**2-3– Select & Summarize** 

In order to fully maximize an in-database toolset, alteryx also provided an in-db version of the commonly used tools in alteryx designer. This makes it easier to apply grouping, selection, and cleaning functions in the back-end data source without having to move the data out of the database. For this lesson, we are going to discuss how to apply the in-db version of the formula, select, summarize and how to create an outgoing data stream from in-db process to regular tools.

**Select In-DB tool** is used to select, deselect, rename, and reorder fields in an In-DB workflow. This tool is commonly used to limit the number of fields in an In-Database data stream in order to significantly improve runtimes and performance.

We are going to continue working with the in-db workflow we’ve built from the previous lesson. From the In-Database palette, drag a select in-db tool and place it in between the payment table connection and the first inner join. The select in-DB has simpler configuration as compared to the regular select tool. Select in-db does not have the option to change data type, size and does not show the description of each field. You also can’t save the in-db field configurations into an yxft file nor can you use decimal separators for string and numeric type conversion. Since this is an in-db procedure, you can simply select, deselect fields, re-order them and rename them. Let’s remove the fields “staff\_id” and “last\_update” since they are not needed downstream. Uncheck the checkbox for the 2 fields. Next, move the payment\_date to the top of the list. select “payment\_date” and click the options button. On the menu, navigate to Move and select “Move Highlighted Fields to Top”. You can also re-order columns by using the up and down arrows. Finally, rename the field “amount” to “AMOUNT” all in uppercase. Type “AMOUNT” on the rename column to change its field name. connect a browse in-db tool after the select in-db. Once done, run the workflow. As we can see from the first 100 records, we have reduced the data from payment table to 5 columns, renamed amount, and placed the payment\_date as the first column on our dataset.

**Summarize In-DB tool** is used to summarize data by grouping, summing, counting, counting distinct fields, and more. The output contains only the result of the calculations.

From the In-db palette, drag a summarize in-db tool and place it between the 2nd inner join (the one joined using store id) and the browse tool. The configurations of this tool look exactly alike to that of the regular summarize tool except for the fact that it has less aggregation methods available. Lets set this tool to show the total amount of sales from each store. Select the field “store\_name” and set GROUP BY as action. Then select “AMOUNT” and set the action to NUMERIC > SUM. This takes the total value of “AMOUNT” for each individual “store\_name”. Rename the output value of Sum\_AMOUNT to “total\_sales” in lowercase by typing the new name on the Output Field Name column. After completing the configuration, run the workflow to show the results on the browse in-db tool. The output displayed is summarized by each store where in Fresh Mart Texas has 37001.52 sales while Fresh Mart New Mexico has 30414.99.

As another example, lets look at the most recent date of transaction for each customer. Remove the 2 actions in the summarize in db tool by selecting both rows and pressing the minus button. Then select “customer\_id” and set the aggregation method to GROUP BY. Next, select “payment\_date” and choose the aggregation method MAX. Once all actions are complete, run the workflow. The output contains 599 since we grouped it by customer ID. The max\_payment\_date on the other hand contains the latest date or their most recent transaction.

Aside from the aggregation method of group by and sum, a few other aggregation methods can be selected using the Add button. Same as the summarize tool, this summary functions will differ based on the data types of the fields selected. The following methods are available for in database process:

* Count – computes the total number of records in the group
* Count distinct – counts the unique records in a group
* Count null - counts the unique records in a group that are null
* Count Non null – computes the number of records that are NOT null.
* Min – returns the smallest value
* Max – returns the largest value
* Average – calculates the mean by taking the sum of all values divided by the total number of values.
* Standard Deviation – gets the standard deviation of the group.
* Variance – calculates by taking the Standard Deviation and multiplying it times itself (StdDev^2)

**Formula In-DB** tool is used to create or update fields in an In-DB data stream with an expression using the native language of the database, like SQL. You can employ this tool to modify data within a database based on an expression or data relationship. While most In-DB tools do not require SQL commands, this tool requires SQL for advanced processing.

On this workflow, we are going to add a formula tool after the summarize tool in order to create a new field that will contain the number of days since last transaction. Drag a formula in-db tool and place it between the summarize and browse. The settings and functions for this tool is entirely different from that of a regular formula tool. The first configuration of the tool is the Output Field. You can directly type a field name here, or choose to update an existing field by selecting one from the dropdown. Since we are going to create a new field, we will type the name “days\_since\_last\_visit”. Next, set the data type on the dropdown. Our new field will contain whole numbers with no decimal places so we will select INT32. If an existing field is selected, the Type dropdown is disabled and is used for reference only. Next, apply the SQL Expression. First, type in “DATEDIFF(CURDATE(),” then click the button “Insert Field” and select Max\_payment\_date. Once done, add a closed parentheses on the end of the expression. Normally, a SQL expression that creates a new field would have the select command in the script, but for this tool you simply need to input the functions for the field. The function used here; **DateDiff** returns the number of days between two date values. The first date we’ve set is the **CURDATE()** which returns the current date as of running the workflow. The second date was from the existing field that we have as output from the summarize in-db and was inserted in the expression by enclosing it in single quotes. Run the workflow to show the new field. The results on the browse in-db displays the new column. You can also add more columns by adding a new row on the list. The tool runs this list from top to bottom. You can use the arrow keys to adjust the order. There is also a nifty function for checking in the bottom of the configuration. “Validate expression here at runtime” provides information about errors contained in the expression. unchecked, the query is not sent until the In-Database workflow is complete and any errors are reported in the Messages display in any downstream tools.

But what if you need to apply reporting tools after the formula in-db? We can’t directly apply regular tools without using a bridge between the local designer and the back-end process in database. In this case we can apply the Data Stream Out tool. **Data Stream Out** is used to stream data from an In-DB workflow out to Alteryx Designer.

Drag a data stream out and connect it to the formula in-db. The data stream out tool has a blue input anchor to take in the in-db stream and a green output anchor to push it out to a regular alteryx tool. Its configurations are optional as well. If you check the “Sort records before streaming out” the incoming data is sorted by the field specification. Check this function then set it to sort by “days\_since\_last\_visit” in descending order. Next, add a table tool from the reporting tool tab and connect it to the output of the data stream out. Let’s just leave its formatting to the default. To view the table, add a browse tool after it. Once all the tools are placed, run the workflow. On the report tab, we can see the table output that we’ve generated based on the in-database process.

Aside from outputting the data stream, you can also generate the in-db workflow information using the Dynamic Output In-DB tool. **Dynamic Output In-DB** is used to output information about the In-DB workflow to fields in a standard workflow stream.

For example, we simply want to see the complete SQL script for the whole in-db process. Drag a Dynamic Output In-DB and connect it to the output of the formula tool. To configure the tool, you simply need to check the fields you want on the output. Check the query and connection name then add a browse tool after it. Once done, run the workflow.

The “Query” field contains the complete sql script which includes all of the union, join, summarize, select and formula that we’ve applied in the in-db process.