



IBM DB2® 9.7

DB2 Fundamentals and Data Studio Hands-On Lab

Data Management Emerging Partnership and Technologies

IBM Canada Lab

Contents

1. INTRODUCTION	4
2. OBJECTIVES	4
3. SUGGESTED READING	4
4. GETTING STARTED: THE BASICS OF IBM DATA STUDIO	5
4.1 ECLIPSE FUNDAMENTALS	5
4.1.1 DATA ORGANIZATION – WORKSPACES AND PROJECTS	5
4.1.2 USER INTERFACE – VIEWS AND PERSPECTIVES	7
4.2 ENVIRONMENT SETUP REQUIREMENTS	9
4.3 INITIAL STEPS	9
4.4 LAUNCHING DATA STUDIO	11
4.5 A SLIGHT PAUSE – INTRODUCING DATAGURU.CA	12
4.5.1 USERS	12
4.5.2 TECHNOLOGIES	13
4.5.3 USER_TECHNOLOGIES	13
4.6 DATABASE CONNECTIONS	13
4.6.1 CREATING A NEW CONNECTION	13
4.6.2 DISCONNECTING AND RECONNECTING	15
4.7 EXPLORING THE DATA SOURCE EXPLORER	16
4.7.1 FILTERING BY SCHEMA	16
4.7.2 VIEWING DATA OBJECT PROPERTIES	17
4.7.3 EXECUTING QUERIES AGAINST THE DATABASE	18
4.7.4 VIEWING AND EDITING DATA	19
4.7.5 UPDATING AND VIEWING TABLE STATISTICS	20
4.7.6 ANALYZING QUERIES	21
4.7.7 GRAPHING VALUE DISTRIBUTIONS	23
4.8 SUMMARY	24
5. WORKING WITH DATA DEVELOPMENT PROJECTS	24
5.1 CREATING A DATA DEVELOPMENT PROJECT	25
5.2 CREATING AND EXECUTING SQL SCRIPTS	26
5.2.1 CREATING AN SQL SCRIPT MANUALLY	26
5.2.2 CREATING A SQL SCRIPT USING SQL BUILDER	27
5.3 CREATING A SQL STORED PROCEDURE	30
5.4 CREATING A JAVA STORED PROCEDURE	32
5.4.1 CREATING A TABLE	33
5.4.2 LOADING A TABLE	35
5.4.3 ALTERING A TABLE	36
5.4.4 CREATING THE STORED PROCEDURE	38
5.5 CREATING AN XQUERY SCRIPT	40
5.5.1 RETURNING AN XML DOCUMENT	41

5.5.2	COMBINING XML AND RELATIONAL DATA	42
5.6	CLEAN UP	43
6.	SUMMARY	43

1. Introduction

With the advent of IBM® Data Studio comes a major advance in the way DB2® developers and administrators alike carry out their day to day functions. Historically, depending on the tasks to be completed, it was common to switch back and forth between disparate tools such as Control Center, Health Monitor, Developer Workbench, and even the DB2 Command Line Processor (CLP).

The release of IBM Data Studio changes all this, facilitating DB2 administration, design, development, and monitoring all within an integrated, Eclipse-based environment. Data Studio, the same tool that allows tuning of buffer pools and restriction of access to data objects, can now be used to develop data web services and debug stored procedures. By leveraging the power of IBM Data Studio, users are certain to enjoy increased productivity as they find themselves able to perform a majority of their tasks within a single environment.

In this lab, you will explore IBM Data Studio Version 2.2 from the perspective of both a developer and administrator. Starting from the basics of creating projects and establishing database connections, you will advance to developing stored procedures in both SQL and Java™, writing queries using both SQL and XQuery, and creating a basic data web service. Along the way, you will also explore administrative tasks such as creating and modifying data objects, updating table statistics, and so on.

2. Objectives

By the end of this lab, you will be able to:

- ▶ Understand the basics of an Eclipse-based environment
- ▶ Establish a database connection
- ▶ Create a Data Development project
- ▶ Create, view, and alter data objects
- ▶ Create, deploy, and execute stored procedures in SQL and Java
- ▶ Create and execute queries in SQL and XQuery

3. Suggested reading

IBM Data Studio: Get Started with Data Web Services

<http://www.ibm.com/developerworks/edu/dm-dw-dm-0711pauser-i.html>

An introduction to data web services development, deployment, and testing using IBM Data Studio.

IBM Data Studio Information Center

<http://publib.boulder.ibm.com/infocenter/dstudio/v1r1m0/index.jsp>

A repository complete with tutorials on developing and administering with IBM Data Studio.

4. Getting Started: The Basics of IBM Data Studio

This section of the lab introduces you to the basics of IBM Data Studio and how you can quickly get up and running with it. After completing this section, you will be able to:

- ▶ Launch Data Studio
- ▶ Explain the artefacts of an Eclipse-based environment
- ▶ Create a new database connection
- ▶ Disconnect and reconnect to a database
- ▶ Filter the data objects displayed by schema
- ▶ Execute and analyze a simple query
- ▶ View and edit data
- ▶ View and update table statistics
- ▶ Show a value distribution on one or more columns of data

4.1 Eclipse Fundamentals

IBM Data Studio is built upon the Eclipse platform and, as such, is said to be an Eclipse-based development environment. The Eclipse platform is a framework that allows integrated development environments (IDE) to be created; plug-ins exist to allow development in Java, C/C++, PHP, COBOL, Ruby, and more.

Developers using Eclipse will appreciate the familiar look and feel that IBM Data Studio offers. However, those with no prior Eclipse experience will still find it user-friendly and a highly productive tool.

4.1.1 Data Organization – Workspaces and Projects

In an Eclipse-based environment, all development takes place within a *project*, which is a directory that contains all of the source code, graphics, and other collateral. This is a concept with which most are familiar from using other IDE's. In Data Studio, you will typically work with *Data Development* projects, but other project types exist for Java development, web development, and more.

Each project you create must be contained within a *workspace*, which is a directory in your file system. A workspace directory contains subdirectories for each of the projects created within it. For example, Figure 4-1 demonstrates a scenario in which a workspace has been created on the path `/workspace`, and three projects – BankApp, BookStore, and WebSite – have been created within the workspace. Notice that the projects have all been created as subdirectories of `/workspace`.

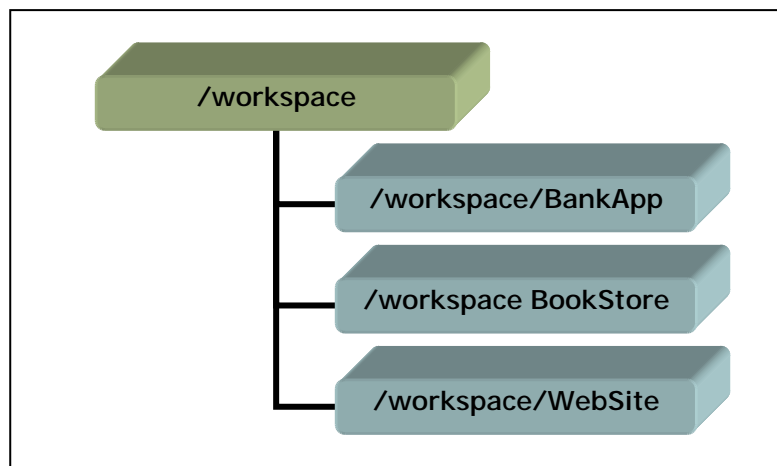


Figure 4-1 – Workspace-project hierarchy

In an Eclipse-based environment, workspaces and projects can easily be navigated through the *Project Explorer* view (we'll cover views in an upcoming section). When an Eclipse-based environment is opened, the user chooses which workspace to use in the dialog displayed in Figure 4-2. It is possible to create a new workspace by entering a new, non-existent path, or to work with an existing workspace by specifying an existing path.

Additionally, users can choose to only work with one particular workspace (and to never be bothered again!) by checking the **Use this as the default and do not ask again** checkbox. Of course, this can always be undone by modifying a setting in the program preferences.

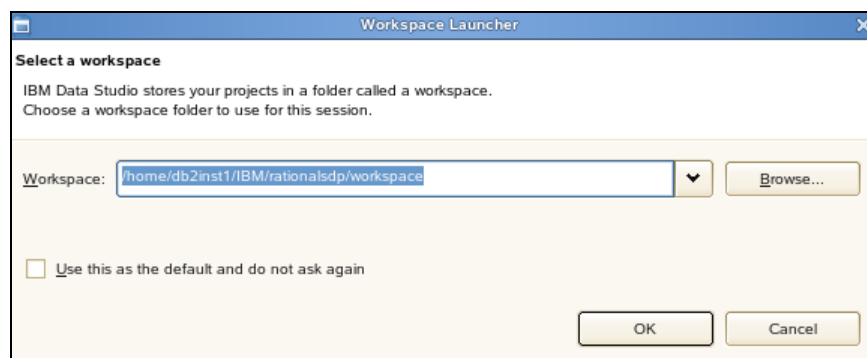


Figure 4-2 – Workspace selection

Figure 4- shows how the workspace hierarchy from Figure 4-1 looks in the Project Explorer.

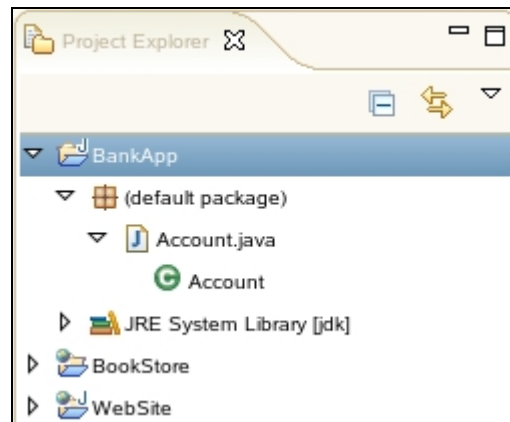


Figure 4-3 – The Project Explorer view

4.1.2 User Interface – Views and Perspectives

Eclipse-based environments offer easy-to-use, customizable graphical interfaces through the use of *views* and *perspectives*. Just as workspaces contain projects, Eclipse perspectives contain views.

In fact, we've already seen an example of a view. In Figure 4-, we saw that the *Project Explorer* view shows all projects in a workspace and files contained within them. A view is nothing more than a task pane – a docked window that allows objects to be viewed and possibly manipulated. There are also many other views, such as the *Data Source Explorer* view, shown in Figure 4-2, which allows users to work with database connections.

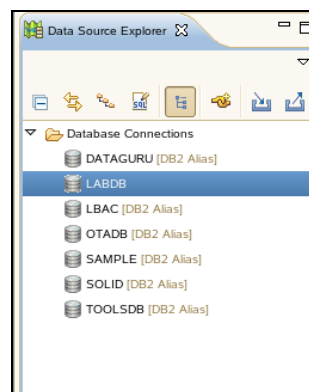


Figure 4-2 – The Data Source Explorer view

Eclipse-based environments define perspectives as a collection of views appropriate for a particular task or line of work. When a perspective is opened, all views associated

with it are opened in the environment, and any other views previously opened are hidden.

In IBM Data Studio, you will generally work with the *Data* perspective shown in Figure 4-4. This perspective provides the *Data Project Explorer* view, *Data Source Explorer* view, *Data Output* view, and others.


To switch between perspectives, click the desired name in the toolbar displayed in Figure 4-3. If the perspective you are looking for is not displayed, simply click the  toolbar icon to bring up a list of available perspectives.



Figure 4-3 – Changing perspectives on the toolbar

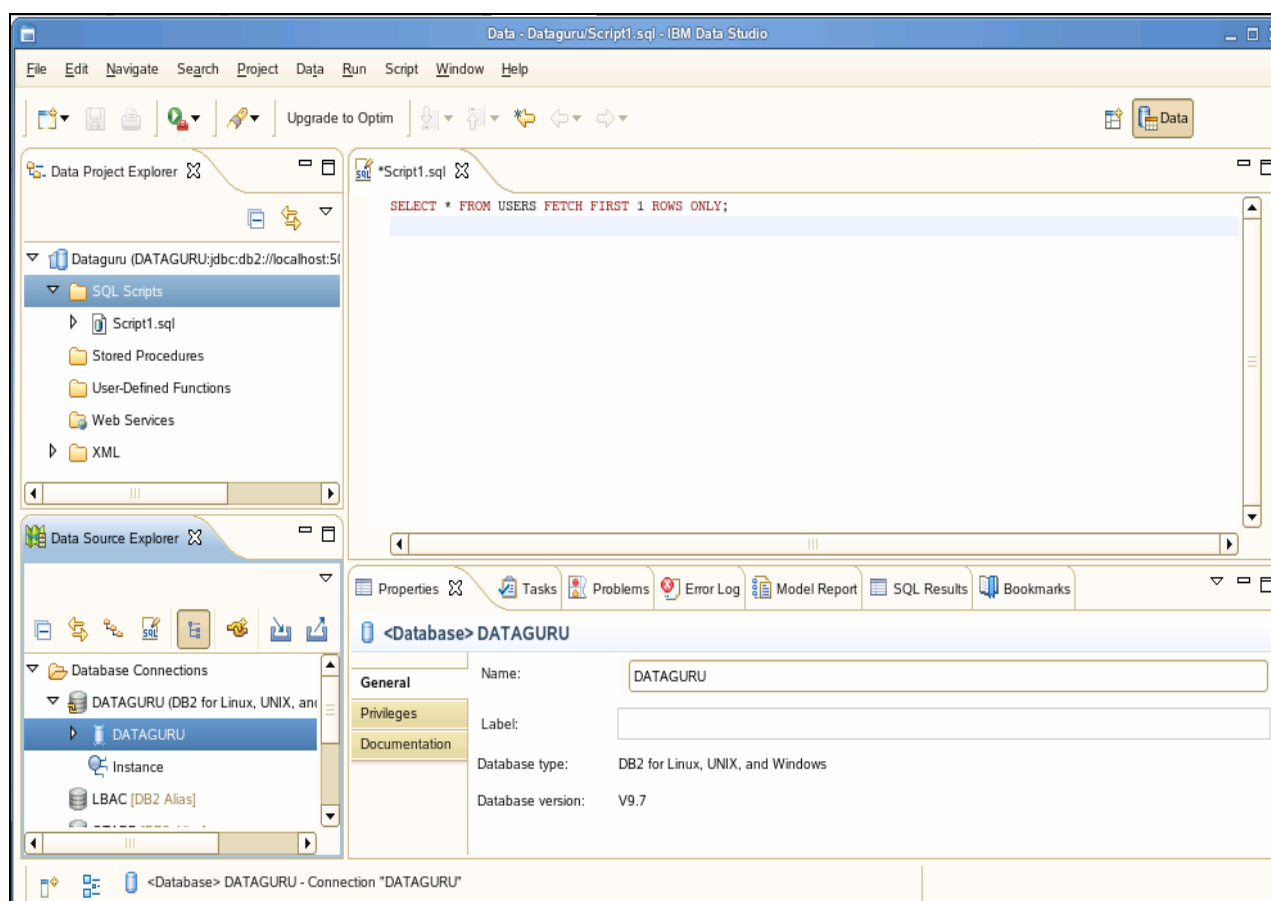


Figure 4-4 – The Data perspective

Eclipse-based environments allow creation of custom perspectives by specifying which views to load.

4.2 Environment Setup Requirements

To complete this lab you will need the following:

- DB2 9.7 Bootcamp VMware® image
- VMware Player 2.x or VMware Workstation 5.x or later

For help on how to obtain these components please follow the instructions specified in **VMware Basics and Introduction** from module 1.

4.3 Initial Steps

1. Start the VMware image by clicking the  button in VMware.

2. At the login prompt, login with the following credentials:

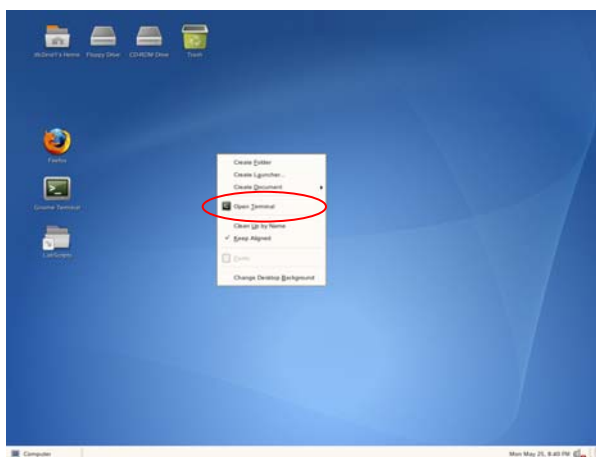
- ▶ Username: **db2inst1**
- ▶ Password: **password**

3. Read and accept the license agreement.

4. Start the graphical interface by issuing the following command at the prompt:

```
db2inst1@db2rules:~> startx
```

5. Open a terminal window by right-clicking on the **Desktop** and choosing the **Open Terminal** item.



6. Ensure that the DB2 Database Manager has been started by issuing the following command at the prompt:

```
db2inst1@db2rules:~> db2start
```

Note: This command will only work if you logged in as the user `db2inst1`. If you accidentally logged in as another user, type `su - db2inst1` at the command prompt password: `password`.

7. On the desktop, double click on the **LabScripts** folder.



8. In the **LabScripts** window that appears, double click on the **DataStudio** folder.
9. Double click on the **open_queries** file and click the **Run** button in the **Run or Display?** dialog.



The last step will load an editor with all the SQL scripts that will be used throughout the lab, saving you from having to type each one manually. When prompted to paste a script into Data Studio, switch to this window, select the appropriate tab, and copy and paste the contents into Data Studio.



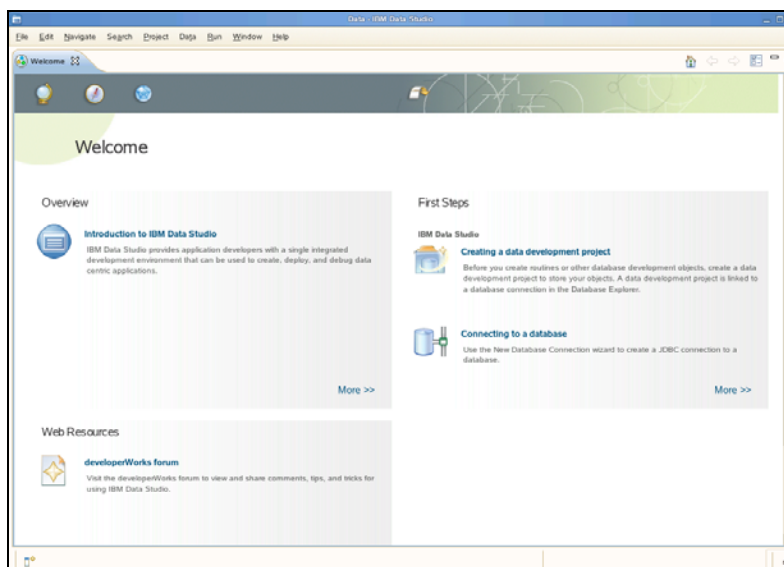
Note: Data Studio has an import feature to import these scripts but for learning purposes it is important to see how to create scripts from scratch.

4.4 Launching Data Studio

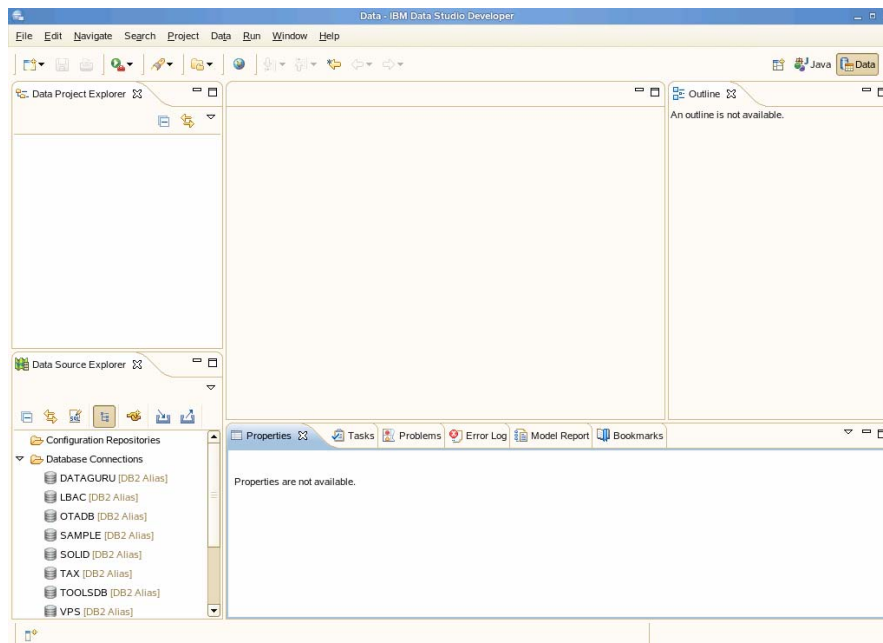
1. Click on the **Computer** button in the bottom left corner of the screen, and select **Data Studio Developer**.



2. In the **Select a workspace** dialog shown in Figure 4-2, accept the default path and check the **Use this as the default and do not ask again** checkbox. Click **OK**.
3. Data Studio will now start with the Welcome homepage.



4. Minimize this window by clicking the minimize button (⏏) located at the top right to bring you into the **Data** perspective as shown below.



4.5 A Slight Pause – Introducing DataGuru.ca

Throughout this lab, development and administration steps will be performed using IBM Data Studio on a database named **DATAGURU**. This database powers a new web site named DataGuru.ca – a fictional recruiting site for database administrators and other data professionals.

The management of DataGuru, Inc. prides the site for its simplicity: users can create accounts, specify the technologies with which they have had experience, and recruiters can search for candidates having skills that they require. As such, the database powering the site has only three tables: **USERS**, **TECHNOLOGIES**, and **USER_TECHNOLOGIES**.

4.5.1 USERS

The **USERS** table stores registration data for the users of DataGuru.ca. This includes personal information about them, along with their 8-character usernames and passwords, allowing them to login to the site. The table schema is shown in Table 4-1.

Column	Type
id	INTEGER
first_name	VARCHAR(20)
last_name	VARCHAR(20)
address	XML
email	VARCHAR(50)

birth_date	DATE
employer	VARCHAR(50)
username	CHAR(8)
password	CHAR(8)
is_recruiter	CHAR(1)
years_experience	INTEGER

Table 4-1 – USERS table

4.5.2 TECHNOLOGIES

The TECHNOLOGIES table stores a list of technologies from which users can pick when completing their profiles on the site. The table schema is shown in Table 4-2.

Column	Type
id	INTEGER
technology	VARCHAR(25)

Table 4-2 – TECHNOLOGIES table

4.5.3 USER_TECHNOLOGIES

Finally, the USER_TECHNOLOGIES table serves as a join table between the USERS and TECHNOLOGIES tables, storing all the technologies that a given user has selected. The table schema is shown in Table 4-3.

Column	Type
user_id	INTEGER
technology_id	INTEGER

Table 4-3 – USER_TECHNOLOGIES table

4.6 Database Connections

Before you can do anything productive with Data Studio, a connection must be established to a database. The Data Source Explorer view in Data Studio allows you to do this. From this view it is possible to interact with and manipulate database artifacts. Since we will be working with the DATAGURU database, let's create a connection to it.

4.6.1 Creating a New Connection

1. In Data Studio navigate to the **Data Source Explorer** view, right-click on the **Database Connections** folder and select **New....**

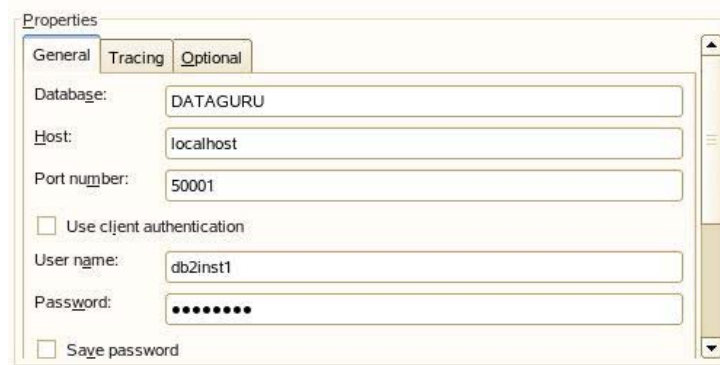
► **Note:** You can also click the  icon in the Data Source Explorer toolbar.

2. Since we're using DB2 on Linux®, select **DB2 for Linux, Unix®, and Windows®**.



3. In the **Properties** pane, you specify the name of the database to which you wish to connect, the host, port number, name of the database instance and the password. Enter the following information:

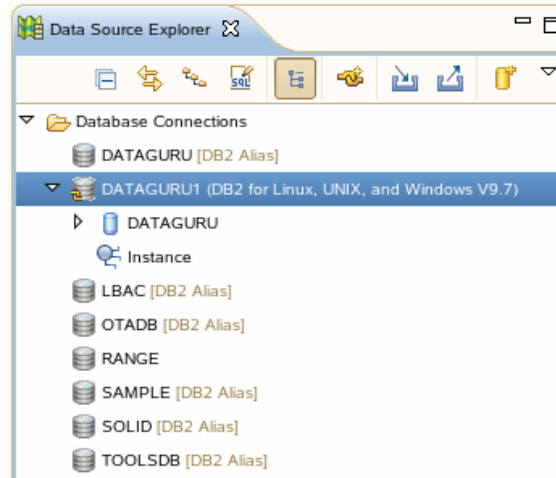
- **Database:** DATAGURU
- **Host:** localhost
- **Port number:** 50001
- **User name:** db2inst1
- **Password:** password



4. Click the **Test Connection** button located on the left. You should receive a message indicating that the connection succeeded. If not, repeat steps 2 – **Error! Reference source not found.**, ensuring that your spelling is correct, and try again. Click **Next** when the test is successful.
5. The next page allows you to filter out the data objects that you see by the schema in which they exist. We'll just leave it as is for now, and see another way to filter by schema later on. Click **Finish** to create the connection.

6. In the Data Source Explorer view, expand the **Database Connection** folder if necessary by clicking the ▶ icon. Notice the **DATAGURU1 (DB2 for Linux, UNIX, and Windows V9.7)** entry. This is the connection that was just created. A “1” was appended to its name because an entry for DATAGURU already exists in the list (we’ll get to that in a moment).

Also notice that the connection icon beside **DATAGURU1** has a chain, while the others don’t. This means that **DATAGURU1** is the only database connection currently open.



4.6.2 Disconnecting and Reconnecting

It turns out that we took the long way around when we created our database connection. If you have DB2 database aliases configured on your system, Data Studio will automatically detect them and configure them as connections in the Data Source Explorer. That’s why when we created a connection to the DATAGURU database, it was named DATAGURU1 – the DATAGURU connection already existed!

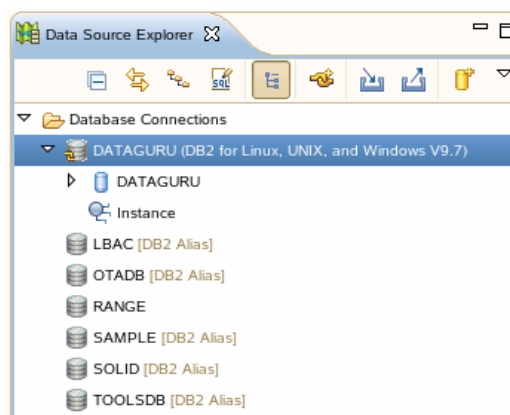
Let’s connect using the DATAGURU connection to see how simple it is to reconnect to a database using an existing connection.

1. In the Data Source Explorer, right-click on the **DATAGURU [DB2 Alias]** node, and select **Connect**.
2. Enter the same credentials as before when the **Database Authorization** dialog appears:
 - ▶ **User ID:** db2inst1
 - ▶ **Password:** password

Click **OK**. Notice that the **DATAGURU** connection icon now has a chain as well, signifying that the connection has been established. If you receive an error here, make sure that the user name and password entered are specified correctly.

Since we are now using the **DATAGURU** connection, let's get rid of the extraneous **DATAGURU1** connection. First, we'll disconnect from it and then delete it. Normally, you could just go ahead and delete it without disconnecting, but DataGuru, Inc. employees like to click things.

3. Right-click on the **DATAGURU1 (DB2 for Linux, UNIX, and Windows V9.7)** node, and select **Disconnect**. Notice that the connection icon no longer has a chain, signifying that the connection has been terminated.
4. Right-click on the **DATAGURU1** node again, and select **Delete**. The Data Source Explorer should now look as follows:



4.7 Exploring the Data Source Explorer

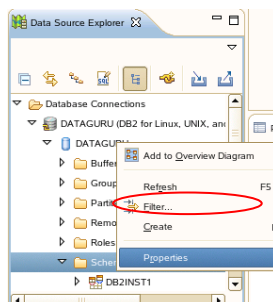
The Data Source Explorer view allows you to do a whole lot more than just connecting and disconnecting from a database. In fact, using the Data Source Explorer, you can perform many of the tasks you might have done in the past with a tool such as Control Center: creating tables and views; modifying columns; viewing table data, and so on. In this section, we'll spend a few moments getting acquainted with the basics of the Data Source Explorer.

4.7.1 Filtering by Schema

1. In the Data Source Explorer, expand the following nodes by clicking the ▶ icons beside them: **DATAGURU [DB2 for Linux...]** > **DATAGURU** > **Schemas**.

Notice that there are many different schemas listed: DB2INST1, NULLID, SQLJ, etc. Because we will only be working with the DB2INST1 schema, let's filter our list to show only this schema.

2. Right-click on the **Schemas** node and select **Filter**. The **Filter** dialog will appear, allowing you to filter either by typing in the name of a schema (or a portion thereof), or by selecting from a list of schemas. For this lab, we will filter by selection:



- ▶ Uncheck the **Disable filter** checkbox.
- ▶ Select the **Selection** radio button.
- ▶ Select **Include selected items** in the drop down list.
- ▶ Check the **DB2INST1** checkbox.
- ▶ Click **OK**.

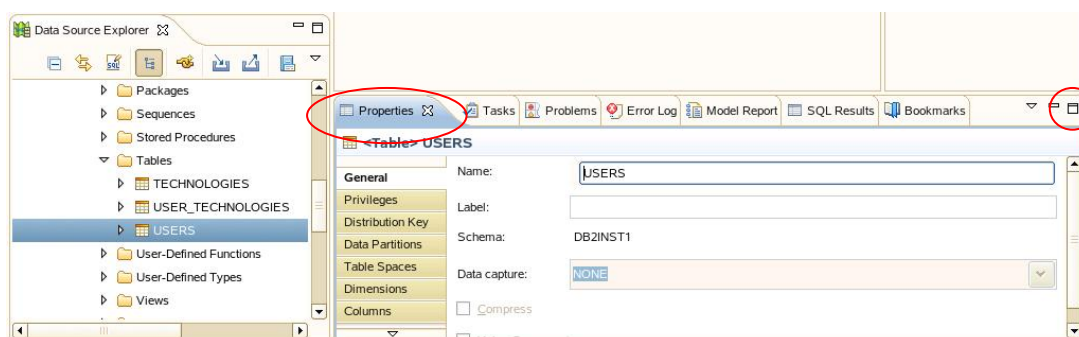



Repeat step 1 and notice that the previous **Schemas** node now reads **Schemas [Filtered]**. Note as well that **DB2INST1** is the only schema that appears in the list.

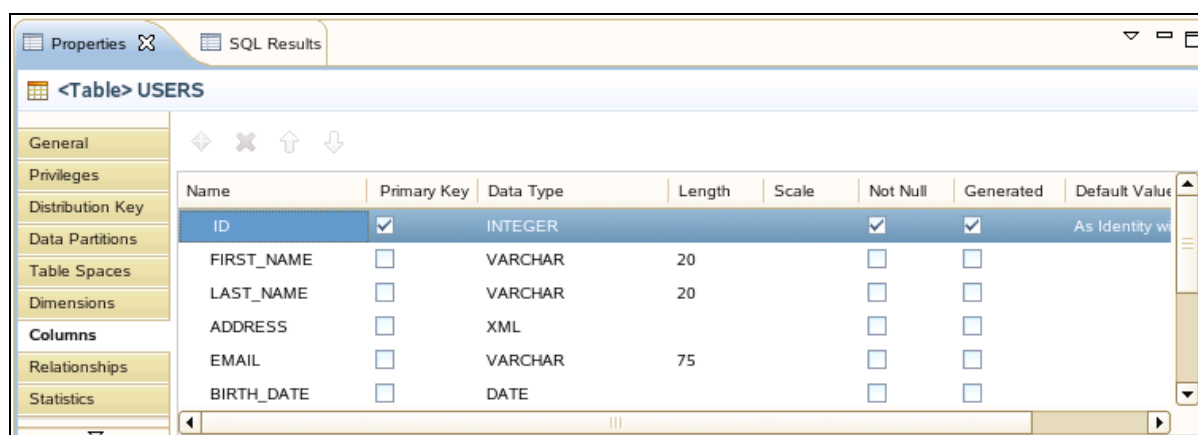
4.7.2 Viewing Data Object Properties


In conjunction with the *Properties* view, the Data Source Explorer allows users to view most properties of nearly every single data object available such as tables, views, packages, sequences, and so on. Let's spend a moment viewing the properties of the **USERS** table to see how the Properties view works.

1. In the **Data Source Explorer**, expand the following nodes by clicking the ▶ icons beside them: **DATAGURU > Schemas [Filtered] > DB2INST1 > Tables**.
2. Select the **USERS** table by clicking on the artefact.
3. Select the **Properties** tab to the right of the **Data Source Explorer**.




4. Expand the **Properties** view by clicking the  icon in its top right corner.
5. Click the **Columns** tab to view a list of the columns in the **USERS** table.



6. Try checking the **Not Null** checkbox beside the **FIRST_NAME** column. Notice that Data Studio does not allow you to do this. This is because the **Properties** view is read-only. We'll see later on how to modify data objects.
7. Spend a few moments getting acquainted with the various properties available in the **Properties** view. When finished, restore the **Properties** view to its original size by clicking the  icon.


4.7.3 Executing Queries against the Database

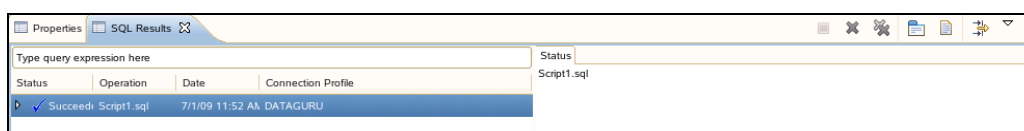
Because no database is worth much unless data can be obtained from it, it's important to understand how to query a database using Data Studio. While we will see other methods of executing queries later on, the method demonstrated here is the fastest way of querying a database.

1. In the **Data Source Explorer** toolbar, click the  icon.
2. In the Select Connection Profile window that appears, select **DATAGURU** and click **Finish**.

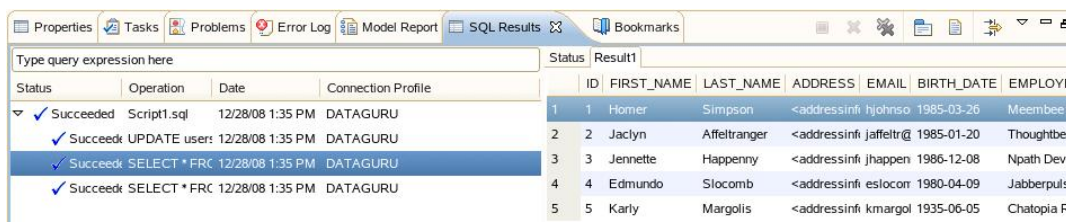
3. A new tab will appear in the main view. You may either type the following query, or copy and paste **Query1.sql** from the editor loaded in section 4.2.



```
UPDATE users SET last_name = 'Simpson' WHERE id = 1;
SELECT * FROM users FETCH FIRST 5 ROWS ONLY;
SELECT * FROM users WHERE last_name LIKE '%Simp%';
```

4. Click the main view to bring focus back to it. From the main menu, select **Run > Run SQL**.
5. Notice that the **SQL Results** view is brought to the foreground at the bottom of the screen. Click the  icon to maximize the view. The SQL Results view should indicate that Script1.sql was successful. In the Status tab to the right, a summary of the statements in Script1.sql are listed.



6. Expand the drop down arrow beside the blue check mark for Script1.sql. As we can see, all three statements were successful. To view the results of the Select statements, click on the desired Select statement and click the **Results** tab on the right.



7. Click the  icon to restore the **SQL Results** view to its original state.
8. Close the **Script1.sql** tab in the main view by clicking its  icon, and click **No** when asked to save changes.

4.7.4 Viewing and Editing Data


In the last section, we executed a query that changed the last name of the user with id 1, retrieved the first five rows from the database, and then retrieved all users whose last names begin with "Simp". For simple tasks like this, the Data Source Explorer has built-in functionality that obviates the need to manually write queries.

1. Expand the artefacts as explained in previous steps to navigate to the **DB2INST1 Schemas [Filtered] > DB2INST1 > Tables**. Right-click on the **USERS** table in the **Data Source Explorer**, and select **Data > Edit**. In the main view, a tab will open showing all data contained in the **USERS** table.


2. Change the first user's last name back to **Johnson**:
 - ▶ Select the **LAST_NAME** column in the first row.
 - ▶ Double-click in the cell until the column becomes editable.
 - ▶ Change Simpson to Johnson and press **Enter**.

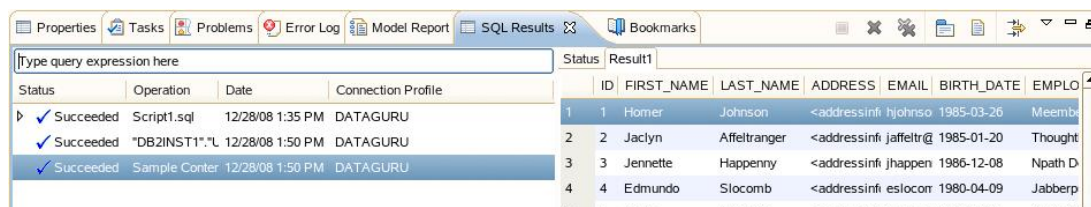


ID [INTEGER]	FIRST_NAME [VARCHAR(20)]	LAST_NAME [VARCHAR(20)]	ADDRESS [XML]	EMAIL [VARCHAR(255)]
1	Homer	Johnson	<addressinfo><street> hjohnson@me...	


3. Close the **USERS** tab by clicking its  icon.
4. When prompted to save changes, click **Yes**.

Now that we've seen how to perform a simple edit to our data without using SQL, let's see the results of the change. Data Studio has a useful function that allows you to see a sample of the data contained in a table. This is useful when you just want to get a feel for what sort of data a given table stores. We'll use this function to verify that our change was made properly.

5. Right-click on the **USERS** table in the **Data Source Explorer**, and select **Data > Sample Contents**.
6. The **SQL Results** view should be brought to the foreground at the bottom of the screen. Click the  icon to maximize it.



Status	Operation	Date	Connection Profile	ID	FIRST_NAME	LAST_NAME	ADDRESS	EMAIL	BIRTH_DATE	EMPLOYEE
✓ Succeeded	Script1.sql	12/28/08 1:35 PM	DATAGURU	1	Homer	Johnson	<addressinfo> hjohnso	1985-03-26		Meemb...
✓ Succeeded	"DB2INST1".L	12/28/08 1:50 PM	DATAGURU	2	Jaclyn	Affetranger	<addressinfo> jaffetr@	1985-01-20		Thought
✓ Succeeded	Sample Conter	12/28/08 1:50 PM	DATAGURU	3	Jennette	Happenry	<addressinfo> jhappen	1986-12-08		Npath D
				4	Edmundo	Slocomb	<addressinfo> eslocor	1980-04-09		Jabberp
				5	Karl	Marronic	<addressinfo> kmarron	1975-06-05		Chaton

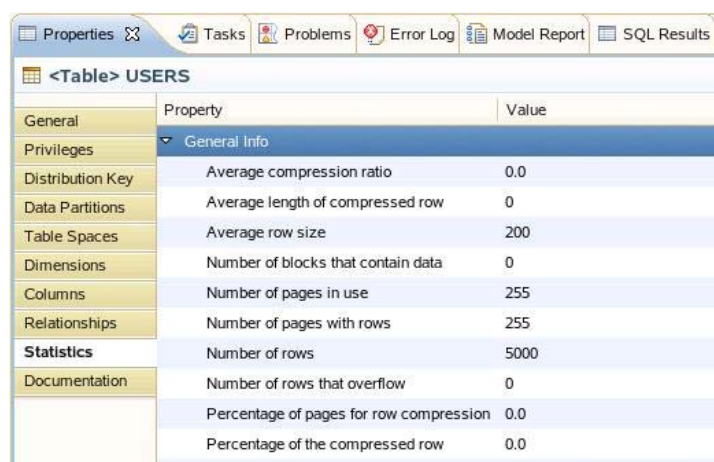
7. Notice that the first user's last name has been changed to **Johnson**. Additionally, scroll to the bottom of the **Results** tab in the **SQL Results** view, and note that only a subset of the table's 5000 rows are displayed when using the **Sample Contents** function.
8. Click the  icon to restore the **SQL Results** view to its original state.

4.7.5 Updating and Viewing Table Statistics

The management of DataGuru, Inc. have made it clear to you that queries to the database powering the web site need to be executed as quickly as possible. You decide to help improve matters by performing a RUNSTATS operation on the tables of the database. RUNSTATS updates catalogue statistics used by the DB2 optimizer to

determine the fastest path to data required by a query. Let's take a look at how to update and view statistics using Data Studio.

1. Select the **USERS** table in the **Data Source Explorer**.
2. Right-click on the **USERS** table in the **Data Source Explorer**, and select **Update Statistics**. The **SQL Results** view will be brought to the foreground, displaying a message indicating that the RUNSTATS operation has completed.
3. Click on the **Statistics** tab in the **Properties** view to the right of the **Data Source Explorer** the following will display:




Property	Value
General Info	
Average compression ratio	0.0
Average length of compressed row	0
Average row size	200
Number of blocks that contain data	0
Number of pages in use	255
Number of pages with rows	255
Number of rows	5000
Number of rows that overflow	0
Percentage of pages for row compression	0.0
Percentage of the compressed row	0.0

4.7.6 Analyzing Queries

Those familiar with DB2 development will have no doubt encountered the Visual Explain tool at some point in their careers. Visual Explain is a Java-based tool that displays a graphical depiction of the access plan that the DB2 optimizer has selected for your query. It is extremely useful in determining why queries are running slowly, and offers insight into how to optimize them.

Fortunately, Data Studio allows developers to execute Visual Explain on their queries right from within the environment. Let's write a query to display all technologies displayed in the TECHNOLOGIES table, along with the number of users that list each particular technology as a skill. We'll then get Visual Explain to analyze it for us. Note that it is always a good idea to perform a RUNSTATS operation on the tables involved in a query before using Visual Explain, so we'll also take care of that.


1. In the **Data Source Explorer** toolbar, click the  icon and select **DATAGURU** as the database and click **Finish**.
2. A new tab will appear in the main view. You may either type the following query, or copy and paste **Query2.sql** from the editor loaded in section 4.2.

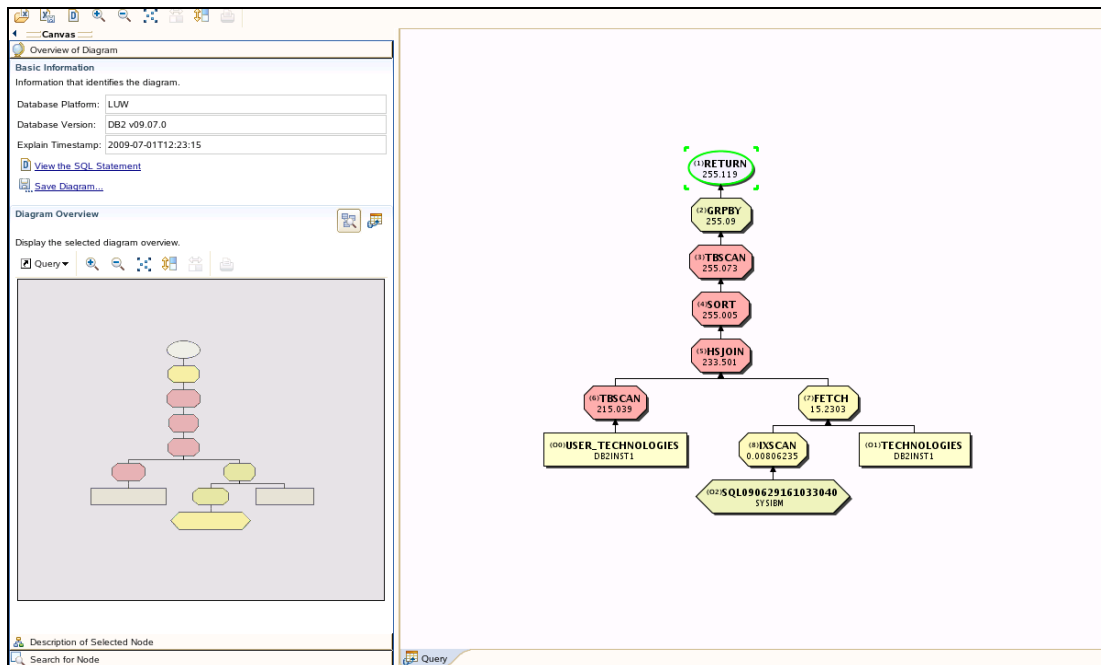
```
SELECT      t.technology, COUNT(*) workers
```


```

FROM      technologies t
INNER JOIN user_technologies ut ON t.id = ut.technology_id
GROUP BY  t.technology
ORDER BY  t.technology

```

3. From the main menu, select **Run > Run SQL**. The results of the query will appear in the **Result1** tab of the **SQL Results** view.
4. Navigate to the **Schemas [Filtered] > DB2INST1 > Tables** and right-click on the **TECHNOLOGIES** table in the **Data Source Explorer** and select **Update Statistics**.
5. Right-click on the **USER_TECHNOLOGIES** table in the **Data Source Explorer** and select **Update Statistics**.
6. Right-click inside the main view of the **Script2.sql** window and select **Open Visual Explain**.
7. In the **Collect Explain Data** window that appears, click **Finish**.
8. Visual Explain will display its analysis of the query. Maximize the **Access Plan Diagram** view by clicking the  icon. Double-click on a node in the graph to view its properties. When finished, close the Visual Explain window.



9. Close the **Script2.sql** tab by clicking its  icon. When prompted to save changes, click **No**.

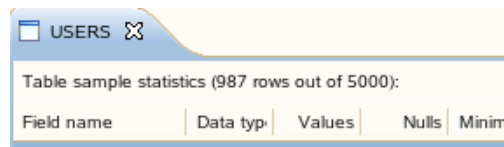
4.7.7 Graphing Value Distributions

The ability to view trends in data is an extremely valuable tool, as it can be used for a multitude of applications. Fortunately, Data Studio offers the ability to graph value distributions data, giving a simple yet effective way of identifying basic patterns and trends that may exist. A value distribution is nothing more than a histogram – a bar chart that tallies the unique values in a given column, and displays how many records have a given value.

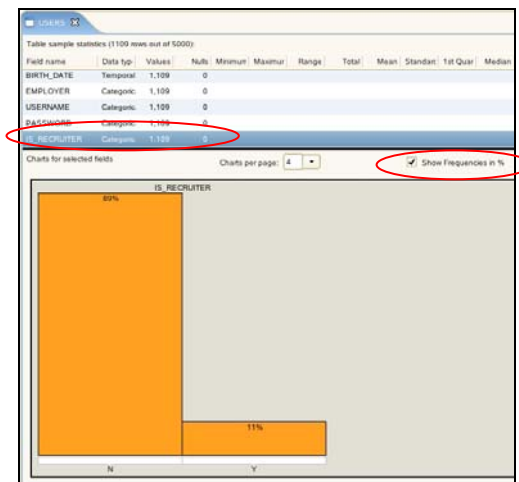
Let's take a look at the proportion of recruiters to job seekers in the DATAGURU database. We'll then take a look at the birth date value distribution to see the age ranges of the users of the site.

1. In the **Data Source Explorer**, right-click on the **USERS** table and select **Value Distributions > Multivariate**.
2. In the main view, a bar chart will appear. Click the ☐ icon to maximize the main view.

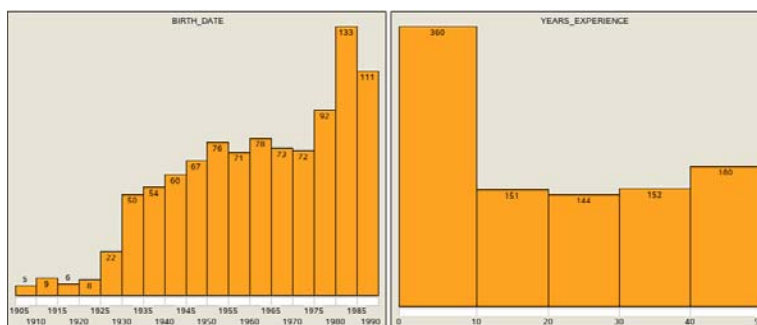
Notice the line **Table Sample Statistics (XXXX rows out of 5000)** at the top of the view. This tells you that not all rows have been used in graphing the distribution – only a representative sample (specifically, XXXX rows) are used. In fact, each time you use the Value Distribution tool, Data Studio will select a different, representative sample size.



3. Select **IS_RECRUITER** in the list of columns above the bar chart.
4. Check the **Show Frequencies in %** checkbox to see percentages on the bar chart. Based on the results of the chart, the users of the site are comprised of approximately 89% job seekers, and 11% recruiters.



5. Select **BIRTH_DATE** in the list of columns above the bar chart. Notice that Data Studio graphs the column by ranges of birth years – it automatically detects that the data is temporal and responds accordingly.
6. Hold down the **Ctrl** key and select **YEARS_EXPERIENCE** in the list of columns above the bar chart. The graphs now appear side by side.



7. Select **LAST_NAME** in the list of columns above the bar chart. You will receive an error message after attempting to do so, since the number of unique last names in the table is simply too great to chart.
8. Close the **USERS** tab by clicking its ✕ icon.

4.8 Summary

In this section, we've seen how to work with the Data Source Explorer view and perform basic operations such as filtering data objects by schema, executing queries, editing data, and so on. We also explored the basics of an Eclipse-based environment to bring you up to speed on perspectives and views, workspaces and projects.

In the next section, we'll focus on working with a new project and developing within it.

5. Working with Data Development Projects

Now that we have a general feel for the basics of IBM Data Studio, it is time to take our skills to the next level and start working on a data development project. Just like regular Eclipse-based projects, data development projects allow you to store files pertinent to your database project including SQL scripts, stored procedures, user-defined functions, XQuery scripts, and more. In this section, we'll focus on creating a project, and developing stored procedures and XML queries using Data Studio. After completing this section, you will be able to:

- ▶ Create a data development project

- ▶ Create and execute a SQL query
- ▶ Create and execute an XML query using XQuery
- ▶ Create, deploy, and execute a SQL stored procedure
- ▶ Create, deploy, and execute a Java stored procedure

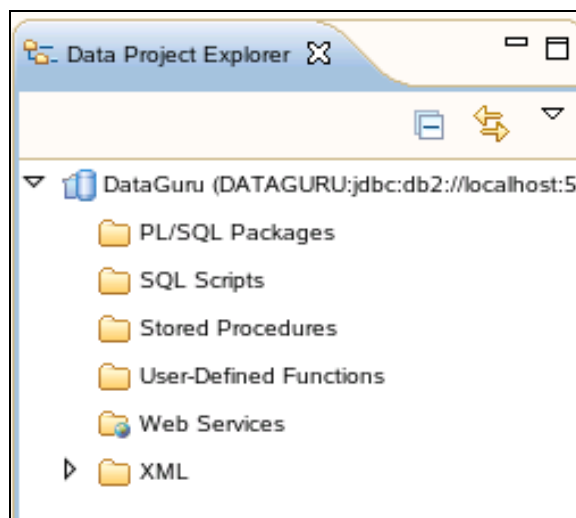
5.1 Creating a Data Development Project

One day while working at DataGuru, Inc., you overhear talk of improvements that need to be made to the DATAGURU database. Expecting your manager to come knocking any minute, you decide to create a data development project to store all work that has been completed for the project.

1. From the main menu, select **File > New > Data Development Project**.
2. In the **Project name** field, enter the value `DataGuru`. Click **Next**.
 - ▶ Select the **DATAGURU** inside the Connections section and click **Finish**.



3. In the **Data Project Explorer** view, expand the **DataGuru** project by clicking the ▶ icon. Notice the empty folders – these are where your project files will be organized according to their types.



5.2 Creating and Executing SQL Scripts

Now that we have a project created, it's time to work with it. In working on a data project, you'll often have the need to write a simple SQL script outside of the context of a stored procedure or user defined function. There are two ways in which SQL scripts can be created in Data Studio: either by typing a query manually or using the Data Studio SQL builder. We'll examine both methods in this section.

5.2.1 Creating an SQL Script Manually

1. In the **Data Project Explorer**, right-click on the **SQL Scripts** folder and select **New > SQL or XQuery Script**.
2. In the Name field, enter the value `ManualQuery`.
3. Select the **SQL and XQuery editor (for scripts that contain one or more SQL and XQuery Statements)** radio button under the **Edit using** heading and click **Finish**. A **ManualQuery.sql** tab will appear in the main view and an icon will appear under the **SQL Scripts** folder in the **Data Project Explorer** view.

4. In the main view, enter the following query to view the names and email addresses of all job seekers that are able to program in Assembly language. You may either type the query, or copy and paste **Query3.sql** from the editor loaded in section 4.2.


```
SELECT      first_name, last_name, email
FROM        users u, user_technologies ut, technologies t
WHERE       u.id = ut.user_id
AND         ut.technology_id = t.id
AND         t.technology = 'Assembler'
AND         u.is_recruiter = 'N'
ORDER BY    first_name, last_name
```

5. From the main menu, select **Run > Run SQL**. The results of the query will appear on the **Results** tab of the **SQL Results** view.
6. Close the **ManualQuery.sql** tab by clicking its icon. Click **Yes** when asked to save changes.

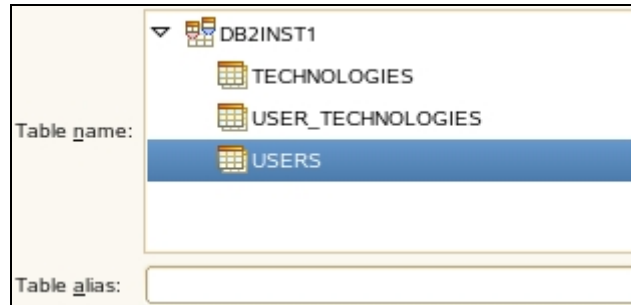
5.2.2 Creating a SQL Script using SQL Builder

It's that easy to create and execute a SQL script manually using Data Studio. However, some might prefer to construct a query visually using the SQL builder provided in Data Studio. Let's take a look at how we could construct and execute the same query as in section 5.2.1 using the SQL builder.

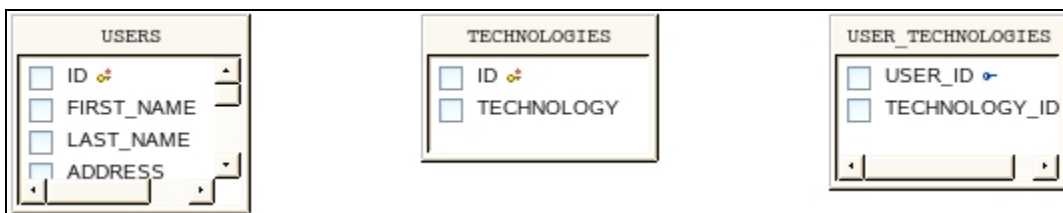
1. Same as in the last section, in the **Data Project Explorer**, right-click on the **SQL Scripts** folder and select **New > SQL or XQuery Script**
2. In the Name field, enter the value `BuilderQuery`.
3. Select the **SQL Query Builder** radio button under the **Edit using** heading, and select **SELECT** in the **Statement type** drop down list.

4. Click **Finish**. A **BuilderQuery.sql** tab will appear in the main view. Additionally, **BuilderQuery.sql** will appear under the **SQL Scripts** folder in the **Data Project Explorer**. Click the  icon to maximize the main view.
5. The main view contains the SQL builder, which is divided into 3 panes. The top pane displays the SQL query as it is being constructed; the middle pane displays the tables present in the query; and the bottom pane shows properties related to the query. Right-click in the **middle pane** and select **Add Table....**

6. Expand the **DB2INST1** schema in the **Add Table** dialog and select **USERS**. Click **OK**.



7. Repeat step 6 for both of the **TECHNOLOGIES** and **USER_TECHNOLOGIES** tables. You should see all three tables in the middle pane.



8. Right-click in a blank area of the **middle pane** and select **Create Join....** Join the **USERS** and **USER_TECHNOLOGIES** tables using the following values:
- ▶ **Source table:** USERS
 - ▶ **Source column:** ID
 - ▶ **Target table:** USER_TECHNOLOGIES
 - ▶ **Target column:** USER_ID
 - ▶ **Join Type:** Inner join

Click **OK**.

Source	
Table (alias)	USERS
Column	ID
Source column type: INTEGER	
Target	
Table (alias)	USER_TECHNOLOGIES
Column	USER_ID
Target column type: INTEGER	
Join type	
<input checked="" type="radio"/> Inner join <input type="radio"/> Left outer join <input type="radio"/> Right outer join <input type="radio"/> Full outer join	

9. Right-click again in the **middle pane** and select **Create Join**. Join the **TECHNOLOGIES** and **USER_TECHNOLOGIES** tables using the following values:

- ▶ **Source table:** TECHNOLOGIES
- ▶ **Source column:** ID
- ▶ **Target table:** USER_TECHNOLOGIES
- ▶ **Target column:** TECHNOLOGY_ID
- ▶ **Join Type:** Inner join

Click **OK**.

10. In the **Columns** tab in the **lower pane**, double-click the first row of the column list. Select **USERS.FIRST_NAME** from the drop down list.
11. Select the **Sort Order** column in the first row, and select **1** from the drop down list. For **Sort Type**, select **Ascending**.



Columns				
Column	Alias	Output	Sort Type	Sort Order
DB2INST1.USERS.FIRST_NAME		<input checked="" type="checkbox"/>	Ascending	1

12. Repeat steps 10 – 11, adding the **USERS.LAST_NAME** and **USERS.EMAIL** columns. Be sure to set the sort order for the **LAST_NAME** column to **2**.

Columns				
Column	Alias	Output	Sort Type	Sort Order
DB2INST1.USERS.FIRST_NAME		<input checked="" type="checkbox"/>	Ascending	1
DB2INST1.USERS.LAST_NAME		<input checked="" type="checkbox"/>	Ascending	2
DB2INST1.USERS.EMAIL		<input checked="" type="checkbox"/>		

13. In the **lower pane**, select the **Conditions** tab and double-click the first row of the conditions list. Select **TECHNOLOGIES.TECHNOLOGY** from the drop down list.
14. Select the **Value** column in the first row and enter the value **Assembler**. Make sure you use the exact upper/lower case combination since it is case sensitive.
15. Select the **AND/OR** column in the first row and select **AND** from the drop down list.
16. Repeat steps 13 – 14, adding a condition that column **USERS.IS_RECRUITER** = **N**. Make sure you use the capital 'N' since it is case sensitive.

Conditions			
Column	Operator	Value	AND/OR
DB2INST1.TECHNOLOGIES.TECHNOLOGY	=	'Assembler'	AND
DB2INST1.USERS.IS_RECRUITER	=	'N'	

17. Restore the main view to its original state by clicking the  icon, and select **Run > Run SQL**. Notice that the results of the query appear in the **Results** tab of the **SQL Results** view.
18. Close the **BuilderQuery.sql** tab by clicking its  icon. Click **Yes** when asked to save changes.

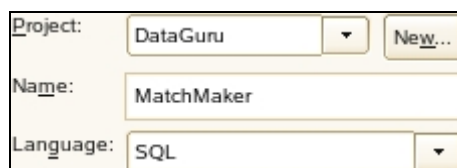
5.3 Creating a SQL Stored Procedure

While profiling the DATAGURU database one day, a certain “matchmaker” query runs quite often against the database. The query is used by recruiters to search for job seekers having skills that they are seeking. It is stored in a CGI script on the DataGuru.ca web server as embedded SQL.

Realizing the potential performance implications of this, you decide to recreate the query as a stored procedure on the database server. Let’s take a look at how you might do that using IBM Data Studio.

1. In the **Data Project Explorer**, right-click on the **Stored Procedures** folder and select **New > Stored Procedure**.
2. In the **New Stored Procedure** dialog that appears, enter the following details:
 - ▶ **Name:** MatchMaker
 - ▶ **Language:** SQL

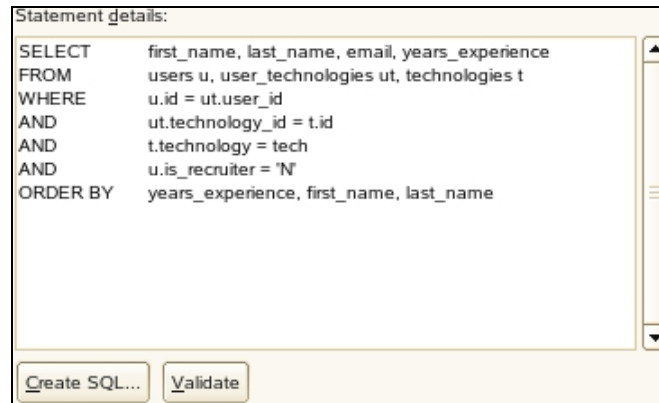
Click **Next**.



3. Highlight the text in the **Statement details** field and replace it either by typing the following query, or copying and pasting **Query4.sql** from the editor loaded in section 4.2.

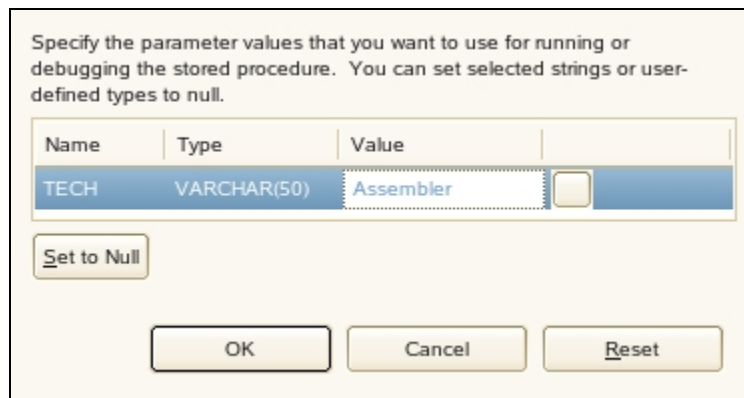
```
SELECT      first_name, last_name, email, years_experience
FROM        users u, user_technologies ut, technologies t
WHERE       u.id = ut.user_id
AND         ut.technology_id = t.id
AND         t.technology = :tech
AND         u.is_recruiter = 'N'
ORDER BY    years_experience, first_name, last_name
```

Click **Validate** to make sure that the statement is using the correct syntax, and then click **Next**.



4. Notice that the `tech` parameter has been added automatically for us as an IN parameter with the data type `VARCHAR(50)`. This is appropriate for our needs, so click **Finish**.
5. The main view will now contain the generated stored procedure. Right-click on the **MATCHMAKER** node under the **Stored Procedures** folder in the **Data Project Explorer** and select **Deploy**.
6. Leave all defaults in the **Deploy Routines** dialog and click **Finish**. The results of the deployment are displayed in the **Status** tab of the **SQL Results** view.
7. Right-click on the **MATCHMAKER** node under the **Stored Procedures** folder in the **Data Project Explorer**, and select **Run**.
8. In the **Specify Parameter Values** dialog, enter the value `Assembler` in the **Value** column. Click **OK**. Make sure you use the exact upper/lower case combination since it is case sensitive.

The results of the query are displayed in the **Results** tab of the **SQL Results** view.

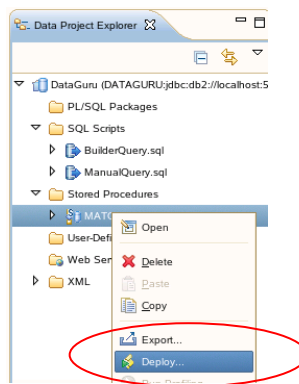



We now have a stored procedure created and deployed to the database server, and we've successfully executed it. But wait a minute – the list of candidates is sorted in ascending order by years of experience. It would probably make more sense to display the most experienced people at the top of the list. Let's change the stored procedure to behave this way.

9. In the **MATCHMAKER** tab in the main view, change the `ORDER BY` line of the query to match the following:

```
ORDER BY U.YEARS_EXPERIENCE DESC, U.FIRST_NAME, U.LAST_NAME;
```

10. Right-click on the **MATCHMAKER** node under the **Stored Procedures** folder in the **Data Project Explorer**, and select **Deploy**. Click **Yes** when asked to save your changes.



11. Leave all defaults in the **Deploy Routines** dialog and click **Finish**. The results of the deployment are displayed in the **Status** tab of the **SQL Results** view.
12. Rerun the stored procedure by repeating steps 7 – 8, and notice that the results of the query are now displayed in the proper order.
13. Close the **MATCHMAKER** tab by clicking its  icon.

5.4 Creating a Java Stored Procedure

The president of DataGuru, Inc. has recently become extremely fascinated with astrology and horoscopes. One morning, he comes to you and insists that the `USERS` table contain everyone's sun sign, as he believes it crucial for a recruiter to make a proper hiring decision. Reluctantly, you comply, but there's just one problem – there are already 5000 users in the table, none of which have astrological information.

Fortunately, you've developed in Java before and decide to write a Java routine to calculate all users' sun signs based on their birth dates. You decide to implement the routine as a stored procedure so that if you ever need to update sign information again, you can just rerun the procedure. Let's take a look at how you might do that.


5.4.1 Creating a Table

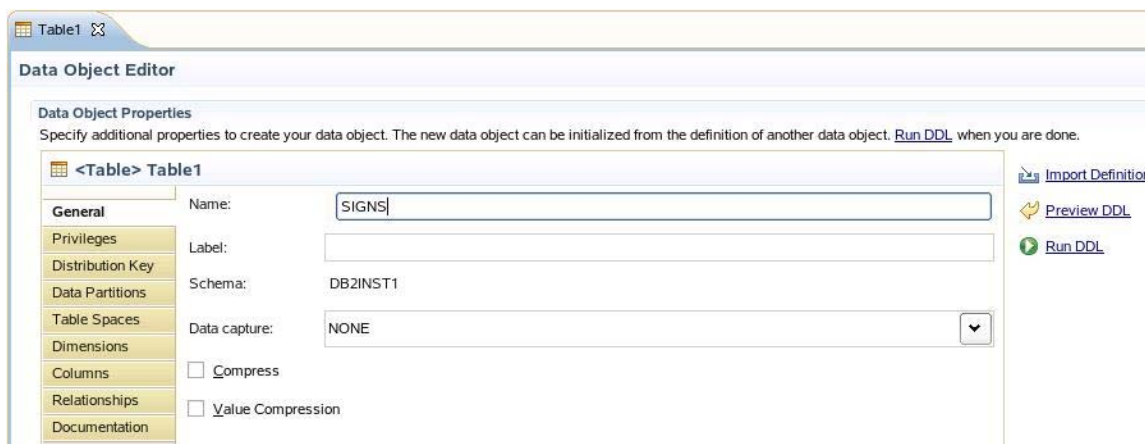
Before we can get down to developing the stored procedure, we are going to need a table to store the names of each sign, along with the dates on which they begin and end. We will store all the dates with the year 2007, for simplicity. The SIGNS table will have the following schema:


Column	Type
id	INTEGER
sign	VARCHAR(12)
start_date	DATE
end_date	DATE

Table 5-1 – SIGNS table

Follow the steps below to create the SIGNS table using the Data Source Explorer.

1. In the **Data Source Explorer**, expand the nodes **DATAGURU > Schemas [Filtered] > DB2INST1**.
2. Right-click on the **Tables** folder and select **Create > Table**.
3. In the main view, click the  icon to maximize the **Data Object Editor**.
4. On the **General** tab of the **Data Object Editor**, enter `SIGNS` in the **Name** field.




5. Select the **Columns** tab and click the  icon to add a new column.
6. Enter the following information for the first column:
 - ▶ **Name:** ID
 - ▶ **Primary Key:** Yes (checked)
 - ▶ **Data Type:** INTEGER
 - ▶ **Not Null:** Yes (checked)

<Table> Table1

General Privileges Distribution Key Data Partitions Table Spaces Dimensions **Columns** Relationships Documentation

Name	Primary Key	Data Type	Length	Scale	Not Null	Generated	Default Value
ID	<input checked="" type="checkbox"/>	INTEGER			<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Import Definition Preview DDL Run DDL

7. Click the  icon to add another column. Enter the following information for the second column:
 - ▶ **Name:** SIGN
 - ▶ **Data Type:** VARCHAR
 - ▶ **Length:** 12
 - ▶ **Not Null:** Yes (checked)
8. Add two more columns named `START_DATE` and `END_DATE`, both of type `DATE`, and set them to **Not Null**.

<Table> Table1

General Privileges Distribution Key Data Partitions Table Spaces Dimensions **Columns** Relationships Documentation

Name	Primary Key	Data Type	Length	Scale	Not Null	Generated	Default Value
ID	<input checked="" type="checkbox"/>	INTEGER			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
SIGN	<input type="checkbox"/>	VARCHAR	12		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
START_DATE	<input type="checkbox"/>	DATE			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
END_DATE	<input type="checkbox"/>	DATE			<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Import Definition Preview DDL Run DDL

9. Click the **Preview DDL** link on the right side of the **Data Object Editor**. Notice that the **DDL** pane displays the Data Definition Language statements that will be generated by the choices you have made.

DDL


DDL statements from the Data Object Properties and Impacted Object sections can be [Previewed](#) here. Alternatively, they can be [Opened](#) with the SQL editor for further editing.

```
CREATE TABLE "DB2INST1"."SIGNS" (
  "ID" INTEGER NOT NULL,
  "SIGN" VARCHAR(5) NOT NULL,
  "START_DATE" DATE NOT NULL,
  "END_DATE" DATE NOT NULL
)
DATA CAPTURE NONE !
ALTER TABLE "DB2INST1"."SIGNS" ADD CONSTRAINT "SIGNS_PK" PRIMARY KEY
("ID")!
```

Preview DDL Open with SQL editor Run DDL

10. Click the **Run DDL** link on the right side of the **Data Object Editor**.



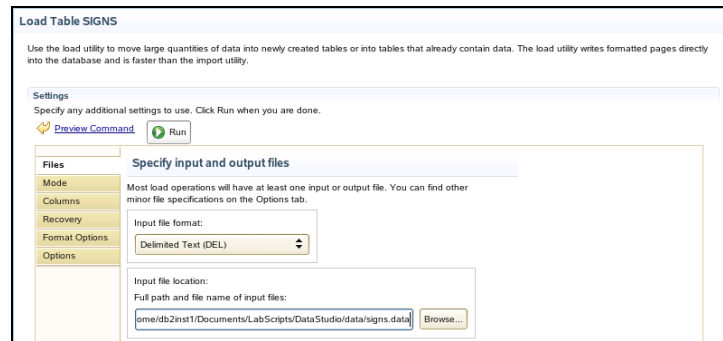
11. Close the **SIGNS** tab in the main view by clicking its  icon. The **Status** tab of the **SQL Results** view shows the output of the table creation.

5.4.2 Loading a Table

Now that we have the SIGNS table, we will need to store some data in it. With Data Studio, loading delimited data is a painless process, as we will see below.

1. In the **Data Source Explorer**, right-click on the **SIGNS** table and select **Data > Load**.
2. You can either navigate to the path by clicking **Browse...** or entering it manually in the **Input File** field of the **Load Data** dialog:

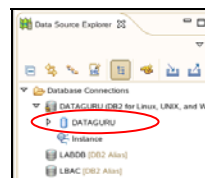
`/home/db2inst1/Documents/LabScripts/DataStudio/data/signs.data`



3. Leave all other defaults in the **Load Data** dialog and click **Run**. The **Messages** tab in the **SQL Results** view should display a message indicating that 12 rows were loaded.



4. After the load it is always better to refresh the database. Right-click on the **DATAGURU** artifact and select **Refresh**.



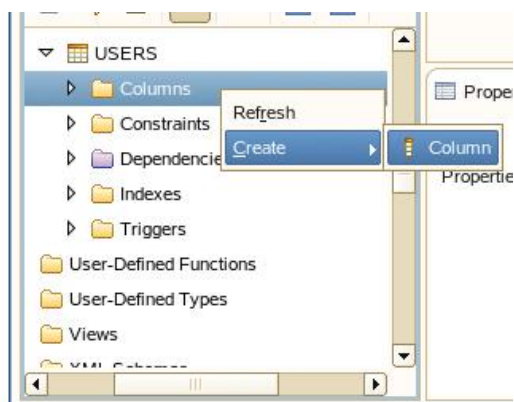
5. Navigate to the tables Schemas **[Filtered] > DB2INST1 > Tables**. Right-click on the **SIGNS** table in the **Data Source Explorer** and select **Data > Sample Contents**. You should now see data from the SIGNS table in the **Results** tab of the **SQL Results** view.

5.4.3 Altering a Table

We still have one thing left to do before we can start developing our stored procedure. We now have the SIGNS table and have populated it with data, but we have no way to relate the USERS table to it. As such, we are going to need to add a SIGN_ID column in the USERS table to act as a foreign key. Our new schema for the USERS table will be as follows:

Column	Type
id	INTEGER
first_name	VARCHAR(20)
last_name	VARCHAR(20)
address	XML
email	VARCHAR(50)
birth_date	DATE
employer	VARCHAR(50)
username	CHAR(8)
password	CHAR(8)
is_recruiter	CHAR(1)
years_experience	INTEGER
sign_id	INTEGER

1. Expand the **USERS** table menu by clicking on the ► icon in the **Data Source Explorer**. Right click on the **Columns** and select **Create > Column**. Click the □ icon to maximize the **Data Object Editor** in the main view.



2. In the **Columns** tab enter `SIGN_ID` as the name of the column.

Data Object Editor

Data Object Properties
Specify additional properties to create your data object. The new data object can be initialized from the definition of another data object. [Run DDL](#) when you are done.

<Column> Column1 [CHAR(5) Nullable]

General Name: [Import Definition](#)

Type Label: [Preview DDL](#)

Documentation [Run DDL](#)

3. Select the **Type** tab on the left and enter `INTEGER` as the **Data type**.

Data Object Editor

Data Object Properties
Specify additional properties to create your data object. The new data object can be initialized from the definition of another data object. [Run DDL](#) when you are done.

<Column> Column1 [CHAR(5) Nullable]

General Data type: [Import Definition](#)

Type Precision: [Preview DDL](#)

Scale: [Run DDL](#)

Default value:

Properties: ☐ Primary Key ☐

4. Click the **Run DDL** link on the right side of the **Data Object Editor**.
5. The results of the DDL execution should be displayed in the **Messages** tab of the **SQL Results** view.

5.4.4 Creating the Stored Procedure

We now have everything we need in order to develop the stored procedure. In a nutshell, the procedure will iterate through each user and check his or her birth date to see if it is included in the `START_DATE` and `END_DATE` ranges of each entry in the `SIGNS` table. Once the appropriate sign is found, the procedure will update that user's record in the `USERS` table with the appropriate sign ID.

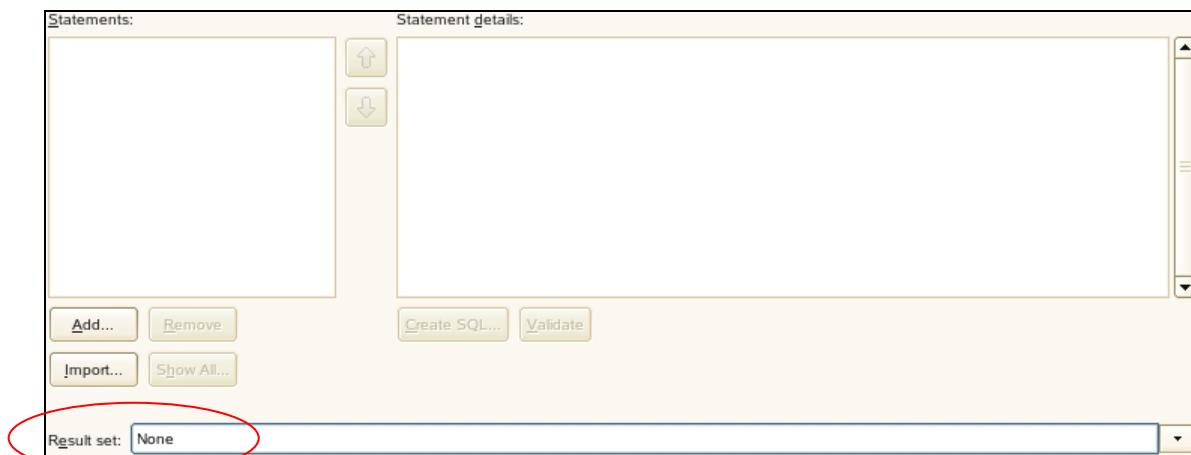
The careful reader may note that each date stored in the `SIGNS` table had the year 2007 for simplicity. To compensate for this, the stored procedure takes each user's birth date and replaces his or her birth year with 2007 in order to perform the comparison. If this makes no sense to you whatsoever, don't worry – the details of the code in the stored procedure are much less important than understanding how easy it is to create a Java stored procedure using Data Studio. Without further ado, let's get to it.

1. In the **Data Project Explorer**, right-click on the **Stored Procedures** folder and select **New > Stored Procedure**.
2. In the **New Stored Procedure** dialog that appears, enter the following details:
 - ▶ **Project:** DataGuru
 - ▶ **Name:** UpdateSigns
 - ▶ **Language:** Java
 - ▶ **Java package:** db2inst1

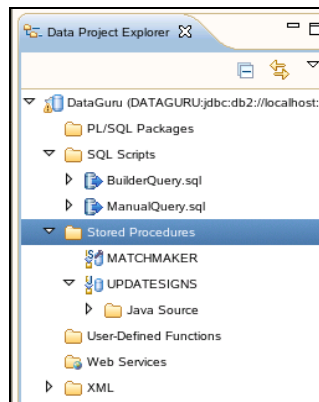
Click **Next**.

The screenshot shows the 'New Stored Procedure' dialog box. The 'Project' field is set to 'DataGuru'. The 'Name' field is 'UpdateSigns'. The 'Language' dropdown is set to 'Java'. Under the 'Java options' section, the 'Java package' is 'db2inst1'. The 'Dynamic SQL using JDBC' radio button is selected, while 'Static SQL using SQLJ' is unselected. Below this, the 'Root package' is 'S941161'. The 'SQLJ translator location' is '/home/db2inst1/sqlib/java/sqlj.zip', with a 'Browse...' button to its right. The 'SQLJ translator class name' is 'sqlj.tools.Sqlj'.

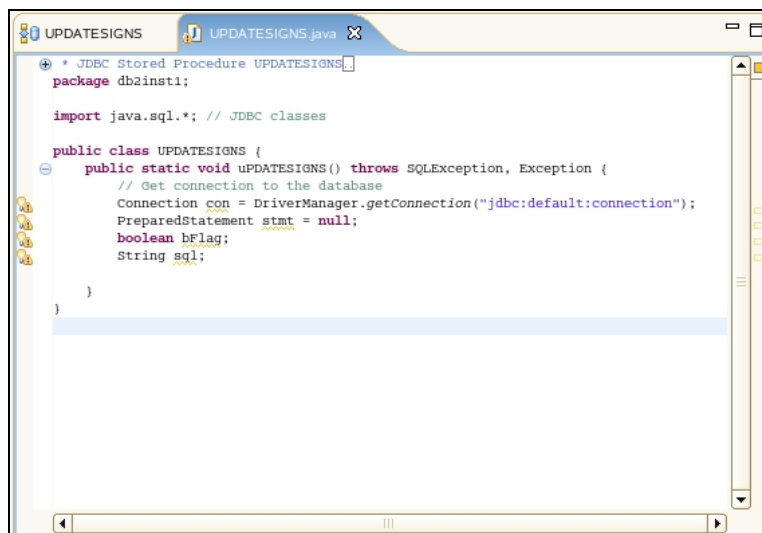
3. Click the **Remove** button to delete the sample statement.
4. Select **None** in the **Result set** drop down list. Click **Finish**.






5. In the **Data Project Explorer**, expand the following nodes:
DataGuru > Stored Procedures > UPDATESIGNS > Java Source



6. Double-click on the **UPDATESIGNS.java** node to open the source code of the stored procedure. Notice that a skeleton has been created for you.



7. Since we have already coded the stored procedure for you, delete all text in the **UPDATESIGNS.java** tab. To do this, press **Ctrl+A** to select all text, and then press the **DELETE** key.
8. From the main menu, select **File > Open File....** In the location field, enter the following path:

```
/home/db2inst1/Documents/LabScripts/DataStudio/java/sp.java
```
9. Click **OK**. In the **sp.java** tab that appears in the main view, press **Ctrl+A** to select all text, and press **Ctrl+X** to cut it.
10. Switch to the **UPDATESIGNS.java** tab in the main view. Make sure all existing code is deleted and press **Ctrl+V** to paste the text from the clipboard.
11. Close the **sp.java** tab by clicking its  icon. Click **No** when asked to save changes.
12. The **UPDATESIGNS.java** file is now complete with all the code we need to create the stored procedure. It is time to deploy and test the stored procedure.
13. Close the **UPDATESIGNS.java** tab by clicking its  icon. Click **Yes** when asked to save changes.
14. In the **Data Project Explorer**, right-click on the **UPDATESIGNS** node under **Stored Procedures** and select **Deploy**.
15. Click **Finish** and wait until you see the message *Succeeded* in the **SQL Results** view.
16. In the **Data Project Explorer**, right-click on the **UPDATESIGNS** node under **Stored Procedures** and select **Run**. Wait until you see the message *Run completed* in the **Status** tab of the **SQL Results** view.
17. You can now use either the command line or Data Studio to view these changes to the **USERS** table and note that each user now has a sign ID in his or her record.
18. Close the **UPDATESIGNS** tab in the main view by clicking its  icon.

5.5 Creating an XQuery Script

Just as Data Studio allows you to create and execute SQL scripts, it allows you to create and execute scripts that query XML data with XQuery as well. In this section, we'll take a look at how to query XML data with XQuery.

You may recall that the **ADDRESSES** column in the **USERS** table is of type XML. In fact, each user's address is stored in this format:


```
<addressinfo>
  <city>CityName</city>
  <prov-state>ProvinceName</prov-state>
  <country>CountryName</country>
</addressinfo>
```


Standard SQL statements alone can only return the full contents of this column in XML format. Fortunately, XQuery allows us to work with XML data and retrieve individual elements from an XML document. We'll see how to do that in the examples that follow.

5.5.1 Returning an XML Document

Our first query will return an XML document that groups each city into its respective province or state. The output will be similar to the following:

```
<prov-state name="Alaska">
  <city>Chicken</city>
  <city>Kenai</city>
  :
</prov-state>
```

1. In the **Data Project Explorer**, right click on the **SQL Scripts** folder and select **New > SQL or XQuery Script**.
2. In the name field, enter the value `CityQuery`.
3. Select the **SQL and XQuery editor** radio button under the **Edit using** heading, and click **Finish**.

4. Click the  icon to maximize the main view and either type the following query, or copy and paste **Query5.sql** from the editor loaded in section 4.2.

```
XQUERY
for $provstate in fn:distinct-values(db2-fn:xmlcolumn('USERS.ADDRESS')
  /addressinfo/prov-state)

  let $cities := db2-fn:xmlcolumn('USERS.ADDRESS')
    /addressinfo/city[../prov-state = $provstate]

  order by $provstate
  return <prov-state name="{ $provstate }">{ $cities }</prov-state>
```

- From the main menu, select **Script > Run SQL**. The query may take a few moments while the grouping is performed. The results of the query, in XML format will be displayed in the **Results** tab of the **SQL Results** view.

Status	Result1
1	<prov-state name='#n/a'><city>Lower Peach Tree</city><city>F
2	<prov-state name='Alaska'><city>Chicken</city><city>Kenai</c
3	<prov-state name='Alberta'><city>Spruce Grove</city><city>Bl
4	<prov-state name='Arizona'><city>Poston</city><city>Gila Ben
5	<prov-state name='Arkansas'><city>Clarkedale</city><city>Hen
6	<prov-state name='British Columbia'><city>Mission</city><cit
7	<prov-state name='California'><city>Forest Ranch</city><city
8	<prov-state name='Colorado'><city>Boulder</city><city>Greene

- Close the **CityQuery.sql** tab by clicking its icon. Click **Yes** when asked to save changes.

5.5.2 Combining XML and Relational Data

It is often convenient to return data from within an XML column in a traditional relational dataset. For example, suppose you want to look into the demographics of DataGuru.ca to find out the top ten provinces and states by number of users on the site. As we know, the provinces and states themselves are stored in the ADDRESS column of the USERS table. In order to get a count of the users in each province and state, however, it is convenient to use an SQL query. Let's take a look at one way to do just that.

- In the **Data Project Explorer**, right click on the **SQL Scripts** folder and select **New > SQL or XQuery Script**.
- In the name field, enter the value **DemographicsQuery**.
- Select the **SQL and XQuery editor** radio button under the **Edit using** heading, and click **Finish**.
- Click the icon to maximize the main view and either type the following query, or copy and paste **Query6.sql** from the editor loaded in section 4.2:

```
SELECT      x.provstate,
            x.country,
            COUNT(*) num_users
FROM        users u,
            XMLTABLE(' $addr/addressinfo' passing U.address as "addr"
                     COLUMNS
                        "PROVSTATE"  VARCHAR(50)   PATH 'prov-state/text()',
                        "COUNTRY"     VARCHAR(50)   PATH 'country/text()'
            ) x
```

```

GROUP BY      x.provstate,
              x.country


ORDER BY      num_users DESC

FETCH FIRST 10 ROWS ONLY;

```

- From the main menu, select **Script > Run SQL**. The results of the query, in tabular format, will be displayed in the **Results** tab of the **SQL Results** view.

Status	Result1
	PROVSTATE COUNTRY NUM_USERS
1	Pennsylvania USA 271
2	Texas USA 239
3	New York USA 213
4	Illinois USA 195
5	Quebec Canada 184
6	New Brunswick Canada 171
7	Ohio USA 154
8	California USA 145
9	Missouri USA 143
10	Ontario Canada 135

- Close the **DemographicsQuery.sql** tab by clicking its  icon. Click **Yes** when prompted to save changes.

5.6 Clean Up

To clean your environment after completing the exercise, if you would like to redo this lab in the future, please execute the following in a terminal window:

```

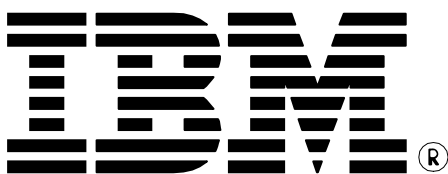
su - root
cd /home/db2inst1/Documents/LabScripts/LBAC
make clean
make install
exit

```

Enter the password “**password**” when prompted.

6. Summary

You can hopefully see by now that IBM Data Studio is a highly productive environment for DB2 development. Over the course of this lab, we’ve seen how fast and easy it is to create and execute SQL and XQuery scripts; develop and test stored procedures in SQL and Java; create and alter database objects; analyze query execution; and graph trends in data.



© Copyright IBM Corporation 2009
All Rights Reserved.

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

Printed in Canada
07/02/2009

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both

UNIX is a registered trademark of The Open Group in the United States, other countries, or both

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

VMware is a trademark of VMware Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of July 2009, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.