Enterprise Edition Only

Big Data Integration Workshop

Contents

VM Login and Setup

Linux OS Login

- See document on Desktop in Docs folder:
    - hds - BDI Workshop Credentials.pdf

-or-

    - bdiw - BDI Workshop Credentials.pdf

Cloudera Control

- Click icon in launcher on bottom screen 

Hue Login (web browser)

- Username: cloudera
- Password: cloudera

Pentaho User Console Login (web browser)

- See document on Desktop in Docs folder:
    - hds - BDI Workshop Credentials.pdf

-or-

    - bdiw - BDI Workshop Credentials.pdf

Start Pentaho and PostGreSQL (terminal window)

- cd /pentaho/current_version
- ./ctlscript.sh start postgresql
- ./ctlscript.sh start baserver

Enterprise Edition Only

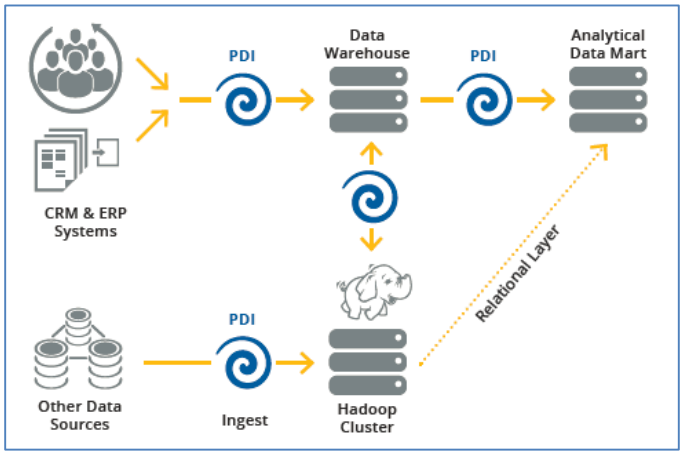
DW Optimization Use Case

What is it?

- Existing Data Warehouse implementations are struggling with rising data volumes, causing performance concerns and impacting a business's ability to meet SLA's. And at the same time, the costs involved in adding additional infrastructure has increased rapidly.
- The optimization solution is intended to relieve pressure on existing infrastructure by offloading less frequently used data, and improve performance by moving data transformation workloads to Hadoop.

Why do it?

- Reduce data warehouse costs by reducing the need for expensive DW infrastructure, and save management costs through the use of low latency data store.
- Meet SLAs to deliver data on time, & empower business users to meet goals on time.
- Satisfy compliance requirements by providing ready access to historical data (such as long-archived financial transaction records).



Value of Pentaho

- Staff savings & productivity: Pentaho Data Integration for big data, along with Pentaho MapReduce means existing data warehouse developers can more easily move and process data between the data warehouse and Hadoop without coding.
- Time to value: Organizations can speed development time with Pentaho MapReduce by up to 15 times versus hand-coding and scripting.
- Faster job execution: Pentaho MapReduce runs in parallel in cluster, delivering faster performance when compared to other scripting tools.

Pentaho components used in this workshop

PENTAHO DATA INTEGRATION (PDI)

PENTAHO ANALYZER

What you will accomplish in this workshop

Part 1 – Building PDI Transformations (3 exercises)
Part 2 – Pentaho Visual MapReduce (3 exercises)

Part 1: Building PDI Transformations

Pentaho Data Integration (PDI) gives users a graphical user interface to build transformations and jobs that solve complex data integration challenges. PDI provides broad connectivity to a variety of sources including relational databases, text files, web services, XML, JSON and other big data technologies. The user interface reduces data integration complexity by eliminating the need to code data extractions, data transformations and data loads.

Part One of the Data Warehouse Optimization use case includes three exercises. The first exercise gives you experience building your first transformation to extract, transform and blend call detail records (CDR) with geographic data based on area code. The second exercise adds data filtering and enrichment steps and a final step to load the transformed CDR data to PostgreSQL. The third and final exercise is the fun part, as you use PDI visualize the CDR data resulting from your hard work in the first two exercises.

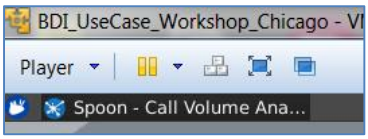
PDI Exercise 1: Create a transformation to blend CDR data and load to PostgreSQL

This first exercise steps you through the process of creating a transformation to transform and blend two CSV files containing call detail records and area code geographies. The first call detail records file, `callrecords_10years.csv`, contains two fields, a call timestamp and the source phone number. The second file, `areacodes.csv`, contains a mapping of area codes to geographies in the United States. In this exercise you blend the CDR records with the geography information.

1. Launch the PDI client-based authoring tool (Spoon) from the launch menu icon  at the bottom of your screen. Click this icon just *once* to launch Spoon.

2. You should see the PDI splash screen appear while PDI loads.

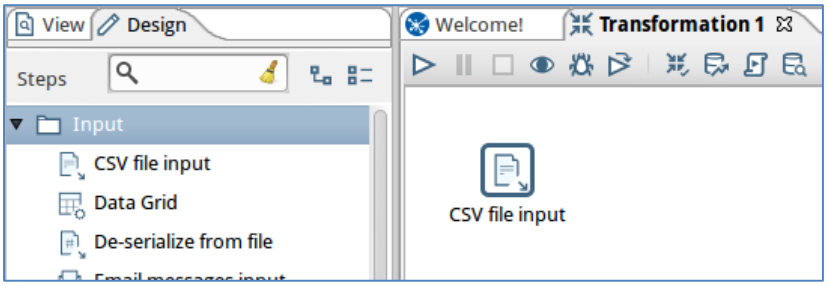
> All open applications appear in the top left section of your screen. Once open, you will see Spoon as shown in the following screenshot.



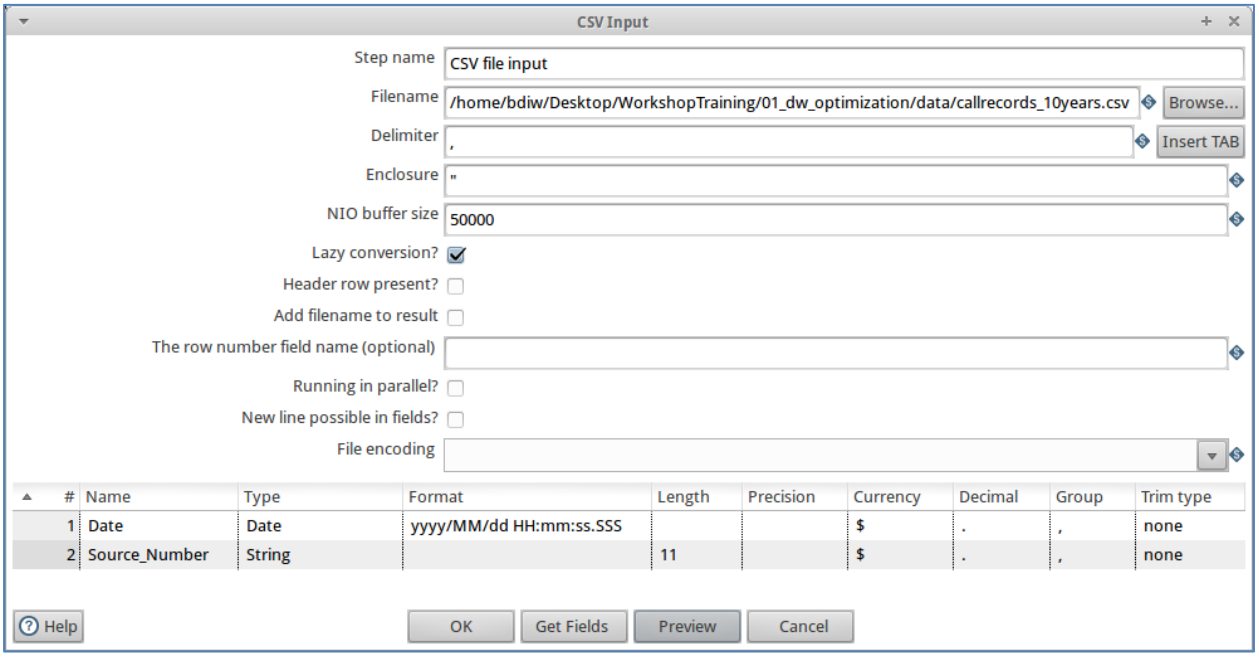
3. From the main menu choose **File | New | Transformation**

We need to access Call Data Records from within a flat file. The file format is a csv file, so you will access the data by configuring a CSV file input step.

4. From the **Design** tab on the left, expand the **Input** folder; then, select and drag **CSV file input** onto the **canvas**



5. Double-click on **CSV file input** to open its properties
6. Browse to the directory `/pentaho/shared_content/WorkshopTraining/01_dw_optimization/data/` and select `callrecords_10years.csv`.
7. Uncheck the **Header Row present?** option
8. Click the **Get Fields** button at the bottom and enter `5,000` for the **Sample Size**.
9. Rename the first field, `Field_000`, to `Date`
10. Rename the second field, `Field_001`, to `Source_Number`
11. Click the **Preview** button to preview the data and then click **Close**. Your **CSV File Input** dialog box should match the following image:



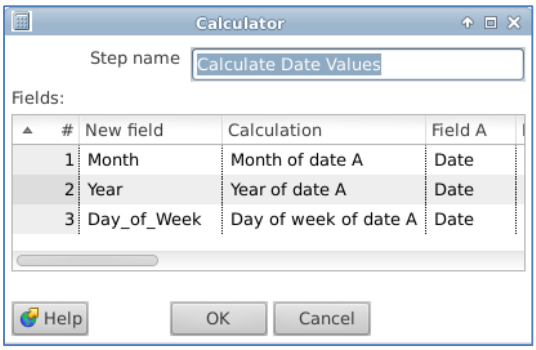
12. Click **OK** to return to the canvas.

Now that we have data, we will calculate the Year, Month, and Day of Week. This will allow the developer to filter the call records in a subsequent step to include only weekend calls.

13. Select and drag the **Calculator** step onto the canvas.
14. To draw a hop between two steps, shift-click the **CSV file input** step and while holding down your mouse key, drag a **hop** over to the **Calculator** step. When prompted, select **Main output of step**.

    Going forward the workbook will simply instruct you to create a hop between two steps without a detailed description as in the previous step 14.

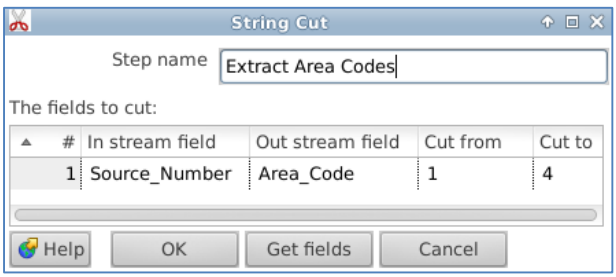
15. Double-click on the **Calculator** step to open its properties.
16. Change the **Step name** to `Calculate Date Values`
17. In the **Fields** section add `Month`, `Year` and `Day_of_Week`. The properties for these fields should match the following screenshot:



18. Click **OK** to return to the canvas.

To derive location information from the data, we must know the Area Code within the phone number.

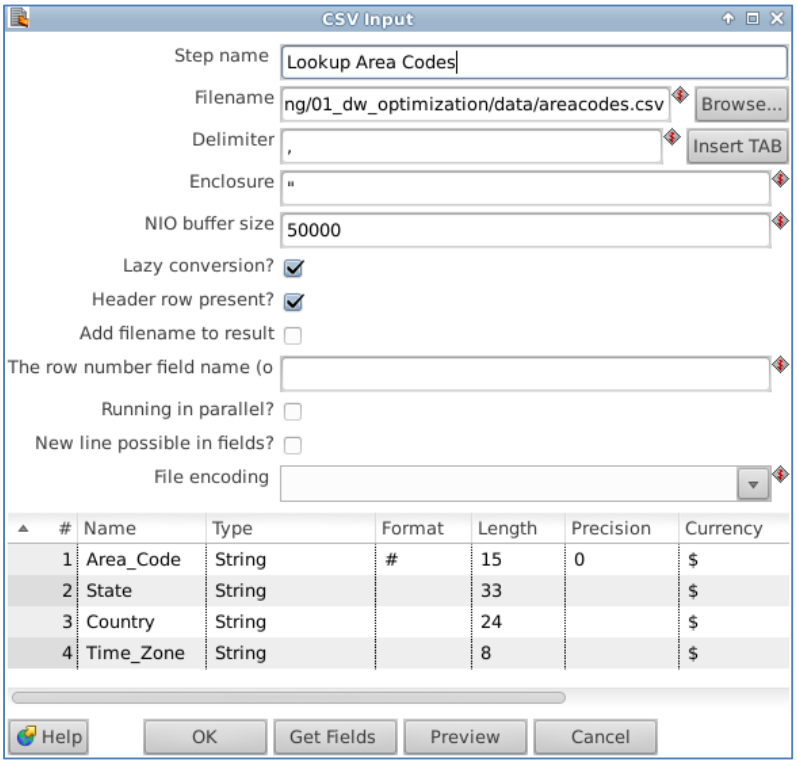
19. Select and drag the **Strings Cut** step onto the canvas.
20. Create a hop between the **Calculate Date Values** step and the **Strings Cut** step.
21. Double-click on the **Strings Cut** step to open its properties.
22. In the **Step name** field, type `Extract Area Codes`. In the **Fields to cut** section select `Source_Number` as the **In Stream field** and type `Area_Code` as the **Out Stream field**. The properties for these fields should match the following screenshot:



23. Click **OK** to return to the canvas.

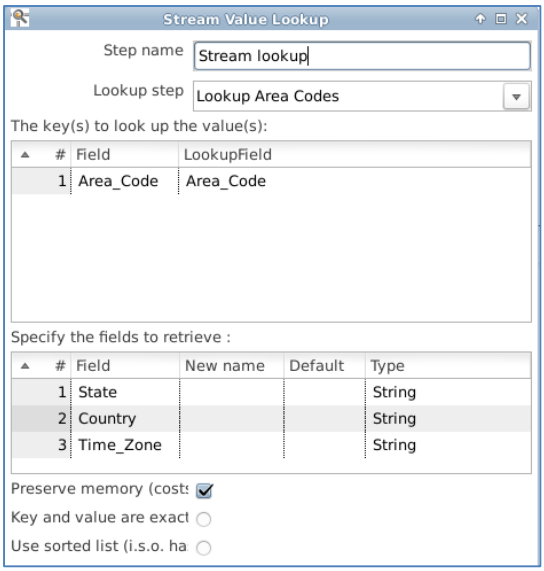
Now that we know the Area Code, we will use a lookup file to map the Area Code to State, Country, and Time Zone.

24. From the **Design** tab on the left, expand the **Input** folder; then, select and drag **CSV file input** onto the **canvas**
25. Double-click on **CSV file input** to open its properties
26. Change the **Step name** to `Lookup Area Codes`
27. Browse to the directory `/pentaho/shared_content/WorkshopTraining/01_dw_optimization/data/` and select `areacodes.csv`.
28. Click the **Get Fields** button at the bottom and enter `5,000` for the **Sample Size**.
29. For the `Area_Code` field, change the **Type** from `Integer` to `String`.
30. Click the **Preview** button to preview the data and then click Close to return to the **CSV File Input** step configuration. Your **CSV File Input** dialog box should match the following image:

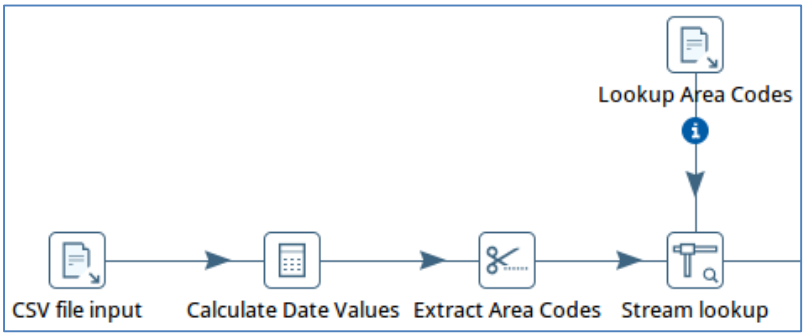


31. Click **OK** to return to the canvas.
32. Expand the **Lookup** folder; then, select and drag **Stream Lookup** onto the canvas.
33. Create a hop from the **Extract Area Codes** step to the **Stream Lookup** step. When prompted choose, **Main Output of Step**, to complete the hop connection.
34. Create a hop from the **Lookup Area Codes** step to the **Stream Lookup** step and choose **Main Output of Step**, to complete the hop connection.

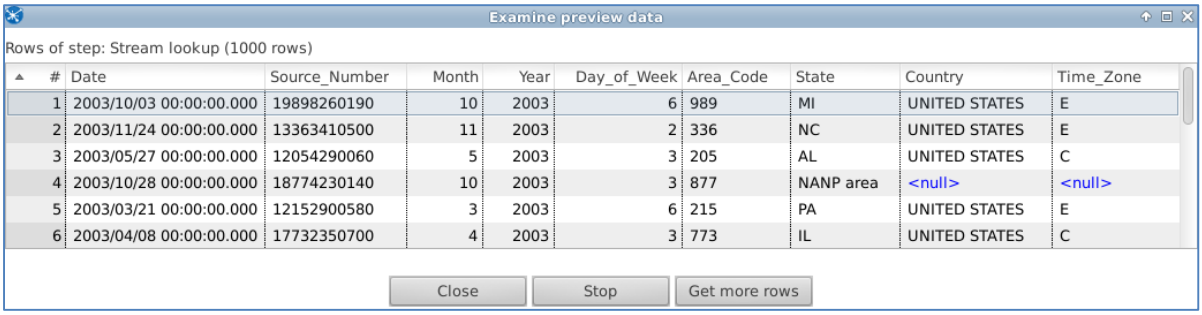
35. Double-click on the **Stream Lookup** step to open its properties.
36. In the **Lookup step** select `Lookup Areas Codes`.
37. In the **Key(s) to Lookup Value(s)** section select `Area_Code` in the **Field** column and select `Area_Code` as the **LookupField** column.
38. Click the **Get Lookup Fields** button to populate the fields to retrieve section at the bottom.
39. Highlight and delete `Area_Code` from the fields to retrieve section. Your **Stream Lookup** dialog box should now look like this:



40. Click **OK** to return to the canvas. Your transformation should match the following screenshot:



41. From the **File** menu, choose **Save**.
42. In the Name field, specify `CDR-RDBMS`, and save to: `/pentaho/shared_content/WorkshopTraining/student_files/01_dw_optimization`.
43. To test your transformation, right-click on the **Stream lookup** step and choose **Preview** from the list.
44. Click the **Quick Launch** button to open the **Examine preview** data dialog. You should see records similar to the image below:



#	Date	Source_Number	Month	Year	Day_of_Week	Area_Code	State	Country	Time_Zone
1	2003/10/03 00:00:00.000	19898260190	10	2003	6	989	MI	UNITED STATES	E
2	2003/11/24 00:00:00.000	13363410500	11	2003	2	336	NC	UNITED STATES	E
3	2003/05/27 00:00:00.000	12054290060	5	2003	3	205	AL	UNITED STATES	C
4	2003/10/28 00:00:00.000	18774230140	10	2003	3	877	NANP area	<null>	<null>
5	2003/03/21 00:00:00.000	12152900580	3	2003	6	215	PA	UNITED STATES	E
6	2003/04/08 00:00:00.000	17732350700	4	2003	3	773	IL	UNITED STATES	C

45. Click the **Stop** button to close the preview data box.

PDI Exercise 2: Enhance the PDI transformation to filter and add call count

This second exercise steps you through adding additional steps to your transformation to filter for U.S. only calls, replace null values, and to add a call count measure field. The final step of this exercise loads the transformed data to PostgreSQL.

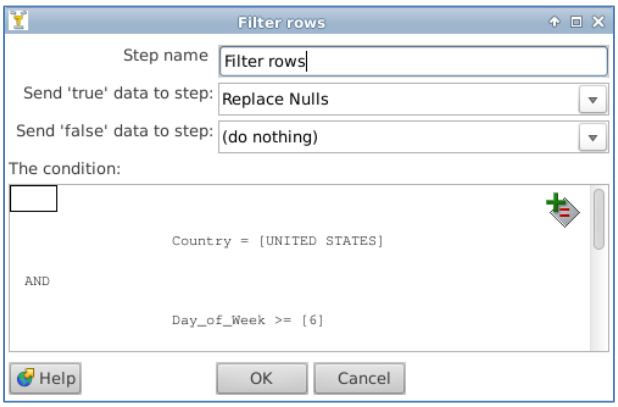
We need to apply a filter to the data to ensure we only get calls placed in the USA and calls made on weekends only, Saturday and Sunday. Calls placed outside of the US and on days Monday through Friday will be discarded.

1. Expand the **Flow** folder; then, select and drag **Filter Rows** onto the canvas.
2. Create a hop between the **Stream Lookup** step and the **Filter Rows** step.
3. Double-click on the **Filter Rows** step to open its properties.
4. Rename this step to **U.S.-only calls**.
5. Click **OK** to return to the canvas.
6. From the **Flow** folder, select and drag **Dummy (do nothing)** onto the canvas above the **U.S.-only calls** step.
7. Create a hop from the **U.S.-only calls** step to the **Dummy (do nothing)** step. Select `Result is False`.
8. Double click the **U.S.-only calls** step to open its properties.
9. Under **The condition** section select the **<field>** on the left. From the pop-up window select `Country` and then click **OK**.
10. Click in the middle box and select "=" as the function and then click **OK**.
11. Click the bottom right **<value>** box and type `UNITED STATES` as the **Value**.

The text comparison is case sensitive. Be sure to enter all caps.

12. Click the **Add condition** icon on the right side to add a filter for Friday and Saturday calls.

13. Click the **null = [ ] area** to add the second condition.
14. Select the **<field>** on the left. From the pop-up window select `Day_of_Week` and then click **OK**.
15. Click in the middle box and select >= as the function.
16. Click the bottom right **<value>** box and type `6` as the **Value.**



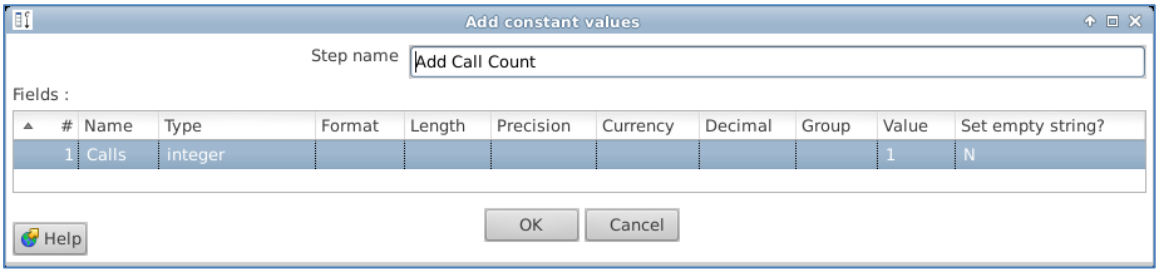
17. Click **OK** to return to the canvas.

There are some Area Code values of 000. We will change those to Null values.

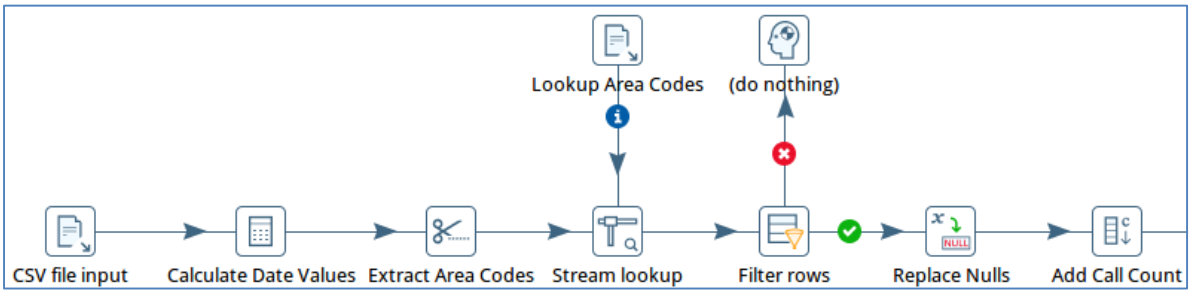
18. Expand the **Utility** folder and drag **Null if…** step onto the canvas.
19. Create a hop between the **U.S.-only calls** step and the **Null if…** step. Select `Result is true` to complete the hop.
20. Double click the **Null if…** step to open its properties.
21. Rename this step to `Replace Nulls`
22. Under the **Fields** section select `Area_Code` and type `000` in the **Value to turn to NULL** field, then click **OK**.

To generate a call count measure, we will add a constant value of 1 to each record so we can aggregate the number of calls with analysis reports.

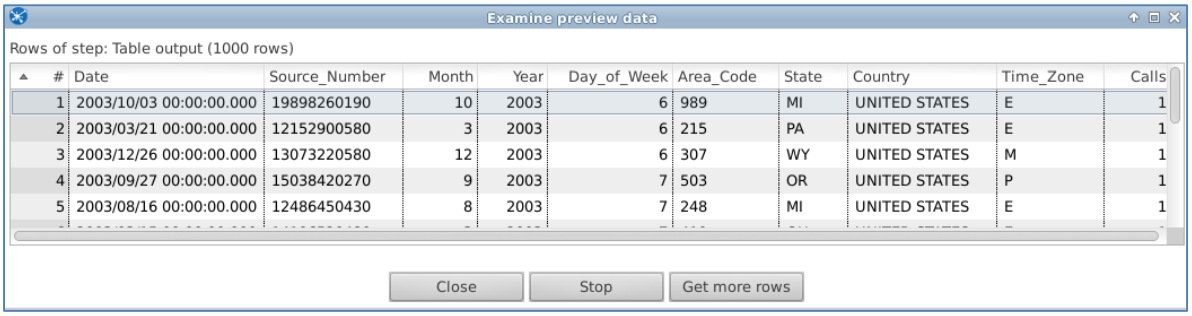
23. Expand the **Transform** folder; then, select and drag the **Add Constants** step onto the canvas.
24. Create a hop between the **Replace Nulls** step and the **Add Constants** step.
25. Double click the **Add Constants** step to open its properties.
26. In the **Step Name** field, type `Add Call Count`.
27. Add a **Field** named `Calls` as a **Type** `Integer` with a **Value** of `1` as illustrated here:



28. Click **OK** to return to the canvas. Your transformation should match the following image:



29. From the **File** menu, choose **Save** to save your work.
30. To test your transformation, right-click on the **Add Call Count** step and choose **Preview** from the list.
31. Click the **Quick Launch** button to open the **Examine preview** data dialog. You should see records similar to the image below:



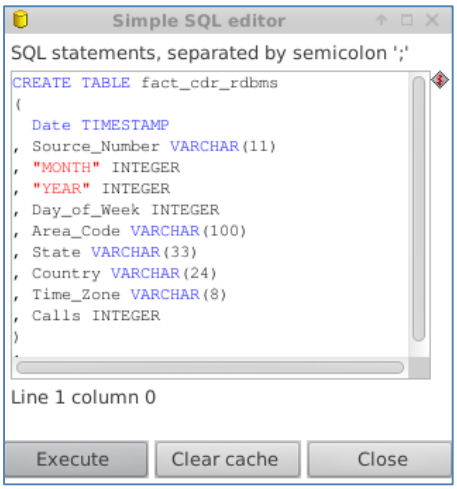
32. Click the **Stop** button to close the preview data box.

Our data is ready to be placed into a database table for reporting. We need to configure a target table within a PostgreSQL database for the enriched call record data.

33. Expand the **Output** folder; then, select and drag **Table output** onto the canvas.
34. Create a hop between the **Add Call Count** step and the **Table output** step.
35. Double-click the **Table output** step to open its properties.
36. For **Connection** select `workshop_postgres`
37. For **Target table** enter `fact_cdr_rdbms`

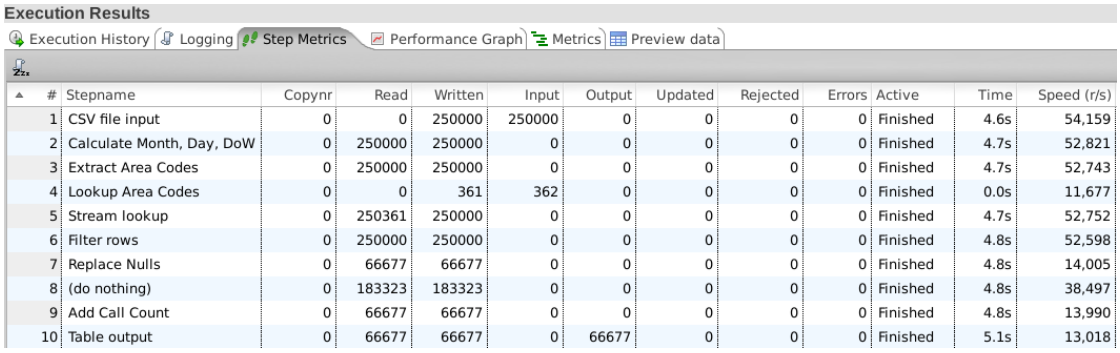
38. Check **Truncate table** and then click the **SQL** button at the bottom to generate a create table SQL statement matching the following image:

    If the `fact_cdr_rdbms` table already exist then your SQL will not match the SQL shown in the following image, and you can close this dialog box without executing any code.



39. Click the **Execute** button to create the `fact_cdr_rdbms` table in PostgreSQL.
40. The **Results of the SQL statements** dialog will appear. Click **OK** to return to the **Table output** configuration dialog box.
41. From the **File** menu, choose **Save**.
42. To load PostgreSQL, execute the transformation by choosing **Action** → **Run** from the main menu or by clicking the run icon  .
43. The **Execute a transformation** dialog will appear. Click the **Launch** button at the bottom.

A green check will appear on the **Table output step** when the transformation finishes without errors. The **Step Metrics** tab shows the 250,000 records input from the **CSV file input** step and 66,677 records output to PostgreSQL.

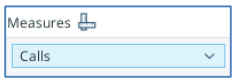


#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	CSV file input	0	0	250000	250000	0	0	0	0	Finished	4.6s	54,159
2	Calculate Month, Day, DoW	0	250000	250000	0	0	0	0	0	Finished	4.7s	52,821
3	Extract Area Codes	0	250000	250000	0	0	0	0	0	Finished	4.7s	52,743
4	Lookup Area Codes	0	0	361	362	0	0	0	0	Finished	0.0s	11,677
5	Stream lookup	0	250361	250000	0	0	0	0	0	Finished	4.7s	52,752
6	Filter rows	0	250000	250000	0	0	0	0	0	Finished	4.8s	52,598
7	Replace Nulls	0	66677	66677	0	0	0	0	0	Finished	4.8s	14,005
8	(do nothing)	0	183323	183323	0	0	0	0	0	Finished	4.8s	38,497
9	Add Call Count	0	66677	66677	0	0	0	0	0	Finished	4.8s	13,990
10	Table output	0	66677	66677	0	66677	0	0	0	Finished	5.1s	13,018

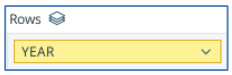
PDI Exercise 3: Create a data table and area chart of the CDR data.

The **Visualize** PDI perspective allows developers to instantly analyze and visualize data from the output of a data transformation. In this exercise, you use the visualize perspective to visualize the newly transformed CDR data by creating a conditionally-formatted table with a custom calculation for CDR annual growth % by year. Next you create and an area chart showing call volumes by year and a drop in call volumes during the 2008 recession.

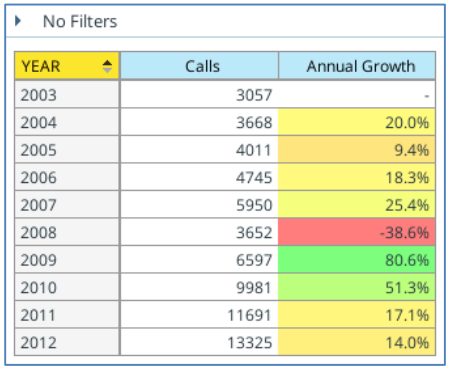
1. Right-click on the **Table output** step and select **Visualize → Analyzer** to launch into the **Visualize** perspective.
2. Under the **Measure** drop down in the **Available fields** section on the left, drag `Calls` to the **Measures** drop zone in the **Layout** section.



3. Under the **Level** drop down in the **Available fields** section on the left, drag `Year` to the **Rows** drop zone in the **Layout** section.



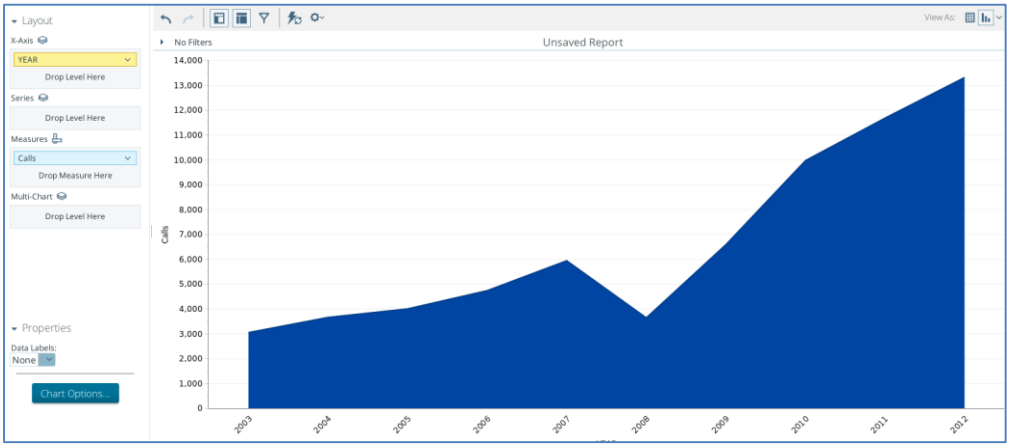
4. Right-click on the column header for `Calls` and select **User Defined Measure → Trend Measure…**
5. In the **Name** field, type `Annual Growth`.
6. In the **Period Type** field, select `Year`.
7. In the **Number of periods** field, type `1` (should be the default).
8. In the **Show trend as** field, select `% of change from previous period`.
9. In the **Decimal Places** field, select `1`.
10. Click **OK** to return to the report.
11. Right-click on the column header for `Annual Growth` and select **Conditional Formatting | Color Scale: Green-Yellow-Red**.
12. Your analysis should now match the following image:



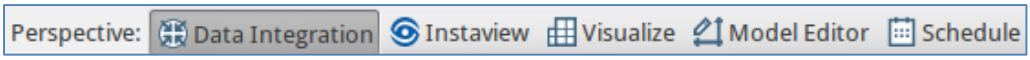
YEAR	Calls	Annual Growth
2003	3057	-
2004	3668	20.0%
2005	4011	9.4%
2006	4745	18.3%
2007	5950	25.4%
2008	3652	-38.6%
2009	6597	80.6%
2010	9981	51.3%
2011	11691	17.1%
2012	13325	14.0%

13. Click on the **Switch to Chart** button on the upper right hand side of the canvas ( ) and select **Area**.

14. Click the `Annual Growth` measure in **Layout** section and choose **Hide from Chart**. The resulting area chart should match the following screenshot:



15. Click the **Data Integration** perspective in the top right area of PDI to return to your transformation.



16. From the **File** menu, choose **Save**.
17. From the **File** menu, choose **Save As** to create a copy of this transformation to be used in the next exercise.
18. In the Name field, specify `CDR-Mapper.ktr`, and save to: `/pentaho/shared_content/WorkshopTraining/student_files/01_dw_optimization`.

Congratulations on completing a PDI transformation to blend, enrich and load CDR data to PostgreSQL for analysis! The next section introduces the concept of Visual MapReduce for running transformations in a Hadoop cluster to leverage the powerful parallel processing capabilities of Hadoop.

Part 2: Pentaho Visual Map Reduce (VMR)

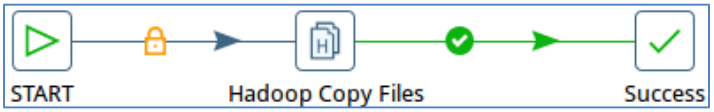
Part Two of the Data Warehouse Optimization use case has you modify your first transformation to create a Visual MapReduce transformation that can be run inside the Hadoop cluster; leveraging the processing power of Hadoop. You convert this transformation to a mapper transformation and use Pentaho to run this as a native mapper job inside of Hadoop. This is powerful because it takes the transformation logic to where the data resides at each data node within the Hadoop cluster. The Pentaho engine and mapper transformation are automatically copied to the data nodes and run as a native MapReduce application.

Part Two includes three exercises. In the first exercise you create a job to load the CDR data to HDFS. The second exercise has you convert your existing transformation to a mapper transformation. The third and final exercise extends the job to add a Visual MapReduce step to execute the mapper transformation created in exercise one against the CDR data loaded to HDFS in exercise two.

MapReduce Exercise 1: Create a PDI Job to load CDR data to Hadoop

This first exercise steps you through creating a job that executes a single job step. You need the CDR data in HDFS to be able to process it with your mapper transformation. This job step loads the raw CDR data to HDFS so that it can be processed in a later exercise.

1. From the main menu choose **File | New | Job**
2. From the **Design** tab on the left, expand the **General** folder and drag **START** and **Success** job steps onto the canvas.
3. Expand the **Big Data** folder and drag **Hadoop Copy Files** onto the canvas between the **START** and **Success** steps.
4. Create a hop between the **Start** step and the **Hadoop Copy Files** step.
5. Create a hop between the **Hadoop Copy Files** step and the **Success** step to match the following image:

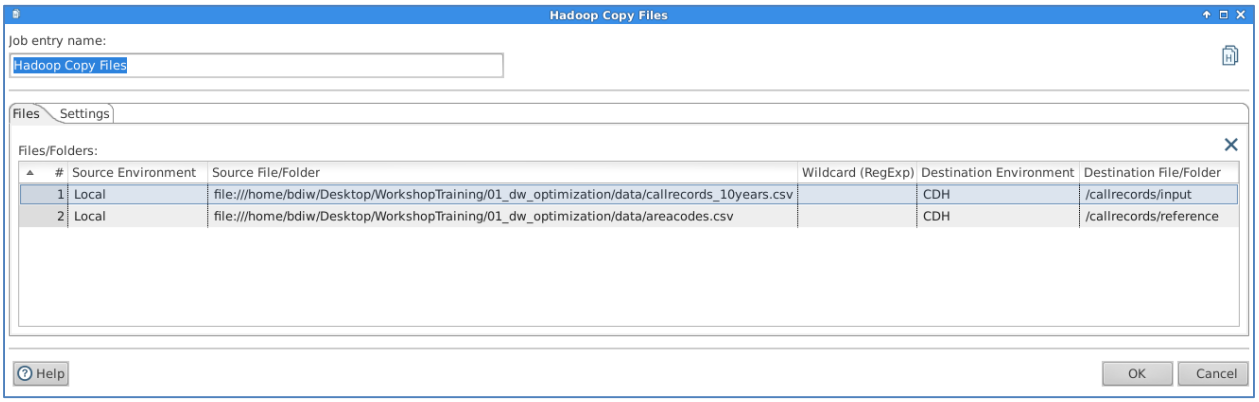


To process data with our mapper transformation inside Hadoop, we need to define the data file that will be copied, and where it will be placed inside HDFS.

6. Double-click on **Hadoop Copy Files** to open its properties
7. On the **Settings** tab check the following two items: **Create destination folder** and **Replace existing files**.
8. In the **Files** tab select the cell beneath the "Source Environment" header on the file tab and select "Local" from the drop down.
9. Select the cell beneath the "Source File/Folder" step and select the ... button.
10. Browse to the following file to select it, and then click the **OK** button to return to the **Copy Files** dialog box.

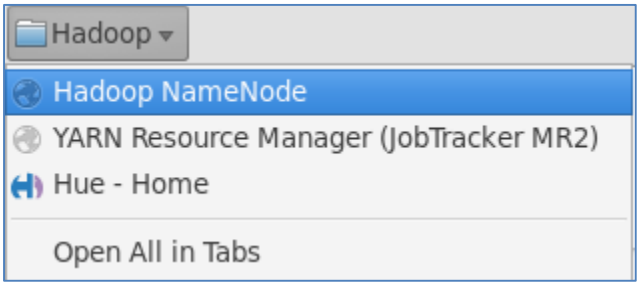
```
file:///pentaho/shared_content/WorkshopTraining/01_dw_optimizati
on/data/callrecords_10years.csv
```

11. Click the **OK** button to return to the **Copy Files** dialog box
12. Select the cell beneath the "Destination Environment" header on the file tab and select "CDH" from the drop down.
13. Select the cell beneath the "Destination File/Folder" step and enter `/callrecords/input`
14. Repeat steps **8-13** to copy the "Source File/Folder" `file:///pentaho/shared_content/WorkshopTraining/01_dw_optimization/data/areacodes.csv` to the "Destination File/Folder" `/callrecords/reference`
15. Your **Hadoop Copy Files** dialog box should match the following image:

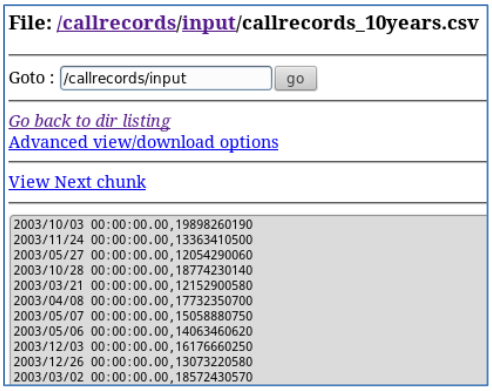


16. Click OK to return to the canvas.
17. From the **File** menu, choose **Save**.
18. In the Name field, specify `CDR-CopyFiles.kjb` and save to the following location: `/pentaho/shared_content/WorkshopTraining/student_files/01_dw_optimization`.
19. From the **Action** menu, choose **Run** and then click **Launch**. On the **Job metrics** tab located in the bottom section of Spoon, you should see the job progress.
20. To view the newly copied CDR data in Hadoop, launch Firefox by single-clicking the Firefox icon at the bottom of your screen.

21. From the Firefox bookmarks bar click **Hadoop NameNode**.



22. Click the following links **Browse the file system** → **callrecords** → **input** → **callrecords_10years.csv**
23. Click the following links **Browse the file system** → **callrecords** → **reference** → **areacodes.csv**
24. If your job executes successfully you will see CDR records as shown in following screenshot:



The file is now ready to be processed within Hadoop and by the mapper transformation previously created in MapReduce Exercise 1.

MapReduce Exercise 2: Create a mapper transformation to process CDR data

This second exercise steps you through modifying your existing transformation to add **Map Reduce Input** and **Map Reduce Output** steps and change the **Stream Lookup** step to query geographic data from HDFS instead of a local CSV file. You will also add additional steps needed to complete the mapper transformation.

1. Make sure that you are working with the newly created `CDR-Mapper.ktr` transformation created in the previous exercise.
2. Delete the following *three* steps: **CSV file input step**, **Lookup Area Codes**, and **Table Output**.

To convert this transformation to a mapper transformation and use Pentaho to run this as a native mapper job inside of Hadoop, we need to define a MapReduce Input and MapReduce Output. This will push the transformation logic to where the data resides at each data node within the Hadoop cluster.

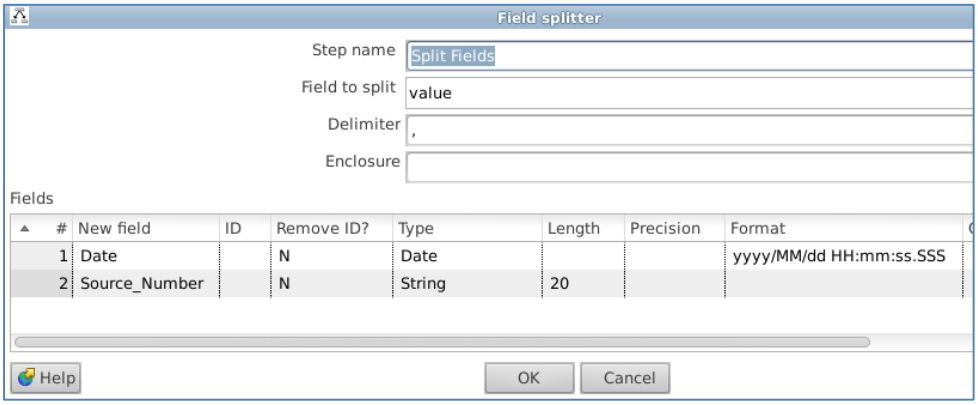
3. Expand the **Big Data** folder and drag the **MapReduce Input** and **MapReduce Output** steps onto the canvas. Place the **MapReduce Input** step at the beginning of your transformation and the **MapReduce Output** step at the end of your transformation.
4. Double-click on **MapReduce Input** step to open its properties.

The MapReduce Input step is introducing two new fields into the stream, **key** and **value**. At this point in the transformation, the **key** field is null, and the **value** field is the full CDR record. In a subsequent step, we will split out the **value** field to Date and Source_Number. The **key** field will be assigned later in the mapper transformation as part of the MapReduce Output configuration, so the Hadoop reducer can get the data in the order we need.

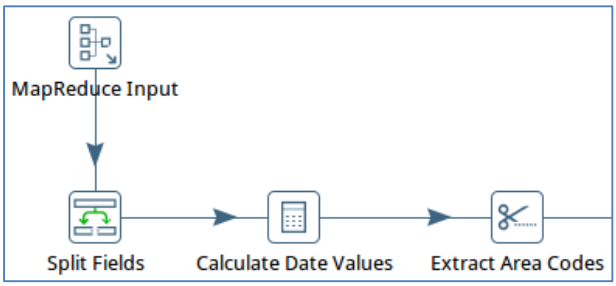
5. For both the **Key field** and the **Value field** select `String` for **Type**.
6. Click **OK** to return to the canvas.

Our input **value** of the Key/Value pair is a comma-delimited record of Date and Source Phone Number. We need to split this data into two defined fields of Date and Source Number.

7. Expand the **Transform** folder and drag **Split Fields** onto the canvas.
8. Create a hop from the **MapReduce Input** step to the **Split Fields** step.
9. Create a hop from the **Split Fields** step to the **Calculate Date Values** step.
10. Double-click on the **Split Fields** step select `value` for **Field to split** and leave "**,**" as the **Delimiter**.
11. In the **Fields** section add `Date` and `Source_Number`.
12. Complete the **Type**, **Length** and **Format** columns for these fields to match the following screenshot:

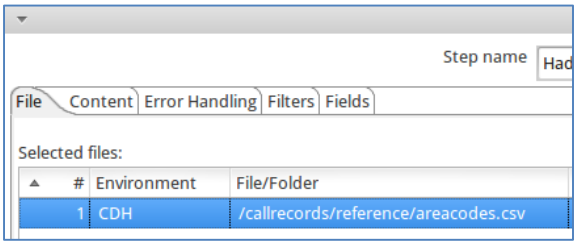


13. Click **OK** to return to the canvas. The first part of your transformation should now match the following image:

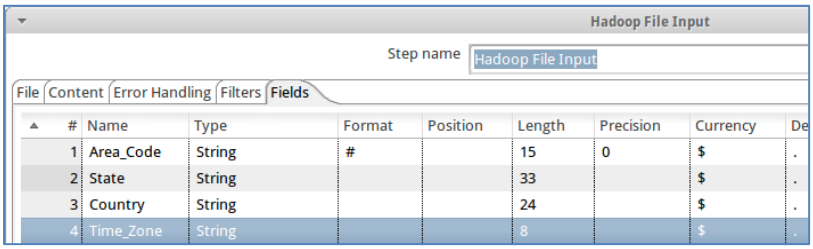


Previously in Part 1, Exercise 1, we used a lookup file on the Pentaho server to map the Area Code to State, Country, and Time Zone. Now we will use a lookup file that is same in data and structure, but is placed on the Hadoop file system HDFS.

14. Expand the **Big Data** folder and select and drag the **Hadoop File Input** step onto the canvas directly above the Stream Lookup step.
15. Create a hop from the **Hadoop File Input** step to the **Stream Lookup** step.
16. Double-click on the **Hadoop File Input** step to open its properties.
17. Select the cell beneath the “Environment” header on the file tab and select “CDH” from the drop down.
18. Select the cell beneath the “File/Folder” step and select the ... button.
19. Browse to the following HDFS directory, `/callrecords/reference` and select the file `areacodes.csv` and then click the **OK** button.
20. Your **Hadoop File Input** dialog box should match the following image:



21. Click on the **Content** tab.
22. Change the **Separator** field to a comma `,`.
23. Click on the **Fields** tab.
24. Click the **Get Fields** button and sample `5000` records.
25. Change the `Area_Code` **Type** from `Integer` to `String`.
26. The dialog box should look like this:



27. Click **OK** to return to the canvas.

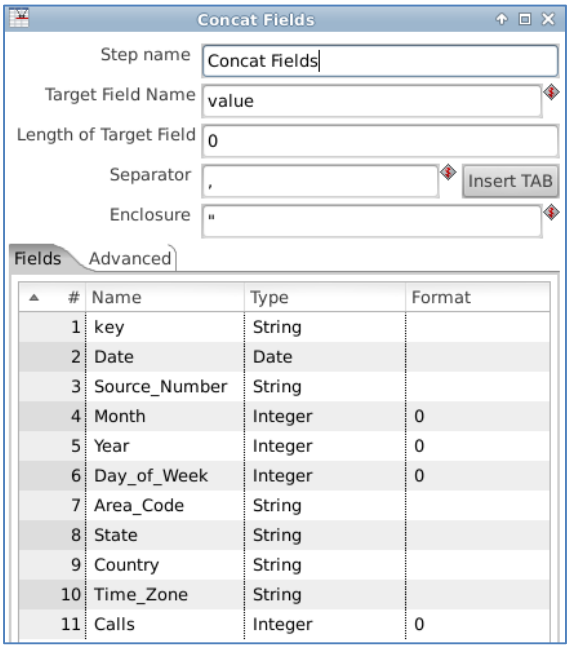
Previously in Part 1, Exercise 1, the **Stream Lookup** step used CSV input from a step named ‘Lookup Area Codes’. This should be changed to reference the new **Hadoop File Input** step we left named as ‘Hadoop File Input’.

28. Double click the **Stream Lookup** step and for the **Lookup step** drop-down, select `Hadoop File Input`.

29. Click **OK** to return to the canvas.

This mapper transformation has generated several new fields. The fields need to be combined into a new target field and delimited. The target field, which we will creatively name 'value', will ultimately be used as the **Value** in the Key/Value pair for the MapReduce output.

30. Expand the **Transform** folder and select and drag the **Concat Fields** step onto the canvas at the end of your transformation.
31. Create a hop from the **Add Call Count** step to the **Concat Fields** step.
32. Double click the **Concat Fields** step to open its properties.
33. In the **Target Field Name** field, type `value` and change the **Separator** field to a comma `,`.
34. Click on the **Get Fields** button.
35. Click on the **Minimal width** button to remove excessive field lengths.
36. Confirm the **Fields** properties match the following and click **OK** to return to the canvas.



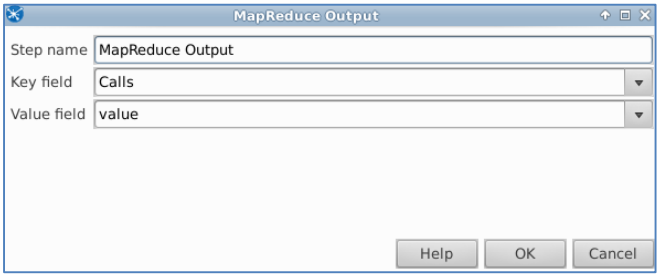
#	Name	Type	Format
1	key	String	
2	Date	Date	
3	Source_Number	String	
4	Month	Integer	0
5	Year	Integer	0
6	Day_of_Week	Integer	0
7	Area_Code	String	
8	State	String	
9	Country	String	
10	Time_Zone	String	
11	Calls	Integer	0

37. Create a hop between the **Concat Fields** step and the **MapReduce Output** step.

The output of MapReduce will be a key/value pair. We can define the output key for any field for which we want to aggregate with a Reducer. The output value will be the contents of the target field we defined in the **Concat Fields** step (Date, Source Number, Area Code, Month, Year, Calls, etc).

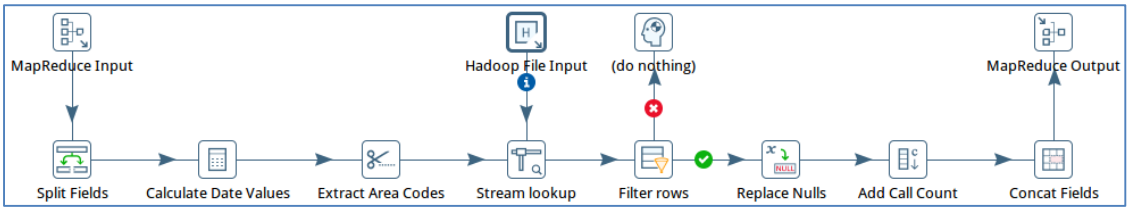
38. Double click the **MapReduce Output** step to open its properties.

39. Select `Calls` for the **Key field** and select `value` for the **Value field**. Your **MapReduce Output** step should match the following image:



40. Click **OK** to return to the canvas.

41. Your completed mapper transformation should match the following image:



42. From the **File** menu, choose **Save**.

43. Do not run this transformation. This transformation will be executed by a job you will created in the last exercise.

MapReduce Exercise 3: Extend the PDI job to execute Visual MapReduce

This final VMR exercise has you extend your existing PDI job with a second job step to execute the mapper transformation you created in the first exercise.

1. Open the `CDR_CopyFiles.kjb` Job you created in the previous exercise.
2. From the **File** menu, choose **Save As**.
3. In the Name field, specify "`CDR-MapReduce-BuildJob`" and save to the following location: `/pentaho/shared_content/WorkshopTraining/student_files/01_dw_optimization`.

To trigger a Pentaho transformation-based MapReduce job within Hadoop, we need to define a **Pentaho MapReduce** step. We will use this step to call our mapper transformation created in Exercise 1, and define parameters such as data inputs and Hadoop cluster connection information.

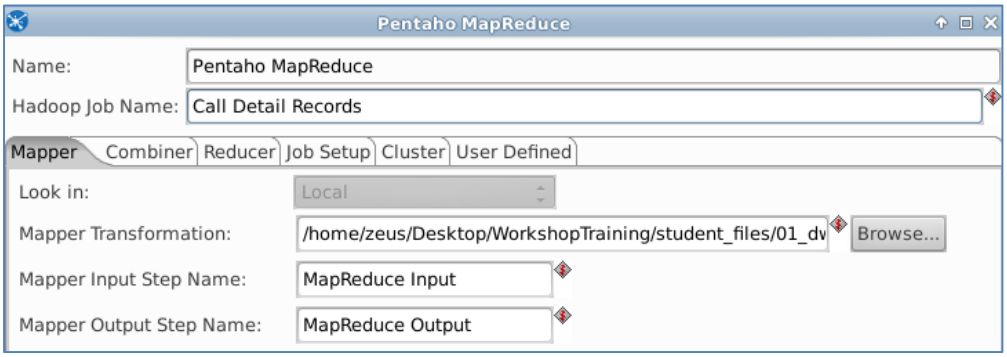
4. From the **Design** tab, expand the **Big Data** folder, then select and drag the **Pentaho MapReduce** step onto the canvas.
5. Select the **Pentaho MapReduce** step and drag it above the connecting arrow between the **Hadoop Copy Files** step and the **Success** step until the arrow becomes highlighted. Once highlighted, drop the **Pentaho MapReduce** step (release the mouse button).

6. Right click on the hop between **Pentaho MapReduce** step and the **Success** step. Select **Evaluation | Follow when result is true**. The job should match the following image:

7. Open the **Pentaho MapReduce** step.
8. In the **Hadoop Job Name** field, specify a name, such as `Call Detail Records`.
9. In the **Mapper** tab, next to the **Mapper Transformation** field, choose the **Browse** button. Navigate to and select this file: `/pentaho/shared_content/WorkshopTraining/student_files/01_dw_optimization CDR-Mapper.ktr`. Click **OK**.

We need to tell Visual MapReduce the names of the MapReduce Input and MapReduce Output steps defined within the mapper transformation created in MapReduce Exercise 1.

10. In the **Mapper Input Step Name** field, type `MapReduce Input`.
11. In the **Mapper Output Step Name** field, type `MapReduce Output`.
12. Your **Pentaho MapReduce** Step should match the following image:



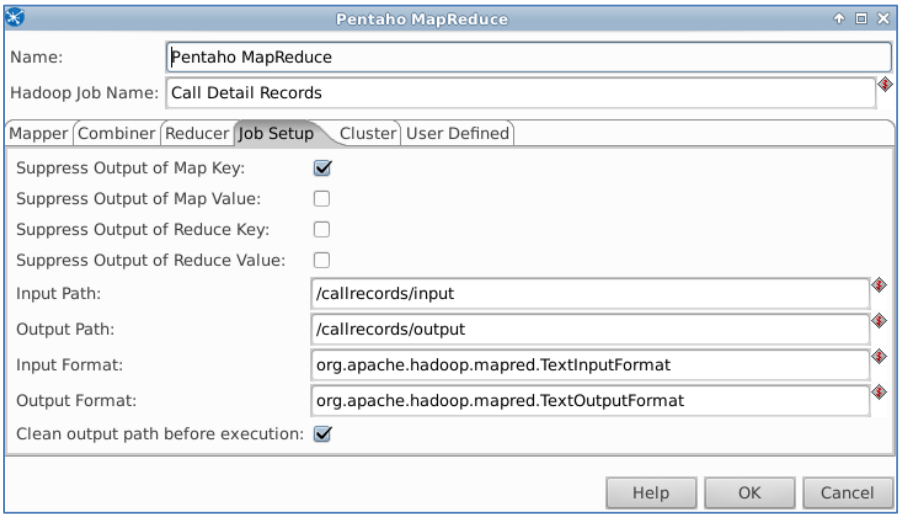
13. Click on the **Job Setup** tab.

We do not want to see the Hadoop-generated mapper key within the Hadoop output data files. And we will remove any files from our output directory before writing new data files to the directory.

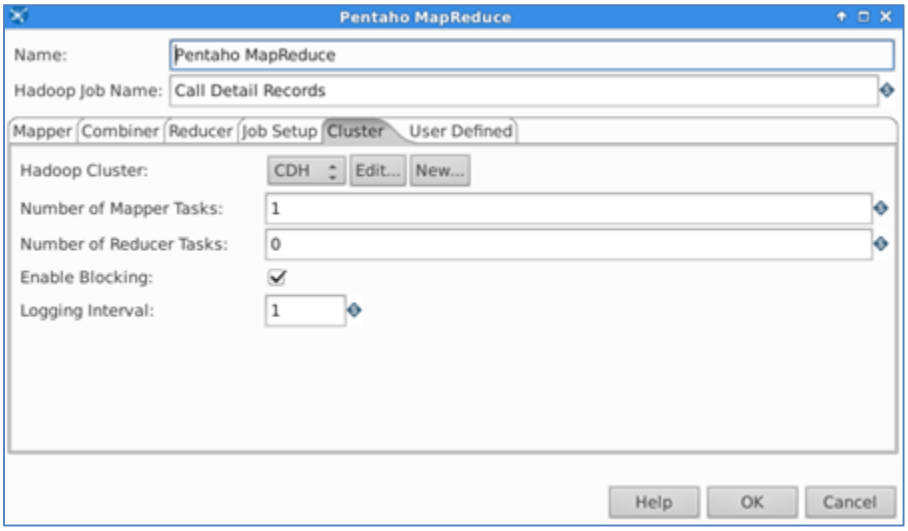
14. Click the check boxes for **Suppress Output of Map Key** and **Clean output path before execution**.

Define the **Input Path** as the directory to which we previously copied the Call Data file to as part of MapReduce Exercise 2.

15. In the **Input Path** field, specify `/callrecords/input`. This is the location in HDFS from which the MapReduce job will retrieve its input data.
16. In the **Output Path** field, specify `/callrecords/output`. This is the location in HDFS where the resulting data processed by the Mappers will be written.
17. In the **Input Format** field, specify `org.apache.hadoop.mapred.TextInputFormat`
18. In the **Output Format** field, specify `org.apache.hadoop.mapred.TextOutputFormat`
19. Your **Job Setup** tab should match the following image:



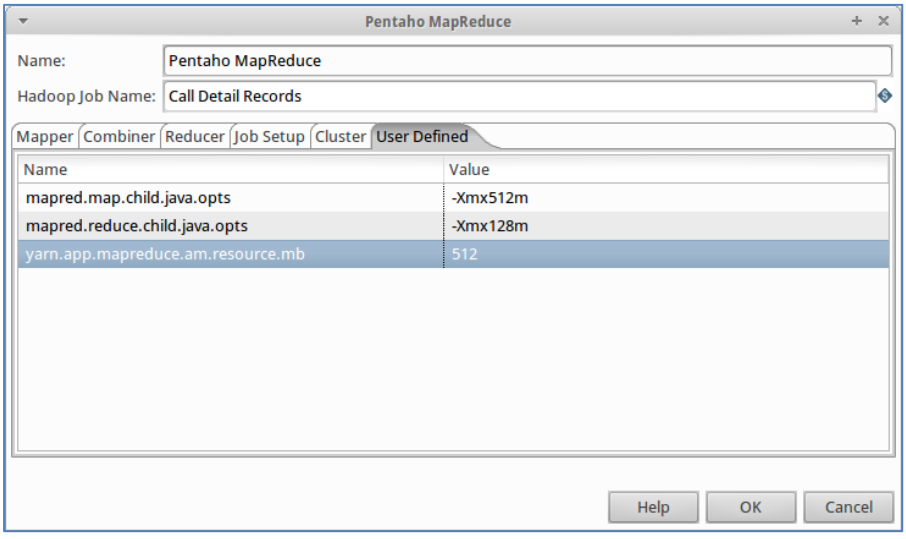
20. Next, click the **Cluster** tab to configure your connection to the Hadoop cluster.
21. In the Hadoop Cluster drop down select “CDH”
22. In the **Number of Mapper Tasks** field, specify `1`.
23. In the **Number of Reducer Tasks** field, specify `0`.
24. Check the box to **Enable Blocking**.
25. Specify a **Logging Interval** of `1` seconds.
26. The **Cluster** tab should match the following image:



27. Click on the **User Defined** tab.

It is important to tune the Java Virtual Machine (JVM) heap size to optimize performance.

28. Click on the first row under the **Name** label. Enter `mapred.map.child.java.opts`.
29. Click in the **Value** field on the same row. Enter `-Xmx512m`.
30. Add another row with **Name** `mapred.reduce.child.java.opts` and **Value** `-Xmx128m`.
31. Add one more row with **Name** `yarn.app.mapreduce.am.resource.mb` and **Value** `512`
32. The **User Defined** tab should look like this:



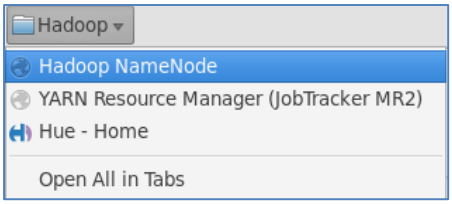
33. Click **OK** to return to the canvas.
34. From the **File** menu, choose **Save**.

35. From the **Action** menu, choose **Run** and then click **Launch**.
36. Click the **Logging** tab located in the bottom section of Spoon to see the Mapper job progress.

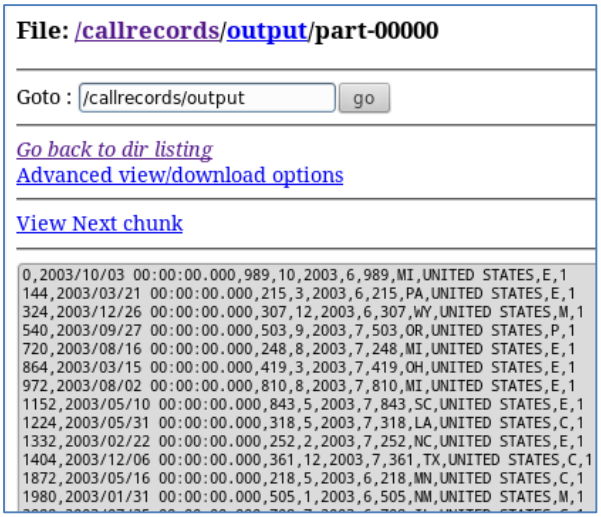
> **Important note**: If the **Logging** tab shows that the mapper is stuck at 4% complete, for example, then your mapper has failed for some reason. The reason for the failure is usually a typing error or missed exercise step. However, the **Logging** tab will not display this reason. To debug this mapper job, you have to go into the mapper task logs via the YARN Resource Manager utility, bookmarked in your Firefox browser. For details on how to debug your mapper job proceed to the optional Exercise 4 on the next page.

37. When the job successfully completes, you will see a green checkmark on the **Success** step.

38. To view the newly processed CDR data in Hadoop, expand Firefox from the top left open applications section if Firefox is already open; otherwise launch it from the bottom launch menu icon.
39. From the Firefox bookmarks bar click **Hadoop NameNode**.

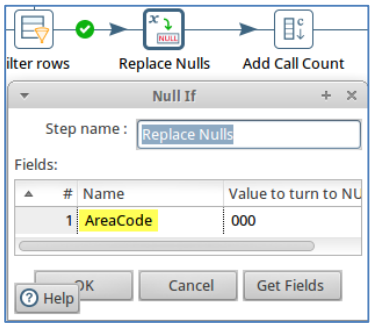


40. Click the following links **Browse the filesystem → callrecords → ouput → part-0000**
41. If your job executes successfully you will see processed and blended CDR records as shown in following screenshot:



(OPTIONAL) MapReduce Exercise 4: Debug a MapReduce job error

If your MapReduce job failed in the previous exercise, you can debug the map task by using the YARN Resource Tracker utility bookmarked in your Firefox browser. This exercise shows you how to debug a spelling issue in the mapper transformation. If you rename the `Area_Code` field in the **Replace Nulls** step to `AreaCode` with no underscore, the mapper transformation will cause your job to fail. You can try it or simply read thru these exercise steps to learn.



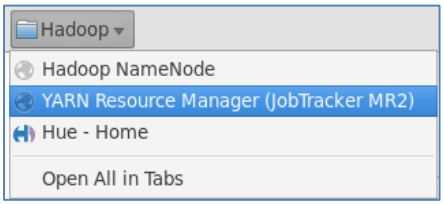
This spelling error will cause your job to fail because the `AreaCode` field does exist in the transformation stream. During execution the **Execution Results** log record starts repeating the % complete value as illustrated in the following screenshot.

```
2015/08/17 20:22:56 - Pentaho MapReduce - Setup Complete: 100.0 Mapper Completion: 2.3178668 Reducer Completion: 0.0
2015/08/17 20:23:06 - Pentaho MapReduce - Setup Complete: 100.0 Mapper Completion: 2.3178668 Reducer Completion: 0.0
2015/08/17 20:23:16 - Pentaho MapReduce - Setup Complete: 100.0 Mapper Completion: 2.3178668 Reducer Completion: 0.0
2015/08/17 20:23:26 - Pentaho MapReduce - Setup Complete: 100.0 Mapper Completion: 2.3178668 Reducer Completion: 0.0
```

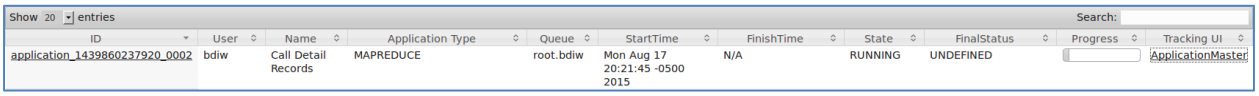
In the previous screenshot you can see the Mapper Completion value of 2.3178668 is repeating. This indicates a problem that should be investigated in the YARN Resource Manager utility. The following steps show how the Cloudera YARN Resource Manager is used to debug the issue.

1. Launch Firefox by single-clicking the Firefox icon at the bottom of your screen.

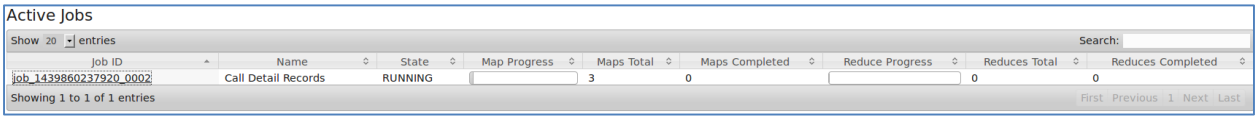
2. From the Firefox bookmarks bar click **YARN Resource Manager**.



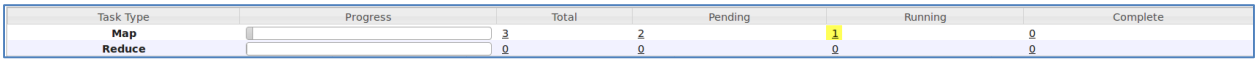
3. The **YARN Resource Manager** home page shows your current running application, `Call Detail Records`, is stuck at 2.3 % complete.



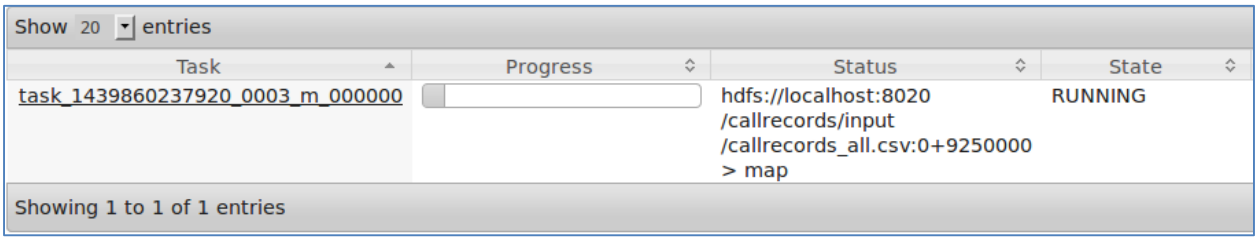
4. Click the **ApplicationMaster** hyperlink to drill into the application. The image below shows that the job is stuck in the map phase. Click the job id link to show map and reduce tasks.



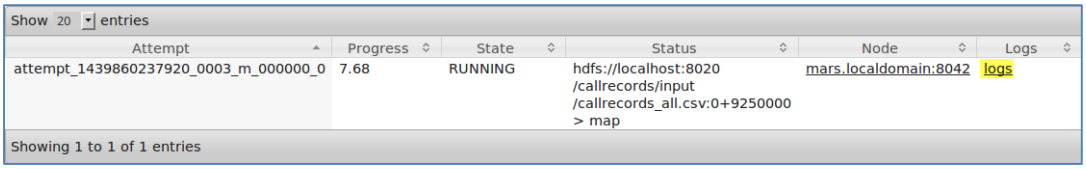
5. Click the **map** hyperlink to drill into the individual map task details.



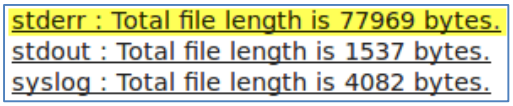
6. Click the **Task** hyperlink to drill into the tasks attempts. As the process continues to run, you might see more than one attempt to run the task.



7. The far right column in the resulting screen provides links for the **Logs**. Click on the log link for the current task attempt.



8. Click the stderr line in the resulting screen.



9. The area code spelling issue is highlighted in the **stderr logs** screenshot below.

10. To fix the issue return to Pentaho and stop the `CDR-MapReduce-BuildJob` job by clicking the green stop icon.

11. Return to your `CDR-Mapper.ktr` transformation and fix the area code spelling in the **Replace Nulls** step.

Congratulations, you have just completed the Data Warehouse Optimization Use Case! This use case is designed to show a development pattern for processing historical CDR data in Hadoop. The next use case, Streamlined Data Refinery, extends the use of Hadoop as a central processing hub for new sources of data that do not necessarily reside in an existing data warehouse.

Enterprise Edition Only

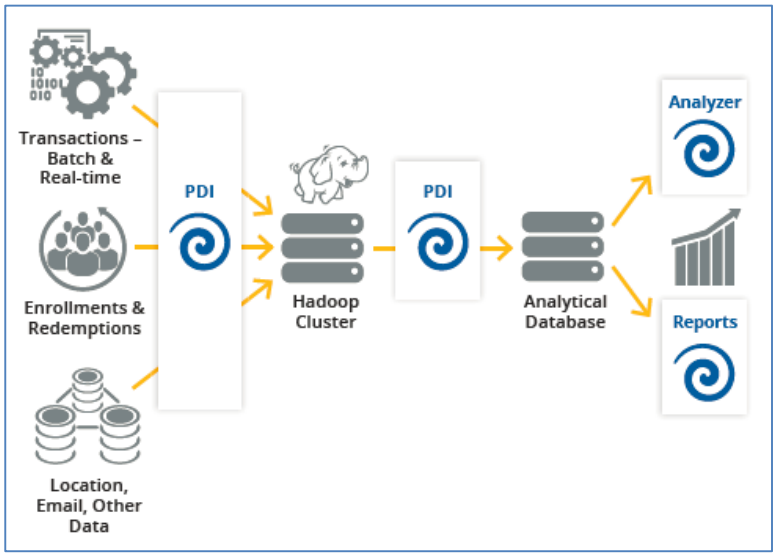
Streamlined Data Refinery Use Case

What is it?

- Streamlined Data Refinery (SDR) is an agile data integration process that includes transformation steps and tools to enable data processing on a low latency data store, such as Hadoop.
- Traditional extract, transform and load (ETL) processes can no longer cope with the variety and volume of data that need to be incorporated in analytics because they are designed for top-down development processes.
- SDR includes self-service analytics along with an analytic database for high speed data querying and visualization.

Why do it?

- Provide business users insight into all data in a timely fashion, including diverse sources at high volume.
- Engineer new data sets for predictive analytics more quickly thanks to rapid ingestion and powerful processing.
- Scale ETL and data management cost savings by utilizing the right technology for the most appropriate purpose.
- Logical next step from Data Warehouse Optimization use case.



Value of Pentaho

- **Data Collaboration**: Using Pentaho Data Integration (PDI), a data developer and business analyst can collaborate in real-time on the creation of new data sets using one product for both data engineering and data visualization.
- **Staff savings & productivity**: Pentaho provides a simple mechanism for creating visual map reduce jobs along with Big Data integration which means data developers can move and process data between Hadoop and a variety of other sources and systems easily and efficiently.

- **Complete Solution**: Broad data integration to accommodate existing architectures plus a powerful array of self-service analytics and visualizations for all end users, whether they be business users, analysts, and/or data scientists!
- **Time to value**: Significantly reduce development time for MapReduce by 15x over typical hand coding and scripting processes.

Pentaho components used in this workshop

PENTAHO DATA INTEGRATION (PDI)

PENTAHO ANALYZER

What you will accomplish in this workshop

- Part 1 – Use PDI and Impala to process data in Hadoop (2 exercises)
- Part 2 – Explore data in PostgreSQL with Pentaho Analyzer (2 exercises)

Part 1: Use PDI and Impala to process data in Hadoop

Now that you have moved the CDR data to HDFS, the next step is to look for other sources of data to take advantage of Hadoop's processing power. Geo-location data feeds from smart phones have traditionally been expensive and difficult to process, but with Hadoop this data can be processed using a streamlined data refinery architecture. Devices register their location periodically on the wireless network. Now instead of estimating the caller's location by an area code lookup, we can more accurately track their location by their geo-location at the time a phone call was made.

Part 1 of the SDR Use Case includes exercises showcasing how to use Impala to join data from multiple tables and load the blended data to both Hadoop and PostgreSQL. The first exercise steps you through the process of creating a PDI job to execute a transformation for loading geo-location data to HDFS and creating an Impala table. The second exercise takes advantage of Impala queries to join two tables into a combined data set. This combined set is saved into Impala first and then loaded to a PostgreSQL database to enable high performance OLAP analysis on the combined data. The combined data is stored in two places to illustrate the flexibility of PDI to orchestrate data into multiple target locations.

PDI Exercise 1: Create a job to load data to HDFS and Impala

This first exercise steps you through the process of creating a PDI job to execute a transformation for loading data to HDFS and creating an Impala table for querying with SQL.

12. From the main menu choose **File | New | Job**
13. From the **File** menu, choose **Save**.
14. In the Name field, specify `SDR_GeoLocation_Impala_Job` and save to this directory: `/pentaho/shared_content/WorkshopTraining/student_files/02_data_refinery`.
15. From the **Design** tab on the left, expand the **General** folder and drag the following three steps on the canvas: **START** , **Success** and **Transformation**.

We need to create a directory on the Hadoop File System (HDFS) to house our geolocation data file.

16. Expand the **File management** folder and drag the **Create a folder** step onto the canvas.
17. Create a hop between the **START** and **Create a folder** steps.
18. Double click the **Create a folder** step to edit its properties. In the **Folder name** field type `hdfs://quickstart.cloudera:8020/demo/data_refinery`.
19. Uncheck the **Fail if folder exists** box.
20. Click **OK** to return to the canvas.

Next, we need to set HDFS permissions to allow Impala write privileges.

21. Expand the **Scripting** folder and drag the **Shell** step onto the canvas to the right of **Create a folder** and double click to open.
22. In **Job entry name** enter **Set HDFS Directory Permissions.**
23. Check the box next to **Insert script.**
24. In **Working directory**, enter /tmp
25. Switch to the **Script** tab and copy/paste the following 2 lines:

```
#!/bin/sh
docker exec $(docker ps -q) sudo -u hdfs hadoop fs -chmod -R 777 /demo/data_refinery
```

26. Click **OK** to return to the canvas.
27. Create a hop between the **Create a folder** and **Set HDFS Directory Permissions** steps
28. Create a hop between the **Set HDFS Directory Permissions** and **Transformation** steps

Once the target directory is created and permissions for Impala access is granted, we need to create and load a text file with geolocation data to HDFS.

29. Double Click the **Transformation** step to edit its properties.
30. In the **Name of job entry** box, type `Load Geolocation Data`.
31. On **Transformation Specification** tab click the browse icon for the **Transformation filename** field and browse to select the following file: `/pentaho/shared_content/WorkshopTraining/02_data_refinery/Solutions/SDR_GeoLocation_HDFS.ktr`

    Note: to save time and reduce redundant work, you are using an *existing* transformation to load the geolocation data to HDFS.

32. Click **OK** to return to the canvas.

We will want to load data to an Impala table. But first we should validate that the target table exists within Impala. The table is named ‘call_detail_geo’.

33. Expand the **Conditions** folder and drag the **Table Exists** step onto the canvas.
34. Create a hop between the **Load Geolocation Data** and **Table Exists** steps.
35. Double Click the **Tables Exists** step to edit its properties.
36. In the **Job entry name** field type `Does Impala table exist?`
37. In the **Connection** field select `Impala`.

Note: this connection to Impala has already been established and is available for use.

38. In the **Table Name** field type `call_detail_geo`.
39. Click **OK** to return to the canvas.

If the Impala table does exist, we will truncate the table data, and load the geolocation data that we previously loaded to HDFS. To load the data, we will execute a SQL script.

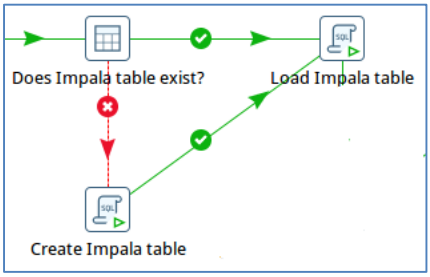
40. Expand the **Scripting** folder and drag a **SQL** step to the right of the **Does Impala table exist?** step.
41. Rename this **SQL** step to `Load Impala Table`

If the Impala table does not exist, we will create the table before loading the geolocation data using a SQL script.

42. Drag a second **SQL** step from the **Scripting** folder and place it below the **Does Impala table exist?** step.
43. Rename this **SQL** step to `Create Impala Table`
44. Create a hop from the **Does Impala table exist?** step to the **Load Impala table** step to the right. This hop should be green in color indicating this path will be taken if the previous step returns a true evaluation.

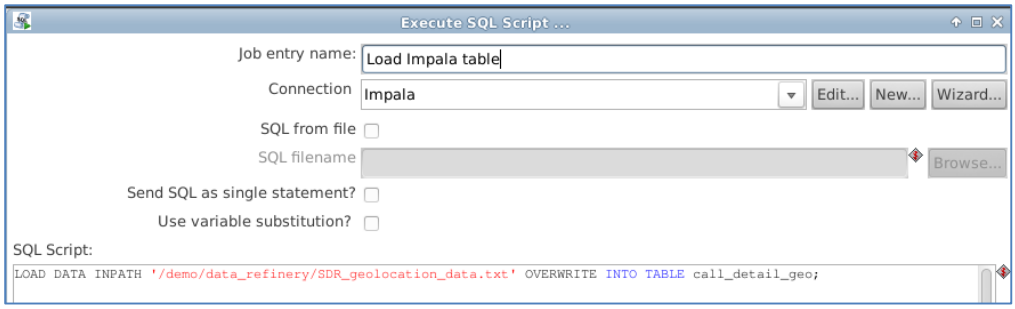
Tip: To change the hop color, click the green arrow.

45. Create a hop from the **Does Impala table exist?** step to the **Create Impala table** step below. This hop should be red in color indicating this path will be taken if the previous step returns a false evaluation.
46. Create a hop from the **Create Impala table** step to the **Load Impala table** step. The hops should match the following image:



47. Double click the **Load Impala table** step to edit its properties.
48. In the **Connection** field select `Impala`.

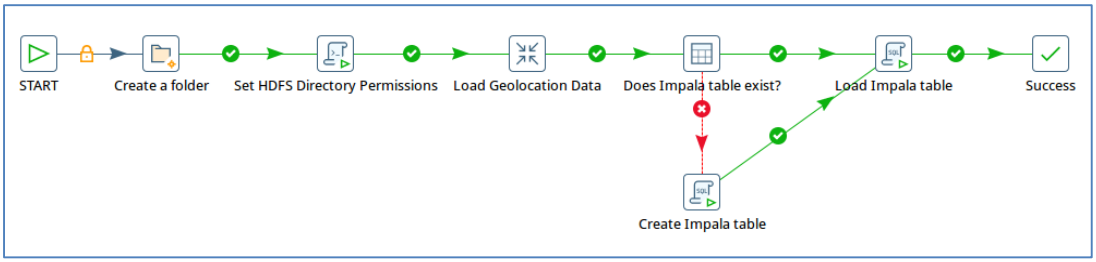
49. In the **SQL Script** section type the following: `LOAD DATA INPATH '/demo/data_refinery/SDR_geolocation_data.txt' OVERWRITE INTO TABLE call_detail_geo;`
50. Your SQL script step should now look like this:



51. Click **OK** to return to the canvas.
52. Double click the **Create Impala table** step to edit its properties.
53. In the **Connection** field select `Impala`.
54. In the **SQL Script** section type or copy/paste the following code which can be found in the `bdi_workshop_code.txt` file on your machine:

```
CREATE TABLE call_detail_geo
(
  calldate STRING,
  source_number STRING,
  home_latitude DOUBLE,
  home_longitude DOUBLE,
  distance DOUBLE,
  direction STRING,
  location_category STRING,
  new_lat DOUBLE,
  new_long DOUBLE
)
row format delimited
fields terminated by '|'
STORED AS TEXTFILE;
```

55. Click **OK** to return to the canvas.
56. Create a hop from the **Load Impala table** step to the **Success** step.
57. Your PDI job should now look like this:



58. From the **File** menu, choose **Save**.
59. Execute the job by choosing **Action** → **Run** from the main menu or by clicking the run icon ▷.
60. The **Execute a job** dialog will appear. Click the **Launch** button at the bottom.
61. A green check will appear on the **Success** step when the job finishes without errors.
62. Keep this job open as it will be used in the next exercise.

PDI Exercise 2: Extend the PDI job to blend data and load to multiple locations

Pentaho Data Integration (PDI) allows you to join data from multiple tables, transform it, and load it to multiple locations. Now that we have both the geolocation data and the call detail records data in Impala, we can take advantage of Impala queries to join two large tables into a combined data set. This combined set is loaded into a new combined Impala table and to a PostgreSQL database to enable high performance OLAP analysis.

63. Make sure the job, `SDR_GeoLocation_Impala_Job`, created in the previous exercise is open.
64. Select the **Success** step and delete it.

Now we need to blend our geolocation data with our call data records, and place the combined data set into a new Impala table using a SQL script.

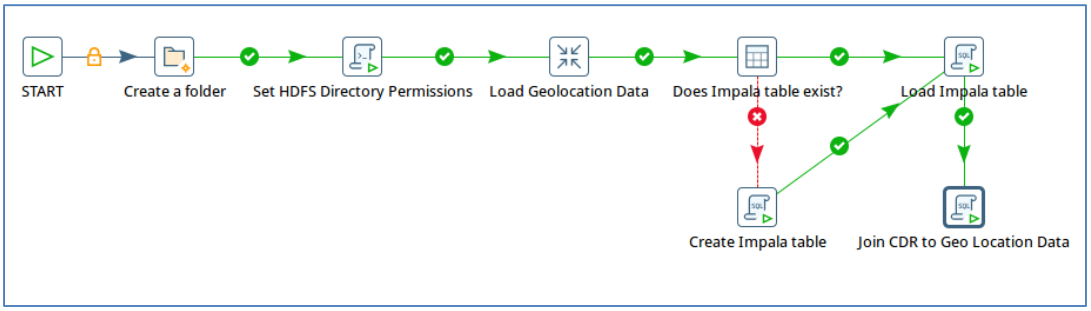
65. From the **Design** tab on the left, expand the **Scripting** folder and drag the **SQL** step onto the canvas below the **Load Impala Table** step.
66. Create a hop between the **Load Impala Table** step and the **SQL** step.
67. Double click the **SQL** step to edit its properties. In the **Step name** field type `Join CDR to Geo Location Data`.
68. In the **Connection** field select `Impala`.
69. In the **SQL** field type or copy/paste in the following code which can be found in the `bdi_workshop_code.txt` file on your machine:

```
DROP TABLE IF EXISTS call_detail_combined;
CREATE TABLE call_detail_combined
  (
```

```
    key STRING
  , source_number STRING
  , call_date STRING
  , call_month INT
  , call_year INT
  , day_of_week INT
  , area_code STRING
  , state STRING
  , country STRING
  , time_zone STRING
  , weekday STRING
  , num_calls INT
  , home_latitude DOUBLE
  , home_longitude DOUBLE
  , distance DOUBLE
  , direction STRING
  , location_category STRING
  , new_lat DOUBLE
  , new_long DOUBLE
  );

INSERT INTO TABLE call_detail_combined
SELECT
    cdr.key
  , cdr.source_number
  , cdr.call_date
  , cdr.call_month
  , cdr.call_year
  , cdr.day_of_week
  , cdr.area_code
  , cdr.state
  , cdr.country
  , cdr.time_zone
  , cdr.weekday
  , cdr.num_calls
  , cdg.home_latitude
  , cdg.home_longitude
  , cdg.distance
  , cdg.direction
  , cdg.location_category
  , cdg.new_lat
  , cdg.new_long
FROM
call_detail_records cdr JOIN call_detail_geo cdg ON
(cdr.source_number = cdg.source_number);
```

70. Click **OK** to return to the canvas. Your job should match the following screenshot.



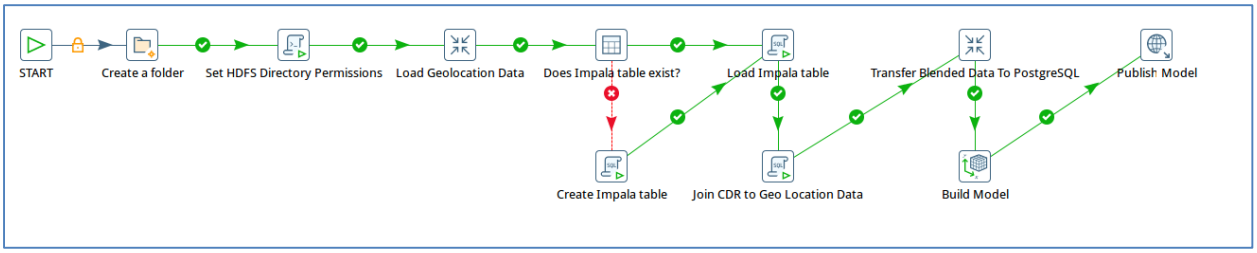
To enable high-performance OLAP analysis and reporting, we would also like to load the combined data set to a PostgreSQL database.

71. From the **Design** tab on the left, expand the **General** folder and drag the **Transformation** step onto the canvas to the right of the **Load Impala Table** step.
72. Create a hop between the **Join CDR to Geo Location Data** step and the **Transformation** step.
73. Double click the **Transformation** step to edit its properties. In the **Name of job entry** field type `Transfer Blended Data to PostgreSQL`.
74. On **Transformation Specification** tab click the browse icon for the **Transformation filename** field and browse to select the following file: `/pentaho/shared_content/WorkshopTraining/02_data_refinery/Solutions/SDR_GeoCDR_Transfer_To_postgres.ktr`

    Note: to save time and reduce redundant work, you are using an *existing* transformation to load the blended data to PostgreSQL.

Now that our data is blended, we need to create and publish a model to the Pentaho Analytics server for web-based dimensional analysis and reporting.

75. From the **Design** tab on the left, expand the **Modeling** folder and drag the **Build Model** step onto the canvas below the **Transfer Blended Data to PostgreSQL** step.
76. Also from the **Modeling** folder drag the **Publish Model** step onto the canvas to the right of the **Transfer Blended Data to PostgreSQL** step.
77. Create a hop between the **Transfer Blended Data to PostgreSQL** step and the **Build Model** step.
78. Create a hop between the **Build Model** step and the **Publish Model** step to match the following screenshot.



Note: the build model and publish model steps will automatically create and publish a metadata model to the analytics server for web-based dimensional analysis on the blended data created by this job.

79. Double click the **Build Model** step to edit its properties. In the **Model Name** field type `SDR Geo Cube` and for the **Output Step** field, confirm it is set `Postgres Output`.
80. Double click the **Publish Model** step to edit its properties.
81. Check **Replace Existing Published Model**.
82. For the **URL** enter: http://localhost:${CURRENT_PENTAHO_BA_PORT}/pentaho/

The ${CURRENT_PENTAHO_BA_PORT} value referenced above is a variable which is storing the current port the Pentaho Business Analytics server is running on. This port may change. This variable value is set in

/home/[OS Uername]/.kettle/kettle.properties

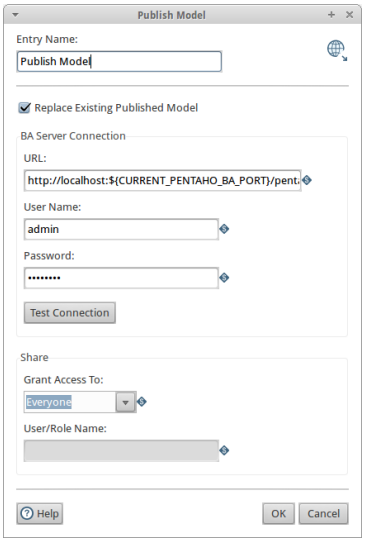
83. Login to the Pentaho User Console

    For Pentaho User Console credentials see the document on the Desktop in the Docs folder:

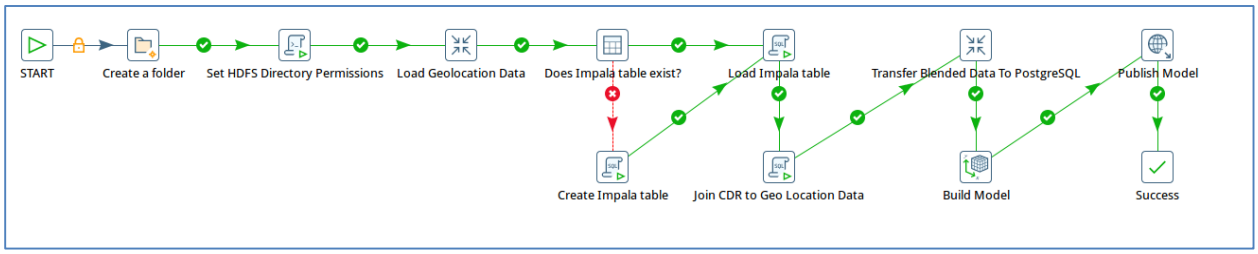
    a. hds - BDI Workshop Credentials.pdf

    -or-

    b. bdiw - BDI Workshop Credentials.pdf
84. The remaining fields can be left as the default
85. Click **Test Connection**.



86. From the **Design** tab on the left, expand the **General** folder and drag the **Success** step onto the canvas below the **Publish Model** step.
87. Create a hop between the **Publish Model** step and the **Success** step to match the following screenshot.



88. From the **File** menu, choose **Save**.
89. Execute the job by choosing **Action** → **Run** from the main menu or by clicking the run icon ▷.
90. The **Execute a job** dialog will appear. Click the **Launch** button at the bottom.
91. A green check will appear on the **Success** step when the job finishes without errors.

Congratulations, you have completed the data integration work for implementing the streamlined data refinery (SDR) use case! The next section contains exercises for analyzing the data you loaded to PostgreSQL.

Part 2: Explore data in PostgreSQL with Pentaho Analyzer

Pentaho Analyzer offers an easy to use, graphical drag-and-drop design environment that can be used by anyone who wants to dynamically explore data to discover anomalies or trends and create visualizations. Part Two of the SDR Use Case includes two exercises showcasing how to Pentaho Analyzer for OLAP analysis against PostgreSQL. Your company is considering introducing a new VOIP service. You need to analyze the geo-location and calling data to determine the calling patterns of your customers. You leverage this information to determine the best markets to launch a VOIP pilot service.

The first exercise has you analyze data to determine where calls originate: from the house, the neighborhood, in town or during travel. This helps to determine which calling product has the most potential for this market. The second exercise plots the calls on a map based on the customers' source area code to determine the highest volume geographical markets to target for a new VOIP service.

Analyzer - Exercise 1: Analyze data for a new VOIP pilot service

In this exercise you create a table and column-line combo chart to aggregate call volume and average distance from home by location categories such as: at home, in the neighborhood, in town, during travel, etc.

> Note: In case the Pentaho BA server is not started, you can start it from command line:
> `/pentaho/current_version/ctlscript.sh start`

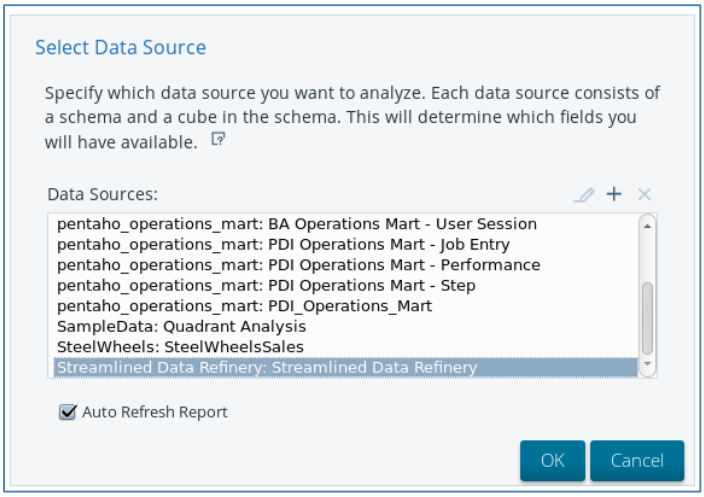
1. With your Firefox browser navigate to `http://localhost:8081/pentaho/Login` to launch the Pentaho User Console (PUC).
2. Login to PUC

   For login credentials, see the document on the Desktop in the Docs folder:

   a. hds - BDI Workshop Credentials.pdf

   -or-

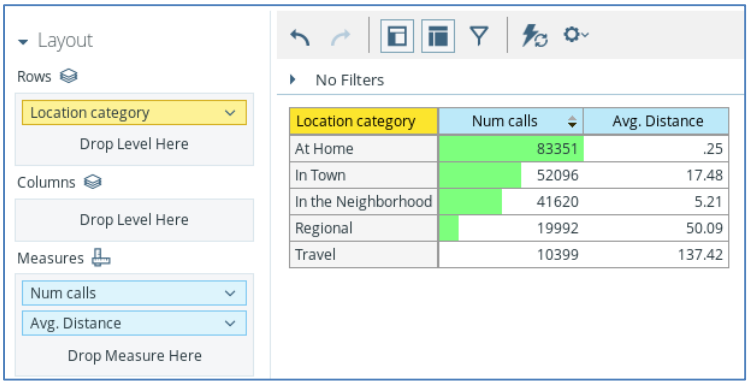
   b. bdiw - BDI Workshop Credentials.pdf
3. Click on the **Create New | Analysis Report** buttons.
4. Select the `Streamlined Data Refinery` data source at the bottom of the **Data Sources** list.



5. This will launch you into a new **Analysis Report** view.
6. From the **Available fields** section on the left, double-click or select and drag `Location Category`, `Num calls`, and `Avg. Distance` to the canvas.

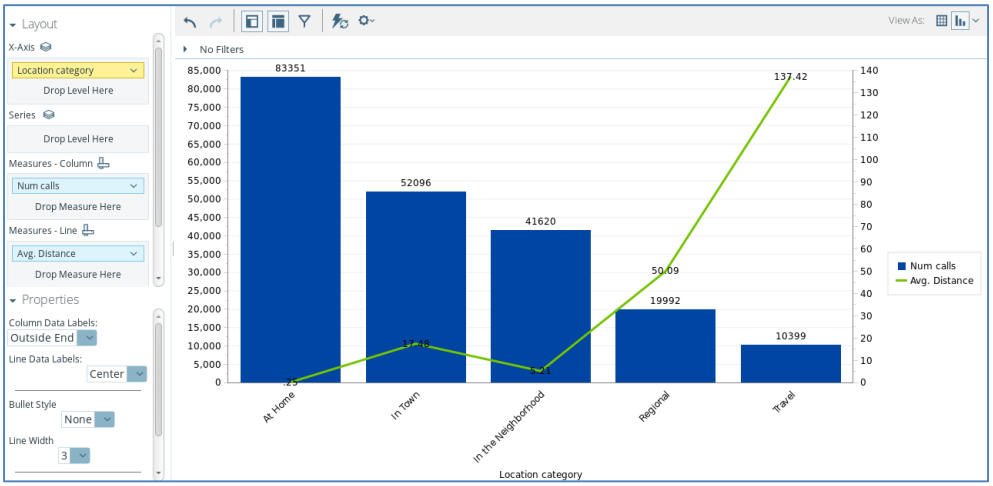
> Notice how the measures automatically aggregate the records in Analyzer.

7. Sort the `Num calls` field in descending order by right clicking the `Num Calls` column header and selecting **Sort Values High->Low**.
8. Add conditional formatting to the `Num Calls` column by right clicking the `Num Calls` column header and selecting **Conditional Formatting | Data Bar: Green**.



> Note that most calls are made `At Home` or `In Town`, and that there is a drop in calls made `In the Neighborhood`. Given the high number of calls made at home, we've verified that a new VOIP calling service for homes may make sense.

9. To visualize the data, click the chart drop down in the top right of your screen and choose **Column-Line Combo**.
10. In the **Layout** section, drag `Avg. Distance` from the **Measures – Column** drop zone down to the **Measures – Line** drop zone.
11. In the **Properties** section, change **Column Data Labels** to `Outside End`, **Line Data Labels** to `Center`, **Bullet Style** to `None`, and **Line Width** to `3`. The resulting chart should match the following image:

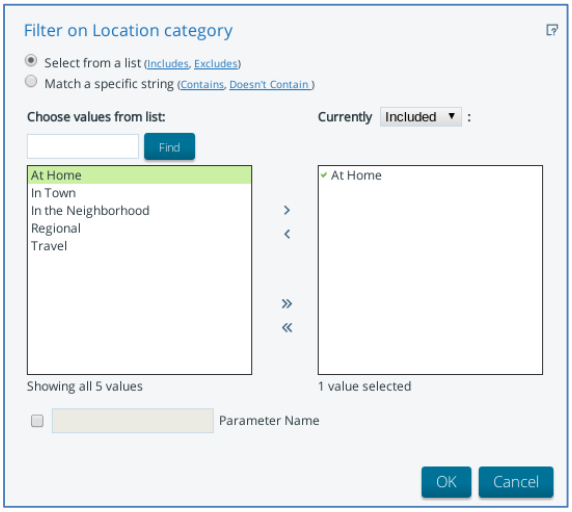


12. Click the Save icon to save the view as `SDR Analyzer Exercise 1` in the default `/home/[usename]` directory.

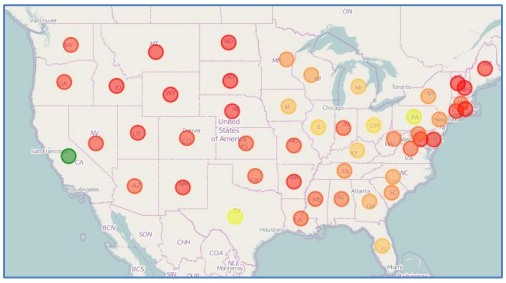
Analyzer - Exercise 2: Analyze and map calling data by geography

Now that you have verified that a VOIP calling plan makes sense, let's determine the geographic region where this service would make the most sense. In this exercise you filter on calls made from home and plot call volume by geography in an interactive map.

1. Click on the **Create New | Analysis Report** buttons.
2. Select the `Streamlined Data Refinery` data source at the bottom of the **Data Sources** list.
3. In the Available Fields section, right-click on `Location category` and choose **Filter** to filter the view where `Location Category` equals `At Home`.

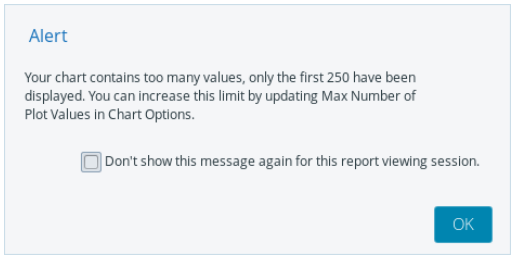


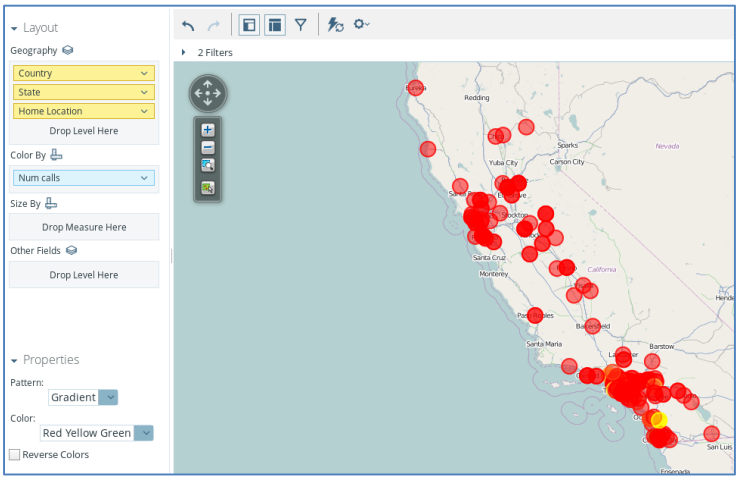
4. Add `Country, State, Num Calls` to the canvas.
5. Click the chart dropdown and select to **Geo Map**.



Note California has the most calls at home as denoted by the darkest green circle.

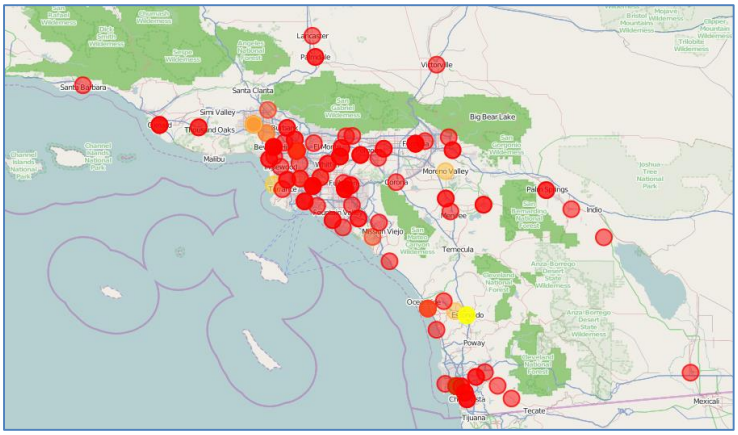
6. Drill into `California` by double clicking on the state's green circle.
7. If presented with the alert below, click OK to display the more detailed map.





If the zoom level is too low, click the center of the navigation controls  to reset the map.

8. Click the lasso filter and then using your mouse to click and drag a lasso filter around the circles representing Southern California, once selected, click the **Keep Only** button.



We can conclude that Southern California is a good place to pilot our VOIP service based on the number of callers who make calls close to home on their cell phones.

Congratulations, you have now completed the SDR use case!

Enterprise Edition Only

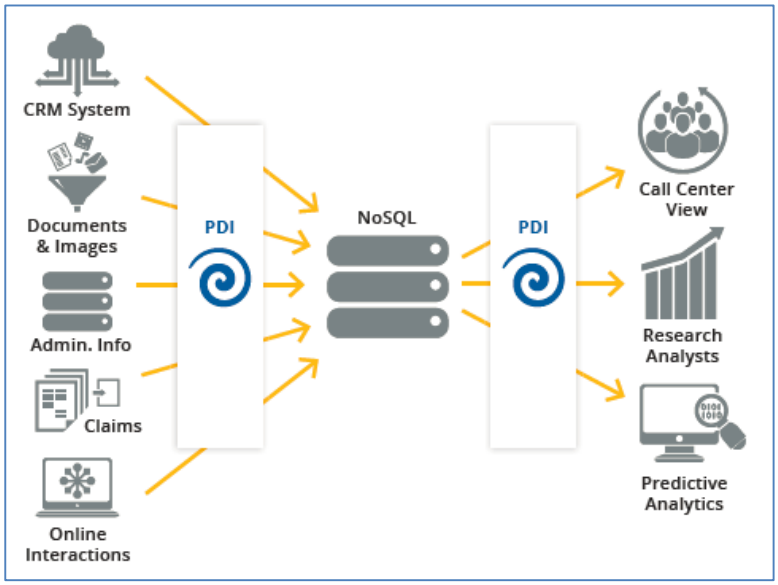
HBase Customer 360 Use Case

What is it?

- Blending a variety of operational & transactional data sources to create an on-demand analytical view across customer touch points and surface the customer experience.
- Providing customer-facing employees and partners with information made available inside everyday line-of-business applications
- Leverages NoSQL database technologies like HBase that provide flexible storage options for single views of data.

Why do it?

- Provide single customer view to sales & services teams, empowering them to grow upsell/cross-sell revenue
- Understand customer perception of your products & services, while decreasing customer churn
- Avoid point-to-point integrations with single repository, and fast queries to improve access to metrics



Value of Pentaho

- Staff savings & productivity: Rapid time to value through drag/drop visual development for big data integration – make big data accessible to all data developers
- Operational Intelligence: Ability to embed analytics into actionable line-of-business applications for each relevant customer-facing role
- Broad & robust data connectivity: Ability to blend traditional sources & big data

- Comprehensive analytics: Visualizations, reports, dashboards, ad hoc analysis provide for all roles

Pentaho components used in this workshop

PENTAHO DATA INTEGRATION (PDI)

ANALYZER

What you will accomplish in this workshop

- Part 1 – Use PDI to create a single view in HBase (4 exercises)
- Part 2 – Visualize data with Analyzer (3 exercises)

Part 1: Use PDI to create a single view in HBase

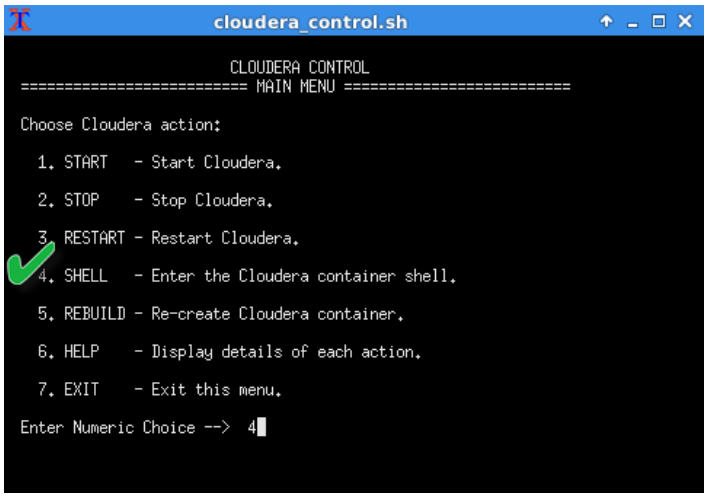
PDI gives users a graphical user interface to build transformations and jobs that solve complex data integration challenges with HBase. PDI provides a scalable solution for migrating to HBase with connectors for all types of disparate data for building out a single customer view. PDI leverages the power of the HBase in-memory database for blending and processing data.

Create the HBase tables

HBase tables must be created via the command line prior to continuing to the exercises. Each table has a corresponding column family. Complete the following steps to create the tables to store the HBase data.

1. Launch the Cloudera Control program by clicking the launch menu icon

2. Enter 4 to select and launch the Cloudera shell (follow prompts)



3. In new terminal, type “hbase shell” and enter the command

4. In hbase shell, create the tables and column families by executing the three commands below

   a. `hbase(main):003:0> create 'customers', 'info'`

   b. `hbase(main):003:0> create 'webevents', 'info'`

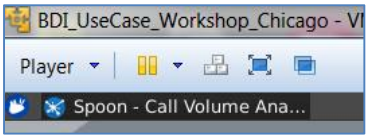
   c. `hbase(main):003:0> create 'pos', 'info'`

PDI Exercise 1: Create a transformation to load customer records to HBase

This first exercise steps you through the process of creating a transformation to load customer master records from a CSV file, calculate the customer age, and then load the results into the HBase customers table.

9. If PDI is not already open, launch it from the launch menu icon at the bottom of your screen. Click this icon just *once* to launch Spoon, the client-based authoring GUI for PDI.

10. You will see the PDI splash screen appear while PDI loads. All open applications appear in the top left section of your screen. You can see Spoon is open in the following screenshot.



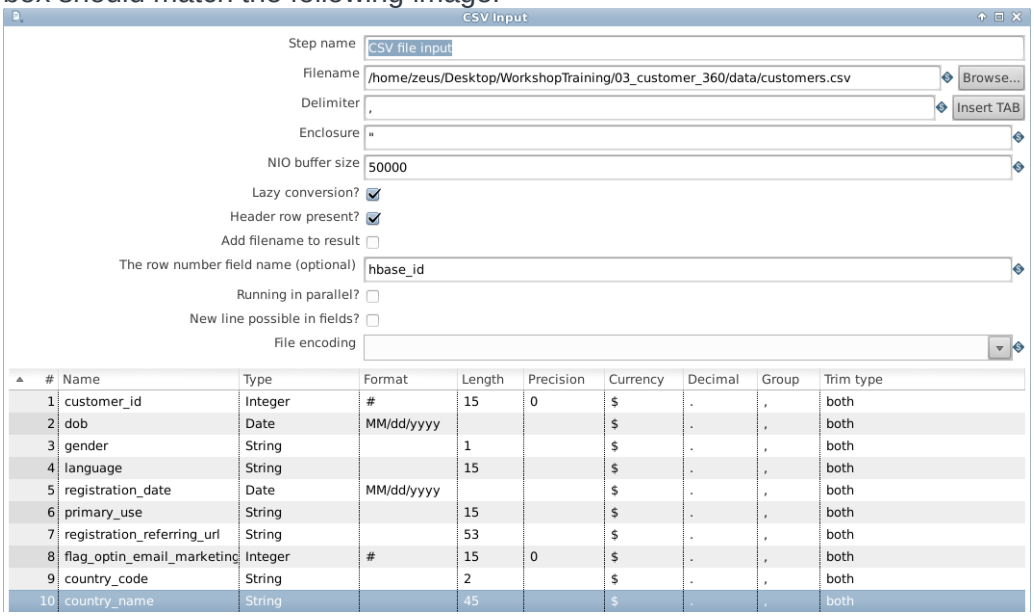
11. From the main menu choose **File | New | Transformation**

12. **File | Save** to `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360_hbase/`**`t_load_customers.ktr`**

We need to access Customer data from within a flat file. The file format is a csv file, so you will access the data by configuring a CSV file input step.

13. From the **Design** tab on the left, expand the **Input** folder and drag **CSV file input** onto the canvas
14. Double-click on **CSV file input** to open its properties
15. Browse to the directory `/pentaho/shared_content/WorkshopTraining/03_customer_360_hbase/data/` and select `customers.csv`.
16. In **The row number field name (optional)** type `hbase_id`
17. Click the **Get Fields** button at the bottom and enter `50,000` for the **Sample Size**.
18. Once the scan is finished, click **Close** to close the scan results.
19. Trim any possible source data character spaces by setting **Trim Type** to `both` for every field.

Click the **Preview** button to preview the data and then click **Close**. Your CSV Input dialog box should match the following image:



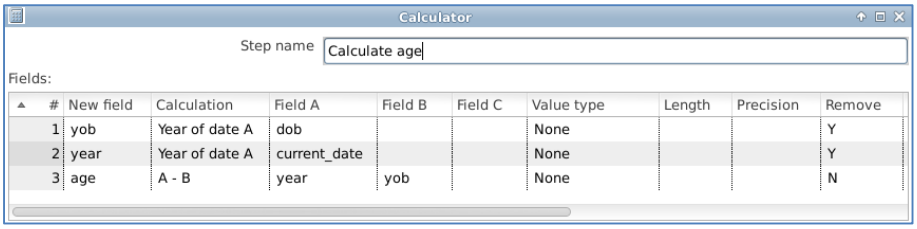
20. Click **OK** to return to the canvas.

To calculate the age of the customer, we need to know the current year and the year in which the customer was born. But to calculate the current year, we need to know the current date. We will use the **Get System Info** step to learn the current date.

21. From the **Design** tab on the left, expand the **Input** folder and drag **Get System Info** onto the canvas.
22. Double-click on **Get System Info** step to open its properties.
23. In the **Name** field type `current_date`
24. In the **Type** field, select from the drop-down menu, `system date (variable)`
25. Click **OK** to return to the canvas.
26. To draw a hop between two steps, shift-click the **CSV file input** step and while holding down your mouse key, drag a **hop** over to the **Get System Info** step. When prompted, select **Main output of step**.

Now that we have the current date, we need to derive the current year from the date. And we need to know the year in which the customer was born, which is based on the customer date of birth. With both years in hand, we can calculate the age of the customer. All of these calculations can be performed with the **Calculator** step.

27. From the **Design** tab on the left, expand the **Transform** folder and drag the **Calculator** step onto the canvas.
28. Draw a hop between the **Get System Info** and **Calculator** steps.
29. Double-click on **Calculator** step to open its properties.
30. In the **Step name** box type `Calculate age`
31. Follow these steps to create the first year of birth calculation: In the **New Field** column type `yob`, in the **Calculation** column select `Year of date A`, in the **Field A** column select the `dob` field, and finally in the **Remove** column select `Y`.
32. Follow these steps to create the second year calculation: In the **New Field** column type `year`, in the **Calculation** column select `Year of date A`, in the **Field A** column select the `current_date` field, and finally in the **Remove** column select `Y`.
33. Follow these steps to create the third customer age calculation: In the **New Field** column type `age`, in the **Calculation** column select `A - B`, in the **Field A** column select the `year` field, in the **Field B** column select the `yob` field, and finally in the **Remove** column, make sure it is set to `N`.
34. Once your **Calculate age** step matches the following image, click **OK** to return to the canvas.



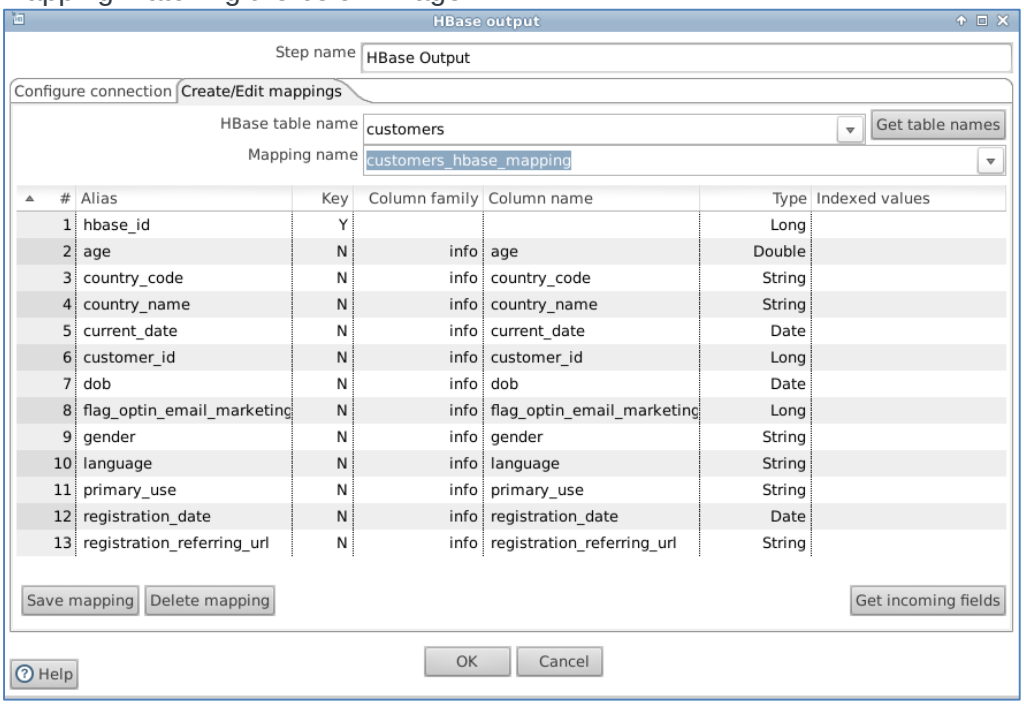
#	New field	Calculation	Field A	Field B	Field C	Value type	Length	Precision	Remove
1	yob	Year of date A	dob			None			Y
2	year	Year of date A	current_date			None			Y
3	age	A - B	year	yob		None			N

Customer data has been enriched with age information, and we are ready to write the data to HBase.

35. From the **Design** tab on the left, expand the **Big Data** folder and drag **HBase Output** onto the canvas.
36. Draw a hop between the **Calculate Age** and **HBase Output** steps. The resulting transformation should match the following image:

37. Double-click on **HBase Output** step to open its properties.
38. In the **Configure connection** tab, from the **Hadoop cluster** drop-down select `CDH`.
39. Select the **Create/Edit mappings** tab
40. Click the **Get table names** button and select `customers` from the list.

41. Click the **Get incoming fields** button and select `hbase_id` as the Key, with the final mapping matching the below image:



#	Alias	Key	Column family	Column name	Type	Indexed values
1	hbase_id	Y			Long	
2	age	N	info	age	Double	
3	country_code	N	info	country_code	String	
4	country_name	N	info	country_name	String	
5	current_date	N	info	current_date	Date	
6	customer_id	N	info	customer_id	Long	
7	dob	N	info	dob	Date	
8	flag_optin_email_marketing	N	info	flag_optin_email_marketing	Long	
9	gender	N	info	gender	String	
10	language	N	info	language	String	
11	primary_use	N	info	primary_use	String	
12	registration_date	N	info	registration_date	Date	
13	registration_referring_url	N	info	registration_referring_url	String	

42. In **Mapping name** enter `customers_hbase_mapping` and then click the S**ave Mapping** button. Click **OK** to exit.
43. Once again, double-click on **HBase Output**
44. In the **Configure connection** tab, click the button **Get table names** and from the **HBase table name** drop-down select `customers`.
45. In the **Configure connection** tab, click the **Get mappings for the specified table** and from the **Mapping name** drop-down select `customers_hbase_mapping`. Click **OK** to exit**.**
46. Click the **Launch** button at the bottom of the **Execute a transformation** dialog box. *NOTE: If you get an error about not being able to connect to HBase, or no records are loading to HBase, restart Cloudera using the Cloudera Control program (option 3).*
47. This transformation will load 19,673 customers into the HBase table customers. You will see green checkmarks on each box when the transformation has succeeded:

PDI Exercise 2: Create a transformation to load web events to HBase

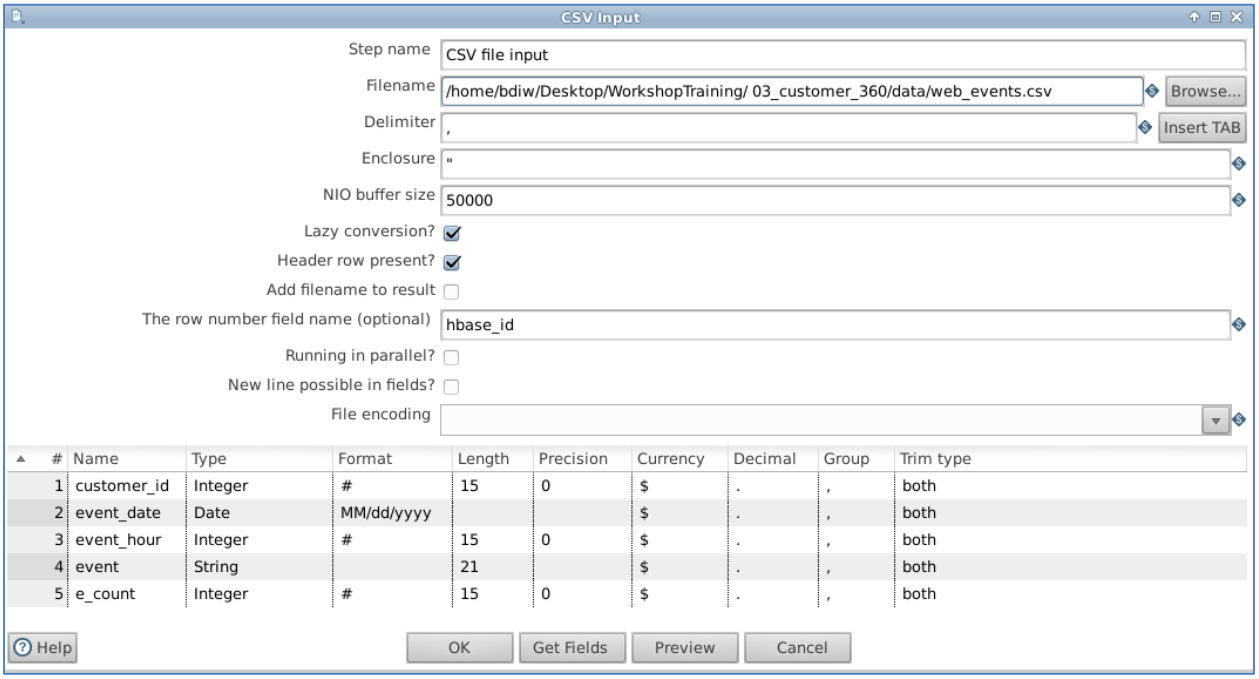
This second exercise steps you through the process of creating a transformation to load the second source of data, customer web clickstream (web event) records, from a CSV file into the `webevents` HBase table created prior to exercise 1.

48. From the main menu choose **File** | **New** | **Transformation**

49. **File | Save** to /pentaho/shared_content/WorkshopTraining/student_files/03_customer_360_hbase/t_load_web_events.ktr

To load customer web events data from a CSV file into the HBase table, first we need to configure a CSV file input step to access the web events.

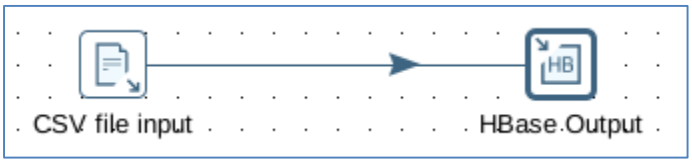
50. From the **Design** tab on the left, expand the Input folder and drag **CSV file input** onto the canvas**.**
51. Double-click on **CSV file input** to open its properties
52. Browse to the directory `/pentaho/shared_content/WorkshopTraining/03_customer_360_hbase/data/` and select `web_events.csv`.
53. In the field “The row number field name (optional)”, enter `hbase_id`
54. Click the **Get Fields** button at the bottom and enter `50,000` for the **Sample Size**.
55. Click **Close** to close the scan results.
56. Click the **Preview** button to preview the data and then click **Close**. Your **CSV Input** dialog box match the following image:



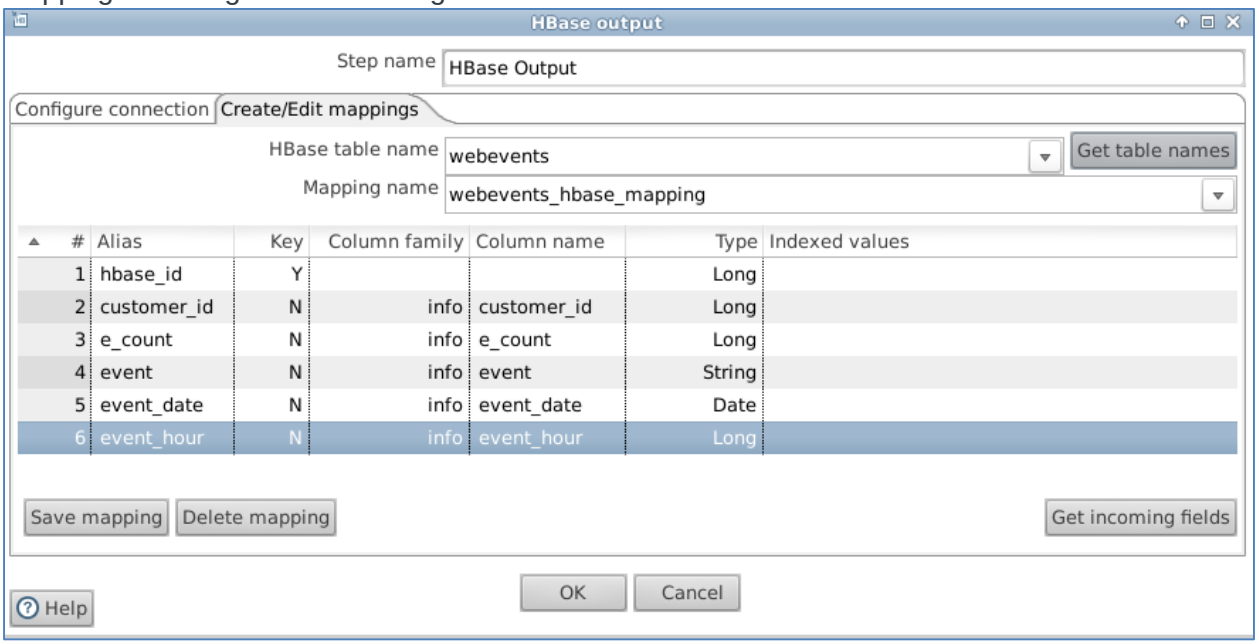
57. Click **OK** to return to the canvas.

The HBase Output step will truncate the target table prior to loading it.

58. From the **Design** tab on the left, collapse the **Input** folder and expand the **Big Data** folder; then, select and drag **HBase Output** onto the canvas.
59. Shift-click the **CSV file input** step and while holding down your mouse key, drag a **hop** over to the **HBase Output** step. When prompted, select **Main output of step**.



60. Double-click on **HBase Output** step to open its properties.
61. In the **Configure connection** tab, from the **Hadoop cluster** drop-down select `CDH 5.3.`
62. Select the **Create/Edit mappings** tab
63. Click on **Get table names** button and select `webevents` from the list.
64. Click the **Get incoming fields** button and enter `hbase_id` as the Key, with the final mapping matching the below image:



65. In **Mapping name** enter `webevents_hbase_mapping` and then click the **Save Mapping** button. Click **OK** to exit.
66. Double-click on **HBase Output.**
67. In the **Configure connection** tab, click the **Get table names** button and from the **HBase table name** drop-down select `webevents`.
68. In the **Configure connection** tab, click the **Get mappings for the specified table** and from the **Mapping name** drop-down select `webevents_hbase_mapping`**.** Click **OK** to exit.
69. Click the **Launch** button at the bottom of the **Execute a transformation** dialog box.

70. This transformation will load 51,419 web event records into the HBase table `webevents`. You will see green checkmarks on each box when the transformation has succeeded:

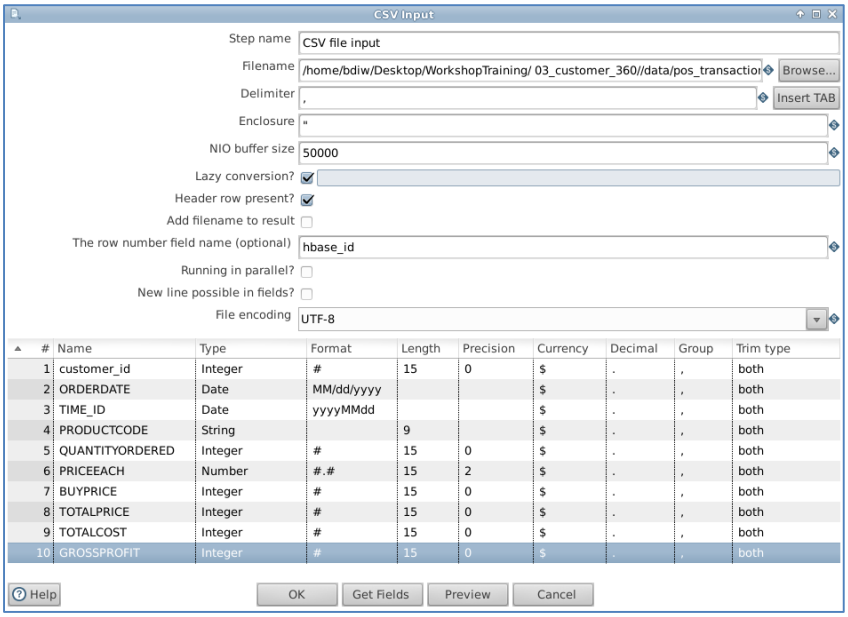
PDI Exercise 3: Create a transformation to load POS transactions to HBase

This third exercise steps you through the process of creating a transformation to load the third and final data source, customer point-of-sale (POS) sale transaction records, from a CSV file into the **`pos`** HBase table created prior to exercise 1. You use the number range step to bin sales orders into buckets based on the size of the order.

71. From the main menu choose **File** | **New** | **Transformation**
72. **File | Save** to `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360_hbase/t_load_pos.ktr`

To blend point-of-sale transactions from a CSV file into the HBase data collection, we need to configure a CSV file input step to access the transactions.

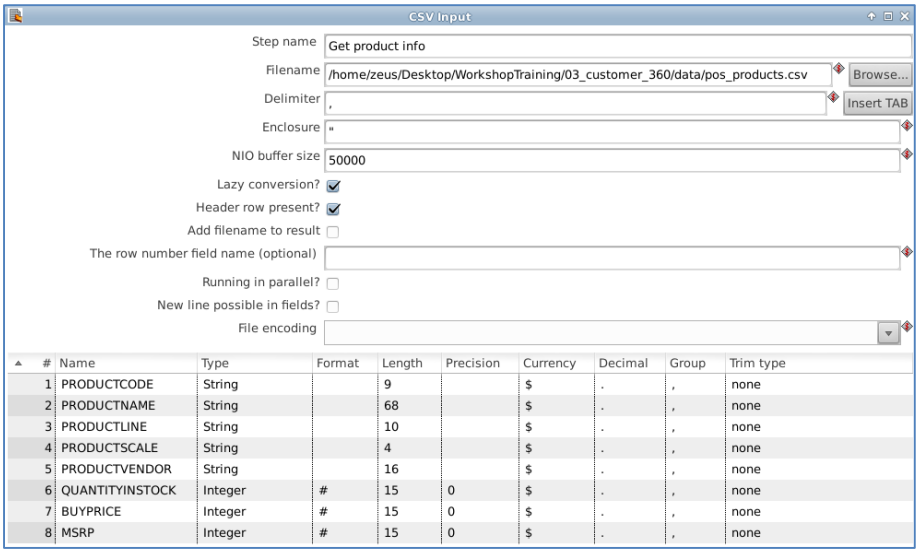
73. From the **Design** tab on the left, expand the **Input** folder and drag **CSV file input** onto the canvas.
74. Double-click on **CSV file input** to open its properties
75. Browse to the directory `/pentaho/shared_content/WorkshopTraining/03_customer_360_hbase/data/` and select `pos_transactions.csv`.
76. In the field **The row number field name (optional)**, enter `hbase_id`
77. Click the **Get Fields** button at the bottom and enter `5,000` for the **Sample Size**.
78. Click the **Preview** button to preview the data and then click **Close**. Your **CSV Input** dialog box should match the following image:



79. Click **OK** to return to the canvas.

The POS file only has product code on the transaction. We need to enhance the data with Product Name, Product Line, and Product Vendor. To do so, we will look up the product information in a flat file to match the transaction product code with its relevant name, line, and vendor.

80. From the **Design** tab on the left, expand the **Lookup** folder and drag **Stream lookup** onto the canvas.
81. Create a hop between the **CSV file input** and **Stream lookup** steps.
82. From the **Design** tab on the left, expand the **Input** folder and drag a second **CSV file input** onto the canvas and place it above the Stream lookup step.
83. Double-click on **CSV file input** to open its properties
84. Change the **Step name** to `Get product info`
85. Browse to the directory `/pentaho/shared_content/WorkshopTraining/03_customer_360_hbase/data/` and select `pos_products.csv`.
86. Click the **Get Fields** button at the bottom and enter `5,000` for the **Sample Size**.
87. Click the **Preview** button to preview the data and then click **Close**. Your **CSV Input** dialog box should look like the following image:

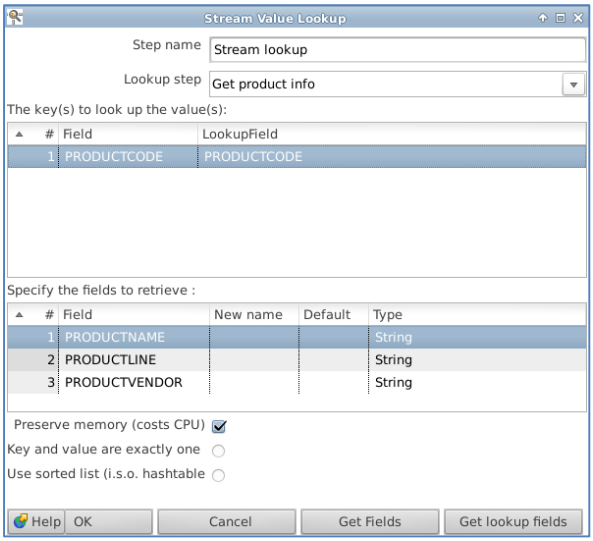


#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	PRODUCTCODE	String		9		$	.	,	none
2	PRODUCTNAME	String		68		$	.	,	none
3	PRODUCTLINE	String		10		$	.	,	none
4	PRODUCTSCALE	String		4		$	.	,	none
5	PRODUCTVENDOR	String		16		$	.	,	none
6	QUANTITYINSTOCK	Integer	#	15	0	$	.	,	none
7	BUYPRICE	Integer	#	15	0	$	.	,	none
8	MSRP	Integer	#	15	0	$	.	,	none

88. Click **OK** to return to the canvas
89. Create a hop from the second **Get product info** step to the **Stream lookup** step.

We need to configure the Stream Lookup to retrieve Product Name, Product Line, and Product Vendor from the lookup file for every product code we find in the POS records.

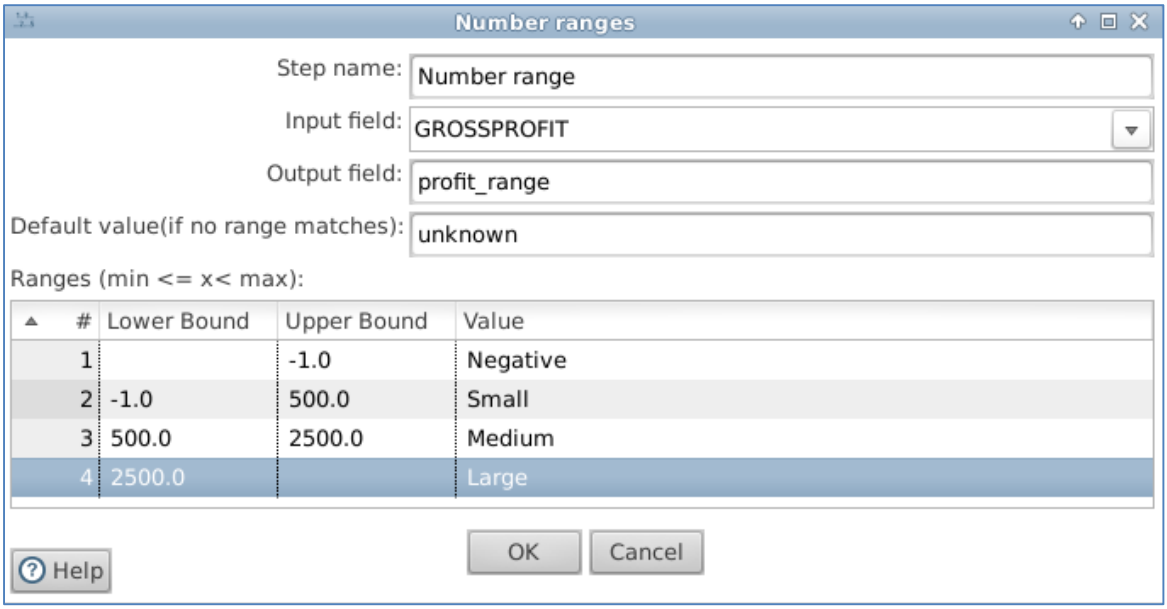
90. Double-click on **Stream lookup** step to open its properties
91. In the **Lookup step** select `Get product info`.
92. In the **Key(s) to Lookup Value(s)** section select `PRODUCTCODE` in the **Field** column and select `PRODUCTCODE` as the **LookupField** column.
93. Click the **Get Lookup Fields** button to populate the fields to retrieve section at the bottom.
94. Highlight and delete the following fields: `PRODUCTCODE, PRODUCTSCALE, QUANTITYINSTOCK, BUYPRICE, MSRP` from the fields to retrieve section. Your **Stream Lookup** dialog box should now match the following image:



95. Click **OK** to return to the canvas.
96. From the **Design** tab on the left, expand the **Transform** folder and drag **Select Values** onto the canvas.
97. Click **OK** to return to the main canvas.

It may be useful to categorize the sale orders into buckets for further analysis, based on order size. We can create a categorization of small, medium, and large.

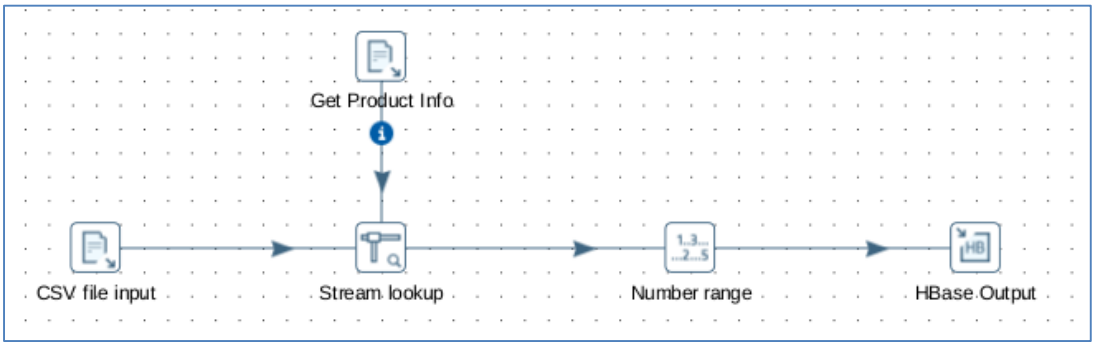
98. From within the **Transform** folder, select and drag **Number Range** onto the canvas.
99. Create a hop between the **Stream lookup** and **Number Range** steps.
100. Double-click on **Number Range** step to open its properties.
101. For **Input field** select `GROSSPROFIT` from the list.
102. For **Output field** type `profit_range`.
103. Enter the following values for **Lower Bound**, **Upper Bound** and **Value** columns.



104. Click **OK** to return to the canvas.

105. From the **Design** tab on the left, expand the **Big Data** folder and drag **HBase Output** onto the canvas.

106. Create a hop between the **Number Range** and **HBase Output** step. The resulting transformation should match the following image:

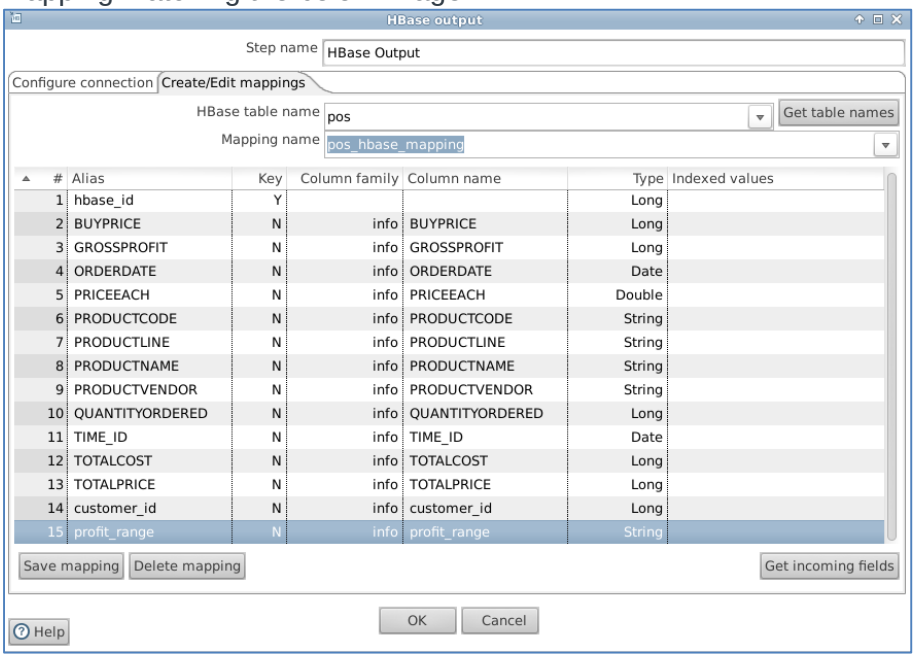


107. Double-click on **HBase Output step** to open its properties. In the **Configure connection** tab, from the **Hadoop cluster** drop-down select `CDH 5.3`.

108. Select the **Create/Edit mappings** tab

109. Click on **Get table names** button and select `pos` from the list.

110. Click the `Get incoming fields` button and define `hbase_id` as the Key, with the final mapping matching the below image:



#	Alias	Key	Column family	Column name	Type	Indexed values
1	hbase_id	Y			Long	
2	BUYPRICE	N	info	BUYPRICE	Long	
3	GROSSPROFIT	N	info	GROSSPROFIT	Long	
4	ORDERDATE	N	info	ORDERDATE	Date	
5	PRICEEACH	N	info	PRICEEACH	Double	
6	PRODUCTCODE	N	info	PRODUCTCODE	String	
7	PRODUCTLINE	N	info	PRODUCTLINE	String	
8	PRODUCTNAME	N	info	PRODUCTNAME	String	
9	PRODUCTVENDOR	N	info	PRODUCTVENDOR	String	
10	QUANTITYORDERED	N	info	QUANTITYORDERED	Long	
11	TIME_ID	N	info	TIME_ID	Date	
12	TOTALCOST	N	info	TOTALCOST	Long	
13	TOTALPRICE	N	info	TOTALPRICE	Long	
14	customer_id	N	info	customer_id	Long	
15	profit_range	N	info	profit_range	String	

111. In **Mapping name** enter `pos_hbase_mapping` and then click the **Save Mapping** button. Click **OK** to exit.

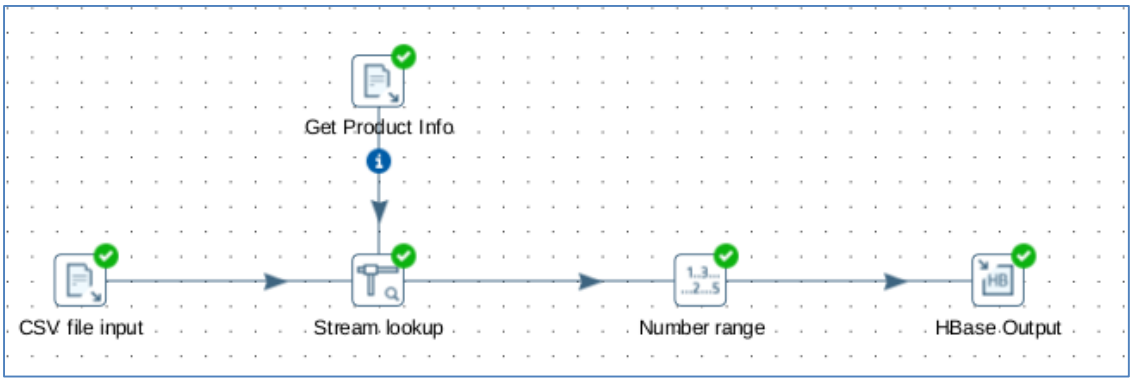
112. Double click **HBase Output**

113. In the **Configure connection** tab, click the **Get table names** button and then from the **HBase table name** drop-down select `pos`

114. In the **Configure connection** tab, click the **Get mappings for the specified table** button and from the **Mapping name** drop-down select `pos_hbase_mapping`. Click **OK** to exit**.**

115. Click the **Launch** button at the bottom of the **Execute a transformation** dialog box.

116. This transformation will load 991 POS records into the HBase table `pos`. You will see green checkmarks on each box when the transformation has succeeded:

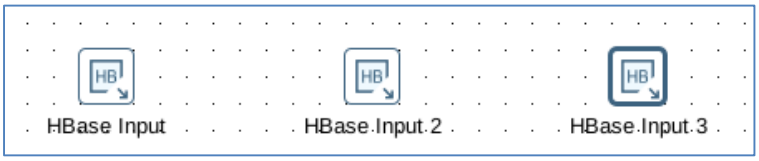


PDI Exercise 4: Create a transformation to merge and annotate the three HBase tables

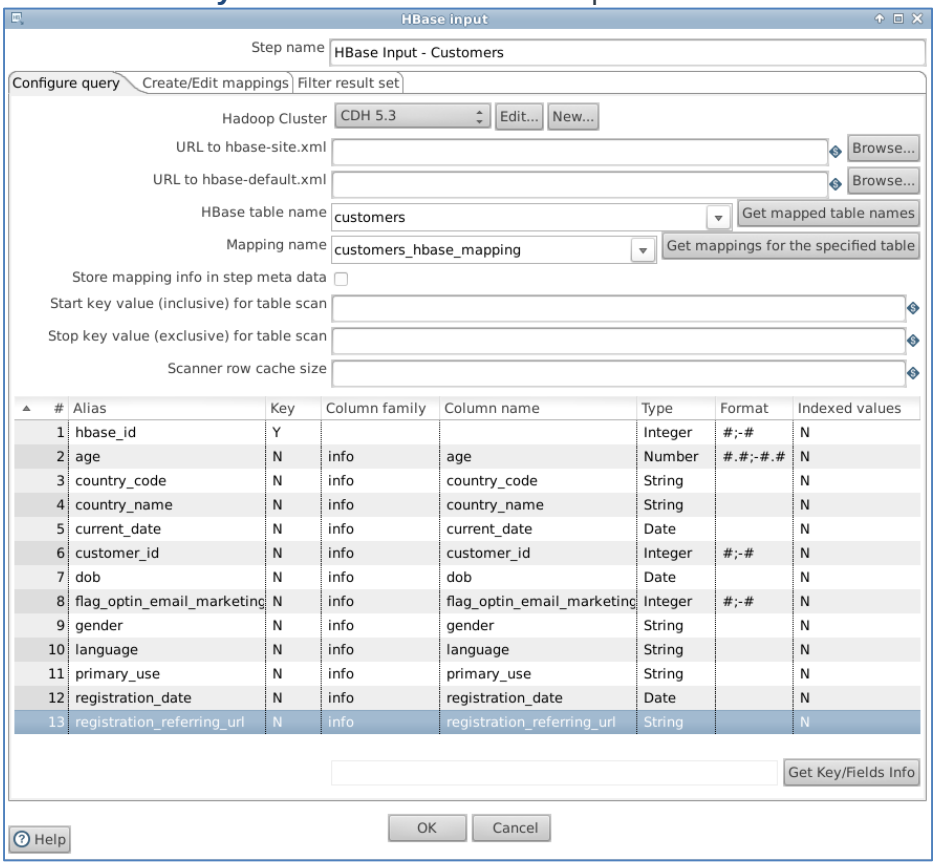
In the fourth exercise, you merge the three HBase tables and prepare them for Analyzer Reports. The final output is to a PostgreSQL database where the data is persisted for reporting.

Note that sections of the final transformation are shown in the following steps with the whole image appearing at the end of this exercise. You will be using a blank, pre-existing transformation to utilize an existing database connection.

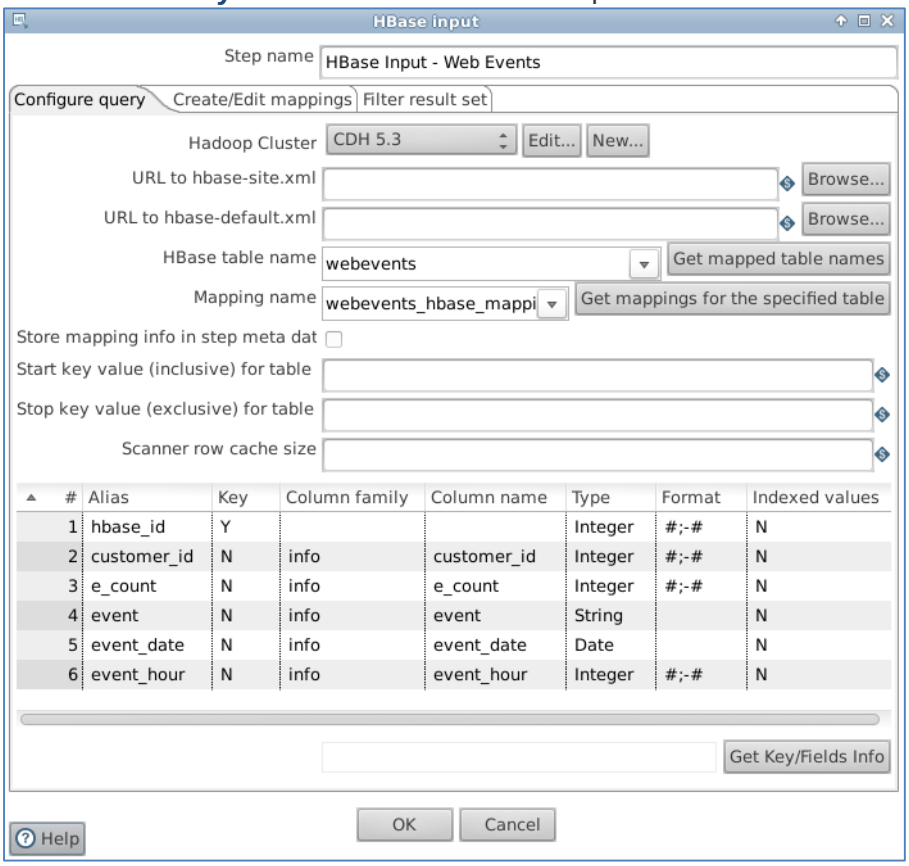
117. **File | Open…** the transformation `/pentaho/shared_content/WorkshopTraining/03_customer_360_mongodb/pdi_transformations/t_load_postgres.ktr`
118. From the **Design** tab on the left, expand the **Big Data** folder and drag **HBase Input** onto the canvas.
119. Repeat step one for a total of 3 HBase Input steps:



120. Double-click the **HBase Input** step to open its properties.
121. Change **Step name** to `HBase Input - Customers`.
122. From the **Hadoop Cluster** drop-down select `CDH 5.3`.
123. Click the **Get mapped table names** button and select `customers` from the drop-down.
124. Click **Get mappings for the specified table** button and select `customers_hbase_mapping` from the drop-down.
125. Click the **Get Key/Fields info** button. The step should now match the image below:

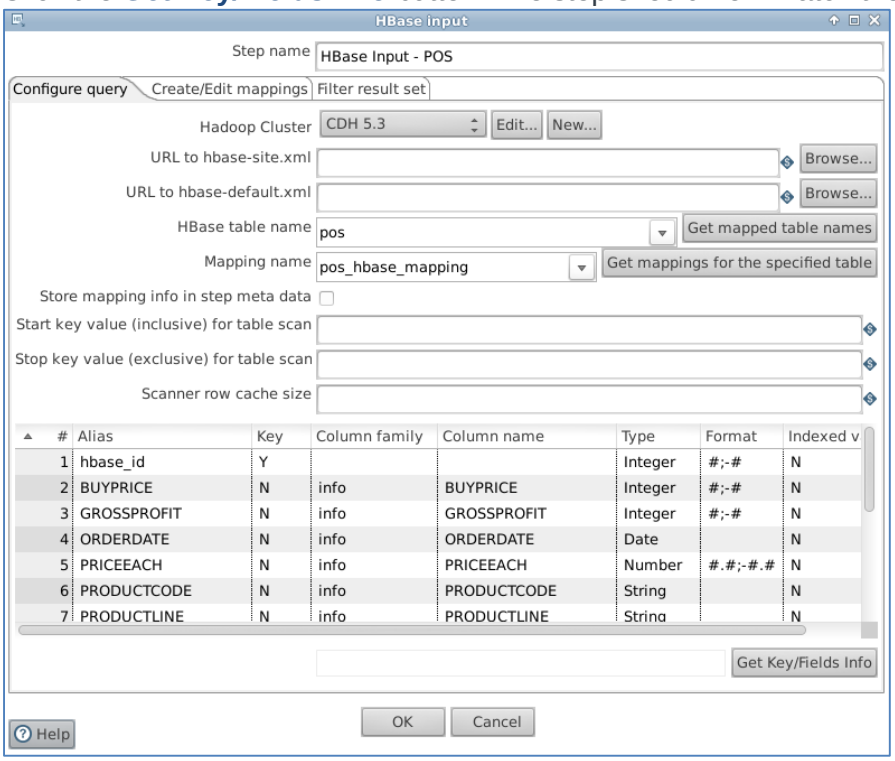


126. Open the **HBase Input 2** step and change **Step name** to `HBase Input - Web Events`.
127. From the **Hadoop Cluster** drop-down select `CDH 5.3`.
128. Click the **Get mapped table names** button and select `webevents` from the drop-down.
129. Click **Get mappings for the specified table** button and select `webevents_hbase_mapping` from the drop-down.
130. Click the **Get Key/Fields info** button. The step should now match the image below:



131. Open the **HBase Input 3** step and change **Step name** to `HBase Input - POS`.
132. From the **Hadoop Cluster** drop-down select `CDH 5.3`.
133. Click the **Get mapped table names** button and select `pos` from the drop-down.
134. Click **Get mappings for the specified table** button and select `pos_hbase_mapping` from the drop-down.

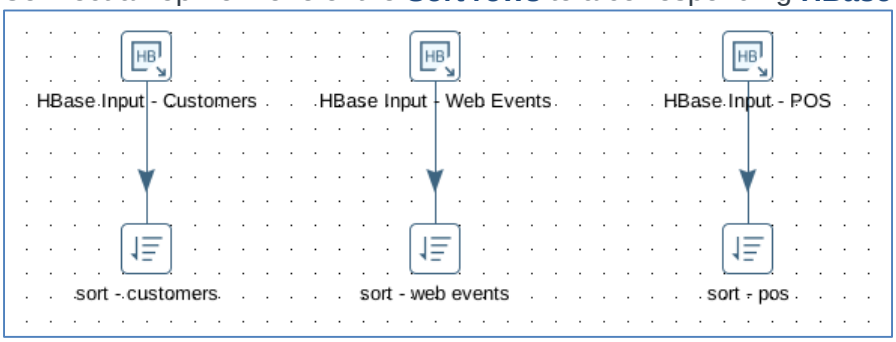
135. Click the **Get Key/Fields info** button. The step should now match the image below:



All three data sources will be merged together on the field customer_id. The "Merge join" requires data to be sorted by the merge key.

136. From the **Design** tab on the left, expand the **Transform** folder and select and drag **Sort rows** onto the canvas.

137. Connect a hop from one of the **Sort rows** to a corresponding **HBase Input**:

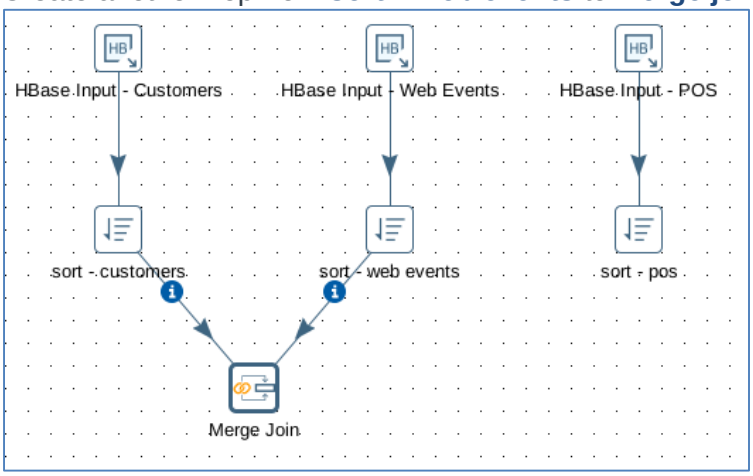


138. Open each **Sort rows**, change the **Step name** per above image and from **Fieldname** select `customer_id`. The data is now sorted and ready to be merged.

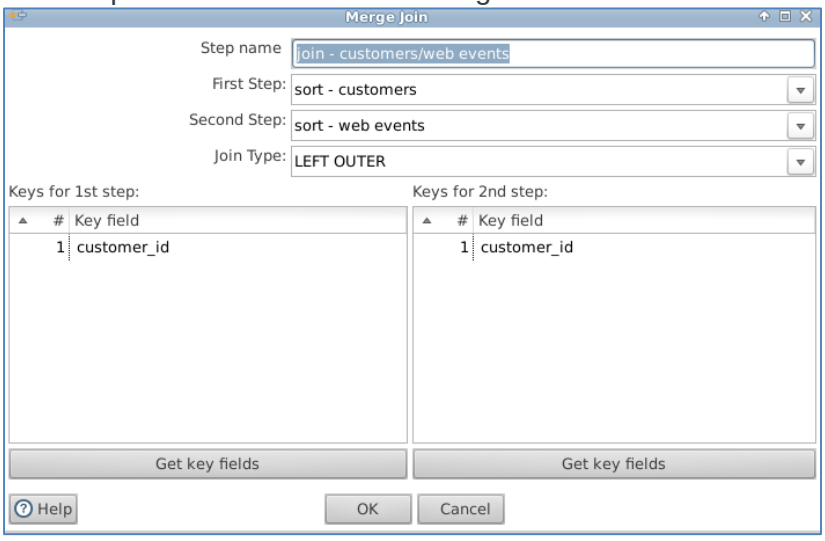
139. From the **Design** tab on the left, expand the **Joins** folder and drag **Merge join** onto the canvas.

140. Create a hop from **sort – customers** to **Merge join**

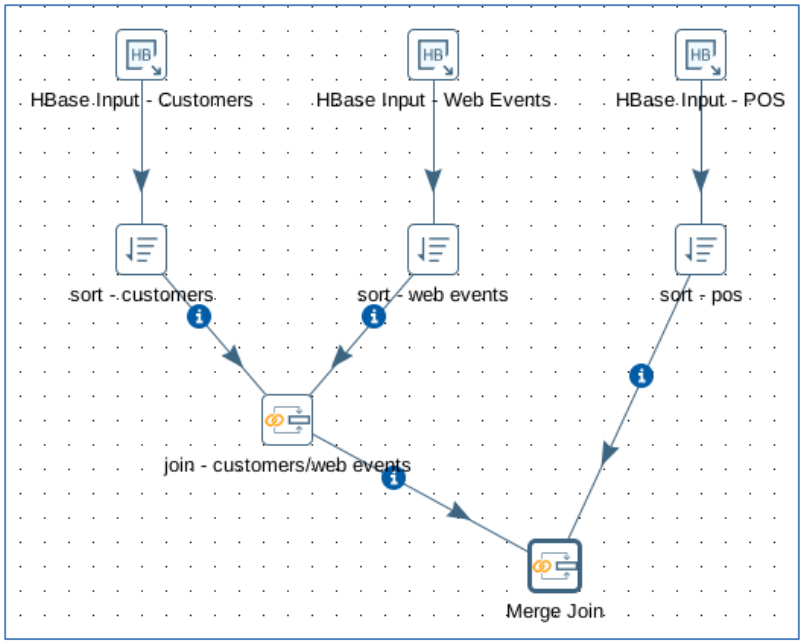
141. Create another hop from **sort – web events** to **Merge join**:



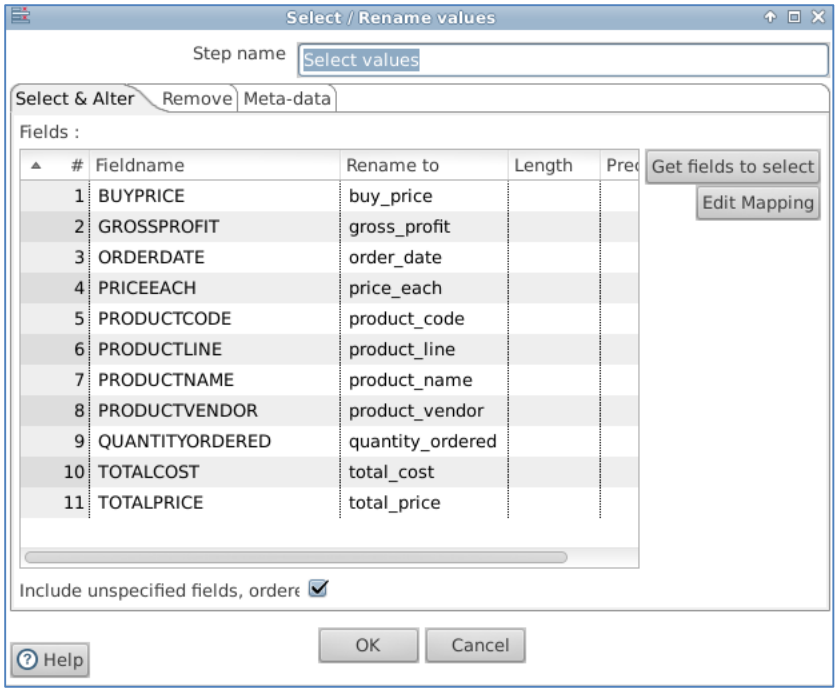
142. Double click **Merge join** to open its properties.
143. Change **Step name** to `join - customers/web events`.
144. From **First step** select `sort - customers`; from **Second step** select `sort - web events`.
145. From **Join Type** select `LEFT OUTER`
146. For **Keys for 1st step** enter `customer_id`; for **Keys for 2nd step** enter `customer_id`. Your step should now match the image below:



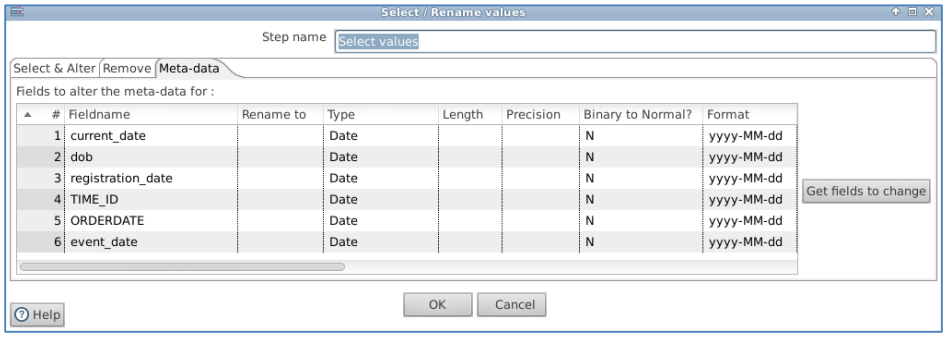
147. Click **OK,** and then click **I understand** in the pop-up warning to exit.
148. From the **Design** tab on the left, expand the **Joins** folder and drag **Merge join** onto the canvas.
149. Create a hop from **join - customers/web events** to **Merge join**.
150. Create another hop from **sort – pos** to step **Merge join**:



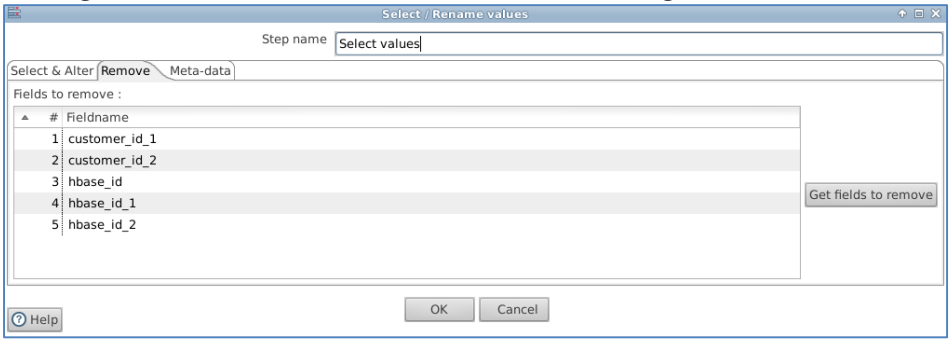
151. Double click **Merge join** to open its properties.
152. Change **Step name** to `join - customer/web/pos`.
153. From **First step** select `join - customers/web events`; from **Second step** select *`sort - pos`*.
154. From **Join Type** select “LEFT OUTER”.
155. For **Keys for 1st step** enter *`customer_id`*; for **Keys for 2nd step** enter *`customer_id`*.
156. Click **OK,** and then click **I understand** in the pop-up warning to exit.
157. From the **Design** tab on the left, expand the **Transform** folder; then, select and drag **Select values** onto the canvas
158. Connect a hop from **join - customer/web/pos** to **Select Values**.
159. Double-click **Select values** to open its properties.
160. In the **Select & Alter** tab we will rename fields so that they are more readable in the reports created in the next section. Ensure that the box is checked next to **Include unspecified fields, ordered by name** and then select the **Fieldname**(s) and make **Rename to** entries per the image below:



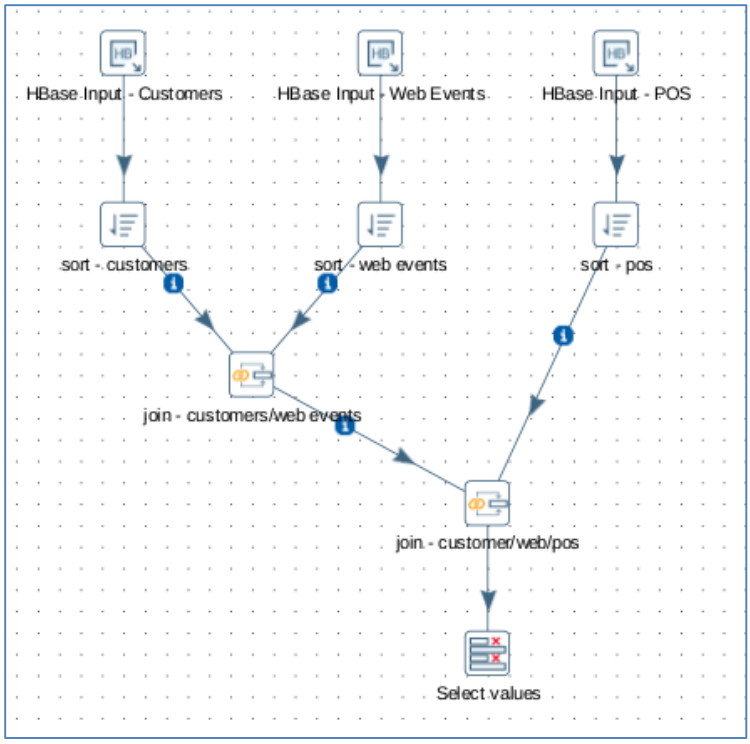
161. In the **Meta-data** tab select the **Fieldname** and enter the **Format** for fields per below image:



162. Next, go to the **Remove** tab and select the following fields in **Fieldname**:



163. Click **OK** to exit. Your transformation should now match the below image:

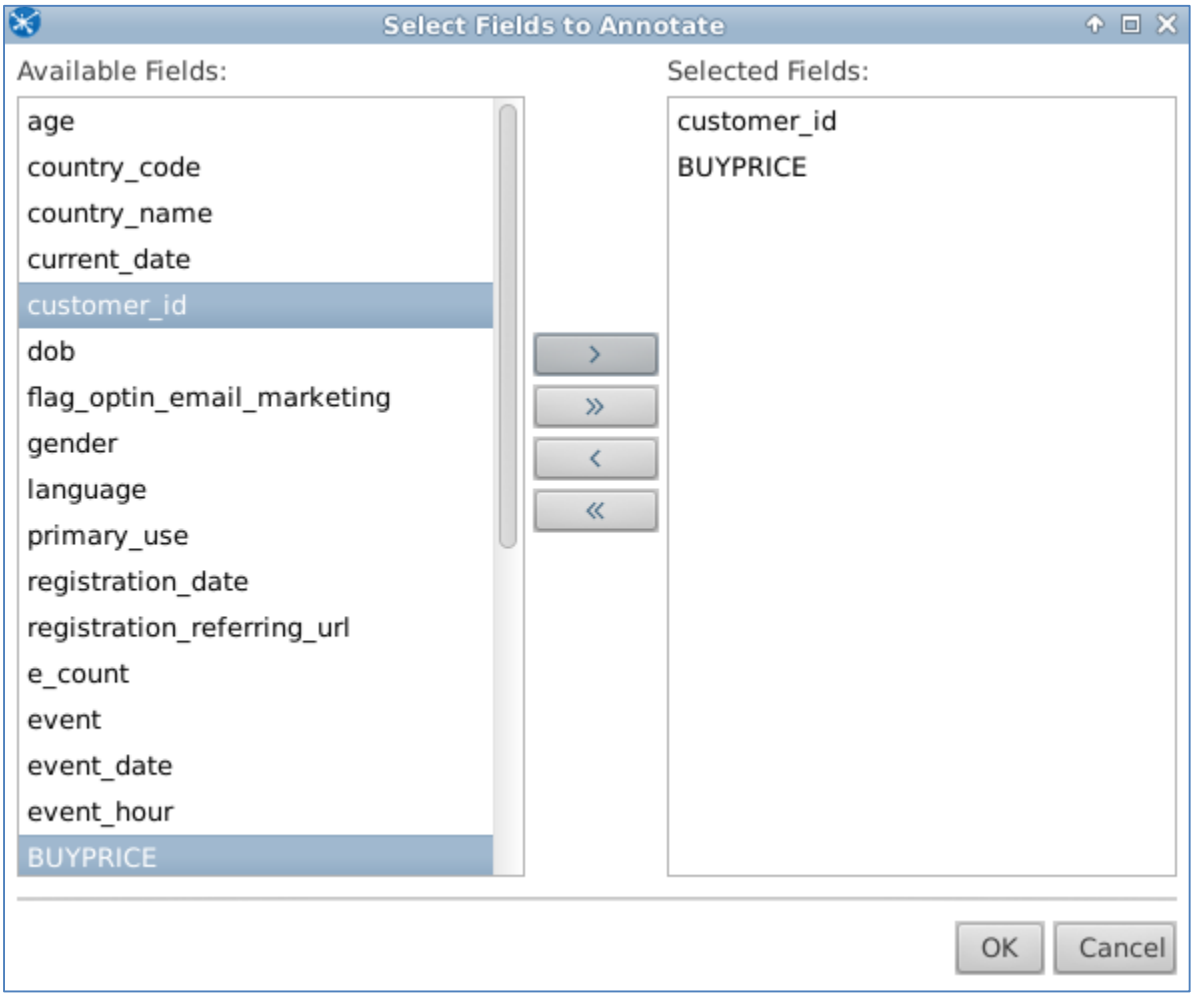


In the next steps, you will partially build the “Annotate Stream” step. This step assists in building the Analyzer reports in the next section by creating aggregations and time measures.

As this is a lengthy and repetitive step to develop, the step source code is provided in a text file. You will be instructed to use the pre-developed step later.

164. From the **Design** tab on the left, expand the **Flow** folder; then select and drag **Annotate Stream** onto the canvas.
165. Connect a hop from **Select values** to **Annotate Stream**.
166. Double-click **Annotate Stream** to open its properties.
167. Click the **Select Fields…** button at the bottom right.
168. In the new window, select `customer_id` from the left pane; hold Ctrl and also select `BUYPRICE`.

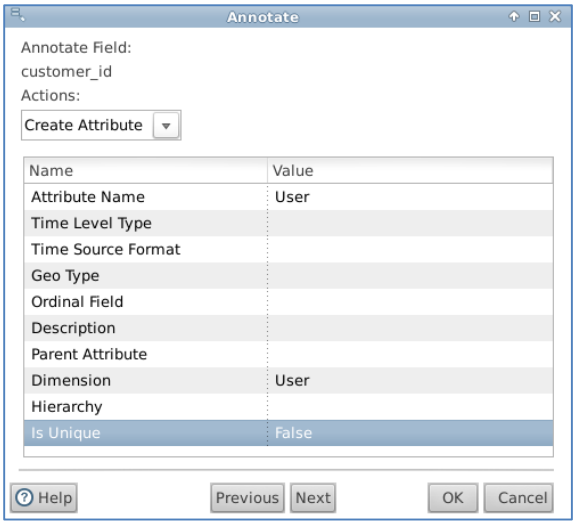
169. Click the single right arrow button [›]. The window should match the image below:



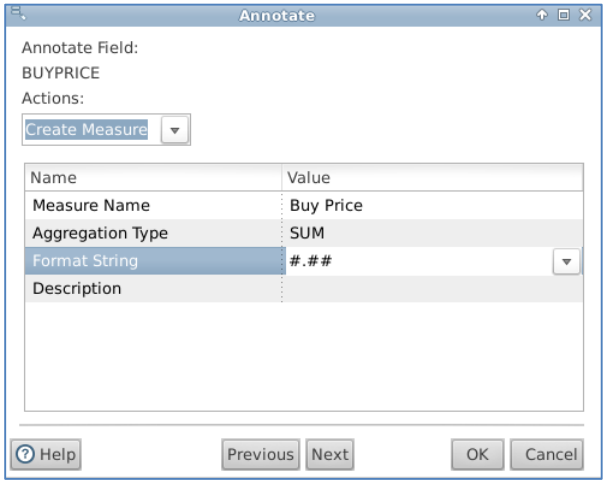
170. Click **OK** to return to the main **Annotate Stream** window.

171. Double-click `customer_id` in the bottom grid.

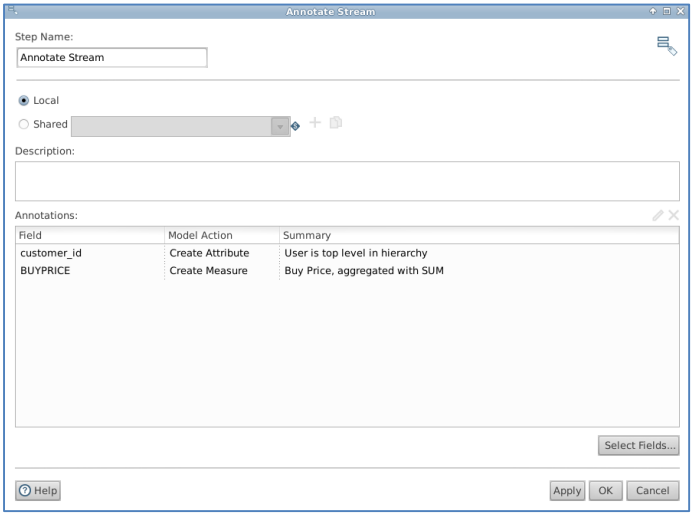
172. From **Actions** select `Create Attribute` and make entries per image below:



173. Click **OK** to exit back to the main **Annotate Stream** window.
174. Next, double click `BUYPRICE` in the grid.
175. From **Actions** select `Create Measure` and make to match the image below:



176. Click **OK** to exit back to the main **Annotate Stream** window

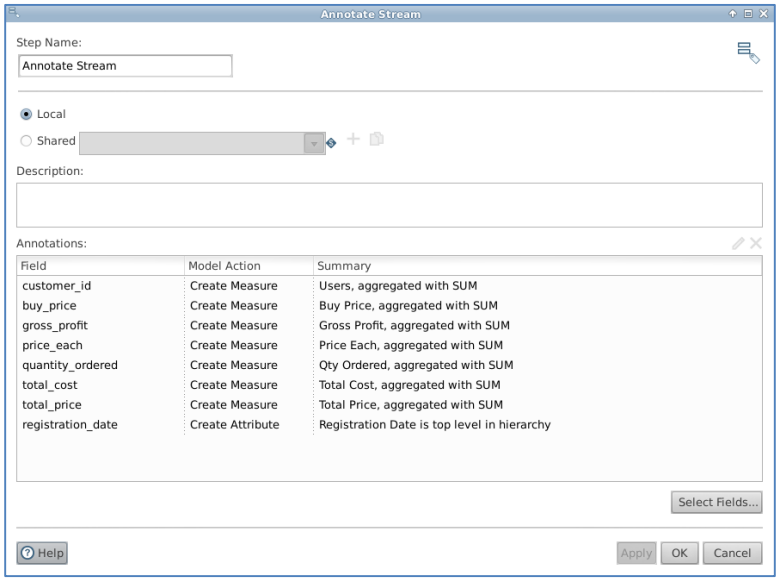


177. Click **Apply** and then click **OK** to exit.

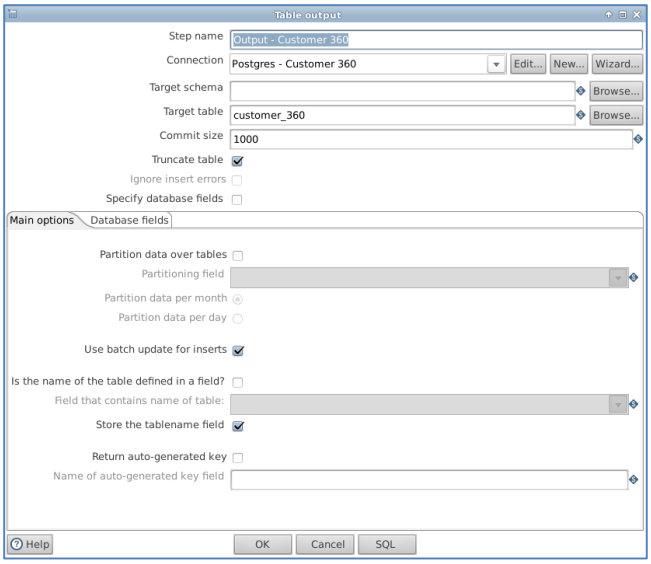
Next, you will paste in the full, pre-developed “Annotate Stream” step. PDI accepts copying/pasting of raw PDI step XML to the canvas to create a step.

178. Delete the **Annotate Stream** step you’ve just created by **right-click | Delete Step**. Select **Yes** in warning prompt.
179. Open the file `/pentaho/shared_content/WorkshopTraining/03_customer_360_hbase/customer_360_annotate_stream_step.txt`
180. Select all of the file contents and copy to clipboard (place cursor in document, Ctrl+a then Ctrl+c)

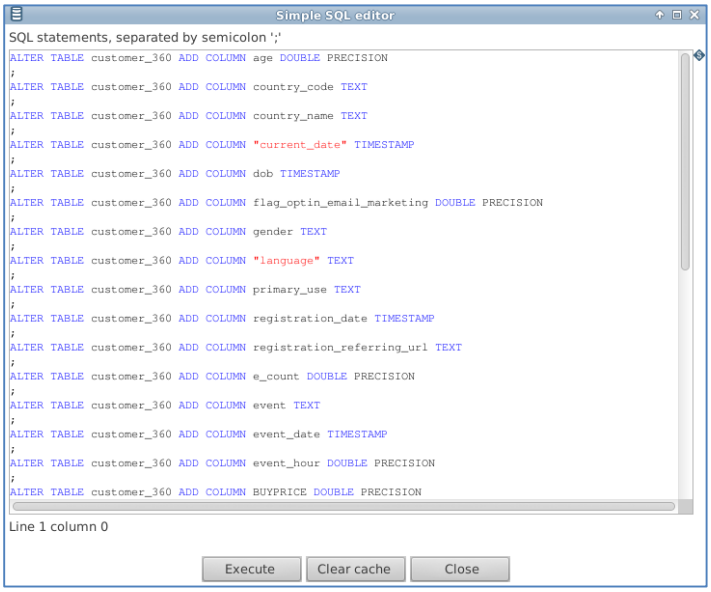
181. Return to Spoon canvas, **right click** | **Paste from clipboard**. You will now see the step **Annotate Stream** on the canvas.
182. Connect a hop from **Select values** to **Annotate Stream**
183. Double click **Annotate Stream** to see the additional development:



184. Click **Cancel** to close.
185. From the **Design** tab on the left, expand the **Output** folder and drag **Table output** onto the canvas.
186. Connect a hop from **Annotate Stream** to **Table Output.**
187. Double-click **Table Output** to open its properties.
188. Change **Step name** to `Output - Customer 360`
189. Under **Connection** select `Postgres - Customer 360`
190. Change **Target table** to `customer_360`
191. Check **Truncate table**
192. Your step should match the image below:



193. While in **Table output**, click the **SQL** button at the bottom right, and a new window will open:

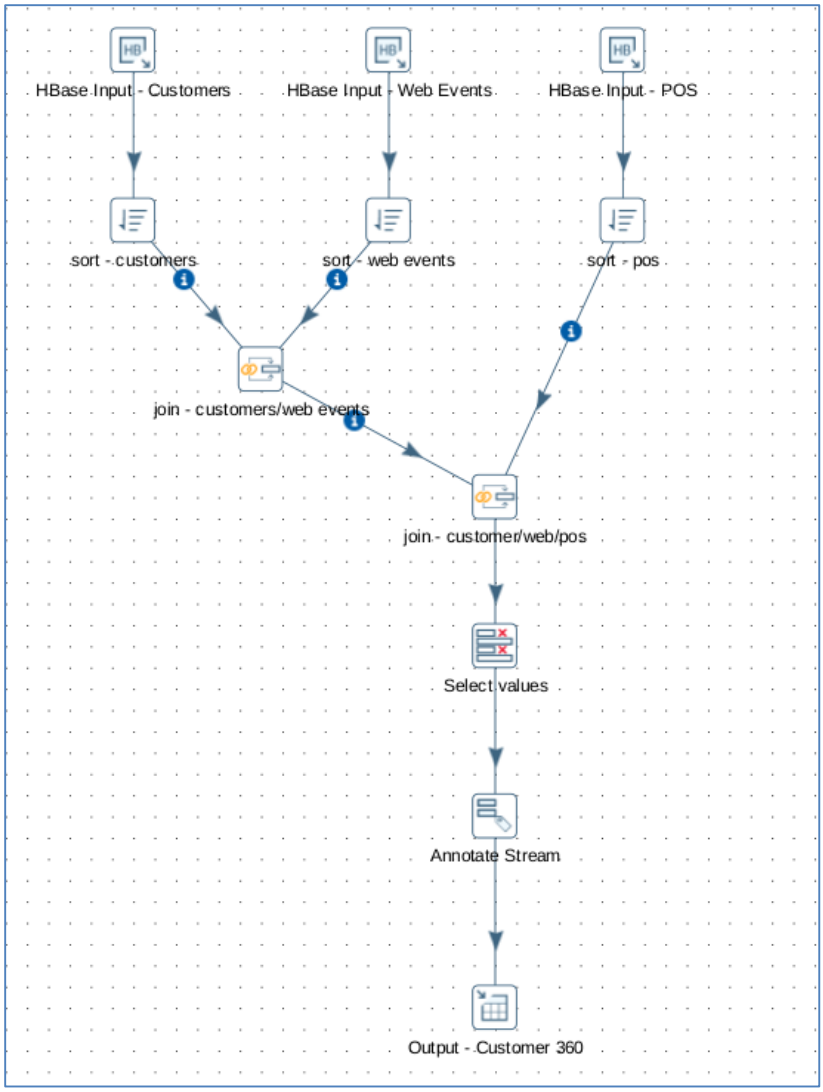


194. Click **Execute.**

195. Click **OK** in new window.

196. Click **Close**.

197. Click **OK** to exit **Table output**. Your final transformation should match the image below:



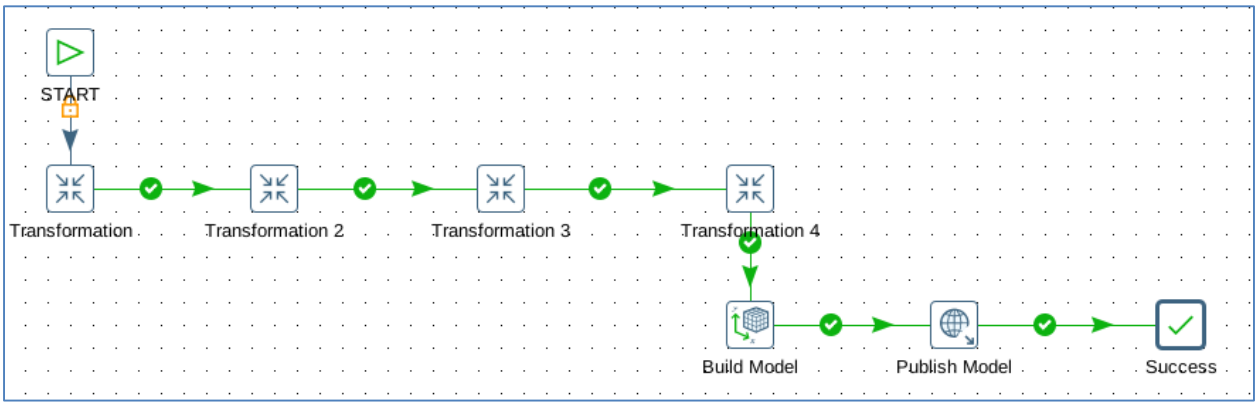
198. Click the **Launch** button at the bottom of the **Execute a transformation** dialog box. 52,772 records should be output.

PDI Exercise 5: Create a job to sequence and automate transformations and publish a model

In this exercise you build a PDI job used to execute all three transformations created in the previous three exercises. PDI jobs can be scheduled for automated processing on a recurring schedule.

199. From the main menu choose **File** | **New** | **Job**.
200. From the **Design** tab on the left, expand the **General** folder; then, select and drag **Start** onto the canvas.
201. While in the General folder, add *three* **Transformation** steps and a **Success** step to the canvas.

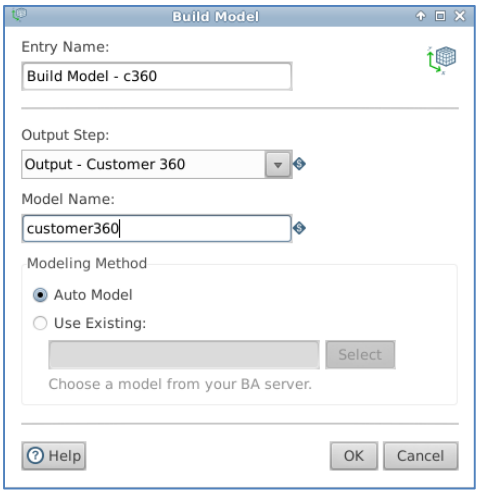
202. Connect each job step with a hop in the following order: **START** | **Transformation** | **Transformation 2** | **Transformation 3 | Transformation 4 | Build Model | Publish Model | Success.**



203. Double-click the first **Transformation** job step to edit the job entry details.
204. Change the **Name of job entry** to `Load HBase - Customers`.
205. For the **Transformation filename** section specify the transformation to run by browsing to `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360_hbase` and choosing `t_load_customers.ktr`.
206. Click **OK** to return to the job canvas.
207. Double-click the **Transformation 2** job step to edit the job entry details.
208. Change the **Name of job entry** to `Load HBase - Web Events`.
209. For the **Transformation filename** section specify the transformation to run by browsing to `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360_hbase` and choosing `t_load_web_events.ktr`.
210. Click **OK** to return to the job canvas.
211. Double-click the **Transformation 3** job step to edit the job entry details.
212. Change the **Name of job entry** to `Load HBase - POS`.
213. For the **Transformation filename** section specify the transformation to run by browsing to `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360_hbase` and choosing `t_load_pos.ktr`.
214. Double-click the **Transformation 4** job step to edit the job entry details
215. Change the **Name of job entry** to `Load Postgres`.
216. For the **Transformation filename** section specify the transformation to run by browsing to `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360_hbase` and choosing `t_load_postgres.ktr`.
217. Double-click the **Build Model** job step to edit the job entry details.

Here, you create the `customer360` model that will be published and used for analysis in Part 2.

218. Change the **Name of job entry** to `Build Model - c360.`
219. From **Output Step** select `Output - Customer 360.`
220. In **Model Name** enter `customer360.` Your step should match the image below:



221. Click **OK** to exit.
222. Double-click the **Publish Model** job step to edit the job entry details.
223. Change the **Entry name** to `Publish - c360`
224. In **URL**, enter `http://localhost:${CURRENT_PENTAHO_BA_PORT}/pentaho/`
225. For **User Name** and **Password**

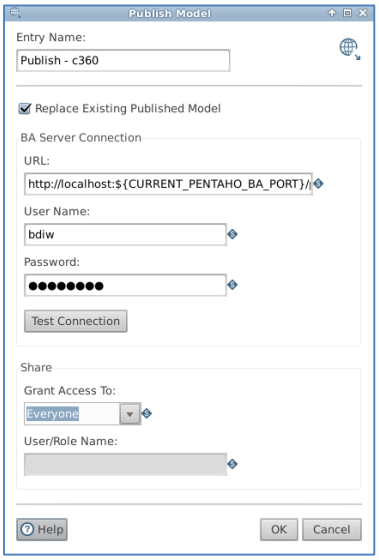
    See Pentaho User Console credentials section in the document on the Desktop in the Docs folder:

    a. hds - BDI Workshop Credentials.pdf

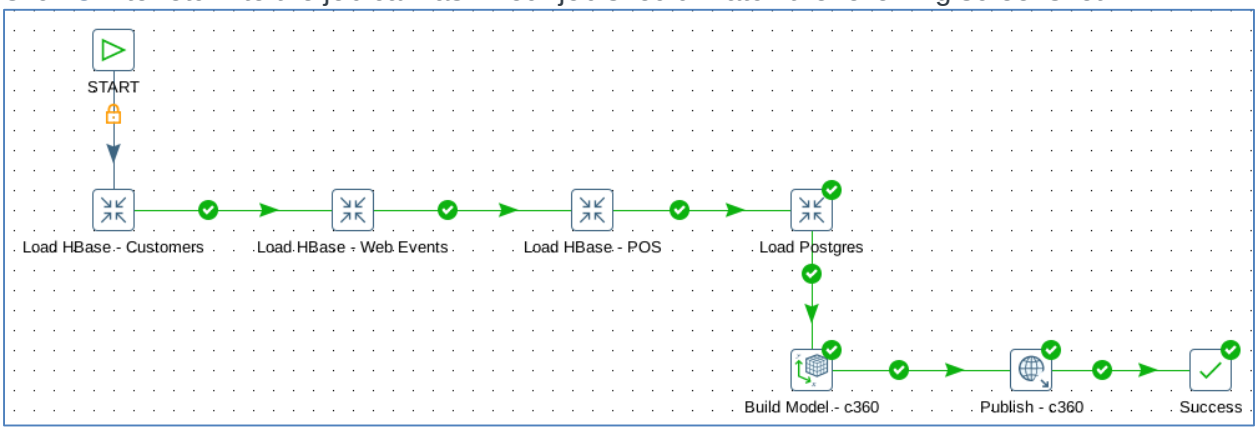
    -or-

    b. bdiw - BDI Workshop Credentials.pdf

226. Your step should match the image below:



227. Click **OK** to return to the job canvas. Your job should match the following screenshot:



228. Use your mouse to lasso-select all five job steps and then right-click any step.
229. From the right-click menu select **Align / Distribute** and then **Snap to grid**.
230. Click the save icon in the toolbar to save your new job as `j_load_customer360.kjb` in the following directory: `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360_hbase`.
231. Run your job by either clicking ▷ or choosing **Action** | **Run** from the main menu. When the job completes without errors, green check marks will appear on each job step.

Part 2: Visualize Data with Analyzer

You now have a `customer360` collection containing customer master data, point-of-sale transactions, and website clickstream event data. Pentaho Analyzer provides analysis using the blended HBase data. The following exercises use Analyzer for HBase to analyze and visualize the data in the `customer360` collection.

Analyzer Exercise 1: Create a scatterplot visualization

This exercise has you create a scatterplot displaying individual products across two measures and then color each point in the scatterplot by the newly calculated field, `Profit_range`.

> Note: The BA server should already be started from the previous exercises, but if you need to start it, execute the following script: `/pentaho/current_version/ctlscript.sh start`.

232. With your Firefox browser navigate to `http://localhost:8081/pentaho/Login` to launch the Pentaho User Console (PUC).
233. Login to PUC

    See Pentaho User Console credentials section in the document on the Desktop in the Docs folder:

a. hds - BDI Workshop Credentials.pdf

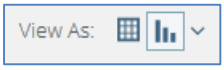
-or-

b. bdiw - BDI Workshop Credentials.pdf

234. Click on the **Create New | Analysis Report** buttons.

235. Select the `customer360: customer360` data source.

236. In the **View As** section in the top right, click the chart icon drop-down and select **Scatter**.



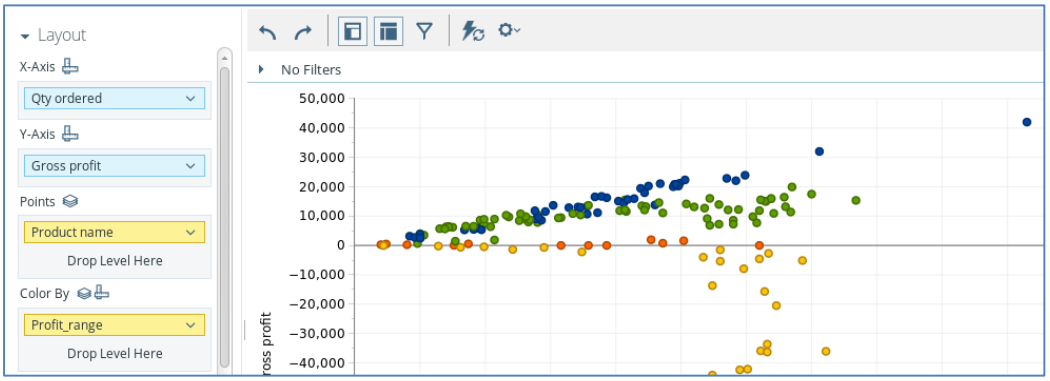
237. Drag `Qty Ordered` to **X-Axis**.

238. Drag `Gross Profit` to **Y-Axis**.

239. Drag `Product name` to **Points**. Analyzer returns a scatterplot view of the products.

240. Drag `Profit range` to **Color By**.

> Profit range was calculated by PDI upon loading the POS data into HBase. This calculation can now be used in visualizations and reports to improve insight.



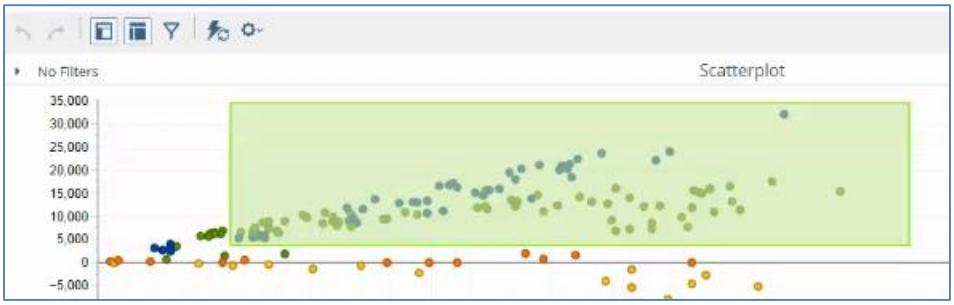
241. Click the Save as button () and save as `Scatterplot` in the following PUC directory: `/home/[PUC Username]`.

Analyzer Exercise 2: Create a high value product list report

This exercise has you lasso-select the high-value products and then create a top 50 product list based on products with the highest gross profit.

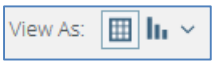
242. Open the **Scatterplot** analysis (if not already open).

243. Lasso-select the points greater than `80 Qty Ordered` and `5000 Gross Profit`.



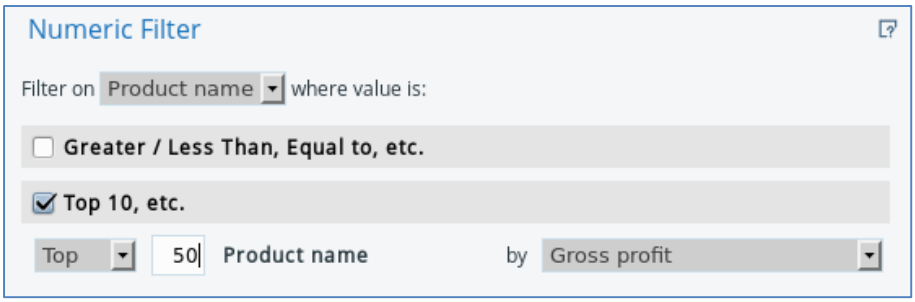
244. Select **Keep Only** from the pop-up menu.

245. From the **View As** menu, click the grid icon to return to a table view



246. Right-click `Product name` and select **Top 10, etc**.

247. In the **Numeric Filter** dialog box, change **Top** 10 to 50 and choose Gross profit in the **by** field.



248. Click **OK**.

249. From the toolbar, click the more actions icon and select **Export → To CSV**.

250. When prompted, accept the defaults and click the **Export** button.

251. Choose **OK** to open the file in Gnumeric, and then save the file to your local disk.

252. Return to PUC and click the **Save as** button in PUC **(** **)** and save as `Top 50 Products` in the following PUC directory: `/home/[PUC Username]`.

Analyzer Exercise 3: Create a linear trend line chart for gross profit

In this exercise you create a linear trend line to trend `Gross profit` across `Order date`. After trending `Gross profit`, you will then create a multi-chart for each `Language`.

253. Click on the **Create New | Analysis Report** buttons.

254. Select the `customer360` data source.

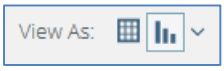
255. Drag `Registration referring url` to **Rows**.

256. Drag `Qty Ordered` to **Measures**.

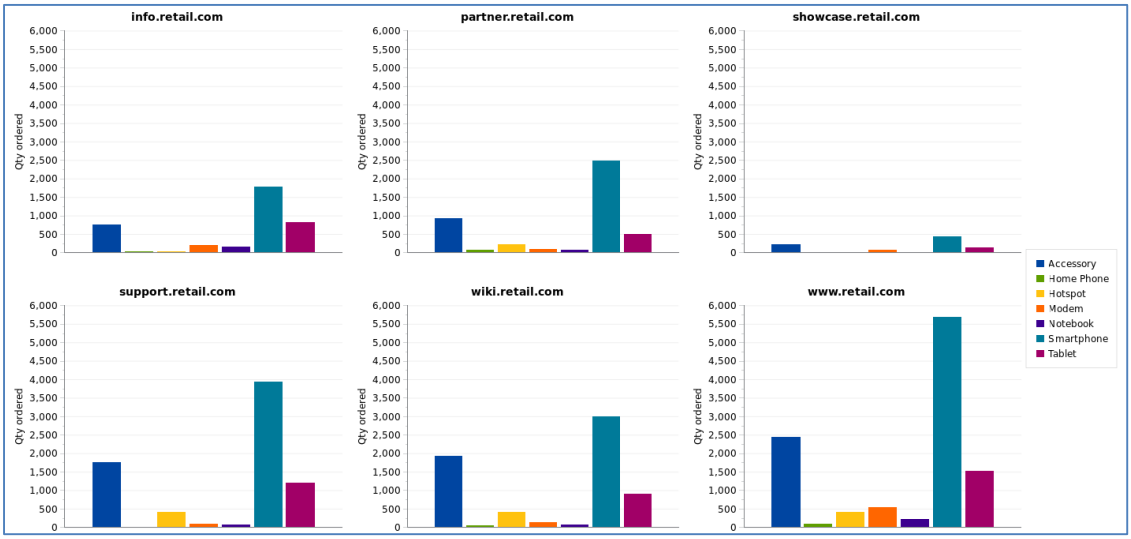
257. Drag `Product line` to **Columns**. Analyzer returns a crosstab view of the `Qty Ordered` as shown in the following image:

	Product line						
	Accessory	Home Phone	Hotspot	Modem	Notebook	Smartphone	Tablet
Registration_referring_url	Qty ordered	Qty ordered	Qty ordered	Qty ordered	Qty ordered	Qty ordered	Qty ordered
info.retail.com	757	41	49	211	168	1,791	840
partner.retail.com	930	85	241	108	82	2,498	511
showcase.retail.com	229	-	-	75	-	452	154
support.retail.com	1,777	-	429	111	72	3,957	1,208
wiki.retail.com	1,950	70	419	151	72	3,016	924
www.retail.com	2,442	104	425	551	227	5,700	1,526

258. In the **View As** section in the top right, click the chart icon drop-down and select **Column**.



259. In the **Layout** section, drag `Registration referring url` from **X-axis** to **Multi-Chart**. The result is a multi-chart by product line for each referring site as shown in the following screenshot.



260. Click the Save as button ( ) and save as `Multi-Chart by Line` in the following directory: `/home/[PUC Username]`.

You have reached the end of the Customer 360 Use Case Workshop and completed the entire Pentaho Big Data Integration Workshop! From all of us at Pentaho, we congratulate you on job well done!

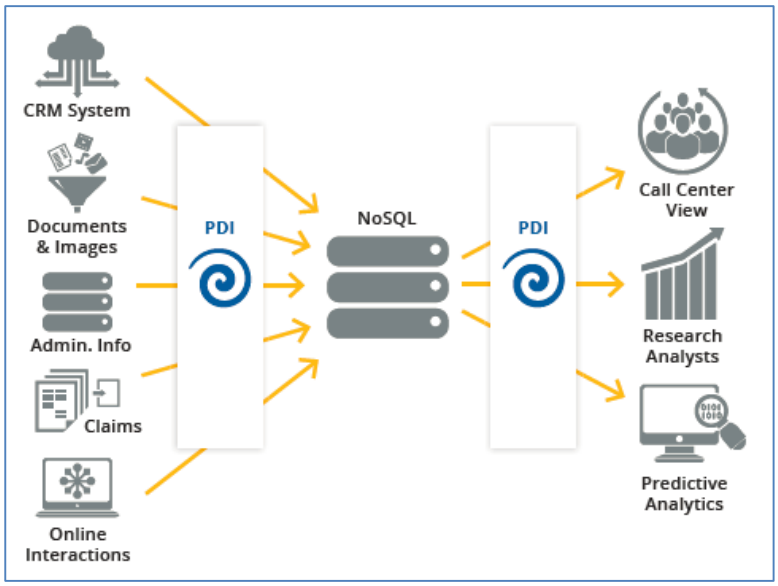
MongoDB Customer 360 Use Case

What is it?

- Blending a variety of operational & transactional data sources to create an on-demand analytical view across customer touch points and surface the customer experience.
- Providing customer-facing employees and partners with information made available inside everyday line-of-business applications
- Leverages NoSQL database technologies like MongoDB that provide flexible storage options for single views of data.

Why do it?

- Provide single customer view to sales & services teams, empowering them to grow upsell/cross-sell revenue
- Understand customer perception of your products & services, while decreasing customer churn
- Avoid point-to-point integrations with single repository, and fast queries to improve access to metrics



Value of Pentaho

- Staff savings & productivity: Rapid time to value through drag/drop visual development for big data integration – make big data accessible to all data developers
- Operational Intelligence: Ability to embed analytics into actionable line-of-business applications for each relevant customer-facing role

- Broad & robust data connectivity: Ability to blend traditional sources & big data
- Comprehensive analytics: Visualizations, reports, dashboards, ad hoc analysis provide for all roles

Pentaho components used in this workshop

PENTAHO DATA INTEGRATION (PDI)

ANALYZER FOR MONGODB

PENTAHO REPORT DESIGNER (PRD)

What you will accomplish in this workshop

- Part 1 – Use PDI to create a single view in MongoDB (4 exercises)
- Part 2 – Visualize data with Analyzer for MongoDB (3 exercises)
- Part 3 – Pentaho Report Designer (4 exercises)

Part 1: Use PDI to create a single view in MongoDB

PDI gives users a graphical user interface to build transformations and jobs that solve complex data integration challenges with MongoDB. PDI provides a scalable solution for migrating to MongoDB with connectors for all types of disparate data for building out a single customer view. PDI leverages the power of the MongoDB aggregation framework query language for aggregating and processing MongoDB data.

Use PDI to create a MongoDB customer data store

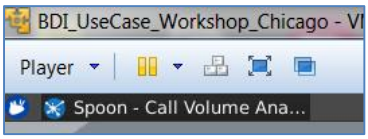
A new MongoDB collection, `customer360`, will store customer master data, point-of-sale (POS) transactions, and website clickstream event data. The customer master records are stored as documents, while the POS transactions and event transactions are stored as data arrays within each customer document.

PDI Exercise 1: Create a transformation to load customer records to MongoDB

This first exercise steps you through the process of creating a transformation to load customer master records from a CSV file, calculate the customer age, and then load the results into a MongoDB collection.

1. If PDI is not already open, launch it from the launch menu icon at the bottom of your screen. Click this icon just *once* to launch Spoon, the client-based authoring GUI for PDI.

2. You will see the PDI splash screen appear while PDI loads. All open applications appear in the top left section of your screen. You can see Spoon is open in the following screenshot.

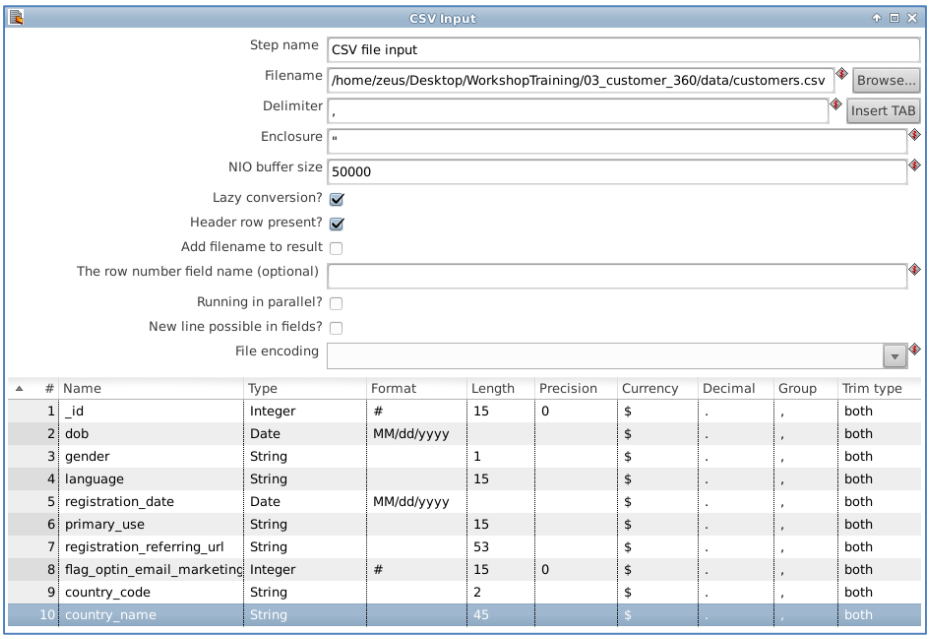


3. From the main menu choose **File | New | Transformation**

We need to access Customer data from within a flat file. The file format is a csv file, so you will access the data by configuring a CSV file input step.

4. From the **Design** tab on the left, expand the **Input** folder and drag **CSV file input** onto the canvas
5. Double-click on **CSV file input** to open its properties
6. Browse to the directory `/pentaho/shared_content/WorkshopTraining/03_customer_360_mongodb/data/` and select `customers.csv`.
7. Click the **Get Fields** button at the bottom and enter `50,000` for the **Sample Size**.
8. Once the scan is finished, click **Close** to close the scan results.

9. Trim any possible source data character spaces by setting **Trim Type** to `both` for every field.
10. Click the **Preview** button to preview the data and then click **Close**. Your CSV Input dialog box should match the following image:



#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	_id	Integer	#	15	0	$	.	,	both
2	dob	Date	MM/dd/yyyy			$	.	,	both
3	gender	String		1		$	.	,	both
4	language	String		15		$	.	,	both
5	registration_date	Date	MM/dd/yyyy			$	.	,	both
6	primary_use	String		15		$	.	,	both
7	registration_referring_url	String		53		$	.	,	both
8	flag_optin_email_marketing	Integer	#	15	0	$	.	,	both
9	country_code	String		2		$	.	,	both
10	country_name	String		45		$	.	,	both

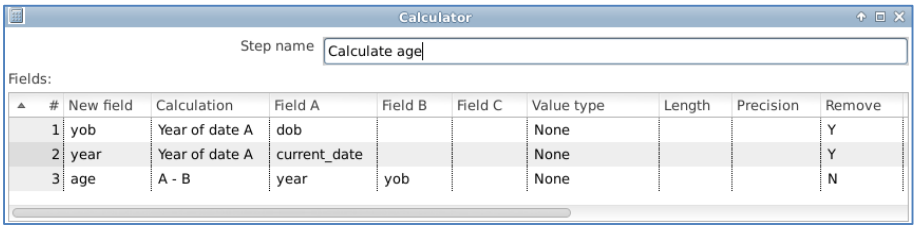
11. Click **OK** to return to the canvas.

To calculate the age of the customer, we need to know the current year and the year in which the customer was born. But to calculate the current year, we need to know the current date. We will use the **Get System Info** step to learn the current date.

12. From the **Design** tab on the left, expand the **Input** folder and drag **Get System Info** onto the canvas.
13. Double-click on **Get System Info** step to open its properties.
14. In the **Name** field type `current_date`
15. In the **Type** field, select from the drop-down menu, `system date (variable)`
16. Click **OK** to return to the canvas.
17. To draw a hop between two steps, shift-click the **CSV file input** step and while holding down your mouse key, drag a **hop** over to the **Get System Info** step. When prompted, select **Main output of step**.

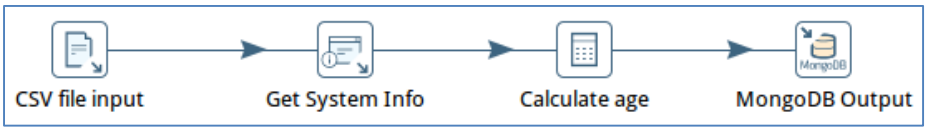
Now that we have the current date, we need to derive the current year from the date. And we need to know the year in which the customer was born, which is based on the customer date of birth. With both years in hand, we can calculate the age of the customer. All of these calculations can be performed with the **Calculator** step.

18. From the **Design** tab on the left, expand the **Transform** folder and drag the **Calculator** step onto the canvas.
19. Draw a hop between the **Get System Info** and **Calculator** steps.
20. Double-click on **Calculator** step to open its properties.
21. In the **Step name** box type `Calculate age`
22. Follow these steps to create the first year of birth calculation: In the **New Field** column type `yob`, in the **Calculation** column select `Year of date A`, in the **Field A** column select the `dob` field, and finally in the **Remove** column select `Y`.
23. Follow these steps to create the second year calculation: In the **New Field** column type `year`, in the **Calculation** column select `Year of date A`, in the **Field A** column select the `current_date` field, and finally in the **Remove** column select `Y`.
24. Follow these steps to create the third customer age calculation: In the **New Field** column type `age`, in the **Calculation** column select `A - B`, in the **Field A** column select the `year` field, in the **Field B** column select the `yob` field, and finally in the **Remove** column, make sure it is set to `N`.
25. Once your **Calculate age** step matches the following image, click **OK** to return to the canvas.



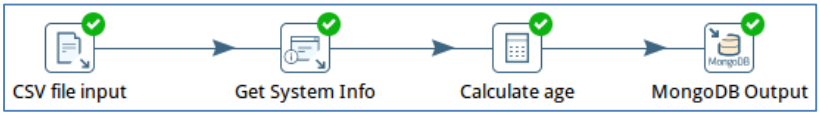
Customer data has been enriched with age information, and we are ready to write the data to a MongoDB collection.

26. From the **Design** tab on the left, expand the **Big Data** folder and drag **MongoDB Output** onto the canvas.
27. Draw a hop between the **Calculate age** and **MongoDB Output** steps. The resulting transformation should match the following image:



28. Double-click on **MongoDB Output** step to open its properties.
29. On the **Output Options** tab, type `customer360` in the **Database** and **Collection** entry boxes.
30. Check **Truncate collection**.

31. On the **Mongo document fields** tab click the **Get fields** button at the bottom to populate the field names.
32. Click **OK** to return to the main canvas.
33. Click the save icon in the toolbar to save your new transformation as `t_load_customers` in the following directory: `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360`
34. Click **OK** to return to the main canvas.
35. Run your transformation by either clicking ▶ or choosing **Action**| **Run** from the main menu.
36. Click the **Launch** button at the bottom of the **Execute a transformation** dialog box. Your transformation has run successfully when you see the green checkmark boxes on each step as shown in the following image. This transformation will load 19,673 customers into the newly created MongoDB collection.



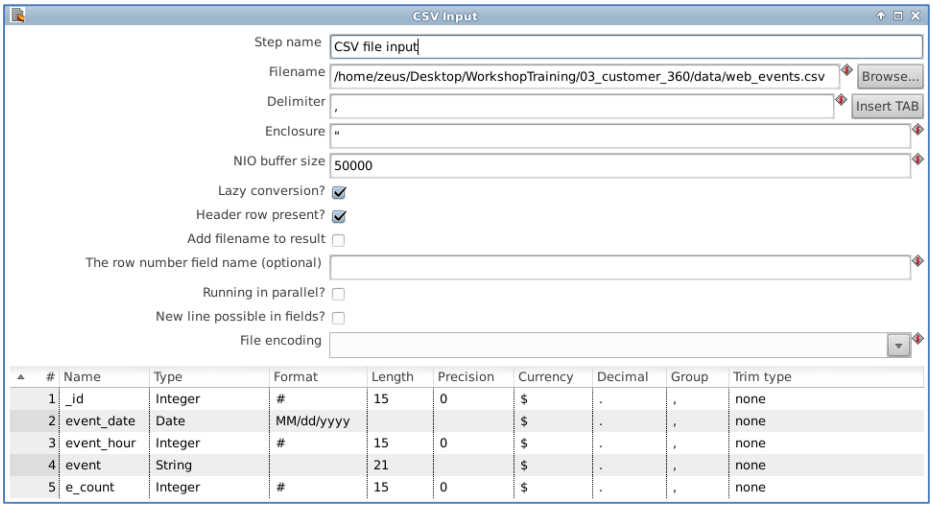
PDI Exercise 2: Create a transformation to load web events to MongoDB

This second exercise steps you through the process of creating a transformation to load the second source of data, customer web clickstream (web event) records, from a CSV file into a MongoDB collection. In this MongoDB output step, you declare the customer "_id" as the lookup key and load the clickstream records to data arrays for each customer.

1. From the main menu choose **File** | **New** | **Transformation**

To blend customer web events data from a CSV file into the MongoDB data collection, first we need to configure a CSV file input step to access the web events.

2. From the **Design** tab on the left, expand the Input folder and drag **CSV file input** onto the canvas**.**
3. Double-click on **CSV file input** to open its properties
4. Browse to the directory `/pentaho/shared_content/WorkshopTraining/03_customer_360_mongodb/data/` and select `web_events.csv`.
5. Click the **Get Fields** button at the bottom and enter `50,000` for the **Sample Size**.
6. Click **Close** to close the scan results.
7. Click the **Preview** button to preview the data and then click **Close**. Your **CSV Input** dialog box match the following image:



8. Click **OK** to return to the canvas.

The customer web events records should be appended to the appropriate customer records already loaded the MongoDB collection. We will use customer "_id" as the lookup key to join the web events records to data arrays we will name "event_data []" for each customer.

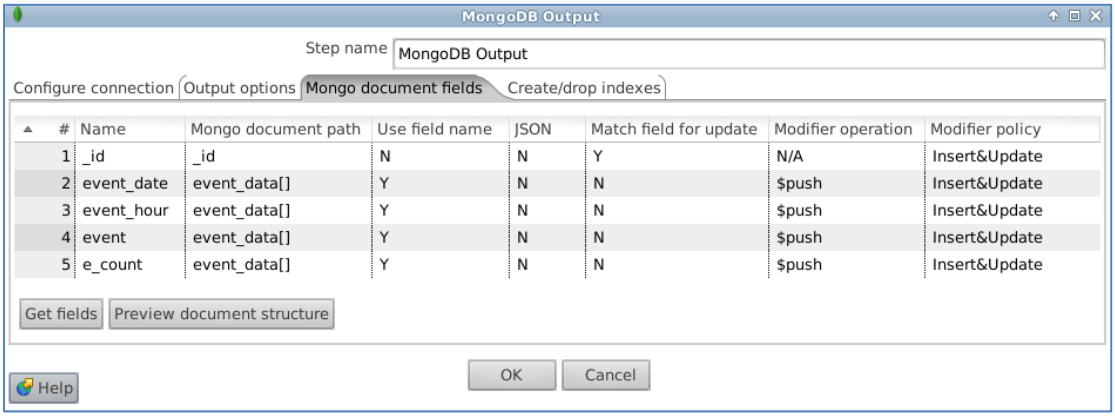
9. From the **Design** tab on the left, collapse the **Input** folder and expand the **Big Data** folder; then, select and drag **MongoDB Output** onto the canvas.
10. Shift-click the **CSV file input** step and while holding down your mouse key, drag a **hop** over to the **MongoDB Output** step. When prompted, select **Main output of step**.

11. Double-click on **MongoDB Output** step to open its properties.
12. On the **Output Options** tab, click the **Get DBs** button and select `customer360` from the list. Click the **Get collections** button and select `customer360` from the list.
13. IMPORTANT - Check **Update, Upsert** and **Modifier Update**.
14. On the Mongo document fields tab click the Get fields button at the bottom to populate the field names and then enter the following values in the Mongo document path column.

```
Row #1 - _id
Row #2 - event_data[]
Row #3 - event_data[]
Row #4 - event_data[]
Row #5 - event_data[]
```

15. On the first row change **Use field name** to `N` and **Match field for update** to `Y`.
16. On rows 2 thru 5, change **Modifier operation** to `$push`.
17. For every row change **Modifier policy** to `Insert&Update`.

18. Your Mongo document fields should match the following screenshot.



19. Click **Preview document structure** to validate the settings.
20. Click **OK** to return to the main canvas.
21. Click the save icon in the toolbar to save your new transformation as `t_load_web_events` in the following directory: `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360`
22. Run your transformation by either clicking ▷ or choosing **Action** | **Run** from the main menu. In the Step Metrics tab you should see the execution results from loading 51,419 clickstream records into MongoDB.

PDI Exercise 3: Create a transformation to load POS transactions to MongoDB

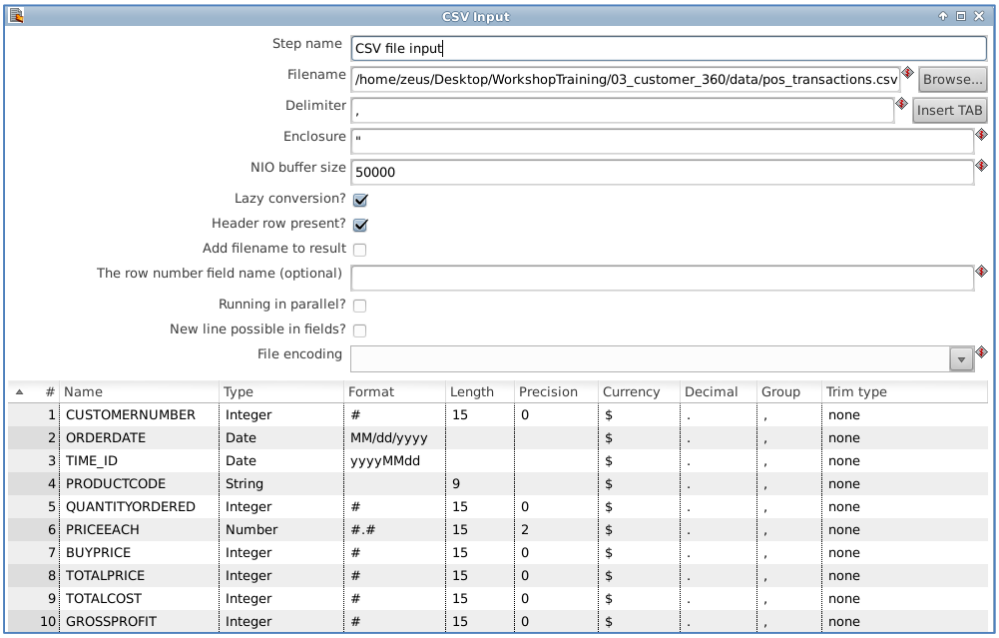
This third exercise steps you through the process of creating a transformation to load the third and final data source, customer point-of-sale (POS) sale transaction records, from a CSV file into a MongoDB collection. You use the number range step to bin sales orders into buckets based on the size of the order. In this MongoDB output step, you declare the customer "_id" as the lookup key and load the POS records to POS data arrays for each customer.

1. From the main menu choose **File** | **New** | **Transformation**

To blend point-of-sale transactions from a CSV file into the MongoDB data collection, we need to configure a CSV file input step to access the transactions.

2. From the **Design** tab on the left, expand the **Input** folder and drag **CSV file input** onto the canvas.
3. Double-click on **CSV file input** to open its properties
4. Browse to the directory `/pentaho/shared_content/WorkshopTraining/03_customer_360_mongodb/data/` and select `pos_transactions.csv`.
5. Click the **Get Fields** button at the bottom and enter `5,000` for the **Sample Size**.

6. Click the **Preview** button to preview the data and then click **Close**. Your **CSV Input** dialog box should look like the following image:

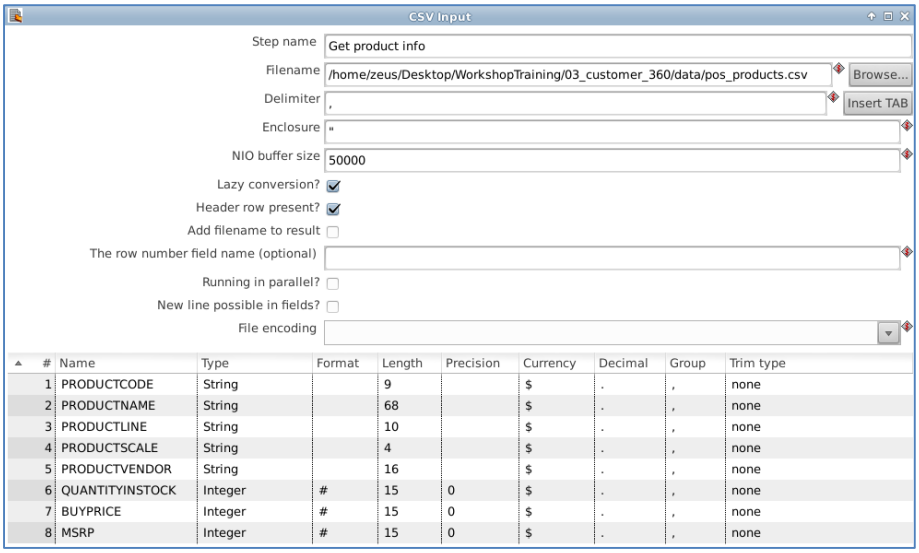


#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	CUSTOMERNUMBER	Integer	#	15	0	$	.	,	none
2	ORDERDATE	Date	MM/dd/yyyy			$	.	,	none
3	TIME_ID	Date	yyyyMMdd			$	.	,	none
4	PRODUCTCODE	String		9		$	.	,	none
5	QUANTITYORDERED	Integer	#	15	0	$	.	,	none
6	PRICEEACH	Number	#.#	15	2	$	.	,	none
7	BUYPRICE	Integer	#	15	0	$	.	,	none
8	TOTALPRICE	Integer	#	15	0	$	.	,	none
9	TOTALCOST	Integer	#	15	0	$	.	,	none
10	GROSSPROFIT	Integer	#	15	0	$	.	,	none

7. Click **OK** to return to the canvas.

The POS file only has product code on the transaction. We need to enhance the data with Product Name, Product Line, and Product Vendor. To do so, we will look up the product information in a flat file to match the transaction product code with its relevant name, line, and vendor.

8. From the **Design** tab on the left, expand the **Lookup** folder and drag **Stream lookup** onto the canvas.
9. Create a hop between the **CSV file input** and **Stream lookup** steps.
10. From the **Design** tab on the left, expand the **Input** folder and drag a second **CSV file input** onto the canvas and place it above the Stream lookup step.
11. Double-click on **CSV file input** to open its properties
12. Change the **Step name** to `Get product info`
13. Browse to the directory `/pentaho/shared_content/WorkshopTraining/03_customer_360_mongodb/data/` and select `pos_products.csv`.
14. Click the **Get Fields** button at the bottom and enter `5,000` for the **Sample Size**.
15. Click the **Preview** button to preview the data and then click **Close**. Your **CSV Input** dialog box should look like the following image:

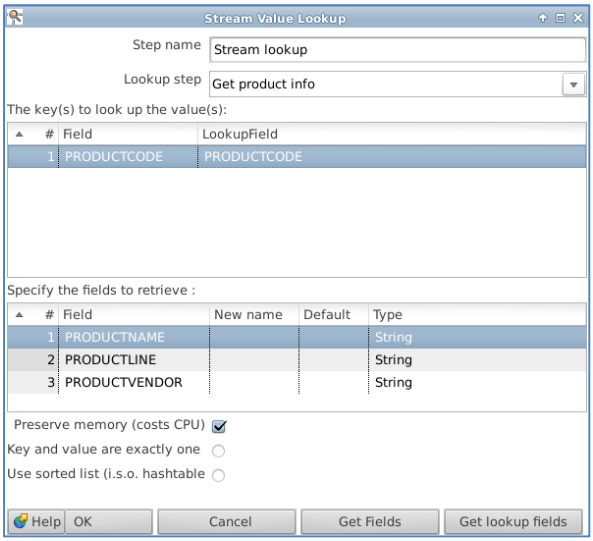


#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	PRODUCTCODE	String		9		$	.	,	none
2	PRODUCTNAME	String		68		$	.	,	none
3	PRODUCTLINE	String		10		$	.	,	none
4	PRODUCTSCALE	String		4		$	.	,	none
5	PRODUCTVENDOR	String		16		$	.	,	none
6	QUANTITYINSTOCK	Integer	#	15	0	$	.	,	none
7	BUYPRICE	Integer	#	15	0	$	.	,	none
8	MSRP	Integer	#	15	0	$	.	,	none

16. Click **OK** to return to the canvas
17. Create a hop from the second **Get product info** step to the **Stream lookup** step.

We need to configure the Stream Lookup to retrieve Product Name, Product Line, and Product Vendor from the lookup file for every product code we find in the POS records.

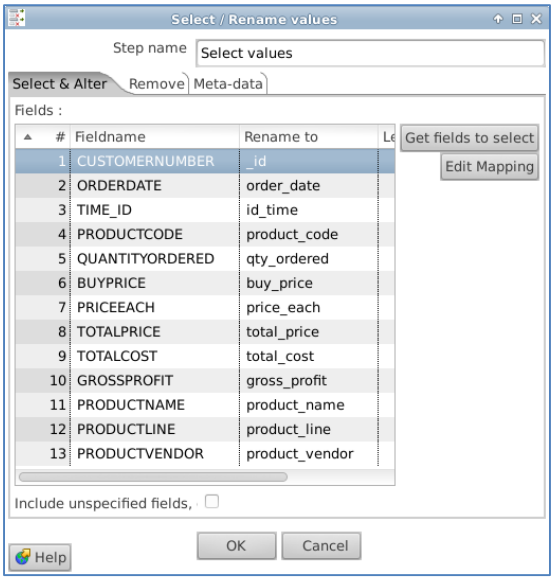
18. Double-click on **Stream lookup** step to open its properties
19. In the **Lookup step** select `Get product info`.
20. In the **Key(s) to Lookup Value(s)** section select `PRODUCTCODE` in the **Field** column and select `PRODUCTCODE` as the **LookupField** column.
21. Click the **Get Lookup Fields** button to populate the fields to retrieve section at the bottom.
22. Highlight and delete the following fields: `PRODUCTCODE, PRODUCTSCALE, QUANTITYINSTOCK, BUYPRICE, MSRP` from the fields to retrieve section. Your **Stream Lookup** dialog box should now match the following image:



23. Click **OK** to return to the canvas.

CUSTOMERNUMBER is the same as customer "_id" within the MongoDB collection. We prefer to rename this field and all the other fields to match and standardize on a lowercase naming convention used in the MongoDB collection target.

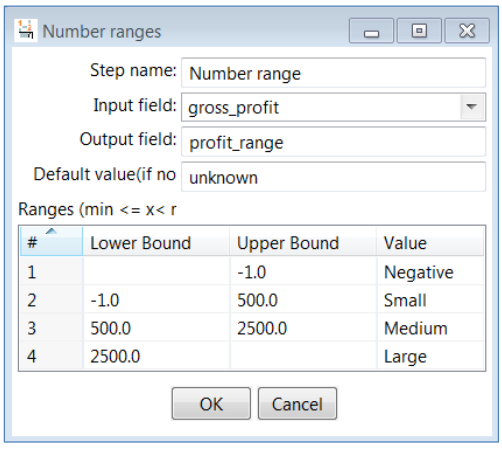
24. From the **Design** tab on the left, expand the **Transform** folder and drag **Select Values** onto the canvas.
25. Create a hop between the **Stream lookup** and **Select values** steps.
26. Double-click the **Select Values** step. On the **Select & Alter** tab, click **Get fields to select** to add all the available stream fields.
27. To rename `CUSTOMERNUMBER`, click inside the **Rename to** column and enter `_id`.
28. Rename all the remaining fields to standardize on a lowercase naming convention and match the other data sets. The result should match the following image:



29. Click **OK** to return to the main canvas.

It may be useful to categorize the sale orders into buckets for further analysis, based on order size. We can create a categorization of small, medium, and large.

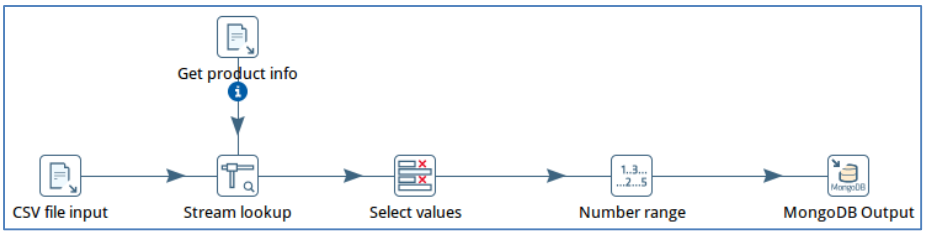
30. From within the **Transform** folder, select and drag **Number Range** onto the canvas.
31. Create a hop between the **Select values** and **Number Range** steps.
32. Double-click on **Number Range** step to open its properties.
33. For **Input field** select `gross_profit` from the list.
34. For **Output field** type `profit_range`.
35. Enter the following values for **Lower Bound**, **Upper Bound** and **Value** columns.



36. Click **OK** to return to the canvas.

The point-of-sale records should be appended to the appropriate customer records already loaded to the MongoDB collection. We will use customer "_id" as the lookup key to join the POS records to data arrays we will name "pos_data []" for each customer.

37. From the **Design** tab on the left, expand the **Big Data** folder and drag **MongoDB Output** onto the canvas.
38. Create a hop between the **Number Range** and **MongoDB Output** step. The resulting transformation should match the following image:



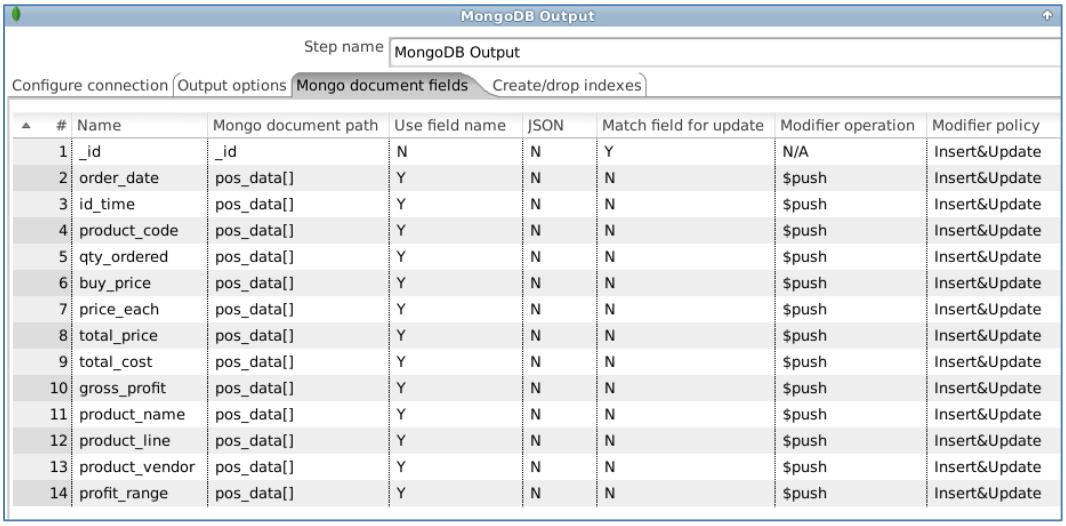
39. Double-click on **MongoDB Output step** to open its properties.
40. On the **Output Options** tab, click the **Get DBs** button and select `customer360` from the list. Click the **Get collections** button and select `customer360` from the list.
41. IMPORTANT - Check **Update, Upsert** and **Modifier Update**.
42. On the **Mongo document fields** tab click the **Get fields** button at the bottom to populate the field names.
43. Click **OK** to return to the main canvas.

    By clicking **OK** the dialog will register your changes and set default field values to be used in subsequent steps.

44. Double-click the **MongoDB Output** step again, and on the **Mongo document fields** tab, enter the following values in the **Mongo document path column**.

```
Row #1 - _id
Row #2-#14 - pos_data[]
```

45. On the first row change **Use field name** to `N` and **Match field for update** to `Y`.
46. On rows 2 thru 14, change **Modifier operation** to `$push`. You can use the right-click option on the mouse button to copy/paste values.
47. For every row change **Modifier policy** to `Insert&Update`.
48. Your Mongo document fields should match the following screenshot.



#	Name	Mongo document path	Use field name	JSON	Match field for update	Modifier operation	Modifier policy
1	_id	_id	N	N	Y	N/A	Insert&Update
2	order_date	pos_data[]	Y	N	N	$push	Insert&Update
3	id_time	pos_data[]	Y	N	N	$push	Insert&Update
4	product_code	pos_data[]	Y	N	N	$push	Insert&Update
5	qty_ordered	pos_data[]	Y	N	N	$push	Insert&Update
6	buy_price	pos_data[]	Y	N	N	$push	Insert&Update
7	price_each	pos_data[]	Y	N	N	$push	Insert&Update
8	total_price	pos_data[]	Y	N	N	$push	Insert&Update
9	total_cost	pos_data[]	Y	N	N	$push	Insert&Update
10	gross_profit	pos_data[]	Y	N	N	$push	Insert&Update
11	product_name	pos_data[]	Y	N	N	$push	Insert&Update
12	product_line	pos_data[]	Y	N	N	$push	Insert&Update
13	product_vendor	pos_data[]	Y	N	N	$push	Insert&Update
14	profit_range	pos_data[]	Y	N	N	$push	Insert&Update

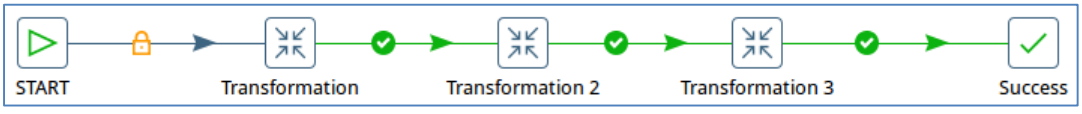
49. Click **Preview document structure** to validate the settings.
50. Click **OK** to return to the main canvas.
51. Click the save icon in the toolbar to save your new transformation as `t_load_pos` in the following directory:

    `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360`
52. Run your transformation by either clicking ▶ or choosing **Action** | **Run** from the main menu. In the Step Metrics tab you should see the execution results from loading 991 POS records into MongoDB.

PDI Exercise 4: Create a job to sequence and automate 3 transformations

In this exercise you build a PDI job used to execute all three transformations created in the previous three exercises. PDI jobs can be scheduled for automated processing on a recurring schedule.

1. From the main menu choose **File** | **New** | **Job**.
2. From the **Design** tab on the left, expand the **General** folder; then, select and drag **Start** onto the canvas.
3. While in the General folder, add *three* **Transformation** steps and a **Success** step to the canvas.
4. Connect each job step with a hop in the following order: **START** | **Transformation** | **Transformation 2** | **Transformation 3** | **Success**.



5. Double-click the first **Transformation** job step to edit the job entry details.
6. Change the **Name of job entry** to `Load Customers`.

7. For the **Transformation filename** section specify the transformation to run by browsing to `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360` and choosing `t_load_customers.ktr`.
8. Click **OK** to return to the job canvas.
9. Double-click the **Transformation 2** job step to edit the job entry details.
10. Change the **Name of job entry** to `Load Web Events`.
11. For the **Transformation filename** section specify the transformation to run by browsing to `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360` and choosing `t_load_web_events.ktr`.
12. Click **OK** to return to the job canvas.
13. Double-click the **Transformation 3** job step to edit the job entry details.
14. Change the **Name of job entry** to `Load POS`.
15. For the **Transformation filename** section specify the transformation to run by browsing to `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360` and choosing `t_load_pos.ktr`.
16. Click **OK** to return to the job canvas. Your job should match the following screenshot.

17. Use your mouse to lasso-select all five job steps and then right-click any step.
18. From the right-click menu select **Align / Distribute** and then **Snap to grid**.
19. Click the save icon in the toolbar to save your new job as `j_load_mongo360` in the following directory: `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360`.
20. Run your job by either clicking ▷ or choosing **Action** | **Run** from the main menu. When the job completes without errors, green check marks will appear on each job step.

Part 2: Visualize Data with Analyzer for MongoDB

You now have a `customer360` collection containing customer master data, point-of-sale transactions, and website clickstream event data. Pentaho Analyzer provides multi-dimensional analysis a single MongoDB collection. The following exercises use Analyzer for MongoDB to analyze and visualize the data in the `customer360` collection.

Analyzer Exercise 1: Create a scatterplot visualization

This exercise has you create a scatterplot displaying individual products across two measures and then color each point in the scatterplot by the newly calculated field, `Profit_range`.

Note: The BA server should already be started from the previous exercises, but if you need to start it, execute the following script: `/pentaho/current_version/ctlscript.sh start`.

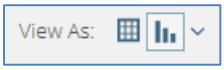
1. With your Firefox browser navigate to `http://localhost:8081/pentaho/Login` to launch the Pentaho User Console (PUC).
2. Login to PUC

   See Pentaho User Console credentials section in the document on the Desktop in the Docs folder:

   a. hds - BDI Workshop Credentials.pdf

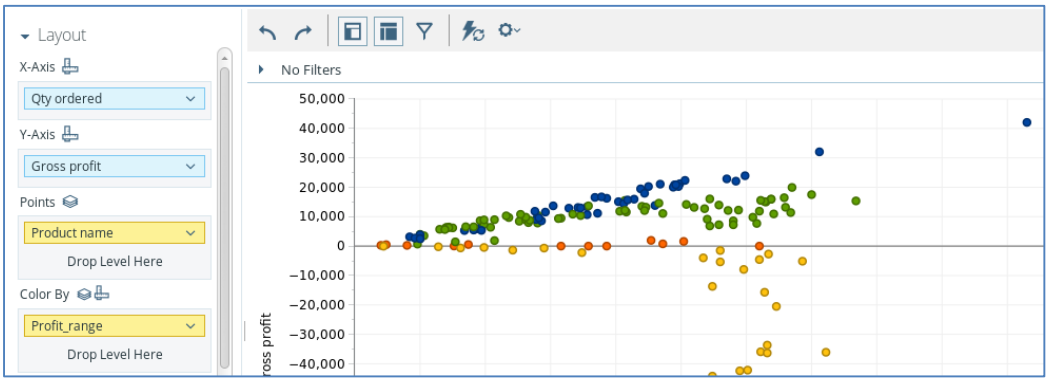
   -or-

   b. bdiw - BDI Workshop Credentials.pdf
3. Click on the **Create New | Analysis Report** buttons.
4. Select the `Customer360Mongo: Customer360PoS` data source.
5. In the **View As** section in the top right, click the chart icon drop-down and select **Scatter**.



6. Drag `Qty ordered` to **X-Axis**.
7. Drag `Gross profit` to **Y-Axis**.
8. Drag `Product name` to **Points**. Analyzer returns a scatterplot view of the products.
9. Drag `Profit_range` to **Color By**.

Profit range was calculated by PDI upon loading the POS data into MongoDB. This calculation can now be used in visualizations and reports to improve insight.

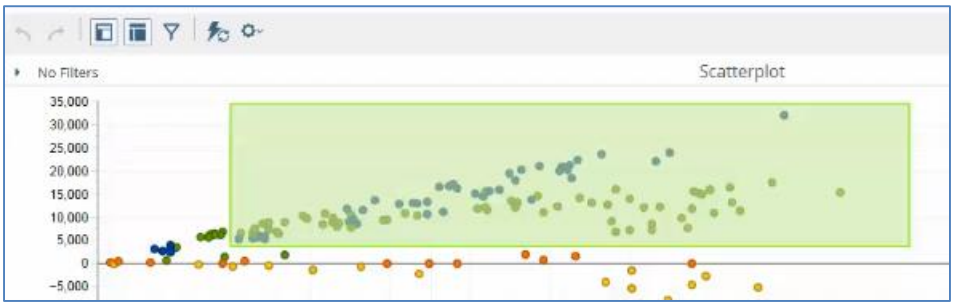


10. Click the Save as button () and save as `Scatterplot` in the following PUC directory: `/home/[PUC Username]`.

Analyzer Exercise 2: Create a high value product list report

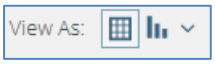
This exercise has you lasso-select the high-value products and then create a top 50 product list based on products with the highest gross profit.

1. Open the **Scatterplot** analysis (if not already open).
2. Lasso-select the points greater than `80 Qty ordered` and `5000 Gross profit`.

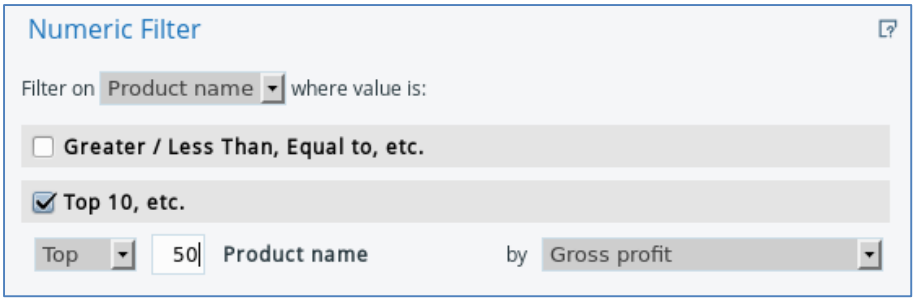


3. Select **Keep Only** from the pop-up menu.

4. From the **View As** menu, click the grid icon to return to a table view



5. Right-click `Product name` and select **Top 10, etc**.
6. In the **Numeric Filter** dialog box, change **Top** 10 to 50 and choose Gross profit in the **by** field.



7. Click **OK**.
8. From the toolbar, click the more actions icon and select **Export → To CSV**.
9. When prompted, accept the defaults and click the **Export** button.
10. Choose **OK** to open the file in Gnumeric, and then save the file to your local disk.
11. Return to PUC and click the **Save as** button in PUC **(** **)** and save as `Top 50 Products` in the following PUC directory: `/home/[PUC Username]`.

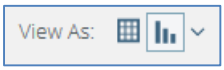
Analyzer Exercise 3: Create a linear trend line chart for gross profit

In this exercise you create a linear trend line to trend `Gross profit` across `Order date`. After trending `Gross profit`, you will then create a multi-chart for each `Language`.

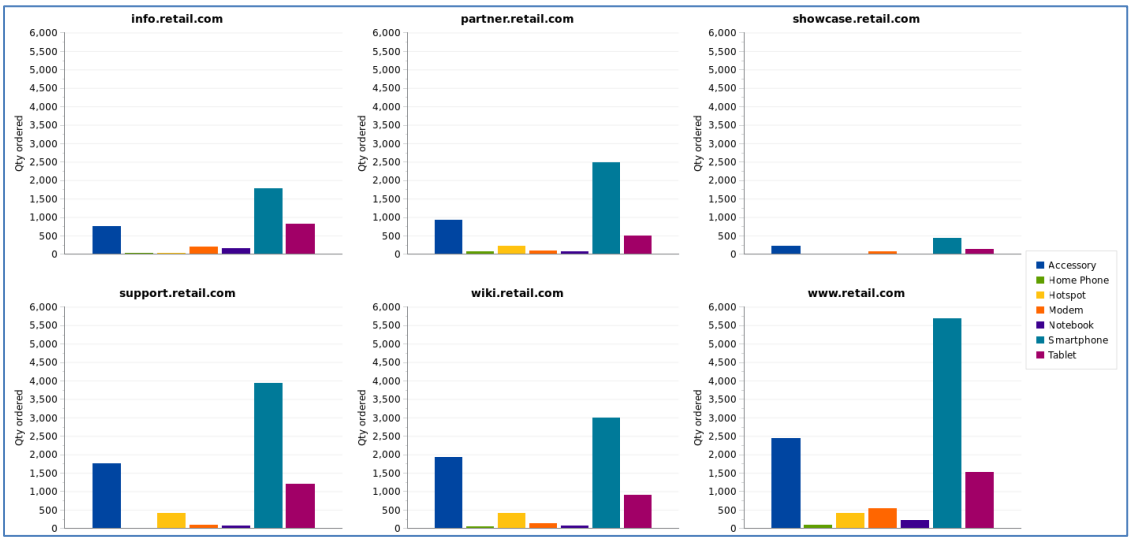
12. Click on the **Create New | Analysis Report** buttons.
13. Select the `Customer360PoS` data source.
14. Drag `Registration_referring_url` to **Rows**.
15. Drag `Qty ordered` to **Measures**.
16. Drag `Product line` to **Columns**. Analyzer returns a crosstab view of the `Qty ordered` as shown in the following image:

	Product line						
	Accessory	Home Phone	Hotspot	Modem	Notebook	Smartphone	Tablet
Registration_referring_url	Qty ordered	Qty ordered	Qty ordered	Qty ordered	Qty ordered	Qty ordered	Qty ordered
info.retail.com	757	41	49	211	168	1,791	840
partner.retail.com	930	85	241	108	82	2,498	511
showcase.retail.com	229	-	-	75	-	452	154
support.retail.com	1,777	-	429	111	72	3,957	1,208
wiki.retail.com	1,950	70	419	151	72	3,016	924
www.retail.com	2,442	104	425	551	227	5,700	1,526

17. In the **View As** section in the top right, click the chart icon drop-down and select **Column**.



18. In the **Layout** section, drag `Registration_referring_url` from **X-axis** to **Multi-Chart**. The result is a multi-chart by product line for each referring site as shown in the following screenshot.



19. Click the Save as button () and save as `Multi-Chart by Line` in the following directory: `/home/[PUC Username]`.

Part 3: Pentaho Report Designer (PRD)

Pentaho Report Designer (PRD) gives users a graphical user interface to build highly formatted, multi-page production reports. PDI provides native connectivity to MongoDB and allows users to construct, parameterize, and execute multiple MongoDB queries in a single report.

PRD Exercise 1: Add a MongoDB web metrics query

In this exercise you create a query to unwind the `event_data` array and count the number of website `Visits`, `Purchases` and `Responses`. This query will be used to populate the summary metrics at the top of the report page. A PRD report template is included with the data files to give you a head start developing a PRD report. You will need to launch PRD from the Pentaho applications menu and open this report template in PRD to begin.

1. Launch the PRD client-based report authoring tool from the launch menu icon at the bottom of your screen. Click this icon just *once* to launch PRD.

2. You will see the PRD splash screen appear while PRD loads. All open applications appear in the top left section of your screen. You can see PRD is open in the following screenshot.

3. Once PRD opens you are presented with a **Welcome** startup box. Close the **Welcome** box.
4. Click open icon on the menu toolbar or choose **File→Open** from the main menu, browse to `/pentaho/shared_content/WorkshopTraining/03_customer_360_mongodb/reports` and open `report_template.prpt`.
5. Click the **Data** tab in the top right of your screen, right click **Data Sets** and select **MongoDB** from the list.
6. While in the **MonogDB Data Source** dialog box, click the green circle with a plus symbol to add a new query.
7. Rename `Query 1` to `web_metrics_summary`.
8. Complete the four data source tabs starting with the **Configure connection** tab. Type `localhost` for **Host name** and `27017` for **Port**.
9. On the **Input Options** tab, click **Get DBs** and select `customer360` from the list of databases.
10. Click **Get collections** and select `customer360` from the list of collections.
11. On the **Query** tab, type or paste the following query into the **Query expression** input box. This query is located in the code files provided (/pentaho/shared_content/WorkshopTraining/03_customer_360_mongodb/reports/code.txt) so that you can copy and paste the code.

```
{ $unwind : "$event_data" },
```

```
{ $group : {
  _id: 1,
  Visits : {$sum: { $cond: [ { $eq: [ "$event_data.event", "Visit" ]},1,0]}},
  Purchases  : {$sum: { $cond: [ { $eq: [ "$event_data.event", "Purchase" ]},1,0]}},
  Responses : {$sum: { $cond: [ { $eq: [ "$event_data.event", "Responded to Offer"
]},1,0]}},
}}}
```

12. *Be sure* and check **Query is aggregation pipeline**.
13. On the **Fields** tab, click **Get fields**. Pentaho will scan the `customer360` collection to detect the collection schema and parse the available document fields.
14. Click the **Preview** button to run the query and view the results.

PRD Exercise 2: Add a MongoDB POS query

In this exercise you create a query to unwind the `pos_data` array and count the number of orders within each `profit_range`. This query will be used to populate the summary metrics at the top of the report page.

1. While in the **MonogDB Data Source** dialog box, click the green circle with a plus symbol to add a new query.

2. Rename `Query 2` to `pos_pie_chart`.
3. Complete the four data source tabs starting with the **Configure connection** tab. Type `localhost` for **Host name** and `27017` for **Port**.
4. On the **Input Options** tab, click **Get DBs** and select `customer360` from the list of databases.
5. Click **Get collections** and select `customer360` from the list of collections.
6. On the **Query** tab, type or paste the following query into the **Query expression** input box. This query is located in the code files provided so that you can copy and paste the code.

```
{ $unwind : "$pos_data" },
{ $group : { _id : "$pos_data.profit_range", count : { $sum : 1 } } },
{$sort:{count: -1}}
```

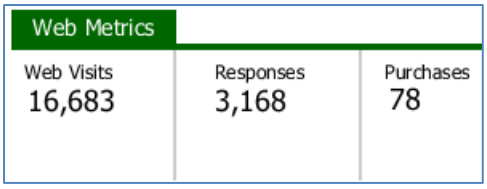
7. *Be sure* and check **Query is aggregation pipeline**.
8. On the **Fields** tab, click Get fields. Pentaho will scan the `customer360` collection to detect the collection schema and parse the available document fields.
9. In the Name column, rename `_id` field to `profit_range`.
10. Click the **Preview** button to run the query and view the results.
11. Click **OK** to return to the workspace.

PRD Exercise 3: Add web metrics fields as report elements

In this exercise you add the newly created query elements to the report by dragging them from the **Data** tab to the report canvas and configuring the alignment and font.

1. On the **Data** tab, right-click the `web_metrics_summary` query and choose **Select Query** to display the query fields.
2. Click the `Visits` query element from `web_metrics_summary` query and drag it underneath the `Web Visits` label on the **Details** band.
3. Click the `Responses` query element from the `web_metrics_summary` query and drag it underneath the `Responses` label on the **Details** band.
4. Click the `Purchases` query element from the `metrics_summary_query` and drag it underneath the `Purchases` label on the **Details** band.
5. Shift-click all three new number fields in the canvas area to select them together for editing.
6. With all three number-fields highlighted, right-click one of the highlighted fields and select **Alignment** → **Top**.
7. With all three number-fields still highlighted, click the **Structure** tab, on the right side of the screen, and locate the **font-size** property in the **Style** tab below.
8. Change the **font** to `SansSerif` and **font-size** to `18`.
9. Click the eye icon on the tab toolbar to preview your report.

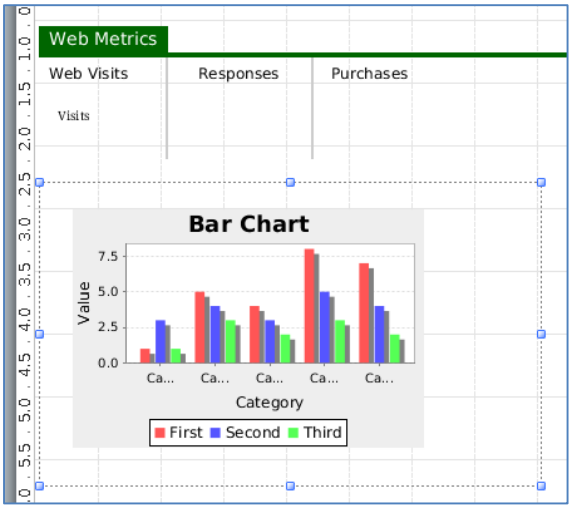
10. Your report should match the following screenshot.



PRD Exercise 4: Create a pie chart from POS data

In this exercise you create a new **Pie Chart** by adding the new POS query elements to a pie chart element and configuring the **Pie DataSet Collector** properties.

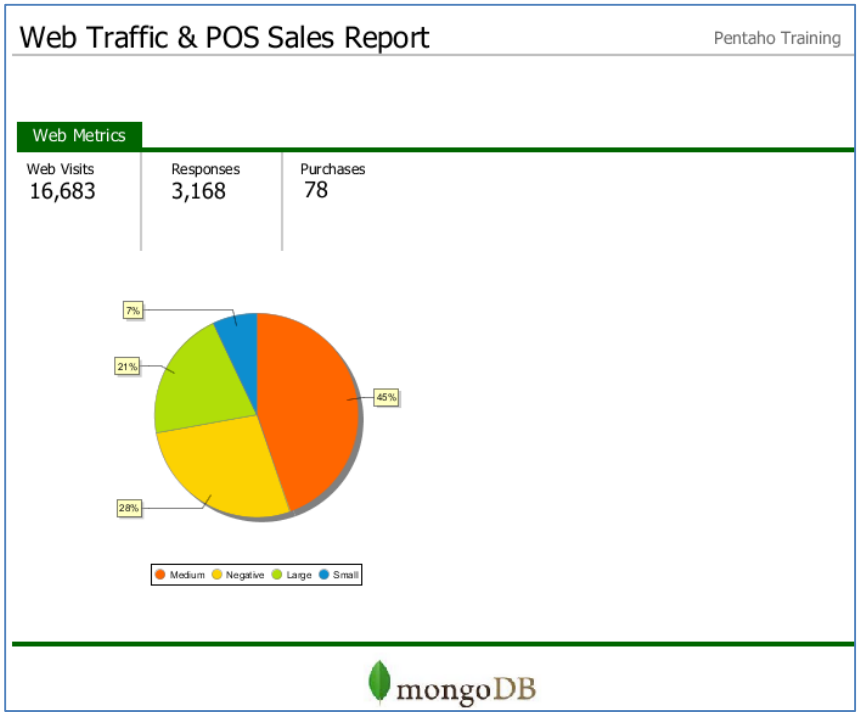
1. Click the sub-report icon  from the **Palette** on the left and drag it directly below the Web Visits metric on the **Details** band.
2. Select **Inline** when prompted to choose a sub-report type.
3. Select `pos_pie_chart` when prompted by the **Select Data Source** dialog box.
4. After choosing a query, a new tab, **<Untitled Subreport>**, will open. Click the chart icon  from the **Palette** on the left and drag it onto the new sub-report's **Report Header** band.
5. Click the **report_template** tab to return to your main report.
6. Resize your sub-report area by clicking and dragging a side of the dotted line box area. You want to resize enough to accommodate the bar chart as shown in the image below:



7. Click the **<Untitled Subreport>** tab to return to the sub-report canvas.
8. Double-click the **Bar Chart** element you just added, and **Edit Chart** dialog box will appear.
9. Select **Pie Chart** from the chart type menu.
10. The **Primary DataSource** tab contains two **Pie DataSet Collector** properties, **value-column** and **series-by-field**, to assign to query fields.
11. Select the **Value** field for **value-column,** and choose the `count` field from the drop-down menu.
12. Select the **Value** field for **series-by-field** and then click the ... button to open the **Edit Array** dialog box.
13. Select `profit_range` from the **Available Items** section and click the add items arrow to add it to the **Selected Items** section.
14. Click **OK** and your **Pie DataSet Collector** properties should match the screenshot below.

Pie DataSet Collector (Chart Data)	
Name	Value
Common	
value-column	count
Series	
series-by-field	[profit_range]

15. Click **OK** again to close the **Edit Chart** dialog and return to your report.
16. Click the eye icon on the tab toolbar to preview your report with the new pie chart.



17. Click the save icon in the toolbar to save your new report as `360_operational_report` in the following directory:

    `/pentaho/shared_content/WorkshopTraining/student_files/03_customer_360_mongodb`

You have reached the end of the Customer 360 Use Case Workshop and completed the entire Pentaho Big Data Integration Workshop! Sit back and take in the fact that you just completed three big data use cases in one day. From all of us at Pentaho, we congratulate you on job well done!