Microsoft Power Platform

Power BI Paginated Reports in a Day

Lab 04A - January 2021 release

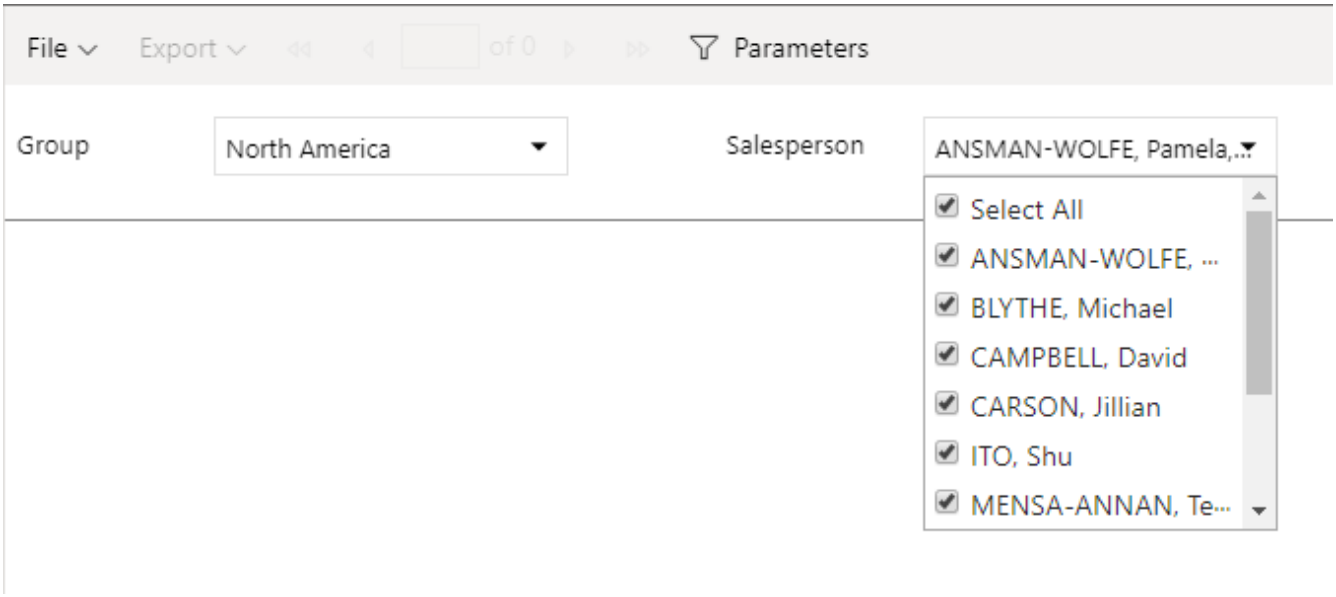
Work with Parameters

Overview

The estimated time to complete this lab is 45 minutes.

In this lab, you will enhance the **Salesperson Directory** report developed in **Lab 03A** by adding parameters.

The cascading parameters will look like the following:



Exercise 1: Work with parameters

In this exercise, you will add and configure report parameters to explore different usage scenarios.

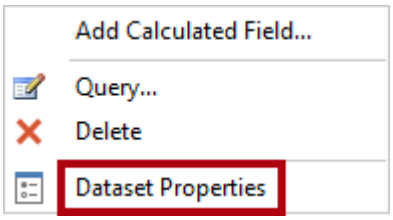
Important: There are many repetitive tasks when developing reports. The labs in this course will progressively reduce the detailed step-by-step instructions when detailed steps have already been provided.

Task 1: Add a query parameter

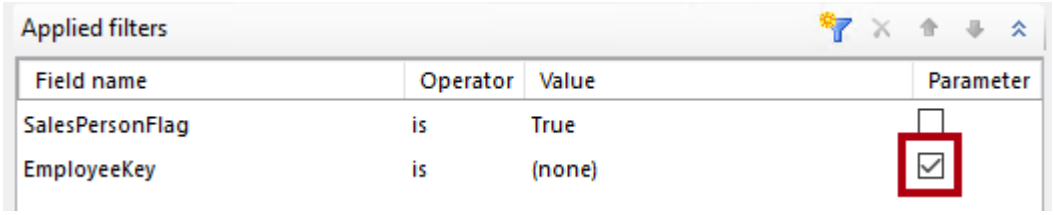
In this task, you will add a query parameter to the **dsMain** dataset of the **Salesperson Directory** report.

*If you didn't successfully complete **Lab 03A** to create the **Salesperson Directory** report, you can open the solution report found in the **<CourseFolder>\PowerBIPRIAD\Lab03A\Solution** folder.*

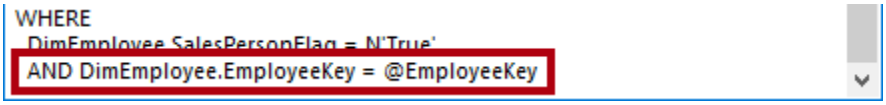
1. In Report Builder, verify that the **Salesperson Directory** report is open from the previous lab.
2. In the **Report Data** pane, right-click the **dsMain** dataset, and then select **Dataset Properties**.



3. In the **Dataset Properties** window, click **Query Designer**.
4. In the **Query Designer** window, in the **Applied Filters** pane, add a new filter for the **EmployeeKey** column.
5. For the **EmployeeKey** filter, check the **Parameter** checkbox.

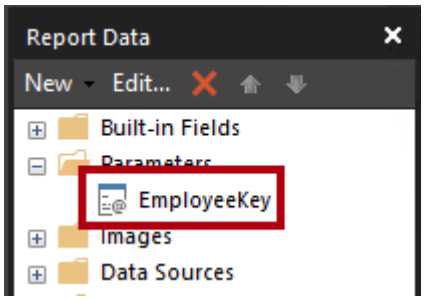


6. Click **OK**.
7. In the **Dataset Properties** window, in the **Query** box, scroll down to reveal the parameterized WHERE clause.



For SQL Server database products, query parameters are prefixed with the at symbol (@). At query execution time, a parameter value will be substituted into the query parameter.

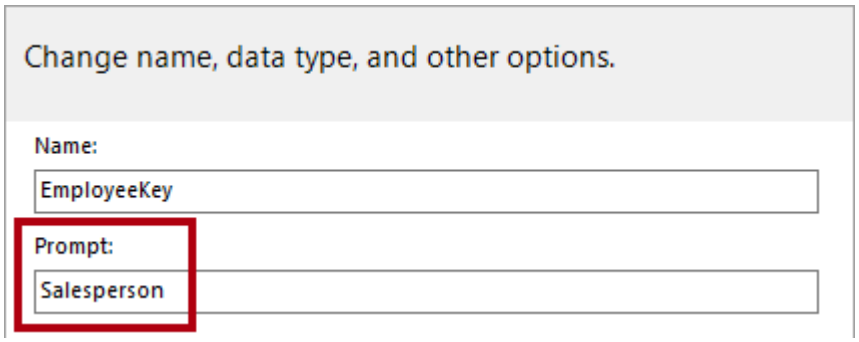
8. Click **OK**.
9. In the **Report Data** pane, expand the **Parameters** folder.
10. Notice the **EmployeeKey** report parameter.



Each query parameter added to a query creates a report parameter. Commonly, report parameters are used to prompt the report user for values, and these values are then mapped to the dataset query parameters.

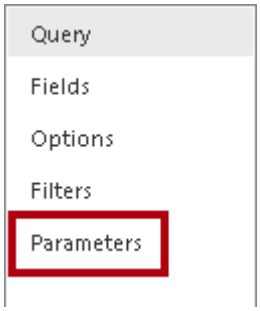
11. In the **Parameters** pane (located above the report designer), notice the **Employee Key** report parameter.

12. In the **Report Data** pane, right-click the **EmployeeKey** report parameter, and then select **Parameter Properties**.
13. In the **Report Parameter Properties** window, in the **Prompt** box, replace the text with **Salesperson**.



14. Click **OK**.
15. In the **Parameters** pane, notice that the report parameter prompt has updated.

16. Open the **dsMain** dataset properties, and then select the **Parameters** page.



17. Notice the mapping of query parameter to report parameter.

    The at symbol (@) used in the shorthand expression denotes that the item is a report parameter.

18. Open the **Expression** window for the **Parameter Value**, and then review the expression.

    *The expression returns the value of the **EmployeeKey** report parameter. So, effectively the dataset query parameter receives the report parameter value entered by the report user.*

19. Close the **Expression** window, and the **Dataset Properties** window.
20. In the page header, right-click the **Subtitle** textbox, and then select **Expression**.
21. In the **Expression** window, in the **Category** list, select **Parameters**.
22. In the **Values** list, double-click the **EmployeeKey** report parameter.
23. Click **OK**.
24. Preview the report (click **Run** on the **Home** ribbon tab).
25. In the **Salesperson** parameter box, enter **272**.
26. Click **View Report** (located at the right of the parameter area).

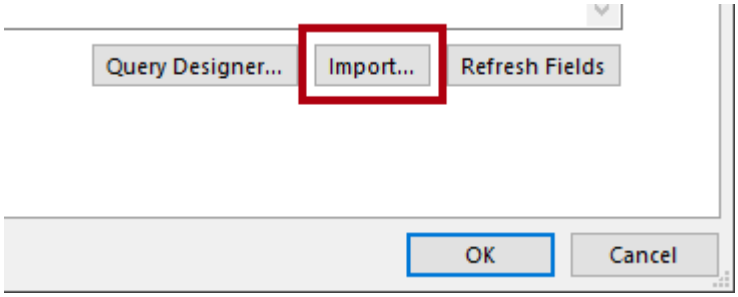
27. Review the rendered report, for the single salesperson named Stephen Jiang.

    It's likely improbably that a report user would know the employee key values required to filter the report. In the next task, you will configure the report parameter to present a dropdown list of salespeople and configure the report subtitle to display the salesperson's name.

Task 2: Configure available values

In this task, you will add a dataset to retrieve a list of salespeople, and then configure the **EmployeeKey** report parameter to present available values based on the dataset.

1. Switch to the report designer.
2. In the **Report Data** pane, right-click the **AdventureWorksDW** data source, and then select **Add Dataset**.
3. In the **Dataset Properties** window, in the **Name** box, replace the text with **dsSalesperson**.
4. To retrieve a prepared query, click **Import**.



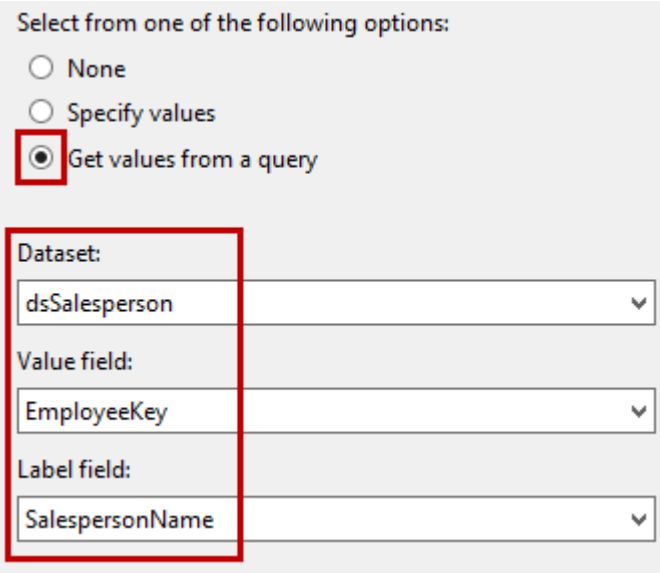
5. In the **Import Query** window, navigate to the **<CourseFolder>\PowerBIPRIAD\Lab04A\Assets** folder.
6. Select the **dsSalesperson_1.sql** file, and then click **Open**.
7. In the **Query** box, review the query statement.

   *The query retrieves all salespeople and returns two columns: the **EmployeeKey** column, and the full name of the salesperson. The query result is sorted in ascending order of the salesperson name.*

8. Click **OK**.
9. Open the **EmployeeKey** report parameter properties, and then select the **Available Values** page.

10. Set the following available value properties:

    - Option: **Get values from a query**
    - Dataset: **dsSalesperson**
    - Value field: **EmployeeKey**
    - Label field: **SalespersonName**



    *The **Value** field is typically assigned a database key value, while the **Label** field is assigned a user-friendly text value. It's possible to use the same field for the **Value** and **Label** fields.*

11. Click **OK**.
12. Modify the expression for the **Subtitle** text box, using the following expression:

VB.NET

```
=Parameters!EmployeeKey.Label
```

13. Preview the report.
14. In the **Salesperson** parameter dropdown list, select **ABBAS, Syed**, and then click **View Report**.
15. Review the rendered report, and notice the improved page header subtitle text.

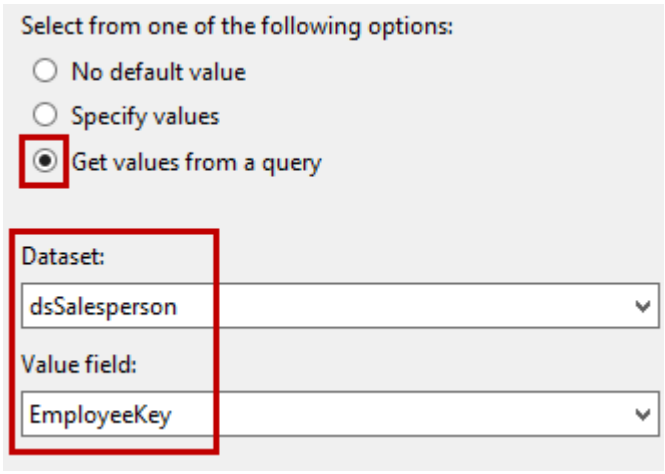
    If a report defines report parameters, it can't be run until all values have been entered. To simplify interacting with the report—or to ensure the report runs immediately upon request—default values can be assigned to report parameters.

Task 3: Configure default values

In this task, you will configure a default value for **EmployeeKey** report parameter.

1. Switch to the report designer.
2. Open the **EmployeeKey** report parameter properties, and then select the **Default Values** page.

3. Set the following default value properties:
   - Option: **Get values from a query**
   - Dataset: **dsSalesperson**
   - Value field: **EmployeeKey**



   Default values can be based on specified values—either constants or expressions—or by dataset query results. When multiple rows are returned by a dataset and the report parameter is not configured to allow multiple values, the first row of the query result is used as the default value.

4. Click **OK**.
5. Preview the report.
6. Notice that the **Salesperson** parameter has been set, and that the report ran automatically.

   The report now displays only a single salesperson. The enhancements you will make in the next task will allow report users to select all salespeople or just a single salesperson.

Task 4: Configure an All item

In this task, you will modify the report datasets to allow selecting all salespeople.

1. Switch to the report designer.
2. Open the **dsSalesperson** dataset properties, and then click **Import**.
3. In the **Import Query** window, select the **dsSalesperson_2.sql** file, and then click **Open**.
4. In the **Query** box, review the query statement that has been modified to become a union query.

   *The first SELECT statement retrieves an "artificial" row that has an **EmployeeKey** value of **-1**, and a caption within parentheses. It returns a key value that doesn't exist in the database. It also defines a label that will appear first when the rows are sorted by the label values. The first SELECT statement is merged with the second, which is the original statement.*

5. Click **OK**.
6. Open the **dsMain** dataset, and import the **dsMain_1.sql** file.
7. In the **Query** box, review the query statement that has a modified WHERE clause.

   *The WHERE clause returns a single employee, or all employees when the **@EmployeeKey** parameter value is **-1**.*

   *This query may not be very efficient as it likely forces a table scan. For large tables, you should avoid this type of predicate by defining the query statement in a stored procedure. The stored procedure logic could branch to different query statements based on the **@EmployeeKey** parameter value.*

8. Click **OK**.
9. Preview the report.
10. Notice that the **Salesperson** parameter now defaults to **(All Salespeople)**, and that all salespeople are retrieved.

    *The report parameter continues to default to the first row of the **dsSalesperson** query, which is now the **(All Salespeople)** row.*

11. Modify the **Salesperson** report parameter to **ABBAS, Syed**, and then click **View Report**.

    When available value lists grow to large sizes, they become impractical and inefficient to present as a single list. To reduce the list size, it's possible to introduce additional parameters that use a cascading behavior to filter other parameter values.

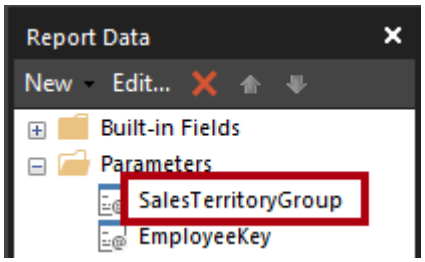
Task 5: Configure cascading parameters

In this task, you will introduce a report parameter to filter the **EmployeeKey** report parameter available values by a sales territory group selected by the report user.

1. Switch to the report designer.
2. Open the **dsSalesperson** dataset properties, and then click **Import**.
3. In the **Import Query** window, select the **dsSalesperson_3.sql** file, and then click **Open**.
4. In the **Query** box, review the query statement and notice the **@SalesTerritoryGroup** query parameter added to the third last line.

   The addition of the query parameter requires the sales territory group value be passed to the query.

5. Click **OK**.
6. In the **Report Data** pane, notice the addition of the **SalesTerritoryGroup** report parameter.

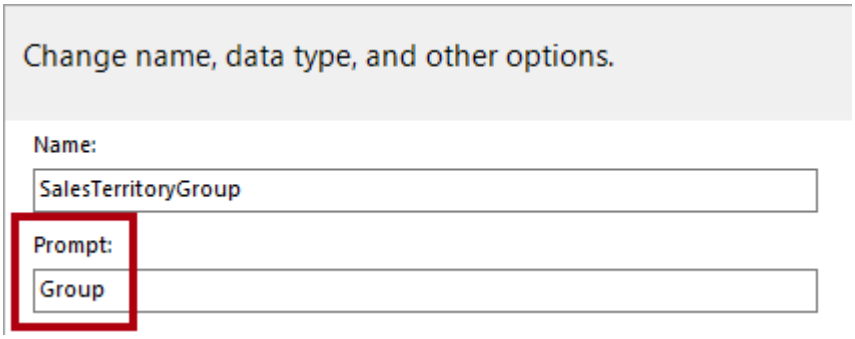


7. To create a new dataset, right-click the **AdventureWorksDW** data source, and then select **Add Dataset**.
8. In the **Dataset Properties** window, in the **Name** box, replace the text with **dsSalesTerritoryGroup**.
9. To retrieve a prepared query, click **Import**.
10. In the **Import Query** window, select the **dsSalesTerritoryGroup.sql** file, and then click **Open**.
11. In the **Query** box, review the query statement.

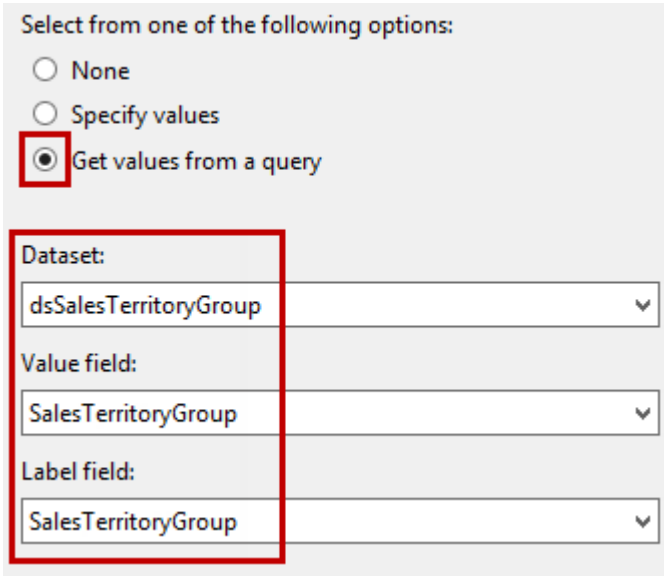
    The query retrieves the distinct sales territory group values.

12. Click **OK**.

13. Configure the **SalesTerritoryGroup** report parameter to prompt the report user for **Group**.



14. Set the following available values properties:
    - Use the **Get Values From a Query** option
    - Set the **Dataset** to **dsSalesTerritoryGroup**
    - Set the **Value Field** to **SalesTerritoryGroup**
    - Set the **Label Field** to **SalesTerritoryGroup**



15. Click **OK**.
16. Modify the **dsMain** dataset query by importing the **dsMain_2.sql** file.

    The WHERE clause also filters the query result using the ***@SalesTerritoryGroup*** *parameter.*

17. Modify the report subtitle text box to use the following expression:

    For convenience, the expression can be copied from the ***<CourseFolder>\PowerBIPRIAD\Lab04A\Assets\Snippets.txt*** *file.*

    VB.NET

    ```
    =Parameters!SalesTerritoryGroup.Value & Iif(Parameters!EmployeeKey.Value = -1, "",
    " - " & Parameters!EmployeeKey.Label)
    ```

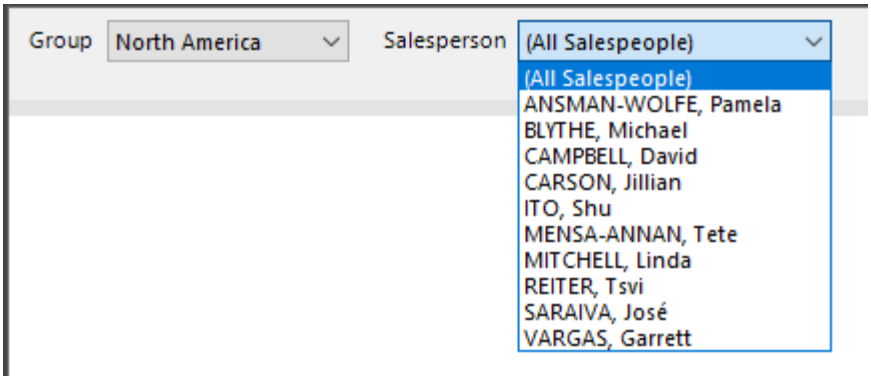
    The expression returns the sales territory group, and then appends the selected salesperson when the "all salespeople" item isn't selected.

18. Preview the report.

19. Notice that the **Salesperson** report parameter is disabled.

    The ***Salesperson*** *report parameter available values cannot be retrieved until a* ***Group*** *report parameter value is selected.*

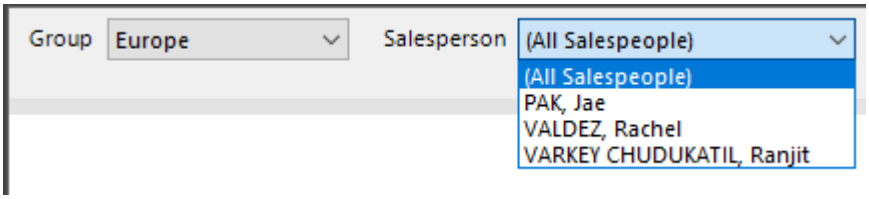
20. In the **Group** parameter dropdown list, select **North America**.



21. In the **Salesperson** parameter dropdown list, notice that 10 salespeople are listed.

22. In the **Group** parameter dropdown list, select **Europe**.

23. In the **Salesperson** parameter dropdown list, notice that three salespeople are listed.



24. Click **View Report**.

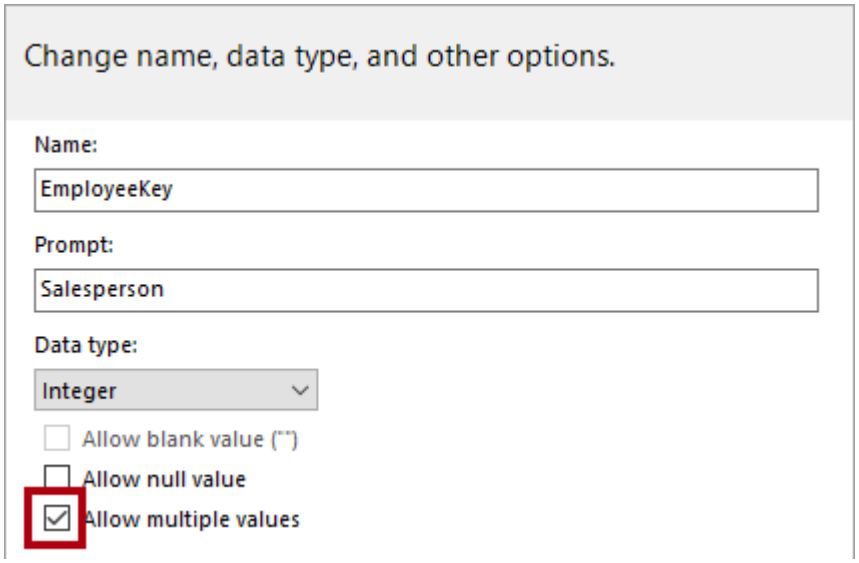
25. Review the rendered report.

    The final enhancement made to the report parameters will be to enable the multi-selection of parameter values.

Task 6: Configure multi-value parameters

In this task, you will enable multi-value selection for the **EmployeeKey** report parameter.

1. Switch to the report designer.
2. Open the **EmployeeKey** report parameter properties, and then check **Allow Multiple Values**.



3. Click **OK**.
4. Preview the report.
5. In the **Group** parameter dropdown list, select **North America**.
6. In the **Salesperson** parameter dropdown list, notice that a multi-select parameter will automatically include a **(Select All)** item.

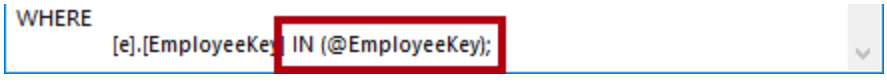
   *All parameter values are selected because the default values are based on all rows of the **dsSalesperson** dataset.*

   Don't view the report, because it will generate an error. You will return to the report designer to continue configuring the parameters.

7. Switch to the report designer.
8. Modify the **dsSalesperson** dataset properties by importing the **dsSalesperson_4.sql** file.

   The query statement no longer includes the "artificial" (All Salespeople) row.

9. Modify the **dsMain** dataset query by importing the **dsMain_3.sql** file.



   *The WHERE clause now filters by the **EmployeeKey** column using the IN operator. The IN operator allows passing a comma-delimited list of values, and Power BI will pass multiple employee key values as a string of comma-separated values.*

10. Modify the report subtitle text box to use the following expression:

    *For convenience, the expression can be copied from the **<CourseFolder>\PowerBIPRIAD\Lab04A\Assets\Snippets.txt** file.*

    VB.NET

    ```
    =Parameters!SalesTerritoryGroup.Value  & Iif(Parameters!EmployeeKey.Count =
    CountRows("dsSalesperson"), "", " - " & Join(Parameters!EmployeeKey.Label, ", "))
    ```

    *The expression also uses the **Join** function to produce a single string of selected values, this time using the **Label** property of the report parameter, and a delimiter value which includes a space. The conditional logic tests the count of selected values, and if it matches the count of rows in the **dsSalesperson** dataset, it doesn't output the delimited list.*

11. Preview the report.
12. In the **Group** parameter dropdown list, select **Europe**.
13. In the **Salesperson** parameter, notice all items are selected—do not select an item.
14. Click **View Report**.
15. Notice that the page header subtitle simply displays the sales territory group name.
16. In the **Salesperson** parameter dropdown list, de-select one salesperson.
17. Click **View Report**.

    Notice that the page header subtitle displays the sales territory group name, and a comma-separated list of the selected salespeople.

Task 7: Publish the report

In this task, you will publish the report to your Power BI workspace.

1. Switch to the report designer.
2. Publish the report to your workspace, overwriting the previously published report.
3. Close Power BI Report Builder.

    *The development of the **Salesperson Directory** report is now complete.*

Summary

In this lab, you enhanced the **Salesperson Directory** report developed in **Lab 03A** by adding parameters.

Terms of use

By using this hands-on lab, you agree to the following terms:

The technology/functionality described in this hands-on lab is provided by Microsoft Corporation in a "sandbox" testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the hands-on lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. Without written permission, you may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this hands-on lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION WITHOUT WRITTEN PERMISSION IS EXPRESSLY PROHIBITED.

THIS HANDS-ON LAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCITONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK If you give feedback about the technology features, functionality and/or concepts described in this hands-on lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

DISCLAIMER This lab contains only a portion of new features and enhancements in Microsoft Power BI. Some of the features might change in future releases of the product.