROW LEVEL أي لم نذكر FOR EACH ROW في عملية الانشاء لان الطلب هنا لا يهتم بما تحتويه كل عملية DML من OLD OR NEW VALUES ولا بعددهم انما بمنع العمليات في أوقات محددة وهذه من الاستخدامات الشائعة للـ STATEMENT LEVEL TRIGGER.**

**نقوم بتجربة بعض العمليات كالتالي:**

|  |  |
| --- | --- |
| **إضافة فلم في غير يوم الخميس او في يوم الخميس لكن ليس في الوقت المحدد:** | **إضافة فلم في يوم الخميس في الوقت المحدد:** |
|  |  |

## الطلب الثاني: **قيمة الايجار في جدول الدفعات يجب أن تكون مساوية لقيمة تكلفة الايجار من جدول الافلام كافة أيام الاسبوع، الا في حال كان يوم الايجار سبت أو أحد تضاف قيمة %15 على تكلفة الايجار.**

CREATE OR REPLACE TRIGGER payment\_rental\_film\_trigger BEFORE

    INSERT ON payment

    FOR EACH ROW

DECLARE

    rate NUMBER(19,0);

    less\_than EXCEPTION;

    great\_than EXCEPTION;

BEGIN

    SELECT rental\_rate INTO rate FROM film

    WHERE id = ( SELECT filmid FROM inventory

            WHERE id = ( SELECT inventoryid FROM rental

                    WHERE id = :new.rentalid ) );

    IF to\_char(sysdate, 'dy') IN ('sat','sun')

    THEN

        rate := rate + rate \* 0.15;

    END IF;

    IF :new.amount < rate THEN raise less\_than;

    ELSIf :new.amount > rate THEN raise great\_than;

    END IF;

EXCEPTION

    WHEN less\_than THEN

        raise\_application\_error(-20500,'Payment amount must be

     as same as the Film''s Rental Rate which is: ' || rate

     || ' but you only enterd: ' || :new.amount);

    WHEN great\_than THEN

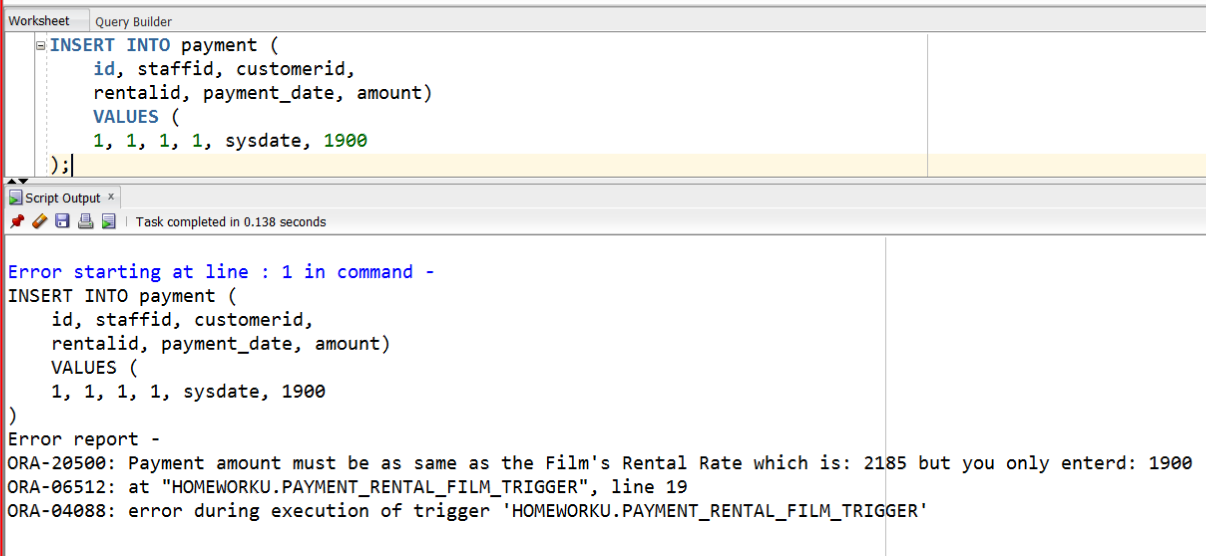
        raise\_application\_error(-20501,'Don''t thief people

     the film''s rental rate is only: ' || rate

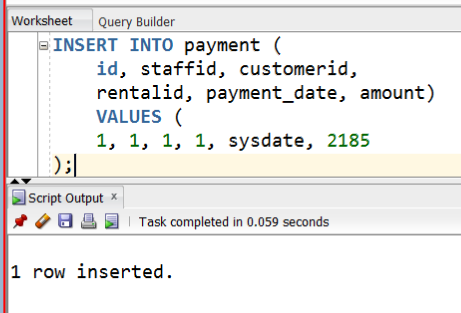
     || ' but you enterd: ' || :new.amount);

END;

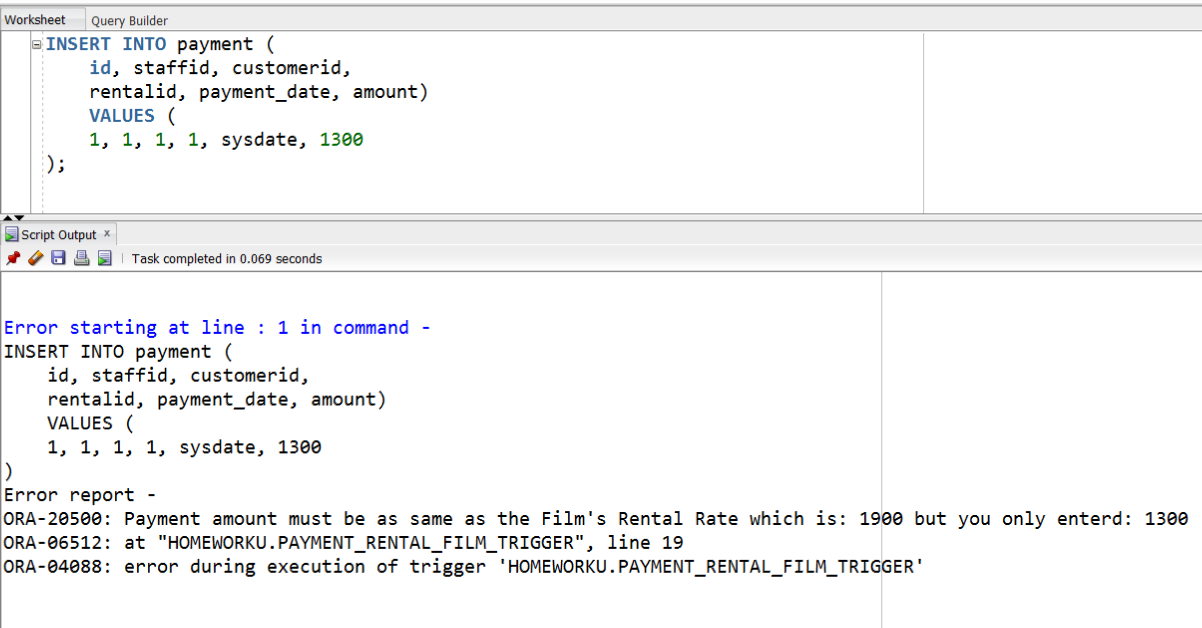
قمنا بعمل استعلام بسيط لجلب سعر ايجار الفلم وذلك بالاستفادة من العلاقات بين الجداول وصولا لجدول الأفلام وبعد ذلك تأكدنا فيما ان كان اليوم هو سبت او أحد واضفنا 15% على سعر الايجار وان لم يكن كذلك سيبقى السعر هو نفسه سعر الايجار للفلم وقمنا برفع خطأ في حال كان اقل او أكثر لان السعر يحب ان يكون متساوي.

تجربة بعض العمليات:

**إضافة دفعة للإيجار بالسعر الموافق لسعر الايجار الخاص بالفلم ولكن في يومي السبت او الاحد:**



**إضافة دفعة للإيجار بسعر مكافئ لسعر الايجار الخاص بالفلم + 15% التي تكون في يومي السبت او الاحد:**



**إضافة دفعة للإيجار بسعر اقل لسعر الايجار الخاص بالفلم في غير يومي السبت او الاحد:**

## الطلب الثالث: **منع تأجير نفس الفيلم لنفس الزبون أكثر من مرة بالشهر.**

CREATE OR REPLACE TRIGGER rental\_film\_once\_trigger BEFORE

    INSERT ON rental

    FOR EACH ROW

DECLARE

    selected\_film NUMBER(10);

    count\_of\_rental\_film\_in\_month number:=0;

    myexc EXCEPTION;

BEGIN

    SELECT filmid INTO selected\_film FROM inventory WHERE

id = :new.inventoryid;

    SELECT COUNT(\*) INTO count\_of\_rental\_film\_in\_month FROM rental

    WHERE customerid = :new.customerid AND inventoryid IN

(SELECT id FROM inventory WHERE filmid = selected\_film)

        AND to\_char(sysdate, 'MM') = to\_char(rental\_date, 'MM');

    IF (count\_of\_rental\_film\_in\_month > 0) THEN

        RAISE myexc;

    END IF;

EXCEPTION

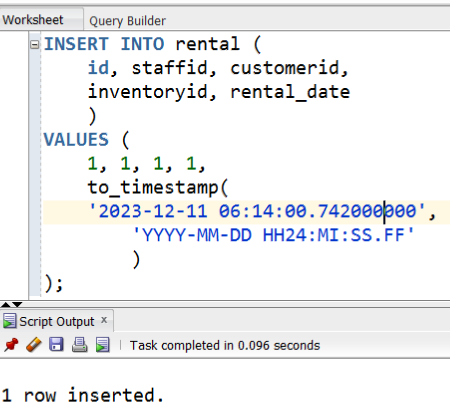
    WHEN myexc THEN

        raise\_application\_error(-20500, 'You Can''t Rental

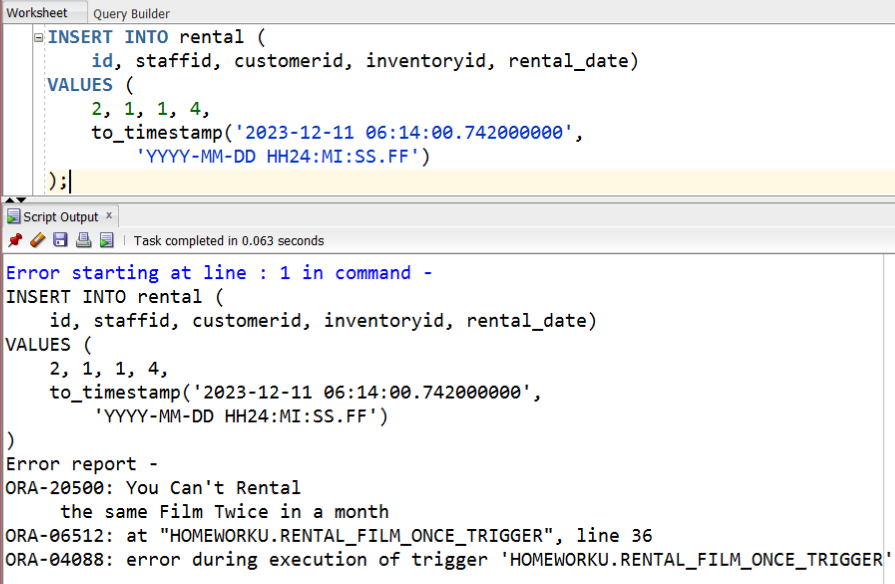
     the same Film Twice in a month');

END;

قمنا أولا بجلب الفلم المراد حجزه ووضعناه ضمن متحول بالاستعلام الأول ثم بالاستعلام الثاني قمنا بجلب عدد الأفلام التي حجزها هذا العميل ولنفس الفلم أيا كان الـ inventory لهذا الفلم وضمن الشهر نفسه ثم اختبرنا انه إذا كان أكبر من الصفر أي لديه حجز سابق لهذا الفلم ضمن الشهر نفسه فيعطي خطأ.

تجربة العمليات لذلك:

**نقوم بإضافة ايجار جديد من اجل العميل صاحب الـ ID رقم 1 للفلم الموجود للـ inventoryid رقم 1 (وهو الفلم الأول) فتتم العملية بنجاح:**



**نقوم بإضافة ايجار جديد من اجل العميل صاحب الـ ID رقم 1 للفلم الموجود للـ inventoryid رقم 4 (وهو ايضاً الفلم الأول) فتفشل العملية لأنه قام بحجز الفلم أكثر من مرة في هذا الشهر:**

## الطلب الرابع: **بناء سجل متابعة لعمليات الإضافة والتعديل لجدول الحجوزات في جدول رديف (يجب بناء الجدول) يسجل قيم جميع الحقول قبل وبعد تنفيذ العملية مع نوع وتاريخ وزمن العملية.**

نقوم ببناء جدول المتابعة أولا وتم اختصار حقول الـ Foreign keys الثلاثة في الجدول للسهولة فقط وإبقاء الحقول الثلاث الأخرى وثم نقوم ببناء أيضا عداد لمعرفة ترتيب عملية التتبع للجدول بالشكل التالي:

|  |  |
| --- | --- |
| CREATE TABLE audit\_rental (  rental\_id NUMBER(10),      old\_rental\_date TIMESTAMP,      old\_return\_date TIMESTAMP,      old\_last\_update TIMESTAMP,      new\_rental\_date TIMESTAMP,      new\_return\_date TIMESTAMP,      new\_last\_update TIMESTAMP,      audit\_seq NUMBER(6),      audit\_kind VARCHAR2(1),      audit\_datetime TIMESTAMP  ); | CREATE SEQUENCE auditseq      MINVALUE 1      MAXVALUE 999999      INCREMENT BY 1      START WITH 1      NOCACHE      ORDER      NOCYCLE; |

**الان نقوم ببناء القادح بالشكل التالي:**

CREATE OR REPLACE TRIGGER audit\_rental\_values\_trigger AFTER

    INSERT OR UPDATE ON rental

    FOR EACH ROW

DECLARE

    rental\_id    NUMBER(10);

    v\_audit\_kind VARCHAR2(1);

    v\_audit\_seq  NUMBER(6);

BEGIN

    IF inserting THEN

        v\_audit\_kind := 'I';

        rental\_id :=:NEW.id;

    ELSIF updating THEN

        v\_audit\_kind := 'U';

        rental\_id :=:OLD.id;

    END IF;

    SELECT auditseq.NEXTVAL INTO v\_audit\_seq FROM dual;

    INSERT INTO audit\_rental VALUES (

        rental\_id,

        :old.rental\_date,

        :old.return\_date,

        :old.last\_update,

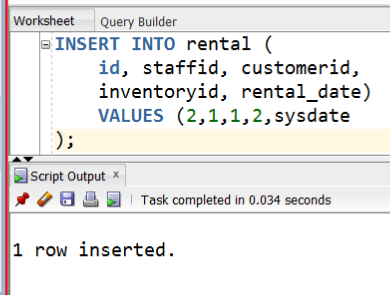
        :new.rental\_date,

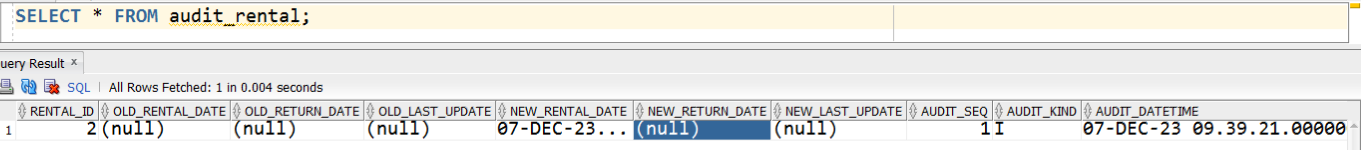
        :new.return\_date,

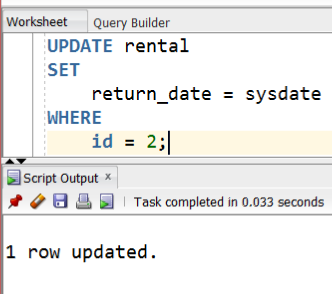
        :new.last\_update,

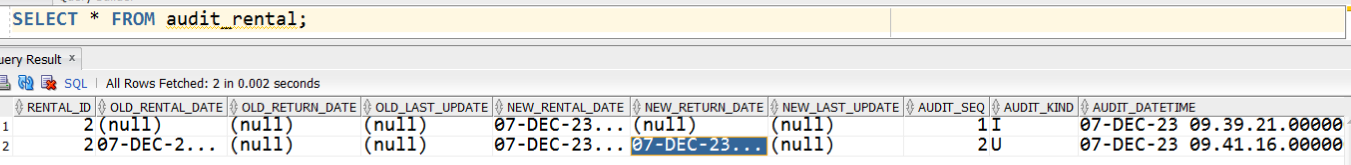
        v\_audit\_seq, v\_audit\_kind, sysdate);

END;

نقوم بإضافة الان سجل جديد الى جدول الحجوزات ومن ثم سنستعلم عن جدول التتبع الخاص بجدول الحجوزات ونرى النتيجة:



سنقوم بتعديل الحقل ونرى النتائج ايضاً:



## الطلب الخامس: **تســــجيل زمــــن دخــــول وخـــروج أي مســــتخدم لقاعــــدة المعطيــــات فــــي جــــدول مناســــب ضـــمن حســــاب homeworku يستثنى من ذلك حسابي System ,Sys.**

نقوم اولاً بإعطاء حساب homeworku سماحية ADMINISTER DATABASE TRIGGER من حساب Sys ليتمكن المستخدم homeworku من انشاء قادح من نوع System Trigger:

نعود لحساب homeworku ونقوم بإنشاء جدول لتتبع عمليات تسجيل الدخول والخروج كالتالي:

CREATE TABLE system\_log (

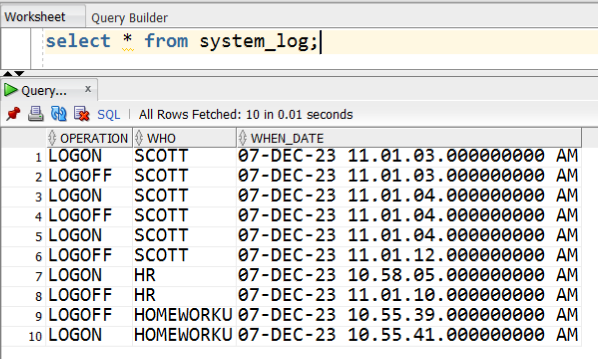
    operation VARCHAR2(30),

    who       VARCHAR2(30),

    when\_date TIMESTAMP);

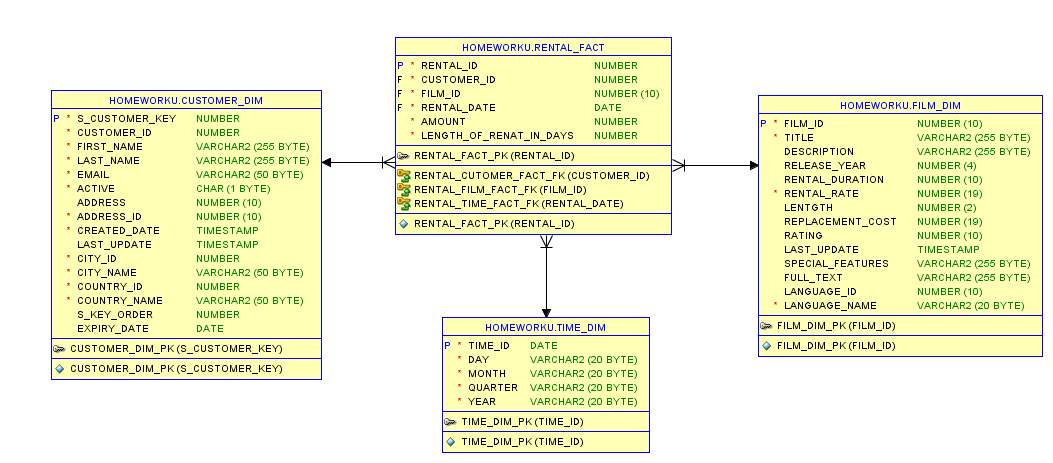
ثم نقوم بإنشاء قادحين الأول للـ LOGON والأخر للـ LOGOFF ولا يمكن جمعهم بقادح واحد لان الـ Logon لا تعمل الا بالـ AFTER والـ Logoff لا يعمل الا بالـ BEFORE:

|  |  |
| --- | --- |
| CREATE OR REPLACE TRIGGER  track\_logoff\_event\_trigger BEFORE LOGOFF ON DATABASE BEGIN      IF user NOT IN ('SYS', 'SYSTEM') THEN INSERT INTO system\_log (operation, who, when\_date) VALUES (ora\_sysevent, user, sysdate);      END IF;  END; | CREATE OR REPLACE TRIGGER  track\_logon\_event\_trigger AFTER LOGON ON DATABASE BEGIN      IF user NOT IN ('SYS', 'SYSTEM') THEN INSERT INTO system\_log (operation, who, when\_date) VALUES (ora\_sysevent, user, sysdate);      END IF;  END; |

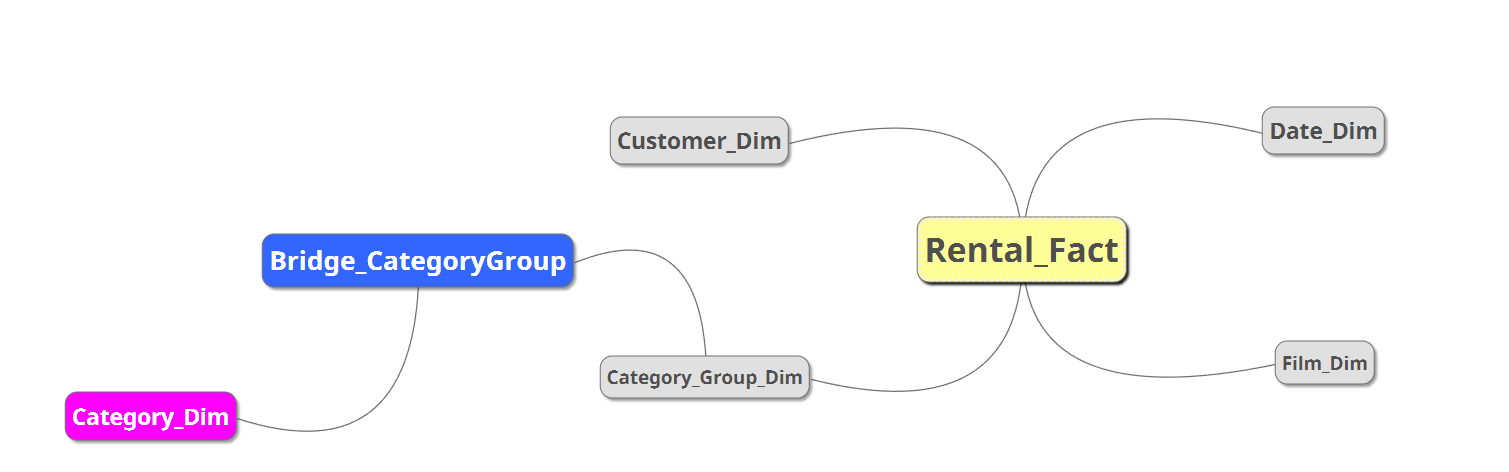


الان لو قمنا بتجريب تسجيل الدخول والخروج بعدة حسابات ومنها الـ SYS AND SYSTEM وقمنا بالاستعلام التالي فستظهر النتائج كالتالي:

# القسم الثالث: **قواعد المعطيات متعددة الابعاد**

سنقوم ببناء المخطط من نوع Star Schema والذي سيمكننا من الحصول على اعلى أداء في الاستعلامات عن الاحصائيات ومحور الـ Fact table هما جدولا الـ Payment And Rental والمخطط بالشكل التالي:

ملاحظة مهمة لسبب اختيار المخطط والابعاد:

لم نقم بتضمين ابعاد نحن لسنا بحاجة لها من قراءتنا لنص الاسئلة في هذا القسم مثل الـ Staff, Store, etc. ومثلا ان كنا بحاجة لمعرفة فئات الأفلام Category وهي تملك علاقة Many-to-Many مع الـ Film هنا سنضطر لجعل المخطط Hybrid أي إضافة تفاصيل للأبعاد لن يتم البحث ضمنها الا عند الحاجة لتفاصيل فئات الفلم مثلا ونقوم على سبيل المثال باستخدام طريقة الـ Bridge Table كـ إحدى الطرق لحل هذه المشكلة والشكل التالي هو مجرد شكل لتوضيحي لكيف سيكون الشكل لمخطط:

نقوم بإنشاء الجداول:

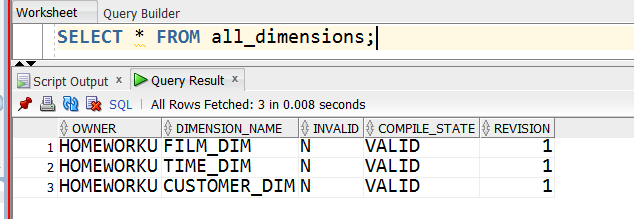
|  |  |
| --- | --- |
| **Customer\_Dim** | **Film\_Dim** |
| CREATE TABLE customer\_dim (    s\_customer\_key  NUMBER NOT NULL,    customer\_id     NUMBER NOT NULL,    first\_name    VARCHAR2(255) NOT NULL,    last\_name       VARCHAR2(255) NOT NULL,    email           VARCHAR2(50) NOT NULL,    active         CHAR(1) NOT NULL,    created\_date   TIMESTAMP NOT NULL,    last\_update   TIMESTAMP,    address\_id     NUMBER(10) NOT NULL,    address       VARCHAR2(50) NOT NULL,    city\_id       NUMBER(10) NOT NULL,    city\_name     VARCHAR2(50) NOT NULL,    country\_id     NUMBER(10) NOT NULL,    country\_name   VARCHAR2(50) NOT NULL,  s\_key\_order NUMBER,    expiry\_date   DATE,    CONSTRAINT customer\_dim\_pk PRIMARY KEY  ( s\_customer\_key )  ); | CREATE TABLE film\_dim (    film\_id         NUMBER NOT NULL,  title           VARCHAR2(255) NOT NULL,    description     VARCHAR2(255),   release\_year   NUMBER(4),    rental\_duration NUMBER(10),    rental\_rate     NUMBER(19, 0) NOT NULL,    lentgth         NUMBER(2),    replacement\_cost NUMBER(19, 0),    rating           NUMBER(10),    last\_update       TIMESTAMP,    special\_features VARCHAR2(255),    full\_text         VARCHAR2(255),    language\_id       NUMBER(10),    language\_name     VARCHAR2(20) NOT NULL,    CONSTRAINT film\_dim\_pk PRIMARY KEY (film\_id)  ); |
| **Rental\_Fact** | **Time\_Dim** |
| CREATE TABLE rental\_fact (    rental\_id     NUMBER NOT NULL,    customer\_id    NUMBER NOT NULL,    film\_id       NUMBER NOT NULL,    rental\_date    DATE NOT NULL,    amount        NUMBER NOT NULL,    length\_of\_renat\_in\_days NUMBER NOT NULL,    CONSTRAINT rental\_fact\_pk PRIMARY KEY ( rental\_id ),    CONSTRAINT rental\_cutomer\_fact\_fk FOREIGN KEY ( customer\_id ) REFERENCES customer\_dim (s\_customer\_key ),    CONSTRAINT rental\_film\_fact\_fk FOREIGN KEY (film\_id ) REFERENCES film\_dim (film\_id ),    CONSTRAINT rental\_time\_fact\_fk FOREIGN KEY (rental\_date ) REFERENCES time\_dim (time\_id )  ); | CREATE TABLE time\_dim (  time\_id DATE NOT NULL,  day     VARCHAR2(20) NOT NULL,  month   VARCHAR2(20) NOT NULL,  quarter VARCHAR2(20) NOT NULL,  year    VARCHAR2(20) NOT NULL,  CONSTRAINT time\_dim\_pk PRIMARY KEY (time\_id)  ); |

## الطلب الأول: **بناء كافة الابعاد مع مراعاة عملية التعديل التاريخية لبعد الزبون، وتحديد جميع الهرميات المناسبة لكل بعد.**

قمنا بإضافة Surrogate Key لجدول الزبون بالإضافة لحقلين إضافيين هما s\_key\_order و expiry\_date وبهذه الحقول سنقوم بحفظ جميع التعديلات التي ستتم للزبون. سيتم ضافة حقل جديد عند أي تغيير في بيانات الزبون وستصبح قيم كل من s\_key\_order بقيمة ترتيب التعديل الذي طرأ أي ان كانت المرة الأول فسيصبح 1 وللمرة الثانية 2 وهكذا وبالنسبة للـ expiry\_date ستصبح بتاريخ حدوث التعديل وهذان الحقلان سيكونان بقيمة null لأخر نسخة لبيانات الزبون وبالنسبة للـ key يوجد عدة طرق لتعبئته وهو ليس مرتبط بأي قيمة من الجدول الأساسي يمكن توليد قيمه من عداد Sequence وملاحظة بهذه الطريقة المستخدمة سيترتب عبء علينا بالاستعلامات التي تخص الزبون لان الزبون قد يوجد له اكثر من حقل في بعد الزبون.

بناء الابعاد وتحديد الهرميات:

|  |  |
| --- | --- |
| **Customer\_Dim** | **Film\_Dim** |
| CREATE DIMENSION customer\_dim    LEVEL s\_customer\_key IS customer\_dim.s\_customer\_key    LEVEL customer\_id IS customer\_dim.customer\_id    LEVEL first\_name IS customer\_dim.first\_name    LEVEL last\_name IS customer\_dim.last\_name    LEVEL email IS customer\_dim.email    LEVEL active IS customer\_dim.active    LEVEL created\_date IS customer\_dim.created\_date    LEVEL last\_update IS customer\_dim.last\_update    LEVEL address\_id IS customer\_dim.address\_id    LEVEL address IS customer\_dim.address    LEVEL city\_id IS customer\_dim.city\_id    LEVEL city\_name IS customer\_dim.city\_name    LEVEL country\_id IS customer\_dim.country\_id    LEVEL country\_name IS customer\_dim.country\_name    LEVEL s\_key\_order IS customer\_dim.s\_key\_order    LEVEL expiry\_date IS customer\_dim.expiry\_date    HIERARCHY hierarchy\_customer (s\_customer\_key        CHILD OF address\_id CHILD OF city\_id CHILD OF country\_id)    ATTRIBUTE address\_id DETERMINES (customer\_dim.address)    ATTRIBUTE city\_id DETERMINES (customer\_dim.city\_name)    ATTRIBUTE country\_id DETERMINES (customer\_dim.country\_name); | CREATE DIMENSION film\_dim      LEVEL film\_id IS film\_dim.film\_id      LEVEL title IS film\_dim.title      LEVEL description IS film\_dim.description      LEVEL release\_year IS film\_dim.release\_year      LEVEL rental\_duration IS film\_dim.rental\_duration      LEVEL rental\_rate IS film\_dim.rental\_rate      LEVEL lentgth IS film\_dim.lentgth      LEVEL replacement\_cost IS film\_dim.replacement\_cost      LEVEL rating IS film\_dim.rating      LEVEL last\_update IS film\_dim.last\_update      LEVEL special\_features IS film\_dim.special\_features      LEVEL full\_text IS film\_dim.full\_text      LEVEL language\_id IS film\_dim.language\_id      LEVEL language\_name IS film\_dim.language\_name      HIERARCHY hierarchy\_film ( film\_id          CHILD OF language\_id      )      ATTRIBUTE language\_id DETERMINES (          film\_dim.language\_name  ); |
| **Time\_Dim** CREATE DIMENSION time\_dim      LEVEL time\_id IS time\_dim.time\_id      LEVEL day IS time\_dim.day      LEVEL month IS time\_dim.month      LEVEL quarter IS time\_dim.quarter      LEVEL year IS time\_dim.year      HIERARCHY hierarchy\_time\_all (day CHILD OF month CHILD OF quarter CHILD OF year)      HIERARCHY hierarchy\_time (day CHILD OF month CHILD OF year); | |

نقوم بالتحقق من وجود الابعاد بالتعليمة التالية:

## الطلب الثاني: **تجزئة جدول Fact بالطريقة التي تراها مناسبة وتعليل ذلك، هل من الممكن أن تكون هذه التجزئة مركبة وضح ذلك.**

نقوم بعمل Drop لجدول الـ Fact بحالتنا لأنه خالي من اية سجلات ثم نقوم ببنائه من جديد وبـ Partition معين لكن ان كان يحوي سجلات فيجب عمل تصدير لها ثم Drop للجدول وانشائه بالتقسيمة المرادة من جديد وعمل استيراد للبيانات وذلك لعدم القدرة على انشاء اقسام (Partitions) للجدول بعد انشائه (وذلك لجميع نسخ oracle التي دون 12c).

نستخدم PARTITION BY RANGE على جدول الـ Fact وذلك للحقل rental\_date كل سنة لحال ويمكن التقسيم كل شهر او او ..الخ ولكن سنعتمد على كل سنة بقسم لحال نظرا الى نوع البيانات لدينا فهي حجوزات أفلام فمن المنطق انه مهما بلغ عدد الحجوزات كبير لن تصل الى ارقام ضمن السنة الواحدة ستبطئ عمل الاستعلامات

**بالشكل التالي:**

CREATE TABLE rental\_fact (

  rental\_id      NUMBER NOT NULL,

  customer\_id    NUMBER NOT NULL,

  film\_id        NUMBER NOT NULL,

  rental\_date    DATE NOT NULL,

  amount         NUMBER NOT NULL,

  length\_of\_renat\_in\_days NUMBER NOT NULL,

  CONSTRAINT rental\_fact\_pk PRIMARY KEY ( rental\_id ),

  CONSTRAINT rental\_cutomer\_fact\_fk FOREIGN KEY ( customer\_id ) REFERENCES customer\_dim (s\_customer\_key ),

  CONSTRAINT rental\_film\_fact\_fk FOREIGN KEY (film\_id ) REFERENCES film\_dim (film\_id ),

  CONSTRAINT rental\_time\_fact\_fk FOREIGN KEY (rental\_date ) REFERENCES time\_dim (time\_id )

)

PARTITION BY RANGE (rental\_date)

(

    PARTITION rentals\_2020 VALUES LESS

THAN(TO\_DATE('01/01/2020','DD/MM/YYYY')),

    PARTITION rentals\_2021 VALUES LESS

THAN(TO\_DATE('01/01/2021','DD/MM/YYYY')),

    PARTITION rentals\_2022 VALUES LESS

THAN(TO\_DATE('01/01/2022','DD/MM/YYYY')),

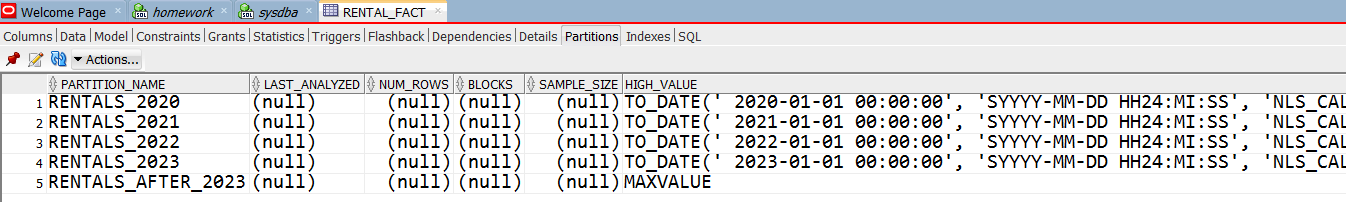
    PARTITION rentals\_2023 VALUES LESS

THAN(TO\_DATE('01/01/2023','DD/MM/YYYY')),

    PARTITION rentals\_after\_2023 VALUES LESS THAN(MAXVALUE)

);

يمكن عمل تجزئة مركبة بمثالنا عن طريق RANGE - HASH حيث من الممكن ان نرى ان تجزئة كل سنة قد تحوي الكثير من البيانات فيمكن إضافة hash وسيكون لحقل rental\_id ولكن التقسيمات للجدول ترى من حجم العمل والتوقعات للـ Business .

سنستعرض الأقسام للجدول التي تم انشائها:

## الطلب الثالث: **بناء VIEW MATERIALIZED يتضمن اسم الفيلم واسم الزبون وفترة الحجز، تتحدث بيانات هذا المنظور بشكل تراكمي.**

نقوم بانشاء MV LOG أولا لكل الجداول التي سنأخذ منها البيانات وهي customer\_dim, film\_dim and rental\_fact بالشكل التالي:

CREATE MATERIALIZED VIEW LOG ON customer\_dim WITH

    PRIMARY KEY,

    ROWID

    INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON film\_dim WITH

    PRIMARY KEY,

    ROWID

    INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON rental\_fact WITH

    PRIMARY KEY,

    ROWID

    INCLUDING NEW VALUES;

وسبب حاجتنا لهذه الـ LOG VIEW بسبب ماتم طلبه بالسؤال من التحديث بشكل تراكمي فكل فترة سنقوم بعمل refresh ليتم تحديث الـ MATERIALIZED VIEW التي سننشئها تالياً:

CREATE MATERIALIZED VIEW film\_rental\_view

    BUILD IMMEDIATE REFRESH FAST ON DEMAND

AS

    SELECT

        f.rowid film\_rid, c.rowid custoemr\_rid, r.rowid rental\_rid,

        f.title          film\_name,

        ( c.first\_name

          || ' '

          || c.last\_name ) customer\_name,

        r.length\_of\_renat\_in\_days

    FROM

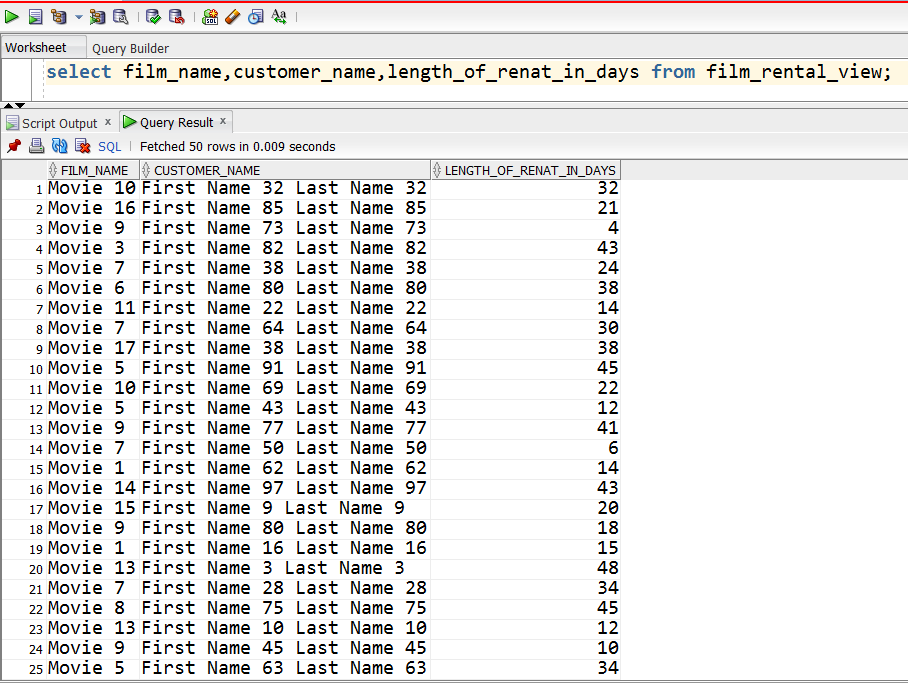
        film\_dim f, customer\_dim c, rental\_fact  r

    WHERE

            r.customer\_id = c.s\_customer\_key

        AND r.film\_id = f.film\_id;

**وضعنا الـ rowid لأنها اجبارية اذا كنا سنقوم بعمل MV من نوع REFRESH FAST والـ MV لا يحوي الا JOINs بدون اية توابع تجميعية**

نستعرض الـ MV بالشكل التالي:

## الطلب الرابع: **عرض قائمة بقيم حجوزات الأفلام على مستوى الشهر والبلد والمدينة بكافة الاحتمالات الممكنة.**

الحل

## الطلب الخامس: **أوجد ترتيب كل سنة وكل فيلم ضمن هذه السنة حسب قيمة التأجير.**

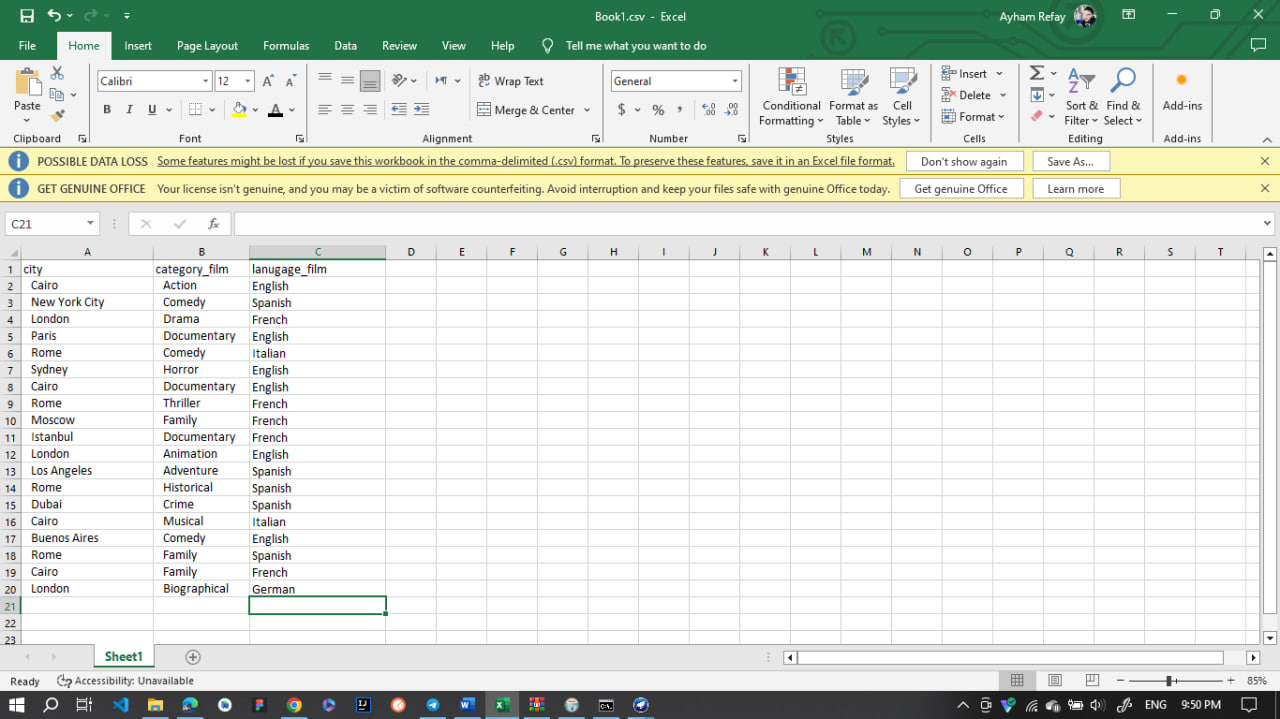
الحل

## الطلب السادس: **يريد مدير الشركة أن يعرف ما هي العالقة ما بين مدينة الزبون وفئة الأفلام ولغة الفيلم لكي يستخدم هذه المعلومة في الإعلانات الموجهة، اقترح طريقة مناسبة لمعرفة هذه العالقة ثم نفذها باستخدام weka.**

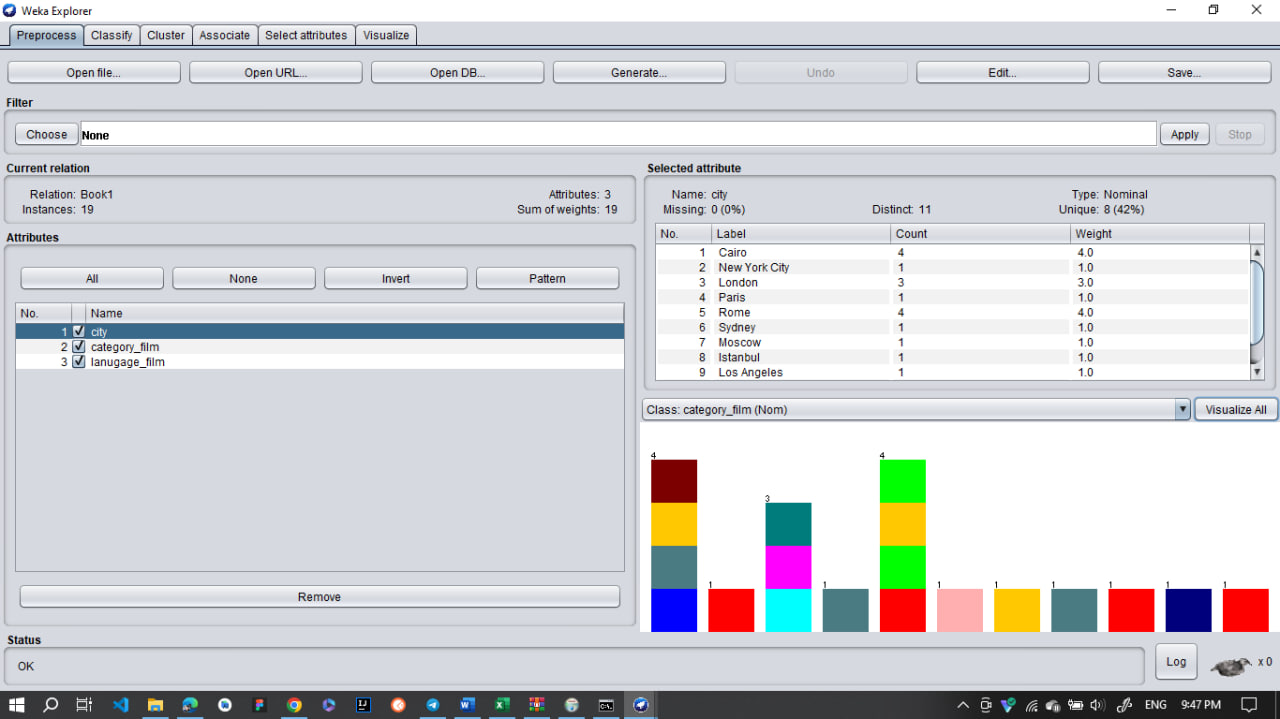
الحل

نقوم بتصدير البيانات لملف Excel بصيغة cvs وندخلها الى tool weka ثم نقوم بتحويل ملف cvs الى صيغة arff

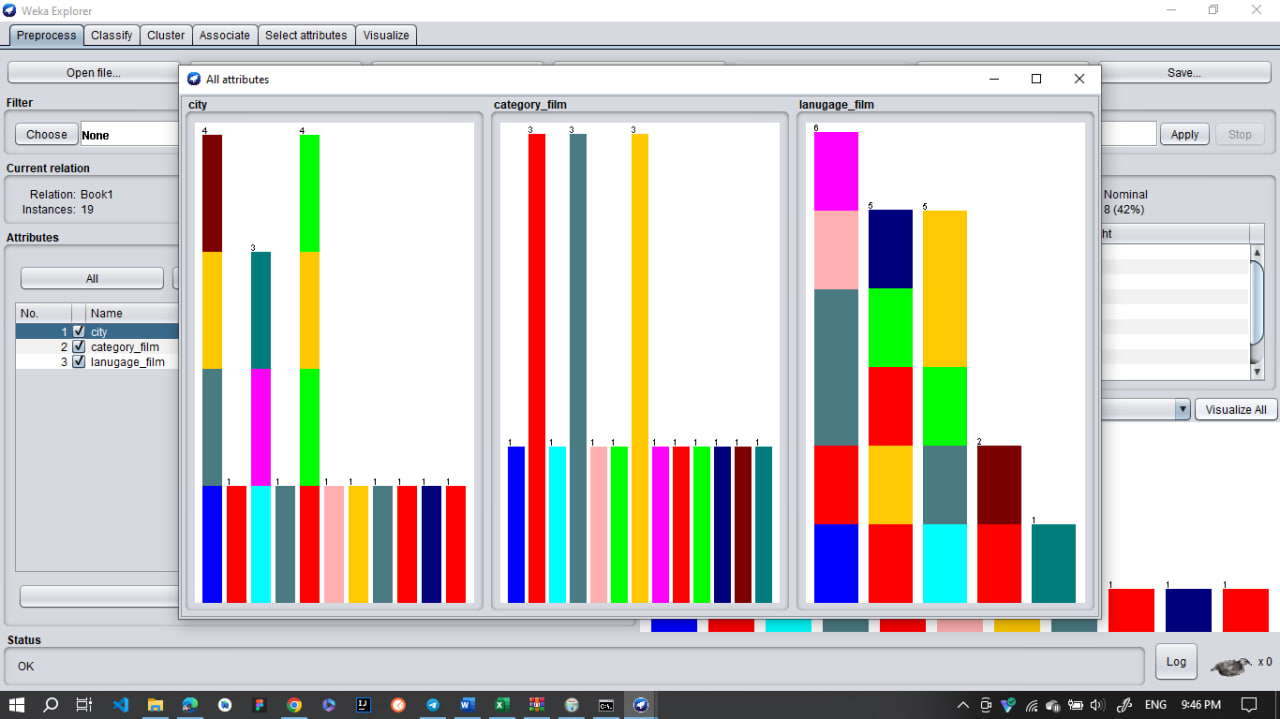
البيانات في ملف excel :



فيكون الخرج كالتالي:



وبتحليل جميع البيانات ينتج:



وبتنفيذ خوارزمية apriori على البيانات المدخلة تكون:

