إذا كانت لك خبرة كافية بالتعامل مع Access وإنشاء استعلاماته وتعرف العلاقة الوطيدة بين الاستعلامات وكود SQL، فهذا لا يكفيك عند الانتقال للعمل مع مشروعات البيانات وقواعد بيانات SQL Server، حيث تختلف لهجة SQL الموجودة داخل SQL Server والمعروفة باسم Transact-SQL أو T-SQL عن لغة SQL الموجودة داخل Access فى بعض السمات، وهو ما سنتعرف عليه معاً فى هذا الفصل.

بانتهاء هذا الفصل ستتعرف على:

- استخدام SQL داخل Access 2007
- القواعد الأساسية للغة SQL
- كتابة استعلامات التحديد باستخدام SQL
- استخدام رموز وعلامات SQL
- ترجمة عبارات SQL داخل نافذة تصميم الاستعلام

استخدام SQL داخل Access 2007

يتناول هذا الفصل ما يعرف بـ Structured Query Language "لغة الاستعلامات التركيبية" وتختصر هكذا SQL، والقواعد المختلفة المستخدمة فى كتابة هذه اللغة ثم نتعرف بعد ذلك على لغة SQL المستخدمة داخل قواعد بيانات SQL Server والمعروفة باسم Transact-SQL وتختصر T-SQL. وقد تعرفنا فى الفصول السابقة من الكتاب على كيفية ترجمة Access للاستعلامات المختلفة وتحويلها إلى عبارات SQL من خلال نافذة تصميم الاستعلام. يمكن اعتبار كلٍ من SQL التى تستخدمها Access فى الاستعلامات والتقارير وT-SQL المستخدمة داخل قواعد بيانات SQL Server لهجات للغة أم هى SQL. وهناك بالطبع تشابهاً كبيراً بين لهجة SQL المستخدمة داخل Access ولهجة T-SQL المستخدمة داخل SQL Server، إلا أن الأولى لايمكنها تدعيم العروض Views والإجراءات المخزنة Stored Procedures والخوادم المرتبطة، وذلك لعدم وجود هذه العناصر داخل Access ووجودها كعناصر أساسية داخل SQL Server. وعلى الجانب الآخر، فإن SQL تدعم التعامل مع دوال لغة VBA الموجودة داخل Access كالدالة CCur() والدالة DatePart() على سبيل المثال. وتحتوى لغة T-SQL حقيقةً على ما يكافئ العديد من دوال VBA إلا أن الصيغة المستخدمة تكون مختلفة بالطبع، وهو ما سنحاول التعرف عليه فى هذا الفصل.

ولغة SQL عبارة عن لغة مشتركة لقواعد بيانات الخادم والعميل، والتى تتميز بأنها لغة قياسية يمكن استخدام عباراتها المختلفة مع جميع أنظمة إدارة قواعد البيانات المتوافقة مع SQL، فهى لغة تطبيقية لقواعد البيانات العلائقية، ولا تعد لغة برمجة، فهى لا تحتوى على عبارات التفرع التى تتحكم فى اتجاه تنفيذ البرنامج، كما أنك لا تستطيع من خلالها إنشاء النماذج التى تستقبل البيانات من المستخدمين مثلاً.

وعلى عكس التوصيات القياسية الخاصة بلغة HTML ولغة XML مثلاً والتى تتغير من آن لآخر من خلال منظمة W3C، فإن إصدارات SQL قليلة وتتغير على فترات متباعدة.

فكانت البداية ب SQL-86 ثم SQL-92 ثم SQL-1999 وأخيراً SQL-2003. ويبنى Access SQL على الإصدار SQL-92 كما يدعم بعض السمات الموجودة فى SQL1999.

وتفيد خلفيتك عن SQL فى فهمك لعملية الاستعلام عن البيانات بشكل كبير، كما يمكنك استخدامها فى تصميم استعلامات عالية الجودة سواءً باستخدام SQL أو باستخدام T-SQL. فبقليل من المعرفة بلغة SQL يمكنك كتابة الاستعلامات الفرعية واستعلامات التوحيد وإنشاء مجموعات السجلات المستخدمة فى تعبئة القوائم ومربعات السرد من خلال لغة VBA. وقد تعرفنا فى الفصول السابقة على أمثلة كثيرة لاستخدام SQL توضح ما يحدث من وراء الستار عند إنشاء استعلام من خلال نافذة تصميم الاستعلام QBE وما بها من أدوات مختلفة، حيث يمكنك استخدام نافذة تصميم الاستعلامات فى الحصول على كود SQL الخاص بأكثر من ٩٠% من الاستعلامات التى تحتاج إليها داخل تطبيقات Access (حيث يجب كتابة استعلامات التوحيد Union Queries والاستعلامات المتداخلة Pass-through Queries من خلال نافذة SQL نفسها).أما فى حالة مشروعات البيانات، فيمكنك الحصول على كود T-SQL الخاص بأكثر من ٧٥% من الاستعلامات التى تحتاج إليها داخل مشروعات البيانات ADP من خلال مصمم المشروعات الرسومى Graphical Project Designer، بينما يجب عليك كتابة بقية الاستعلامات بنفسك. وعلى عكس SQL، تحتوى لغة T-SQL على العديد من العبارات المستخدمة فى التحكم فى تنفيذ الكود كعبارة IF ... ELSE وعبارة WHILE على سبيل المثال لا الحصر. فإذا ما أردت الحصول على معظم السمات الموجودة باستعلامات SQL Server 2005، يجب فى هذه الحالة تعديل عبارات T-SQL من خلال مصمم المشروعات.

يمكنك تعلم عبارات SQL بسهولة شديدة من خلال أمثلة عملية. كلما قمت بإنشاء استعلام من خلال نافذة تصميم الاستعلام أو نافذة مصمم المشروعات، انتقل إلى عرض SQL وشاهد عبارات SQL أو T-SQL المكافئة للاستعلام الذى قمت بإنشائه.

القواعد الأساسية للغة SQL

عند البدء فى تعلم القواعد الأساسية لإحدى اللغات الجديدة، من الأفضل تقسيم هذه القواعد إلى مجموعات مختلفة تبعاً للاستخدام ثم مجموعات أخرى تبعاً للتحدث. ويمكن تقسيم أوامر لغة SQL تبعاً للاستخدام بدايةً إلى ست مجموعات أساسية وهى:

- أوامر الاستعلام عن البيانات Data Query Language (DQL) وتحتوى على الأوامر المستخدمة فى الحصول على البيانات من الجداول وتحديد كيفية عرض هذه البيانات، ويأتى على رأس هذه المجموعة الأمر SELECT.
- أوامر معالجة البيانات Data Manipulating Language (DML) وتحتوى على الأوامر المستخدمة فى معالجة البيانات Data Manipulation Language (DML) كالأمر INSERT المستخدم فى إضافة صفوف جديدة إلى الجداول والأمر DELETE المستخدم فى حذف صفوف من الجداول والأمر UPDATE المستخدم فى تعديل قيم أعمدة معينة بالصفوف الموجودة بالجداول.
- أوامر معالجة الإجراءات Transaction Processing Language (TPL) ومن أمثلتها الأوامر BEGIN TRAN وCOMMIT وROLLBACK المستخدمة فى تنفيذ عدد من أوامر المجموعة السابقة DML فى خطوة واحدة. فإذا فشل تنفيذ أحد الأوامر الموجودة بالمجموعة، يتم على الفور التراجع عن بقية الأوامر، أى يتم تنفيذ مجموعة الأوامر كحزمة واحدة.
- أوامر تعريف البيانات Data Definition Language (DDL) ومن أمثلتها الأوامر CREATE | ALTER TABLE وADD | ALTER COLUMN

وCREATE VIEW المستخدمة فى إنشاء الجداول والعروض. كما تستخدم أوامر DDL أيضاً لتعديل الجداول وإنشاء/حذف الفهارس.

- أوامر التحكم فى مكان المؤشر Cursor Control Language (CCL) وتشمل الأوامر المستخدمة فى اختيار صف معين داخل مجموعة السجلات الناتجة من أحد الاستعلامات مثل الأمر UPDATE WHERE CURRENT.
- أوامر التحكم فى البيانات Data Control Language (DCL) أو أوامر تأمين البيانات Data Security Language وتقوم هذه المجموعة بأداء الوظائف الإدارية التى تحدد أولويات وأذونات استخدام قاعدة البيانات كالأمر GRANT والأمر REVOKE، أو استخدام جداول معينة بقاعدة البيانات أو أوامر SQL معينة.

هذا من حيث الاستخدام، أما من حيث المصطلحات، فيمكن تقسيم مصطلحات SQL إلى المجموعات التالية:

- الأوامر Commands مثل الأوامر SELECT وEXECUTE وCREATE وALTER وتمثل الأفعال المستخدمة فى تنفيذ الأحداث المختلفة.
- المحددات Qualifiers مثل كلمة WHERE المستخدمة فى تحديد أو تقييد البيانات الناتجة من الاستعلام.
- العبارات Clauses مثل عبارة ORDER BY وتستخدم فى تعديل ترتيب البيانات الناتجة.
- المسندات Predicates مثل الكلمات IN وALL وANY وSOME وLIKE وUNIQUE المستخدمة فى اختبار حقائق معينة بقيم البيانات الناتجة من الاستعلام وإرجاع القيمة TRUE "صح" أو FALSE "خطأ" (وأحياناً القيمة NULL).
- المعاملات Operators ويطلق عليها أيضاً مسندات المقارنة مثل علامات = و> و< المستخدمة فى مقارنة القيم وتستخدم مع العبارات التى تحتوى على WHERE وJOIN.

- دوال التجميع Group aggregate functions مثل الدوال AVG() وCOUNT() وMAX() وMIN() وSUM() والتى تقوم بإرجاع قيمة واحدة من مجموعة من القيم.
- دوال التحويل بين أنواع البيانات Data type conversions functions والتى تقوم بتحويل القيم من نوع بيانات معين إلى نوع بيانات آخر، كالدالة CAST() والدالة CONVERT() على سبيل المثال.
- الدوال المساعدة Utility functions وتقوم بإرجاع القيم التى يتم تحديدها من خلال تعبيرات. ويمكنك على سبيل المثال استخدام الدالة NULLIF() لإرجاع القيمة NULL إذا كانت القيمة الناتجة من الدالة TRUE. ومن أمثلة دوال هذه المجموعة دوال معالجة النصوص ودوال التاريخ والوقت.
- الكلمات المحجوزة Reserved words ويتم من خلالها تعديل أحداث العبارات أو معالجة المؤشرات المستخدمة فى التعامل مع الصفوف الناتجة من الاستعلامات، ومن أمثلتها كلمة FOR XML المستخدمة داخل T-SQL والتى تقوم بإرجاع مستند XML بدلاً من مجموعة السجلات الناتجة من الاستعلام.

يتم عادةً كتابة الكلمات المحجوزة داخل SQL بحروف كبيرة، إلا أنها غير حساسة لحالة الأحرف، أى يتم تنفيذها بشكل سليم حتى لو قمت بكتابتها بحروف صغيرة.

كتابة استعلامات التحديد باستخدام SQL

تعتبر عبارة SELECT أهم عبارات SQL على الإطلاق، حيث تستخدم هذه العبارة فى الحصول على مجموعة معينة من السجلات من جدول أو أكثر وتحتوى على الصيغة العامة التالية:

```
SELECT [ALL|DISTINCT|DISTINCTROW] [TOP (n) [PERCENT]]
        [WITH TIES] select_list
        FROM table_names
        [WHERE search_criteria]
        [ORDER BY column_criteria [ASC|DESC]]
```

حيث:

- تمثل كلمة **SELECT** الأمر الأساسى المستخدم فى الاستعلام.
- يتم استبدال **select_list** بقائمة الأعمدة (الحقول) التى ترغب فى استرجاع بياناتها من خلال الاستعلام. وعند استخدام نافذة تصميم الاستعلام فى تعيين الاستعلام، يتم تحديد هذه الأعمدة عن طريق إدراجها من الجداول الموجودة بالجزء العلوى إلى الصف **Fields** بشبكة **QBE** الموجودة بالجزء السفلى بشرط تنشيط مربع الاختيار الموجود بالصف **Show** مع الأعمدة (الحقول) التى ترغب فى تضمينها بالقائمة **select_list**، حيث يتم فصل هذه الحقول باستخدام علامة الفاصلة (,).
- يدل القوسان [ ] فى الجزء **[ALL|DISTINCT|DISTINCTROW]** على أن ما بينهما اختيارى، كما يدل الحرف | على أنك تستطيع استخدام عنصر واحد فقط من العناصر الثلاثة. فى حالة اختيار **ALL** يتم تضمين جميع الصفوف (السجلات) الناتجة من الاستعلام. وفى حالة اختيار **DISTINCT**، يتم حذف الصفوف التى تحتوى على بيانات مكررة. أما فى حالة اختيار **DISTICTROW**، فيتم حذف الصفوف التى تحتوى على بيانات مكررة كما فى حالة اختيار **DISTINCT** مع القدرة على تعديل القيم الناتجة من الاستعلام، وهذا الخيار موجود فى **Access SQL** فقط ولا يوجد فى **T-SQL**.
- يستخدم الجزء الاختيارى **[[PERCENT] (n) TOP]** فى حالة الرغبة فى استرجاع عدد معين أو نسبة معينة من السجلات المفترض الحصول عليها من الاستعلام، حيث يتم استبدال **n** بالعدد المطلوب أو النسبة المطلوبة مع اختيار **PERCENT** فى الحالة الأخيرة. فعلى سبيل المثال، لاختيار أول ١٠٠ صف،

استخدم التعبير (100) TOP، ولاختيار نصف السجلات الناتجة، استخدم التعبير TOP (50) PERCENT. ويمكنك داخل SQL Server 2005 استخدام متغير بدلاً من n.

- يستخدم الجزء الاختيارى [WITH TIES] داخل T-SQL فقط مع العبارة ORDER BY ويتسبب فى إرجاع المزيد من الصفوف الناتجة عن القيم المتساوية داخل الحقل المستخدم بعبارة ORDER BY.
- يستخدم الجزء FROM table_name فى تحديد الجدول أو الجداول مصدر السجلات الناتجة. وعند استخدام نافذة تصميم الاستعلام فى تعيين الاستعلام، يتم تحديد هذه الجداول من خلال المربع الحوارى Add Table. وفى حالة تضمين حقول من أكثر من جدول بالجزء select_list، يجب فى هذه الحالة تضمين هذه الجداول بالجزء table_name، حيث يتم فصل هذه الجداول باستخدام علامة الفاصلة (,). فإذا كان هناك أكثر من حقل بنفس الاسم فى أكثر من جدول، يجب أن يكون كل حقل داخل الجزء select_list مصحوباً باسم الجدول المناسب.
- يحدد الجزء الاختيارى WHERE search_criteria السجلات التى سيتم عرضها من القائمة الناتجة عن طريق استبدال search_criteria بإحدى علامات المقارنة مثل LIKE أو = فى حالة الحقول النصية، > و< و=> في حالة الحقول أو المعاملات الرقمية . فإذا لم تقم بتضمين هذا الجزء داخل عبارة SELECT، يتم إرجاع جميع السجلات الموجودة بالجدول المحدد بالجزء FROM table_name.
- يستخدم الجزء الاختيارى [ORDER BY column_criteria [[ASC|DESC] فى تحديد ترتيب (فرز) السجلات الناتجة تبعاً للقيم الموجودة داخل عمود (حقل) أو أكثر من الأعمدة الموجودة بالاستعلام، كما يمكنك استخدام كلمة ASC للحصول على الفرز التصاعدى أو كلمة DESC للحصول على الفرز التنازلى، والأولى هى القيمة الافتراضية.

عند إضافة الحقول من جدولين أو أكثر ولم تقم بربط الجداول من خلال العبـــارة WHERE Table1.Field1 = Table2.Field2 أو من خلال العبارة JOIN، يتم إرجاع جميع السجلات الموجودة بكل جدول من هذه الجداول وهو ما يعمل على تكدس الشبكة واختناقها إذا تم تنفيذ الاستعلام من خلال الشبكة، لذا يجب مراعاة ذلك عند تصميم الاستعلامات.

استخدام رموز وعلامات SQL

بالإضافة إلى معاملات المقارنة المستخدمة داخل التعبيرات المختلفة، تحتوى SQL على عدد من الرموز والعلامات الأخرى كالفاصلة والفاصلة المنقوطة والأقواس والنقطة، حيث لكل عنصر من هذه العناصر مدلوله الخاص داخل SQL وذلك كما يلى:

- الفاصلة (,) وتستخدم فى فصل العناصر المختلفة الموجودة بقائمة واحدة كمـــا فى أسماء الحقول. ففى حالة وجود ثلاثة حقول على سبيل المثال، يتم كتابة هذه الحقول داخل عبارة SQL هكذا Name, Address, Age.
- الأقواس المربعة [ ] وتستخدم للإحاطة بأسماء الحقول التى تحتوى على مسافات أو كلمات محجوزة أو أى رموز أخرى كما فى الحقل [Order Details]، فلا يصح كتابة هذا الحقل داخل عبارة SQL بدون هذه الأقواس. كما يجب استخدام هذه الأقواس مع معاملات الإدخال Input Parameters التى يتم استخدامها داخل استعلامات Access من خلال نافذة التصميم.
- النقطة [.] وتستخدم فى فصل أسماء الكائنات ذات التصنيفات المتداخلة. فعلـــى سبيل المثال، فى حالة تضمين حقول من أكثر من جدول داخل الاستعلام، يتم فصل كل حقل عن اسم الجدول المصاحب باستخدام النقطة هكذا مثلاً:

```
[Order Details] . Order_no
```

- معرفات النصوص وهى علامة التنصيص (') وعلامـــة التنصـــيص المزدوجـــة (") المستخدمة فى كتابة النصوص داخل عبارات SQL، ومن الأفضل استخدام العلامة ' بدلاً من " نظراً لاستخدام الأولى فى كتابة عبارات SQL داخل كود VBA.

- الرموز الشاملة **Wildcards** وهى الرمز ? المستخدم فى الدلالة على غياب حرف واحد والرمز * المستخدم فى الدلالة على غياب أكثر من حرف.
- معرف التاريخ والوقت، وهو الرمز #، حيث يجب وضع قيم التاريخ والوقت داخل عبارات **SQL** بين الرمزين # #.
- معرفات المتغيرات (:) و(@)، حيث يتم استخدام المعرف : للإشارة إلى المتغيرات فى **ANSI SQL** بينما يتم استخدام المعرف @ للإشارة إلى المتغيرات فى **T-SQL**، من المهم أن تعرف أن **SQL** لا تدعم أيٍ من المعرفين.
- الرمز ! ويستخدم كبديل للمعامل **NOT**. يتم استخدام التركيب =! داخل **ANSI SQL** للإشارة إلى عدم التساوى، بينما يتم استخدام <> داخل **SQL** و**T-SQL** للإشارة إلى ذلك، كما تدعم **T-SQL** أيضاً استخدام =!.

ترجمة عبارات SQL داخل نافذة تصميم الاستعلام

حينما تقوم بإنشاء استعلام تحديد باستخدام العبارة **SELECT** من خلال نافذة تصميم الاستعلام، يقوم **Access** بترجمة التصميم الذى قمت به إلى كود **SQL** المصاحب. وكذلك الحال فى مشروعات البيانات، يقوم مصمم المشروعات بترجمة الاستعلام إلى كود **T-SQL** المناسب. فإذا قمت بتغيير الكود الناتج من خلال نافذة **SQL** أو **T-SQL**، تنعكس التغييرات التى قمت بها على الاستعلام فى نافذة تصميم الاستعلام أو مصمم المشروعات على الترتيب.

إنشاء كود SQL

لإنشاء كود **SQL** المكافئ لاستعلام **Access** داخل قاعدة البيانات **Sales.accdb**، تابع معنا الخطوات الآتية:

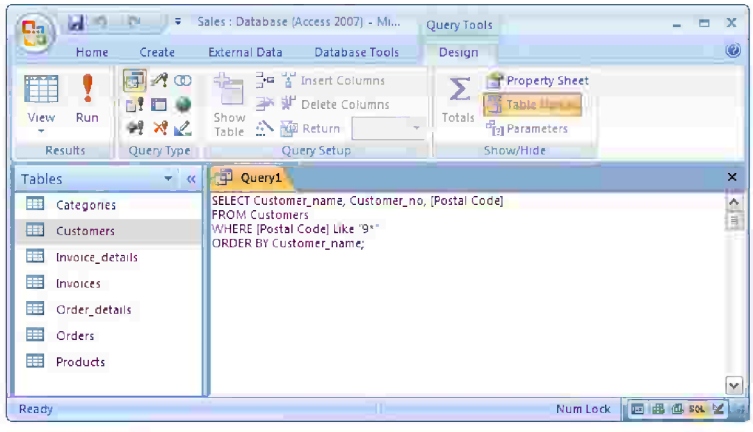
١. تأكد من فتح قاعدة البيانات **Sales.accdb** وإلا قم بفتحها ثم انقر التبويب **Create** بالجزء العلوى من الشاشة وانقر زر تصميم الاستعلامة  **Query Design**، وحينئذٍ يتم فتح استعلام جديد بنافذة تصميم الاستعلام.

٢. قم بإغلاق المربع الحوارى Show Table ثم انقر الزر SQL View بالجانب الأيسر من شريط الأدوات، وحينئذٍ تظهر نافذة استعلام بعنوان Query1 محتويةً على كلمة SELECT.

٣. قم بتعديل عبارة SQL كما يلى (انظر شكل ١-١٧):

```
SELECT Customer_name, Customer_no, [Postal Code]
FROM Customers
WHERE [Postal Code] Like "9*"
ORDER BY Customer_name;
```

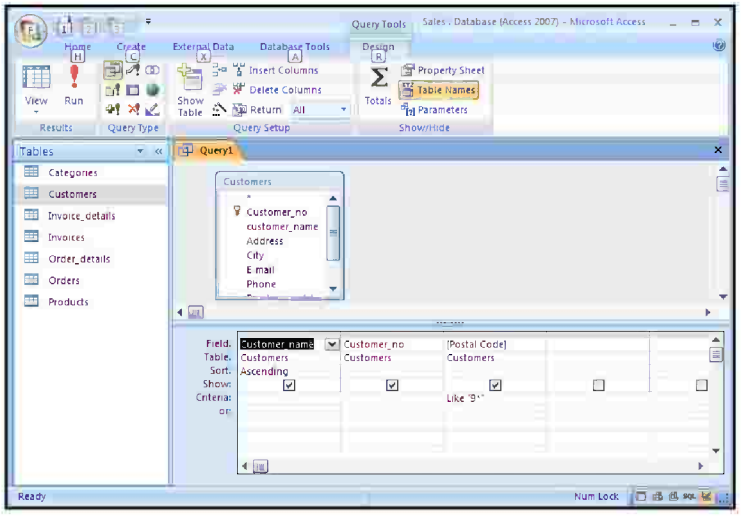
حيث تقوم هذه العبارة بعرض الأعمدة Customer_name وCustomer_no وPostal Code من الجدول Customers للعملاء الذين يبدأ العمود Postal Code فيها بالرقم 9 مع ترتيب السجلات الناتجة تبعاً لاسم الشركة ممثلاً فى العمود Customer_name.



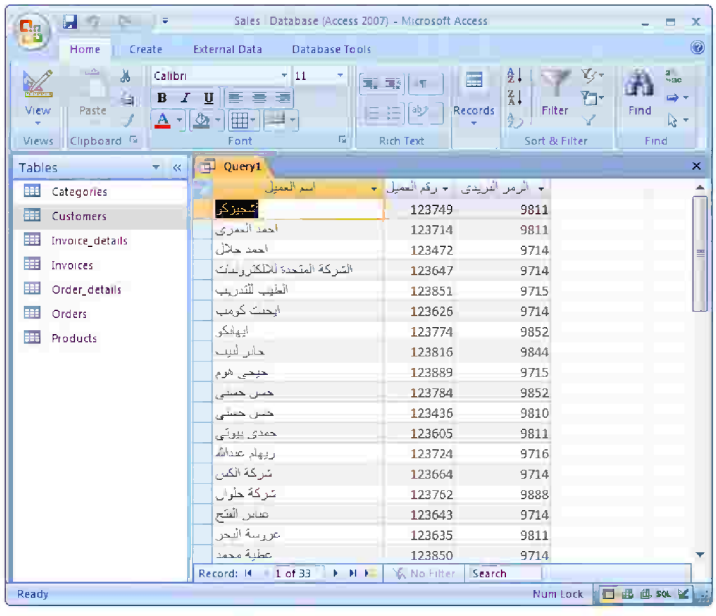
شكل ١-١٧ كتابة عبارة SQL من خلال نافذة عرض SQL.

٤. انقر الزر View بأقصى يسار شريط الأدوات ثم اختر Design View من القائمة الناتجة لعرض شبكة QBE وبها الاستعلام المكافئ لعبارة SQL التى قمت بتعيينها (انظر شكل ٢-١٧).

٥. انقر الزر Run بالجانب الأيسر من شريط الأدوات لتشغيل عبارة SQL التى قمت بإدخالها وبالتالى عرض السجلات الناتجة من الاستعلام (انظر شكل ٣-١٧).



شكل ٢-١٧ الاستعلام المكافئ لعبارة SQL داخل نافذة تصميم الاستعلام.



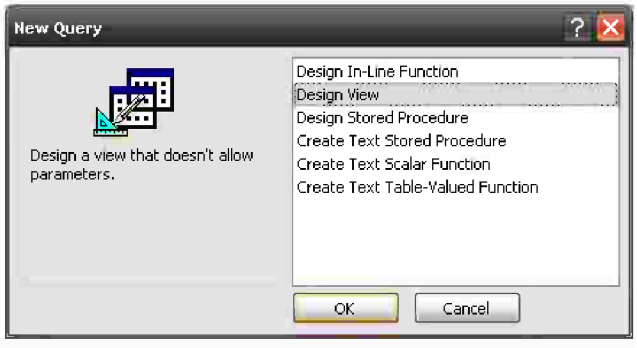
شكل ٣-١٧ نتيجة تشغيل عبارة SQL داخل نافذة عرض الاستعلام.

العمل مع العروض (SQL Server Views)

تشبه العروض Views داخل SQL Server الاستعلامات SELECT داخل Access، إلا أنك تستطيع داخل SQL Server الحصول على نتائج الاستعلامات باستخدام الإجراءات المخزنة Stored Procedures والدوال السطرية Inline Functions. العرض View عبارة عن جملة SELECT تستخدم كبديل لاستعلام التحديد QueryDef داخل SQL Server.

لإنشاء أحد العروض من عبارة SQL باستخدام مصمم المشروعات داخل مشروع بيانات (ADP)، تابع معنا الخطوات الآتية:

١. افتح المشروع SalesProjectCS.adp ثم انقر التبويب Create ومنه انقر الزر Query Wizard. يظهر المربع الحوارى New Query (انظر شكل ٤-١٧).



شكل ٤-١٧ المربع الحوارى New Query.

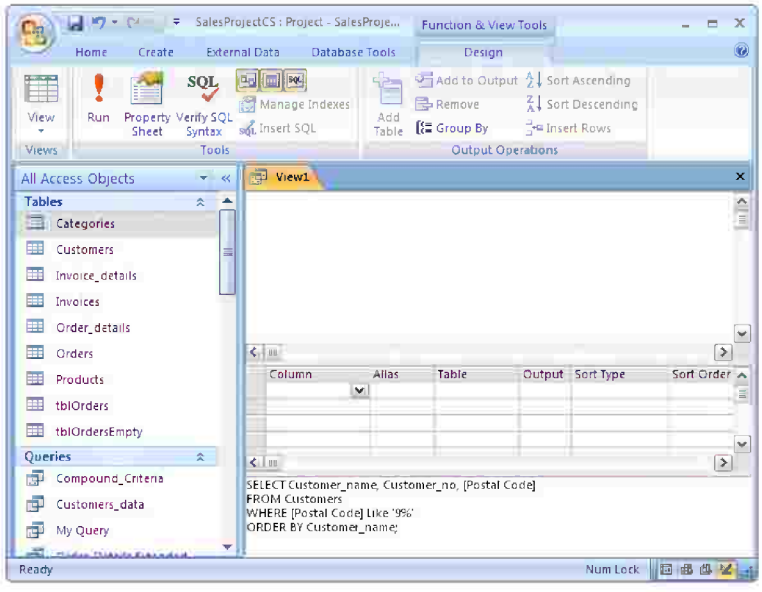
٢. اختر Design View من مربع السرد ثم انقر زر Ok لإغلاق المربع الحوارى New Query وإظهار المربع الحوارى Add Table.

٣. أغلق المربع الحوارى Add Table ثم نشط التبويب Design ثم انقر الزر  بالجزء Tools لإظهار منطقة نص SQL التى تحتوى على عبارة SELECT ... FROM.

٤. قم بكتابة عبارة SQL بنافذة SQL بالجزء السفلى من الشاشة، وعلى فرض أننا نرغب فى استخدام نفس عبارة SQL السابقة، سنقوم بإدخال بعض التعديلات على العبارة كى تتناسب مع T-SQL هكذا (انظر شكل ٥-١٧):

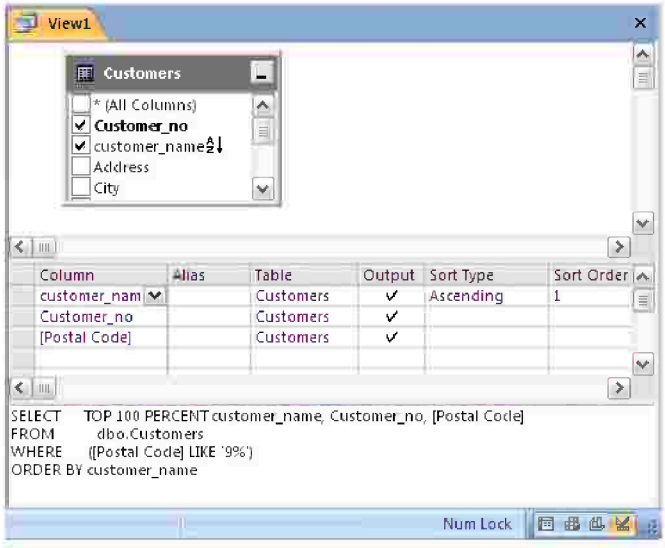
```
SELECT Customer_name, Customer_no, [Postal Code]
FROM Customers
WHERE [Postal Code] Like '9%'
ORDER BY Customer_name;
```

حيث قمنا باستبدال المعيار "9*" بالجزء LIKE داخل SQL بالجزء '9%' ، أى استبدال الأقواس المزدوجة بالأقواس المفردة واستبدال الرمز * بالرمز %.



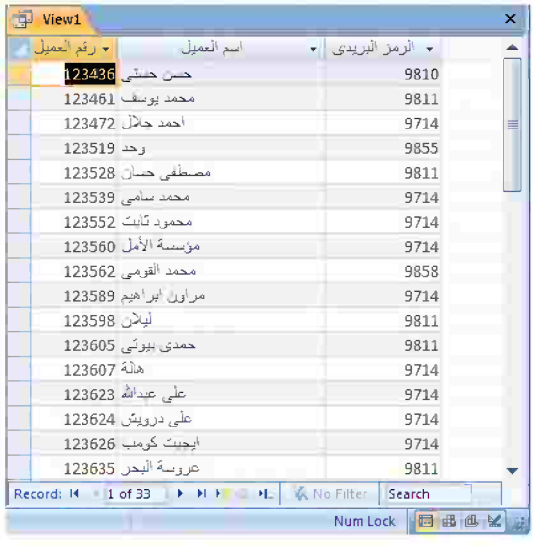
شكل ٥-١٧ كتابة عبارة SQL.

٥. انقر الزر SQL ثم انقر زر Ok رداً على مربع الرسالة الناتج. يظهر الاستعلام الناتج من عبارة SQL بشبكة QBE بالجزئين العلوى والأوسط (انظر شكل ٦-١٧).



شكل ٦-١٧ عرض الاستعلام بشبكة QBE.

٦. احفظ العرض باسم مناسب ثم انقر زر التشغيل ! لعرض نتيجة الاستعلام بنافذة عرض صفحة البيانات (انظر شكل ٧-١٧)، حيث يجب داخل قواعد بيانات SQL Server حفظ العروض أولاً قبل تشغيلها.



شكل ٧-١٧ عرض الاستعلام داخل صفحة البيانات.

استخدام الدوال التجميعية والدوال السطرية

يمكنك استخدام الدوال التجميعية (Aggregate Functions) فى الحصول على الإجماليات والمتوسطات الحسابية والبيانات الإحصائية لمجموعة من السجلات تحتوى على قيمة مشتركة، وفى هذه الحالة يجب استخدام عبارة GROUP BY داخل عبارة SQL وذلك لتحديد السجلات (الصفوف) التى يتم تطبيق هذه الدوال عليها. كما يمكنك تصفية نتيجة عبارة GROUP BY باستخدام كلمة HAVING. والصيغة العامة لهذه العبارة كما يلى:

```
SELECT [ALL|DISTINCT] [TOP (n) [PERCENT]]
        Aggregate_function(field_name) AS alias_name
        [, select_list]
FROM table_names
[WHERE search_criteria]
GROUP BY group_criteria
[HAVING aggregate_criteria]
[ORDER BY column_criteria]
```

حيث:

- تتضمن select_list الدالة التجميعية aggregate_function التى تحتوى على معامل عبارة عن field_name حيث يجب أن يحتوى هذا الحقل على قيمة رقمية.
- يتم استبدال alias_name بعنوان للعمود الجديد الذى يحتوى على القيمة الناتجة من الدالة التجميعية، وهو يكافئ :

alias :aggregate_function (field name) بشبكة الاستعلام داخل Access.

- يستخدم GROUP BY group_criteria فى إنشاء العمود أو الأعمدة التى يتم تجميعها (أى تطبيق الدالة عليها)، حيث تظهر عبارة GROUP BY فى هذا العمود بالصف Totals بشبكة الاستعلام. وهذه العبارة أساسية فى حالة الاستعلامات التجميعية.

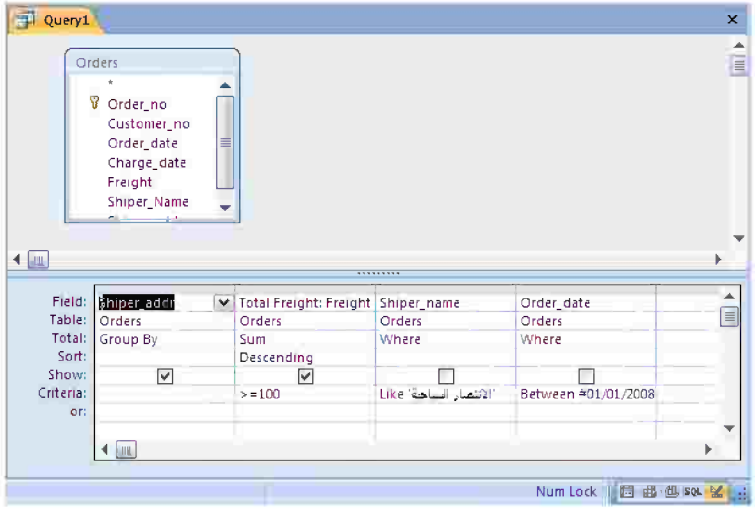
- يستخدم الجزء HAVING aggregate_criteria فى تطبيق معيار أو أكثر على العمود الذى يحتوى على الدالة التجميعية، حيث يتم تطبيق الشرط aggregate_criteria بعد الانتهاء من عملية التجميع نفسها. وهذا الجزء اختيارى.
- يعمل الجزء WHERE search_criteria قبل إجراء عملية التجميع، أى قبل تنفيذ الدالة. وهذا الجزء اختيارى، إلا أنه نادراً ما يخلو منه أى استعلام تجميعى.

للتعرف على كيفية استخدام الدوال التجميعية، قم بتنفيذ الكود التالى داخل قاعدة البيانات Sales.accdb من خلال استعلام بالطريقة التى ذكرناها منذ قليل:

```
SELECT      Orders.Shiper_addr,
            Sum(Orders.Freight) AS [Total Freight]
FROM Orders
WHERE (((Orders.[Shiper_name]) Like "الانتصار السياحية")
AND ((Orders.[Order_date]) Between #1/1/2008# And
#12/31/2008#))
GROUP BY Orders.Shiper_addr
HAVING (((Sum(Orders.[Freight]))>=100))
ORDER BY Sum(Orders.Freight) DESC;
```

تقوم هذه العبارة باسترجاع عمودين، الأول عنوان شركة الشحن (العمود Shiper_addr) بينما يحتوى الثانى على إجماليات مصروفات الشحن لكل منطقة فى سنة ٢٠٠٨ على ألا تقل هذه المصروفات عن ١٠٠.

يوضح شكل ٨-١٧ طريقة عرض تصميم الاستعلام المكافئ للكود السابق.



شكل ٨-١٧ كود SQL السابق بطريقة عرض تصميم الاستعلام.

إذا قمت الآن بتشغيل هذا الاستعلام، تحصل على ثلاثة صفوف لمناطق (عناوين) شركات الشحن المختلفة. فإذا قمت بحذف الجزء HAVING من العبارة (أى عدم اشتراط ألا تقل المصروفات عن ١٠٠)، تحصل على عدد أكبر من الصفوف فى هذه الحالة.

أما داخل SQL Server، فيمكنك استخدام العروض التى تدعم المعاملات من خلال نوع من الدوال يسمى Table-valued functions أو TVFs أو الدوال السطرية. للتعرف على كيفية استخدام عبارة SQL السابقة داخل SQL Server، تابع معنا الخطوات التالية:

١. افتح مشروع البيانات SalesProjectCS.adp إذا لم يكن مفتوحاً بالفعل ثم نشط التبويب Create بالجزء العلوى من الشاشة وانقر الزر Query Wizard، يظهر المربع الحوارى New Query (راجع شكل ٤-١٧).

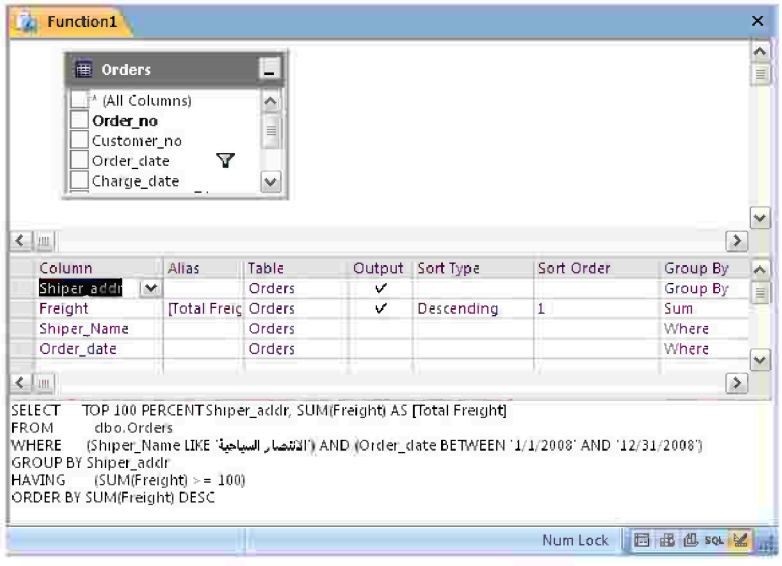
٢. اختر Design In-Line Function من مربع السرد ثم انقر زر Ok لإغلاق المربع الحوارى New Query. تظهر النافذة Function1 وبها المربع الحوارى Add Table.

٣. انقر زر Close لإغلاق المربع الحوارى Add Table ثم قم بتنشيط التبويب Design ثم انقر الزر  بالجزء Tools لإظهار منطقة نص SQL التى تحتوى على عبارة SELECT ... FROM.

٤. قم بكتابة عبارة SQL بنافذة SQL بالجزء السفلى من الشاشة، وعلى فرض أننا نرغب فى استخدام نفس عبارة SQL السابقة، سنقوم بإدخال بعض التعديلات على العبارة كى تتناسب مع T-SQL هكذا (لاحظ استبدال # فى التواريخ بعلامة التنصيص '):

```
SELECT        Orders.Shiper_addr,
Sum(Orders.Freight) AS [Total Freight]
FROM Orders
WHERE (((Orders.[Shiper_name]) Like "الانتصار السياحية")
AND ((Orders.[Order_date]) Between '1/1/2008' And '12/31/2008'))
GROUP BY Orders.Shiper_addr
HAVING (((Sum(Orders.[Freight]))>=100))
ORDER BY Sum(Orders.Freight) DESC;
```

٥. انقر الزر  للتأكد من صحة الكود الذى قمت بإدخاله ثم انقر زر Ok رداً على مربع الرسالة الناتج. يظهر الاستعلام الناتج من عبارة SQL بشبكة QBE بالجزئين العلوى والأوسط (انظر شكل ٩-١٧).



شكل ٩-١٧ عرض الاستعلام بشبكة QBE.

٦. قم بحفظ الدالة باسم مناسب ثم انقر زر التشغيل ! لعرض نتيجة الاستعلام بنافذة عرض صفحة البيانات، حيث نحصل فى هذه الحالة على نفس النتيجة السابقة التى حصلنا عليها فى حالة استعلام Access.

تستخدم الدوال السطرية بدلاً من الإجراءات المخزنة فى الحصول على العرض المصاحب، وذلك لأن العروض لا تدعم استخدام المعاملات، كما أن مجموعات البيانات الناتجة من هذه الدوال قابلة للتحديث بعرض ورقة البيانات على عكس مجموعات البيانات الناتجة من الإجراءات المخزنة.

ربط الجداول من خلال كود SQL

يتم ربط جدولين أو أكثر داخل نافذة تصميم استعلامات Access أو داخل نافذة مصمم المشروعات باستخدام التركيب JOIN ... ON والذى يتم من خلاله تحديد الجداول المرتبطة والحقول المستخدمة فى عملية الربط، حيث يحتوى التركيب على الصيغة العامة التالية:

```
SELECT [ALL|DISTINCT]  select_list
```

```
FROM
table_name [INNER|LEFT [OUTER]] |FULL [OUTER] ]
    JOIN join_table ON join_criteria
[table_name [INNER|LEFT [OUTER]|RIGHT [OUTER] ] |FULL
[OUTER] ]
    JOIN join_table ON join_criteria]
[WHERE search_criteria]
[ORDER BY column_criteria]
```

حيث:

- يستخدم الجزء

table_name [INNER|LEFT [OUTER]] |FULL [OUTER]] JOIN join_table

فى تحديد اسم الجدول المرتبط مع بقية الجداول المذكورة فى **table_name**. فإذا قمت بإنشاء ربط ذاتى لنفس الجدول عن طريق تضمين نسختين من قائمة حقول نفس الجدول، يتم تمييز الجدول الثانى عن الأول بإضافة شرطة تحتية (_) ورقمٍ ما.

يجب أن يسبق عبارة **JOIN** داخل استعلامات **Access** إحدى الكلمات الخاصة بأنواع الربط المختلفة **INNER** أو **FULL** أو **LEFT** أو **RIGHT**، بينما استخدام **INNER** اختيارى فى كود **T-SQL**.

- يستخدم الجزء **ON join_criteria** فى تحديد الحقلين المستخدمين فى عملية الربط وكذلك العلاقة بين هذه الحقول، حيث يحتوى التعبير **join_criteria** عادةً على المعامل = ويقوم بإرجاع القيمة **TRUE** أو القيمة **FALSE**.

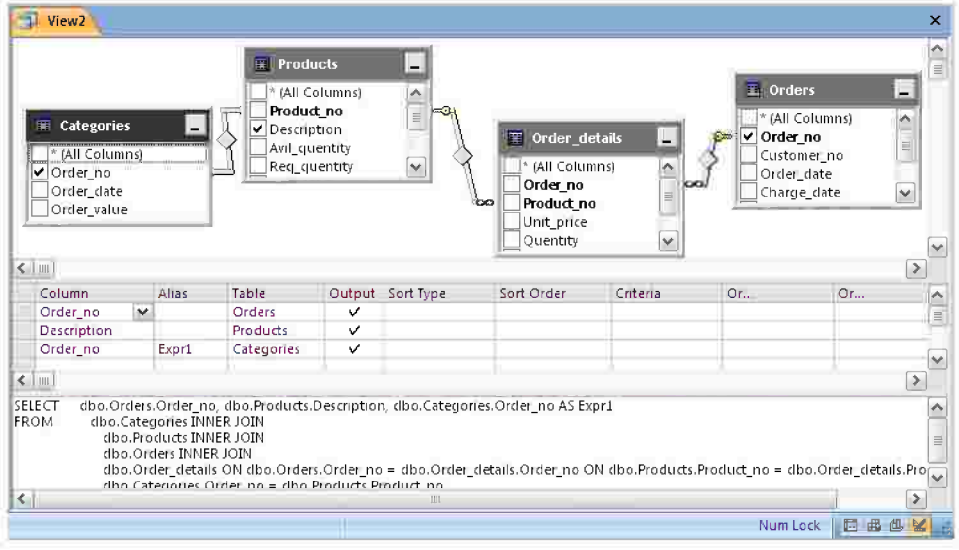
ويتوقف عدد عبارات **JOIN** التى يتم استخدامها داخل عبارة **SQL** على عدد الجداول المستخدمة فى عملية الربط، حيث تقل هذه العبارات بمقدار واحد عن عدد الجداول. أى لربط جدولين، يتم استخدام عبارة **JOIN** واحدة، بينما يتم استخدام عبارتين لربط ثلاثة جداول، وهكذا.

للتعرف على كيفية ربط الجداول باستخدام كود **SQL**، دعنا نرى العبارة التالية والتى يتم فيها إنشاء عملية ربط داخلى بين أربعة جداول وهى الجداول **Orders** و **Order Details** و**Products** و**Categories**:

```
SELECT Orders.Order_no, Products.Description,
Categories.Order_no
FROM Categories INNER JOIN (Products INNER JOIN (Orders
INNER JOIN Order_details ON Orders.Order_no =
Order_details.Order_no) ON Products.Product_no =
Order_details.Product_no) ON Categories.Order_no =
Products.Product_no;
```

تقوم هذه العبارة بإرجاع قائمة المنتجات من الجدول **Products** والمجموعة الخاصة بها من الجدول **Categories** وذلك مع كل طلب صرف داخل الجدول **Orders**.

ويمكنك نسخ هذه العبارة من نافذة عرض **SQL** داخل **Access** إلى مصمم المشروعات الخاص بعرض أو إجراء مخزن أو دالة سطرية للحصول على نفس النتيجة. يوضح شـــكل ١٠-١٧ كود **T-SQL** المكافئ للكود السابق داخل نافذة عرض داخل **SQL Server**.



شكل ١٠-١٧ ربط الجداول من خلال كود **T-SQL**

وبدلاً من استخدام **JOIN** فى ربط الجداول، يمكنك أيضاً استخدام **WHERE** للوصول إلى نفس النتيجة، إلا أن هذه الطريقة غير قابلة للتحديث. فعلى سبيل المثال، للحصول على نفس النتيجة السابقة باستخدام **WHERE**، يمكنك استخدام عبارة **SQL** التالية:

```
SELECT Orders.Order_no, Products.Description,
Categories.Order_no
```

```
FROM Orders, Order_details, Products, Categories
WHERE Orders.Order_no = Order_details.Order_no
AND Products.Product_no = Order_details.Product_no
AND Categories.Order_no = Products.Product_no;
```

ولكن إذا قمت بنسخ هذا الكود إلى مصمم المشروعات داخل SQL Server ثم قمت بنقر زر SQL ، فسيقوم المصمم بتحويل عبارة WHERE إلى عبارات JOIN المصاحبة ونحصل على الكود التالى:

```
ELECT dbo.Orders.Order_no, dbo.Products.Description,
dbo.Categories.Order_no
FROM dbo.Orders INNER JOIN
dbo.Order_details ON dbo.Orders.Order_no =
dbo.Order_details.Order_no INNER JOIN
dbo.Products ON dbo.Order_details.Product_no =
dbo.Products.Product_no INNER JOIN
dbo.Categories ON dbo.Products.Product_no =
dbo.Categories.Order_no
```

فإذا أردت الاحتفاظ بعبارة WHERE داخل الكود كما هى دون تحويل، يجب كتابة الاستعلام باستخدام إحدى أدوات تنفيذ الاستعلام مثل نافذة استعلام SQL Server Management Studio على سبيل المثال.

كتابة استعلامات التوحيد Union Queries

يمكنك من خلال استعلامات التوحيد تجميع البيانات الناتجة من أكثر من استعلام SELECT فى مجموعة واحدة، وهذا ما يحدث عادةً مع الجداول متشابهة التركيب كالجدول Customers والجدول Suppliers على سبيل المثال، حيث يتم إنشاء هذا النوع من الاستعلامات سواءً داخل Access أو داخل SQL Server من خلال كود SQL على عكس الاستعلامات الأخرى التى يتم إنشاؤها من خلال نوافذ التصميم.

وتحتوى استعلامات التوحيد على الصيغة العامة التالية:

```
SELECT field_list
UNION SELECT field_list
[GROUP BY group_criteria]
[HAVING aggregriate_criteria]
[UNION SELECT field_list
```

```
[GROUP BY group_criteria]
[HAVING aggregriate_criteria] ]
[UNION ...]
[ORDER BY column_criteria]
```

حيث:

- يجب أن يتساوى عدد الحقول الموجودة بالقائمة field_list داخل الجزء SELECT مع الحقول الموجودة بكل field_list داخل كل جزء UNION وإلا ستحصل على رسالة خطأ بمجرد تنفيذ هذا الكود.
- يجب ترتيب الحقول الموجودة داخل القائمة field_list ترتيباً واحداً فى جميع عبارات SELECT وإلا ستحصل على نتائج غير سليمة، حيث لا تظهر رسائل خطأ فى هذه الحالة. وفى حالة استعلامات توحيد Access، يتم ترقية نوع البيانات فى حالة اختلاف نوع بيانات الحقل المصاحب فى عبارة SELECT أخرى. أما SQL Server فلا يدعم عملية التحويل التلقائى لأنواع البيانات.
- يمكنك استخدام ORDER BY بعد عبارة UNION SELECT، كما يمكنك استخدام GROUP BY وHAVING داخل كل عبارة SELECT وعبارة UNION SELECT.

يوضح كود SQL التالى استعلام توحيد يقوم بتجميع الصفوف من جدولين، الأول Orders والآخر Invoices:

```
SELECT    Customer_no, Order_date, 'Order' AS Relationship
FROM      Orders
UNION
SELECT    Customer_no, Invoice_date, 'Invoice'
FROM      Invoices
ORDER BY Order_date
```

أما فى كود SQL التالى:

```
SELECT      Customer_no, Charge_date AS Code, Order_date,
'Order' AS Relationship
FROM        Orders
UNION
```

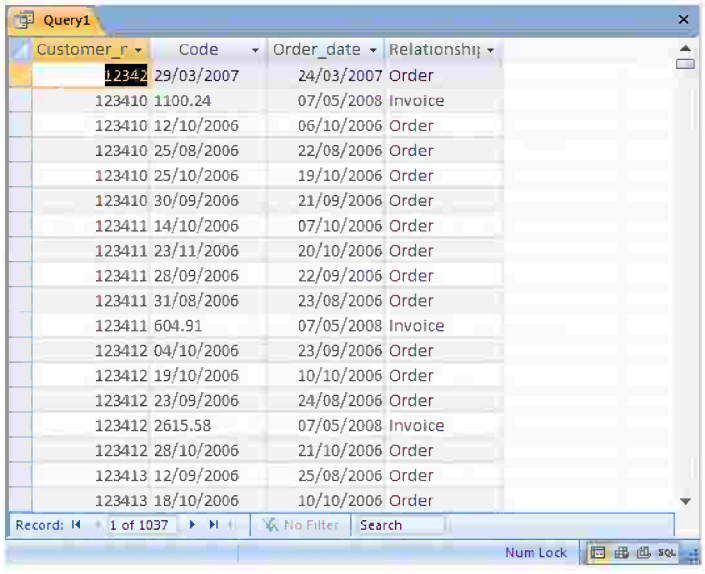
```
SELECT      Customer_no, Invoice_total AS Code, Invoice_date,
'Invoice'
FROM        Invoices
ORDER BY Order_date
```

فقد قمنا بتضمين حقلين بنوعى بيانات مختلفين، الأول هو الحقل **Charge_date** ويحتوى على بيانات من النوع **datetime**، والثانى هو الحقل **Invoice_total** ويحتوى على بيانات من النوع **float**، وهذا ما يدعمه **Access** حيث يقوم بإجراء تحويلات أنواع البيانات متى أمكن ذلك. أما داخل **SQL Server**، فيجب أن تقوم بإجراء عملية التحويل بنفسك من خلال الدالة **CAST()** أو الدالة **CONVERT()** وإلا يتم استخدام نوع بيانات الحقل الأول (أى **datetime**) كنوع بيانات للحقل الثانى المصاحب فنحصل على النوع **datetime** بدلاً من القيمة **float**. فعلى سبيل المثال، للحصول على نفس نتيجة استعلام التوحيد السابق، يتم استخدام كود **T-SQL** التالى:

```
SELECT      Customer_no, Charge_date AS Code, Order_date,
'Order' AS Relationship
FROM        Orders
UNION
SELECT      Customer_no, CAST(Invoice_total AS float) AS Code,
Invoice_date, 'Invoice'
FROM        Invoices
```

كما لا يمكنك فى استعلامات التوحيد بكود **T-SQL** استخدام العبارة **ORDER BY**، وإنما يمكنك تعيين حقول الفرز من خلال الخاصية **Order By** بصفحة الخصائص.

للتعرف على نتيجة تنفيذ الكود السابق، انقر الزر  ثم قم بالانتقال إلى طريقة عرض الاستعلام (انظر شكل ١١-١٧)، حيث تظهر نفس النتيجة سواءً مع كود **SQL** أو كود **T-SQL**.



شكل ١١-١٧ استعلام التوحيد بطريقة عرض البيانات.

يمكنك استخدام استعلامات التوحيد فى إضافة الخيار (All) أو أى خيارات أخرى عند تعبئة مربعات السرد ومربعات السرد والتحرير باستخدام استعلام تحديد. ففى عبارة SQL التالية على سبيل المثال يتم إضافة (All) إلى نتيجة الاستعلام المستخدم فى تعبئة مربع سرد وتحرير مستخدم فى اختيار أوامر الصرف من دولة معينة أو جميع الدول (الخيار All فى هذه الحالة):

```
SELECT City FROM Customers
UNION SELECT '(All)'
FROM Customers
ORDER BY City
```

والسبب فى استخدام الأقواس حول كلمة All هو جعل هذا الخيار فى بداية العناصر الناتجة من الاستعلام.

تنفيذ الاستعلامات الفرعية

يمكنك داخل SQL وT-SQL كتابة استعلام تحديد يحتوى بداخله على استعلام تحديد آخر داخل الجزء WHERE وبالتالى نحصل على ما كان يعرف فى الإصدارات السابقة من

Access بالاستعلامات المتداخلة. وتحتوى الاستعلامات الفرعية على الصيغة العامة التالية:

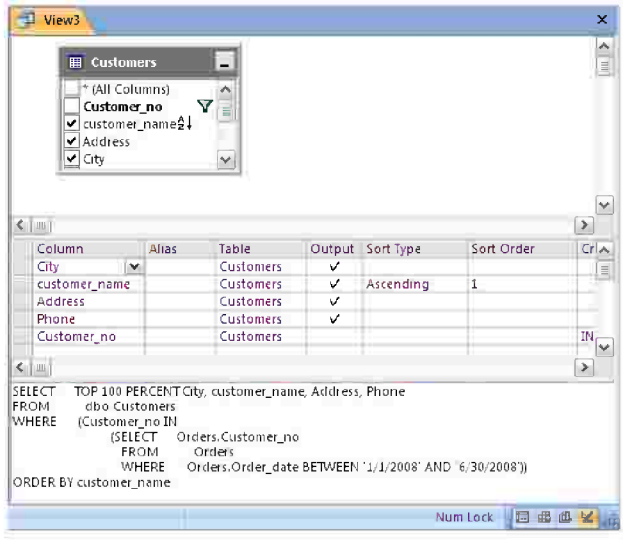
```
SELECT field_list
FROM table_list
WHERE [table_name.]field_name
IN (SELECT select_statment
      [GROUP BY group_criteria]
      [HAVING aggregate_criteria]
      [ORDER BY sort_criteria] )
[ORDER BY sort_criteria]
```

فعلى سبيل المثال، يوضح كود T-SQL التالى استعلاماً فرعياً بسيطاً يقوم بإرجاع أسماء وعناوين العملاء الذين لهم طلبات صرف فى الفترة من ٢٠٠٨/١/١ إلى ٢٠٠٨/٦/٣٠:

```
SELECT Customers.City, Customers.Customer_name,
       Customers.Address, Customers.Phone
  FROM Customers
  WHERE Customers.Customer_no
    IN (SELECT Orders.Customer_no
        FROM Orders
        WHERE Orders.Order_date
            BETWEEN '1/1/2008' AND '6/30/2008')
  ORDER BY Customers.Customer_name
```

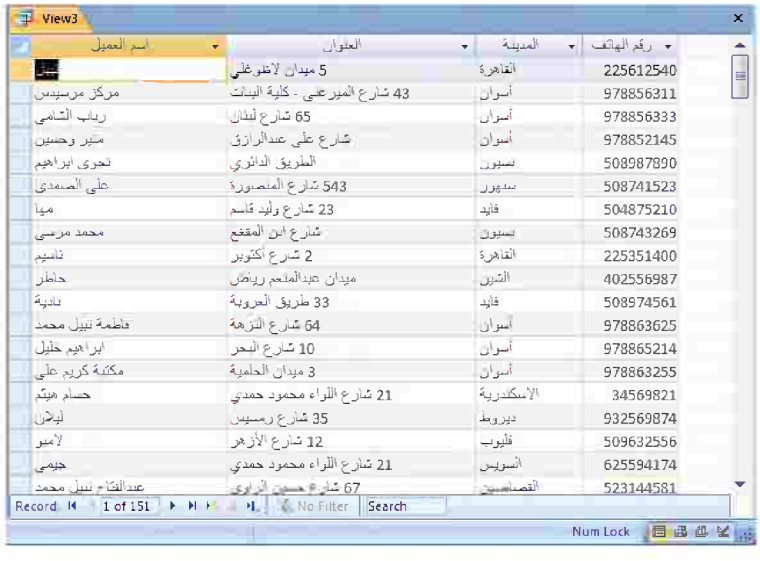
فى هذا الكود، يقوم الاستعلام الفرعى (الذى يبدأ بعد كلمة IN) بإرجاع قيم العمود Customer_no من الجدول Orders ومقارنتها بقيم العمود Customer_no بالجدول Customers.

وعلى عكس استعلامات التوحيد، يدعم عرض تصميم الاستعلامات داخل Access ومصمم المشروعات داخل SQL Server التصميم الرسومى للاستعلامات الفرعية، حيث يتم كتابة IN متبوعةً بعبارة SELECT الفرعية بالعمود Criteria داخل قوسين (انظر شكل ١٢-١٧).



شكل ١٢-١٧ تمثيل الاستعلامات الفرعية من خلال مصمم المشروعات.

يوضح شكل ١٣-١٧ نتيجة تنفيذ عبارة T-SQL السابقة.



شكل ١٣-١٧ نتيجة تنفيذ عبارة T-SQL السابقة.