**2-2– Blending**

**Data Blending** is the process of pulling data from multiple sources and combining them in order to create an actionable analytic dataset. Using two or more datasets can reveal valuable information that would otherwise go unnoticed if the data were not combined. On regular alteryx workflows, Join and Union tools are used to bring data together but we can’t employ them for in-db workflow. For this lesson, we are going to discuss the join and union tools counterpart in-db as well as the tools needed to bring in data outside the database.

**Join In-DB Tool** is used to combine 2 In-DB data streams based on common fields via an outer or inner join. Use this tool to blend 2 database tables.

We will use the Join in-DB Tool to join this customer table and the payment table to get all rental transactions with their customer information. The 2 source tables are already in place using connect in-db so we only need to add the join tool to connect them. From the In-database tool tab, drag the Join In-DB tool and place it into the canvas. Connect the customer table to its left anchor, then connect the payment table to its right anchor. **Compared to the Join Tool which has 3 output anchors, the Join In-DB tool only has a single output. Thus, its out**put will only show the matched data that was generated from the desired join type. To configure the tool, first, we need to set the join fields from both left and right sources. From the left data source, choose the field “customer\_id” as the join field. Since our second data source has the same field name, alteryx automatically selected it as the join field for the right. You can also add other join fields by indicating the new fields on another row. Do take note that this tool restricts which field types can be joined together. The selected join fields should have the same data type. String fields can only be joined to other strings and numeric fields can only be joined to other numeric fields. Mismatching data types will result in error messages once the workflow is run. In addition, double and float data types are not recommended as join fields due to rounding errors. The customer\_id we’ve assigned is set as an integer for both tables so we are in the clear. The 2nd configuration for the Join in-db tool is the Join Type. A Venn diagram is also displayed to help users understand how each join works. Let’s choose Inner Join as our first example. According to the diagram, this join will only output the overlapping data. Unmatched records will not be included in the output. Before running the workflow, let's insert a browse in-db tool and connect it to the output anchor of the join in-db tool. Set the browse first number to 0 so we can check all joined records. Once done, run the workflow.

As we can see from the result, 16 049 were matched using customer ID. The combined data contains all of the fields from both left and right input wherein the duplicate fields were renamed with an R\_ prefix. Next, edit the join in-db configuration and try the “Left Outer Join”. Its venn diagram shows that aside from the overlap, the left circle is also filled in. This means that the left outer join will show all of the records from the left input including the records that has matched with the right. Run the workflow to see the output. The left outer join produced 16,072 records. 16 049 of its was from the matched records and the remaining 23 records were not matched from the left input. You can check which records were unmatched from the left customer table by looking at the rows that has null values on the fields that came from the right input such as payment\_id and staff\_id. Next, change the join type to “Right Outer Join”. As we can see from the diagram, it covers the whole right circle which indicates that this type of join contains all of the records from the right input, including the matched records. Run the workflow to view the results. From the output, we can see that the first 16 records have null values on the fields that came from the left input (such as the first and last name of customer), but the right-side columns have data. Basically, these records are unmatched and are only available on the right payment input. The final join type is called “Full Outer Join”. This will output all of the records from both left and right input regardless if they matched or not.

Having an in-database workflow does not mean that you can’t bring in data from spreadsheets, APIs and other forms of non-database sources. Data Stream In Tool can help you with that. **Data Stream In is used to bring data from Alteryx Designer into the In-DB workflow. This tool is especially useful for scenarios when you only need to blend in a small table from another source and combine it to that of an in-db process.**

We are going to use the same set of tools we used from the join in-db example and change the join type to an inner join. Next, remove the connection between the join in-db tool and the browse tool. Next, add another join in-db tool and place it between the first join in-db and the browse tool. Once done, connect the left anchor of the join in-db to the output of the first join. The data we are going to bring in is from a text input which has the store information, staff id, and address id. Its data types were already corrected in the select tool so we can now insert a data stream in the tool to be able to join its data to our in-db process. Drag a Data Stream In tool from the In-Database palette and place it into the canvas. Connect it to the output anchor of the select tool and connect its output anchor to the right input of the second join in-db. The tool has 2 types of anchors, a green one for taking in ordinary alteryx tools and a blue output anchor that can be connected to in-db tools.

The Data Stream In tool has 3 configurations. First, we need to set the connection name. Same as the connect in-db tool, you can set it to an existing connection that is already set up, use a saved database connection file or create a new one. We will set this to connect using the MySQL in-db connection that we’ve created in the previous lesson. Next is to set the Creation Mode. In order to bring in external data to an in-db process, the data stream in the tool saves the data to the database so we can process it on the back end. The creation mode configuration lets you select the appropriate option for writing the data. **Create Temporary Table** writes to a temporary table that is available until the end of the session. This option is useful for building In-DB predictive macros because it holds the metadata in place temporarily. Since it is temporary, this disables the Table Name configuration and a unique temporary table name is generated on each run. **Create New Table** option creates a brand-new table on the database. It will not overwrite an existing table. This also enables the last configuration for Table Name to let you customize the table that will be created upon running to workflow. Lastly, **Overwrite Table (Drop)** Completely drops the existing table and creates a new one. We will set this to create a temporary table. For the join field, choose “store\_id” and set Inner Join as the join type. Finally, run the workflow. On the far-right column, we now have the store information combined with our rental transactions’ dataset.

**Union In-DB** is used to combine 2 or more In-DB data streams with similar data structures based on field names or field positions. In the output, each column contains the data from each input.

Let’s union our customer table to the e-commerce customer. Drag a new connect in-db tool to the workflow and place it above the customer input. Set it to connect to the existing mysql in-db connection. Once the visual query builder window has opened, select the sakila schema and double click on ecommerce-customer table. Select all of the columns on the table and hit OK once done. Then, drag a Union in-db tool and connect it to the input. Delete the connection of the customer input to the join tool and connect the customer input to the union in-db tool as its second source. Then connect the output anchor of the union tool to the left anchor of the join in-db.

The union in-db tool only has 2 configurations. First is to choose the preferred configuration mode. You can either **Auto Config by Name** to align the union by field name or **Auto Config by Position** to align fields by their field order in the stream. We will set this to auto config by name since the ecommerce table has the same field names. Next is to configure when fields are different. This process handles nonconforming data fields. You can choose to either Error-stop processing -to throw an error and end the schema, Output all fields – to show all and have null values populate empty fields, or Output common subset of fields – to only show fields that each input has in common. We will choose “Output common subset of fields” then run the workflow. By adding a union, we were able to expand the list of customers that we can join our payment table into.