|  |  |
| --- | --- |
|  | **2016** |
|  | جامعة دمشق  كلية الهندسة المعلوماتية  قسم هندسة البرمجيات ونظم المعلومات |

|  |
| --- |
| **[وظيفة قواعد المعطيات المتقدمة]** |
| إعداد الطلاب: آلاء الحموي محمد باسل الشمالي محمد غانم هانئة المالكي |

# 1. أوراكل:

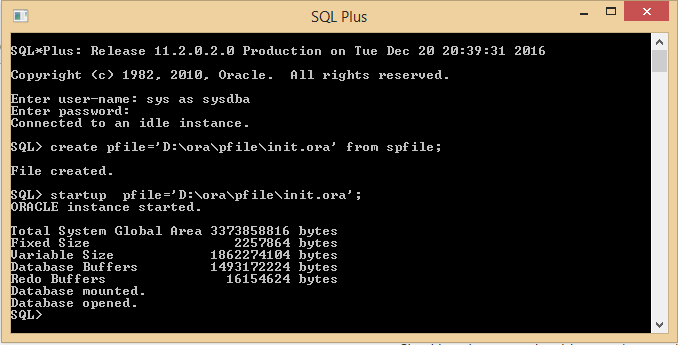
## الطلب الأول:

لإنشاء pfile نستخدم التعليمة التالية:

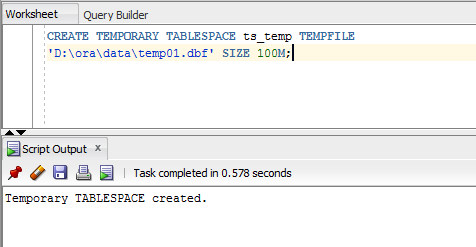
create pfile='D:\ora\pfile\init.ora' from spfile;

الآن نقلع قاعدة البيانات من الـ pfile الذي تم إنشاءه:

STARTUP pfile 'D:\ora\pfile\init.ora';



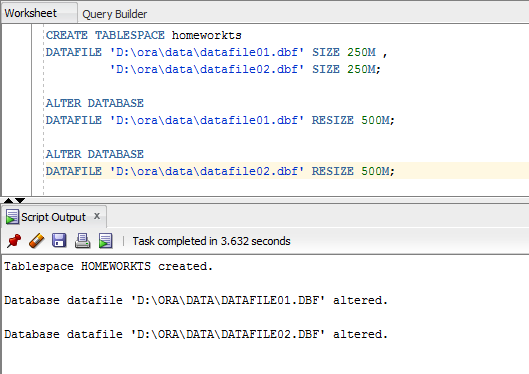
## الطلب الثاني:



## الطلب الثالث:

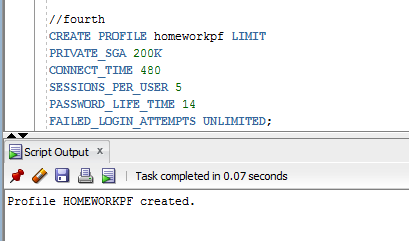
بدايةً، نقوم بإنشاء tablespace حجمه 500 Mega وذلك بجعله مكون من ملفين كل منهما حجمه 250 Mega

بعد ذلك نقوم بزيادة حجمه وذلك بزيادة حجم الملفات التي تكونه

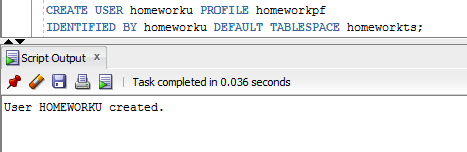


## الطلب الرابع:

إنشاء profile وجعل الذاكرة200 KB ومدة الاتصال 8 ساعات أي480 دقيقة وعدد الجلسات التي يستطيع نفس المستخدم فتحها 5 وعدد الأسابيع التي يجب بعد مرورها تغيير كلمة السر 2 أي 14 يوم وعدد المرات التي يستطيع إعادة محاولة تسجيل الدخول بعد فشلها غير محدود.

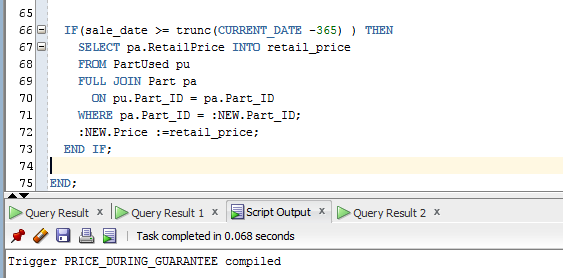
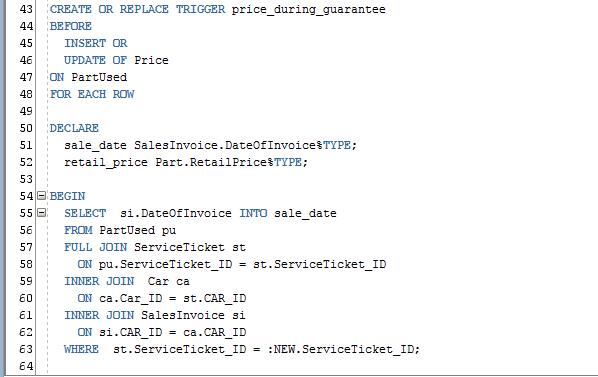


## الطلب الخامس:

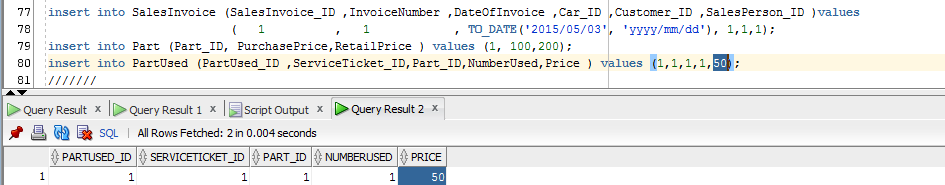


# 2.قواعد المعطيات الفعالة و PL\_SQL:

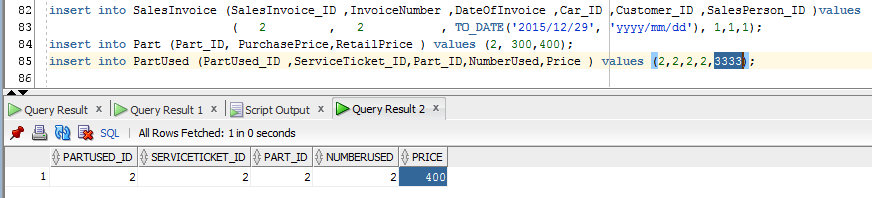
## الطلب الأول:



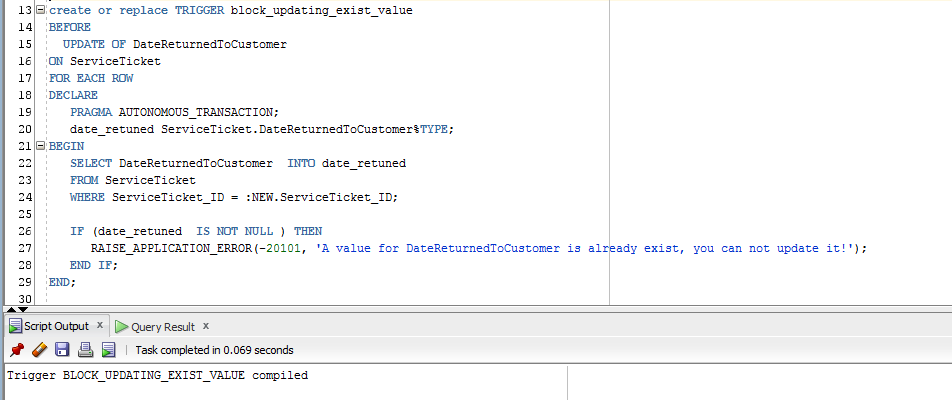
الآن نلاحظ عند إدراج PartUsed تابع لسيارة خارج مدة الكفالة، ييتم تخزين في قاعدة البيانات القيمة المدخلة:



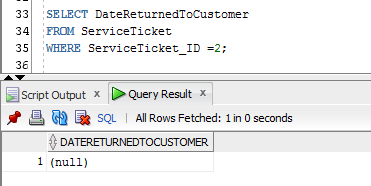
أما عند إدراج PartUsed تابع لسيارة ضمن مدة الكفالة، يتم تخزين قيمة الـ RetailPrice في حقل الـ Price في قاعدة البيانات بدل من القيمة المدخلة:



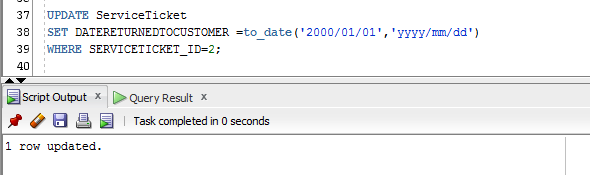
## الطلب الثاني:



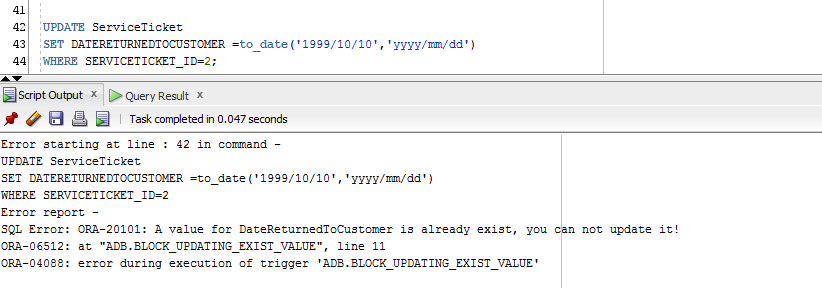
في حال كان لدينا ريكورد لا يملك قيمة:



عند محاولة تعديل هذه القيمة تنجح العملية:

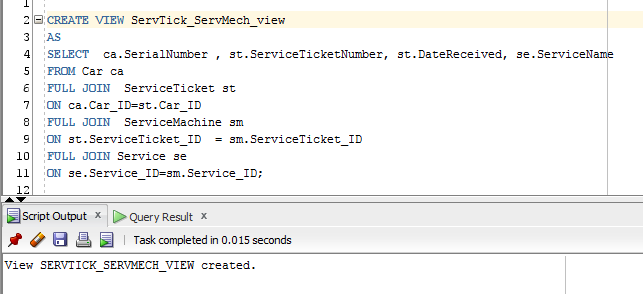


بعد ذلك نحاول تعديل نفس القيمة بعد امتلاكها قيمة سابقة فـ **لا** تتم عملية التعديل بنجاح ويظهر خطأ يمنع إتمام العملية:

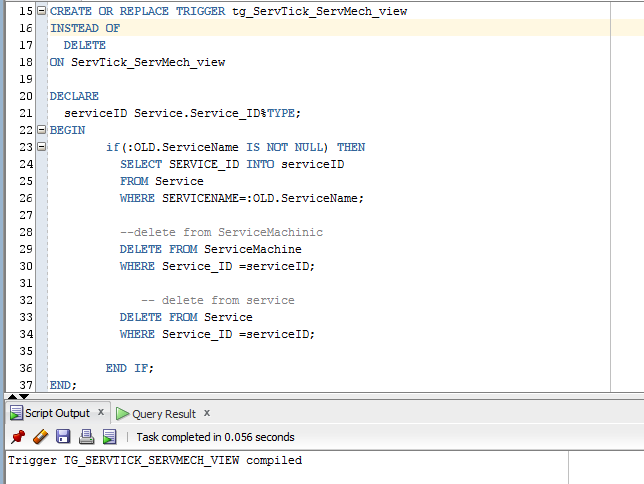


## الطلب الثالث:

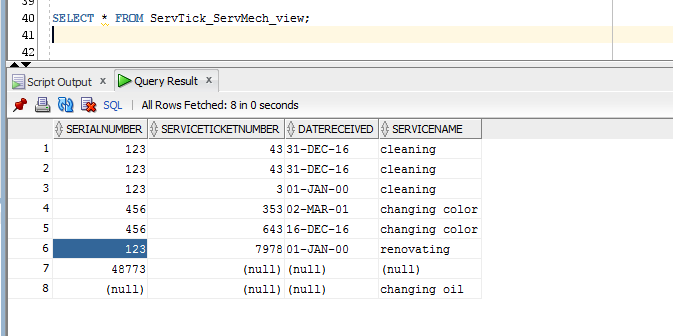
نقوم ببناء view على ثلاثة جداول: Car للتمكن من الوصول إلى رقم السيارة، ServiceTicket للتمكن من الوصول إلى رقم تذكرة الصيانة والتاريخ، ServiceMachine كسر يشكل جسر وصول للخدمة عن طريق أخد ID الخدمة منه، Service للتمكن من الوصول إلى اسم الخدمة.



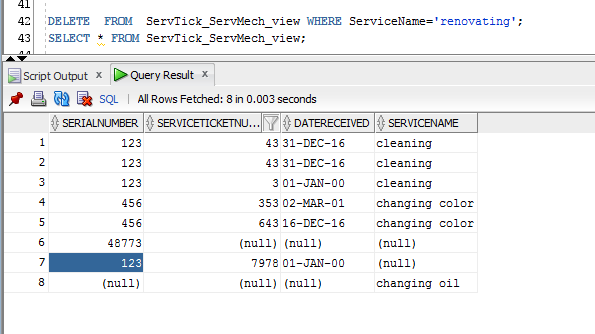
الآن نقوم ببناء قادح من نمط instead of لجعل المنظور editable، عند القيام بأي عملية حذف على المنظور يتم قدح هذا الترغر، وحذف الخدمة "في حال وجودها" وكل ServiceMachanic متعلقة بها.



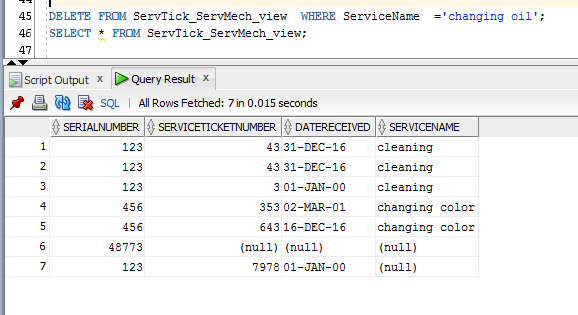
بالوصول إلى المنظور الذي تم بناءه نلاحظ وجود بعض السيارات التي لم تتلق أي خدمة بعد، كما نلاحظ وجود خدمات لم تقدم لأي سيارة بعد:



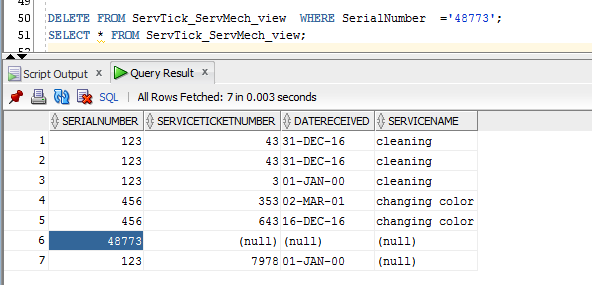
نلاحظ أنه عند حذف أي سجل يتم حذف الخدمة ولا يتم حذف السيارة أو الحقول المتعلقة بتذكرة الصيانة.



وعند حذف خدمة لم تقدّم لأي سيارة بعد، يتم حذف كامل السجل من المنظور

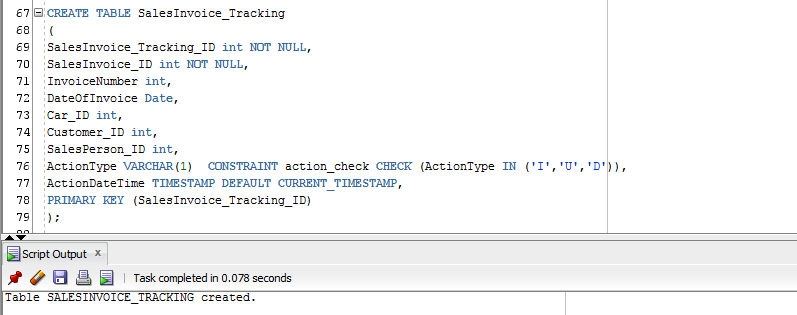


أما عند حذف سجل لا يحوي خدمة من المنظور، لا يحدث شيء

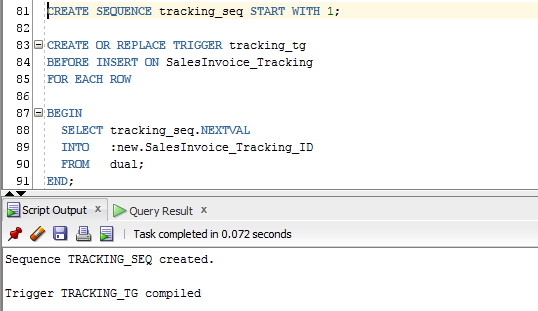


## الطلب الرابع:

يتم بناء جدول SalesInvoice\_Tracking تحفظ فيه جميع القديمة الموجودة في جدول SalesInvoice قبل تعديلها، مع ذكر تاريخ وزمن التعديل:

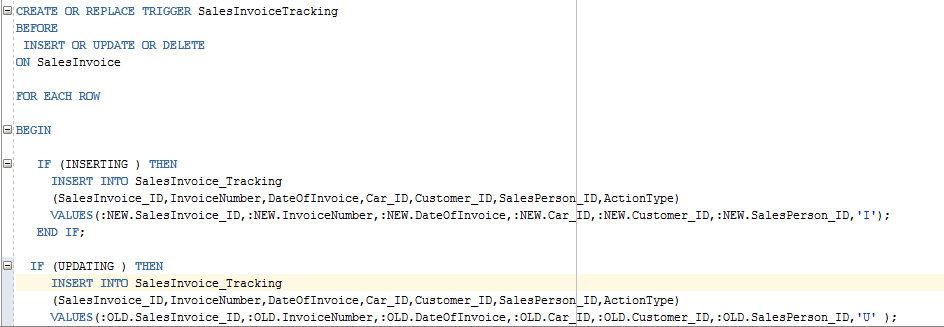


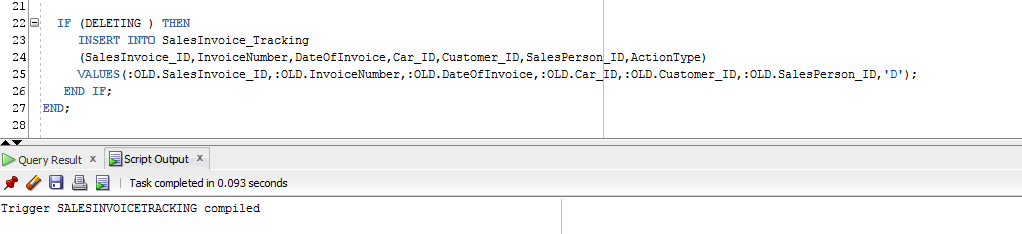
وبما أن النسخة 11 من أوراكل لا تدعم الـ IDENTITY يتم استخدام SEQUENCE للحصول على ID متسلسل أوتوماتيكي وبناء قادح يقدح قبل أي عملية إدخال تتم على الجدول



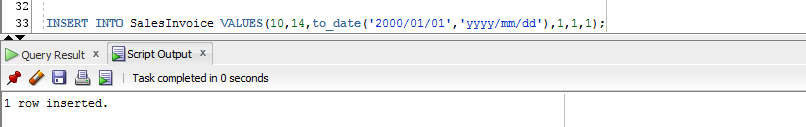
ملاحظة: تم إضافة ميزة الـ IDENTITY في النسخة 12

بعد ذلك يتم بناء قادح على جدول فاتورة البيع SalesInvoice، ويُقدح قبل البدء بأي عملية DML، إذ يقوم بتخزين القيم القديمة في الجدول SalesInvoice\_Tracking مع إضافة تاريخ وزمن التعديل:

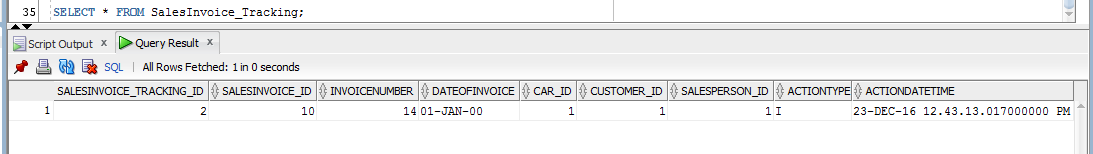


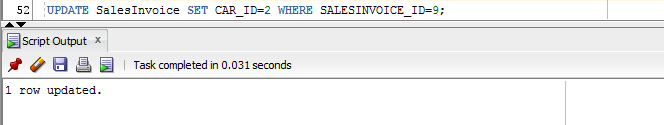


الآن عند القيام بأي عملية إضافة – تعديل – حذف على جدول الفواتير ينعكس ذلك على جدول التتبع:

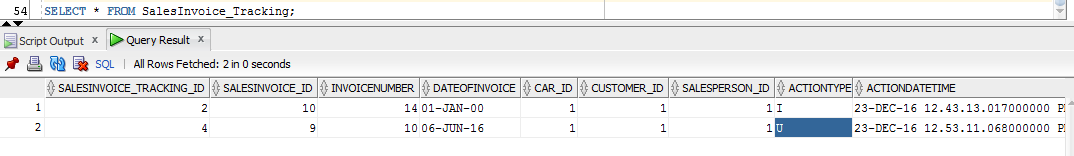


نلاحظ أن القيمة انعكست على جدول المتابعة:

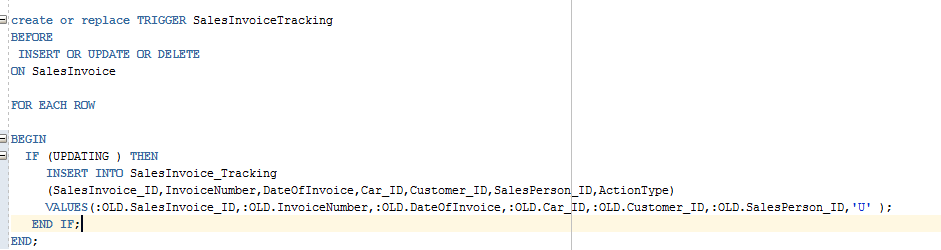


وكذلك الأمر بالنسبة لعملية الـ Update

نلاحظ أن القيمة قبل التعديل قد تم ادراجها في سجل المتابعة:

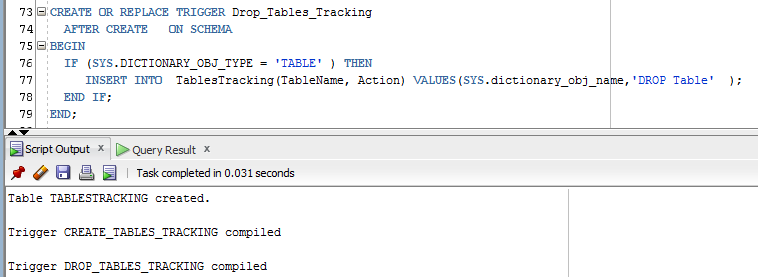
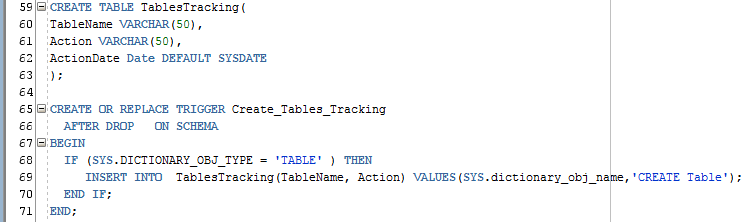


ملاحظة: يمكن جعل المتابعة تتم فقط لعملية التعديل Update فلا يسجل في سجل المتابعة إلا عمليات التعديل، وبذلك يصبح شكل القادح:

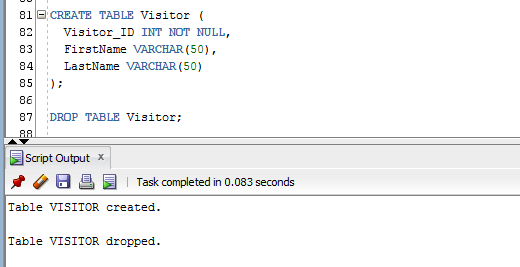


## الطلب الخامس:

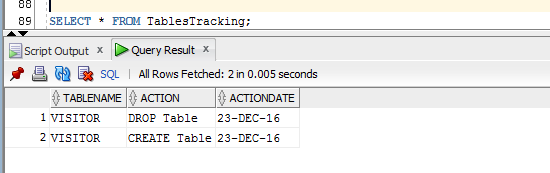
نقوم ببناء الجدول TablesTracking الذي سيضم معلومات عن الجداول التي يتم إنشاءها وحذفها، ثم قوادح من نمط DDL تنقدح في حال القيام بعملية إنشاء أو حذف جدول وتخزن العملية في الجدول TablesTracking



نقوم بعملية إضافة وحذف لنفس الجدول

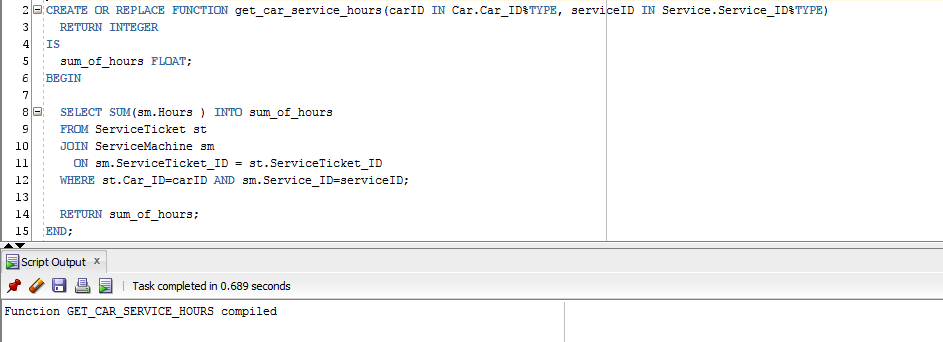


نلاحظ أن نتيجة الاضافة والحدث قد تم الاحتفاظ بها

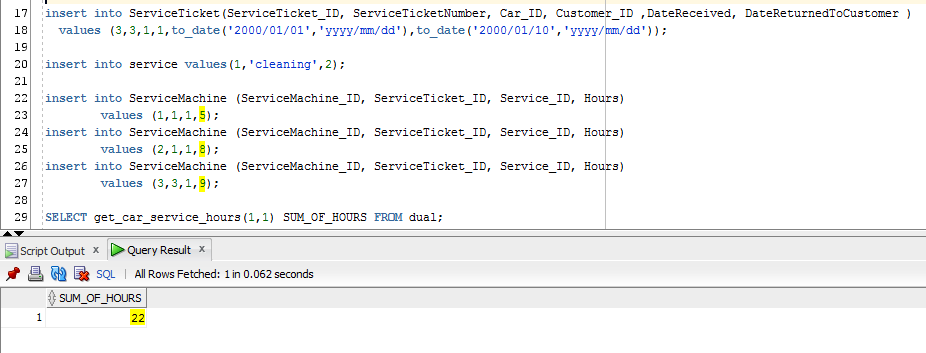


## الطلب السادس:

نقوم بخلق تابع يستقبل رقم الخدمة والسيارة ويعيد عدد الساعات المقضية على هذه الخدمة للسيارة المطلوبة



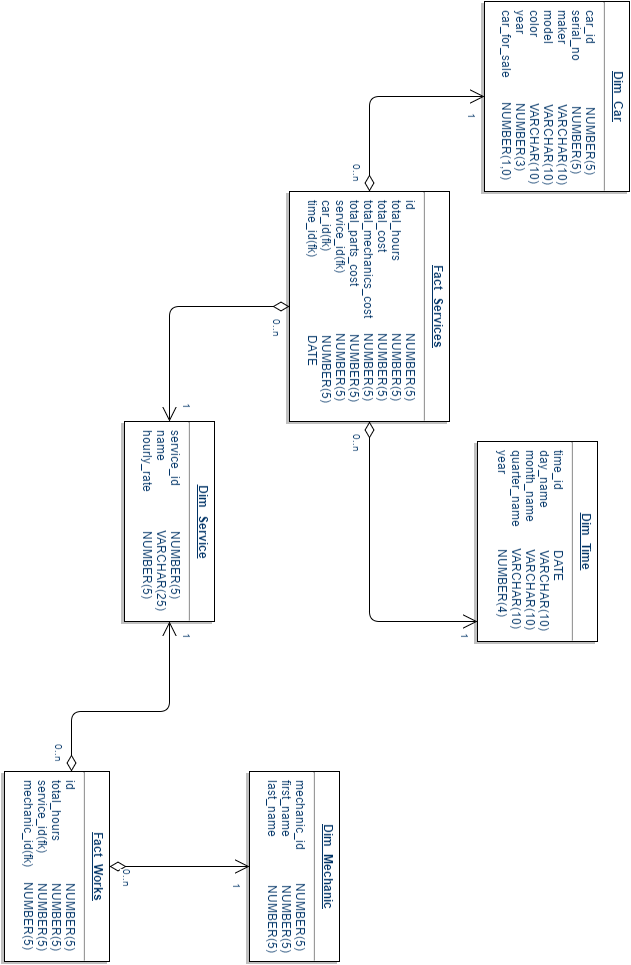
بإدخال ثلاث ريكودرات لخدمة التنظيف للسيارة 1 تشغل كل منها الأوقات 5, 8, 9 نجد أن الزمن الكلي المقضي على هذه الخدمة بالنسبة لهذه السيارة هو 22



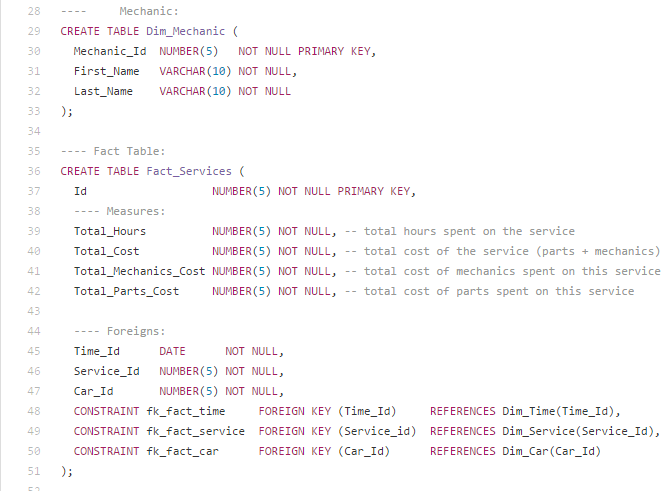
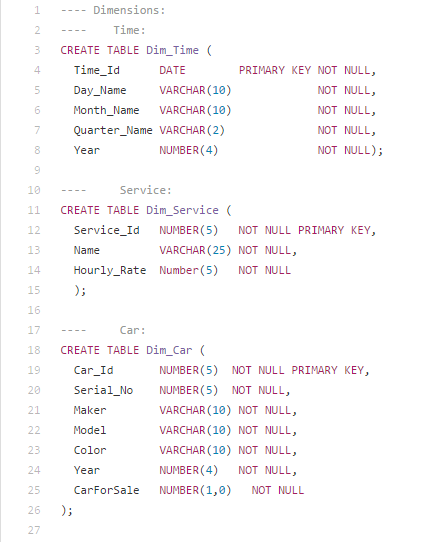
# 3.قواعد المعطيات متعددة الأبعاد:

## الطلب الأول:

قمنا باقتراح بنيتي Star Schema التاليتين بناءً على المتطلبات المستخرجة من أسئلة هذا القسم حيث وجدنا أنه يوجد لدينا عمليتين (Two Business Processes) يجب أن يتم نمذجتهما كــ Fact Tables وهما حقيقة التخديم Services Fact التي تقوم بتقديمها ورشة التصليح التي نقوم بنمذجتها وأيضاً حقيقة الخدمات التي قد تم إنجازها من قبل الميكانيكيين Works Fact وعليه يكون لدينا المخطط التالي:

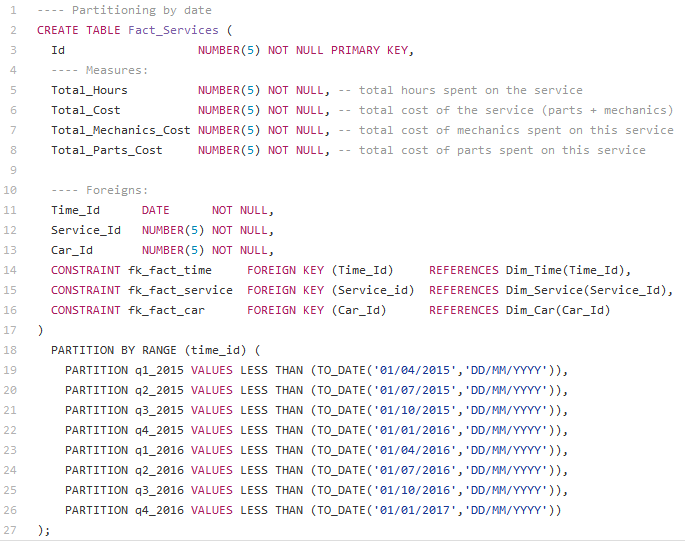


وفيما يلي تعليمات بناء الجداول:



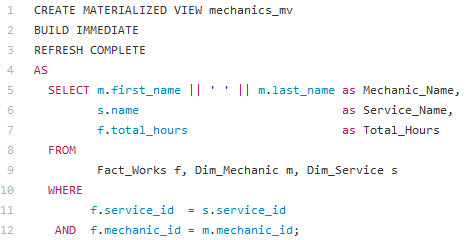
## الطلب الثاني:

# قمنا بتجزئة جدول الحقيقة باستخدام Range على حقل التاريخ، فالطريقة المثالية لتجزئة Historiacl Data هو باستخدام Range:



## الطلب الثالث:

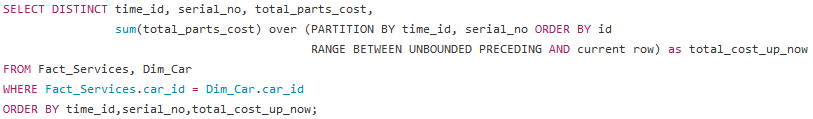
# مع امتلاك المستخدم لصلاحية إنشاء Materialized View يمكننا إنشاء الView المطلوب من خلال التعليمة التالية:



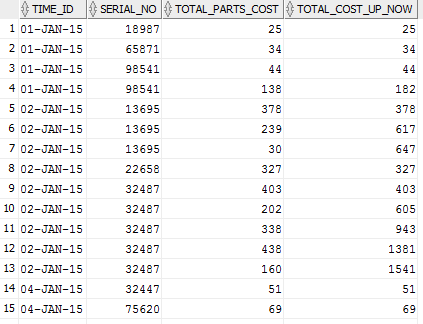
# ولم نحتج لجداول سجلات Log files للـMaster Tables التي قمنا بإنشاء الـMaterialized View منها, لأن نوع التحديث فيه هو Complete.

## الطلب الرابع:

# قمنا بتحقيق المطلوب من خلال استخدام التابع sum() مع over() لتحقيق Sliding Window على كل من time\_id, serial\_no من خلال PARTITION BY وقمنا بجعل النطاق مفتوح البداية حتى السطر الحالي:

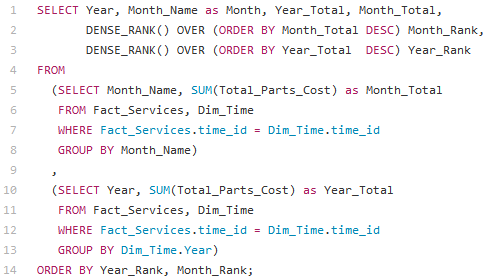


**التنفيذ:**

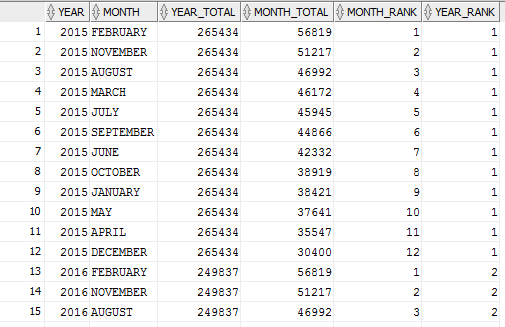


## الطلب الخامس:

# قمنا باستخدام التابع التحليلي DENSE\_RANK() لتحقيق المطلوب, وقد فضلنا DENSE\_RANK() عن RANK() لجعل المرتبة واحدة لكل شهر أو سنة (أي عدم جعل أكثر من شهر أو سنة تحمل نفس المرتية عند تساوي القيم):



**التنفيذ:**



## الطلب السادس:

# لمعرفة الخدمات المتصلة ببعضها، يجب تحديد الخدمات التي تقدم خلال زمنية معينة إلى سيارة معينة وهي مسألة مشابهة لمسألة Market Basket حيث أن الخدمات المقدمة للسيارة تشبه السلع المشتراة، فعلينا ايجاد الخدمات التي تُقدم للسيارة مع بعضها البعض.

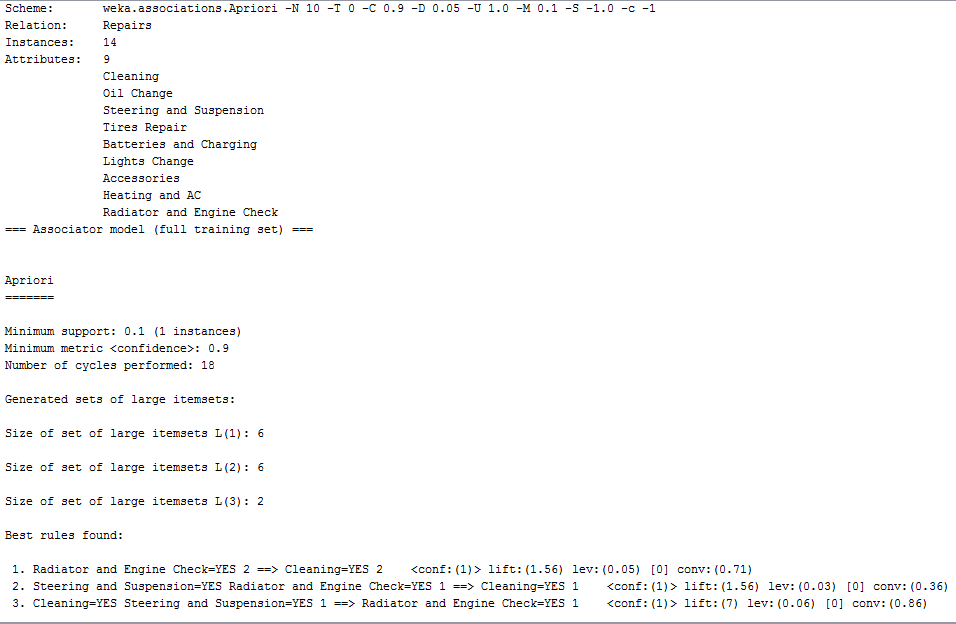
# وبالتالي تكون ال itemset هي مجموعة الخدمات الموجودة في وكالة السيارات، والخورازمية التي سوف نطبقها هي Apriori Association Rule وهي تطبق على داتا Transctional من الشكل التالي:

|  |  |
| --- | --- |
| Cleaning & Oil Change & .. | Car 1 |
| Accessories & .. | Car 2 |
| …. | Car 3 |

يمكن الحصول على الداتا السابقة من خلال استخدام جدول الفاكت الأول، وتكون بنية الملف ARFF كما يلي:

|  |
| --- |
| @relation Repairs  @attribute 'Cleaning' {YES}  @attribute 'Oil Change' {YES}  @attribute 'Steering and Suspension' {YES}  @attribute 'Tires Repair' {YES}  @attribute 'Batteries and Charging' {YES}  @attribute 'Lights Change' {YES}  @attribute 'Accessories' {YES}  @attribute 'Heating and AC' {YES}  @attribute 'Radiator and Engine Check' {YES}  @data  YES,YES,?,?,?,?,YES,?,?  ….. |

فكل سطر يعبر عن سيارة قدمت لها مجموعة من الخدمات خلال يوم معين، فمن أجل كل خدمة في الوكالة يوجد حقل اذا كانت قيمة هذا الحقل YES فالخدمة تم تقدميها للسيارة وفي حال كانت القيمة ? فالخدمة لم يتم تقديمها.

وبعد تطبيق الخوارزمية على داتا معينة نجد ما يلي: