School Of Computer Application

Bachelors of Computer Applications

Babu Banarasi Das University

Lucknow

Academic Session 2025 – 2026

Name : YASHASHVI SRIVASTAVA

Section : BCADS 36

Roll No: 1230258484

Assignment: PREDICTIVE ANALYTICS

Semester: 5th

Submit To : MR. AYUSHMAN BHADAURIA

[illegible]

PHONE: [illegible]

INDEX

Sr. No.	Name of Experiment	Date	Faculty signature	Remarks
1	DEMONSTRATE THE USE OF NODES IN SPSS FOR IMPORTING A DATASET, APPLYING DATA FILTERS TO SELECT SPECIFIC RECORDS, AND EXPORTING THE FILTERED RESULTS.	04/09/25		
2	DEMONSTRATE HOW TO USE NODES IN SPSS TO GATHER INITIAL DATA FOR A TELECOMMUNICATIONS COMPANY.	08/09/25		
3	CREATE A DATA-MINING PROJECT TO PREDICT CHURN IN TELECOMMUNICATIONS FIRM.	10/09/25		
4	DEFINE THE UNIT OF ANALYSIS FOR THE TELECOMMUNICATIONS DATASET IN SPSS USING NODES.	12/09/25		
5	Show the Integration of telecommunications datasets using SPSS.	24/09/25		
6	Demonstrate the working of derive and reclassify nodes for the telecommunications datasets using SPSS.	27/09/25		
7	Identify relationships in the telecommunications data using SPSS.	08/10/25		
8	Use functions to cleanse and enrich telecommunications data	10/10/25		

Practical 08

Project Definition:

In this task, you are working with customer demographic and churn data for a telecommunications company. The goal is to clean, refine, and enrich the dataset by creating new fields and preparing the data for further churn analysis and modelling. The dataset used is telcoxsubset.dat.

Outcomes / Learning:

This practical teaches how to:

- **Clean raw fields** to make data uniform and usable
- **Create new fields** using the **Derive node**
- Apply different derived operations such as **Formula, Conditional, and Flag**
- Professionally prepare customer-level data for future analysis

Required Tool: IBM SPSS Modeler

Working:

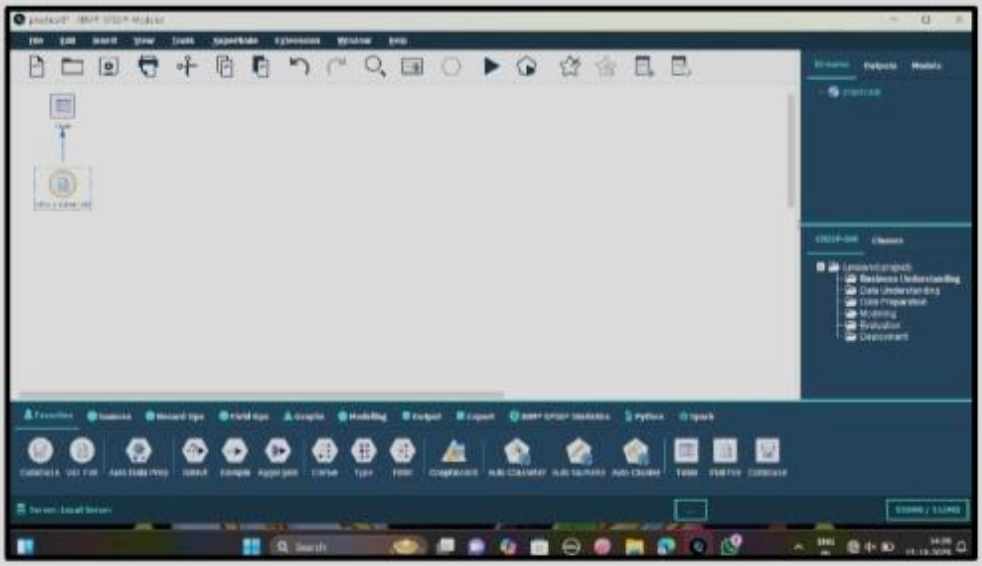
The stream mainly uses the **Derive node** multiple times to:

- Calculate date differences
- Extract month names
- Validate email formats
- Compute revenue-related values

  This process ensures the dataset becomes **clean, consistent, and analysis-ready**.

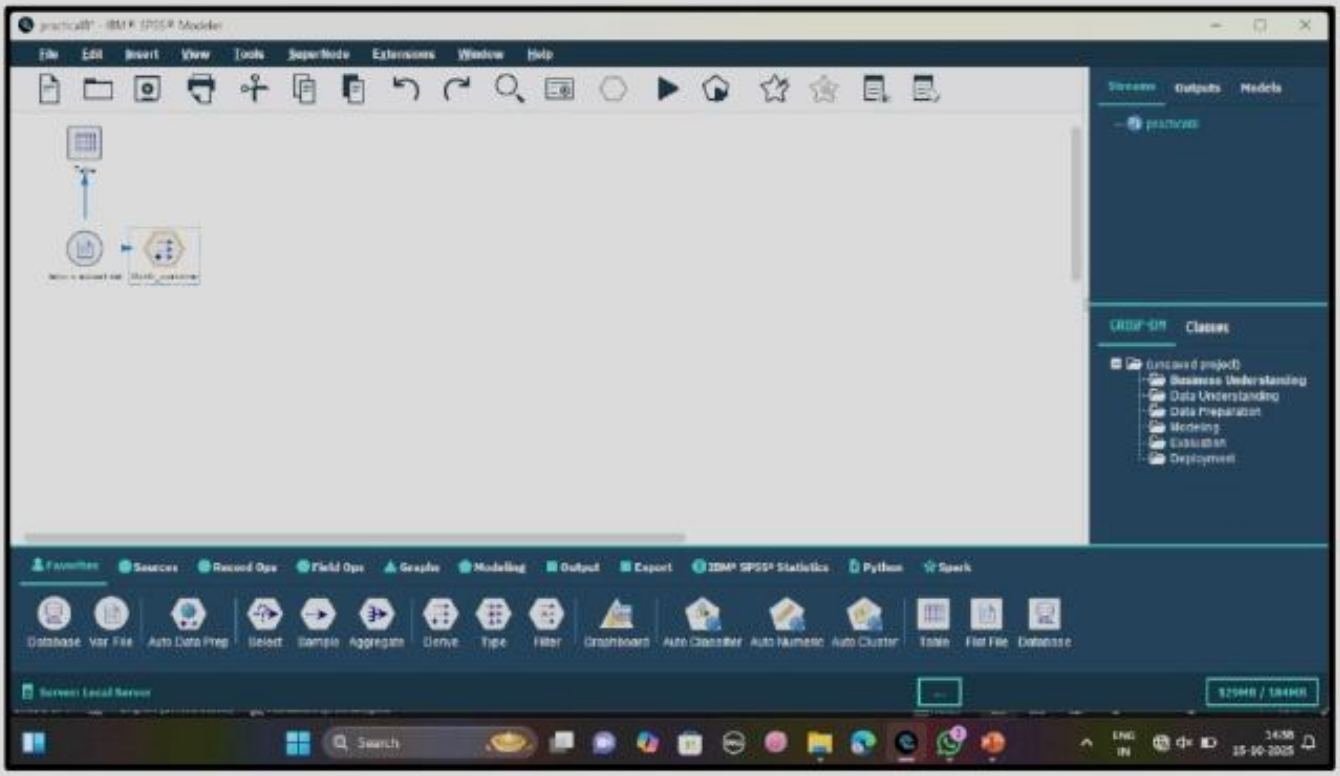
Step 1:

Load the dataset **telcoxsubset.dat** using the **Var. File** node and attach a **Table** node to display the raw data.



Step 2:

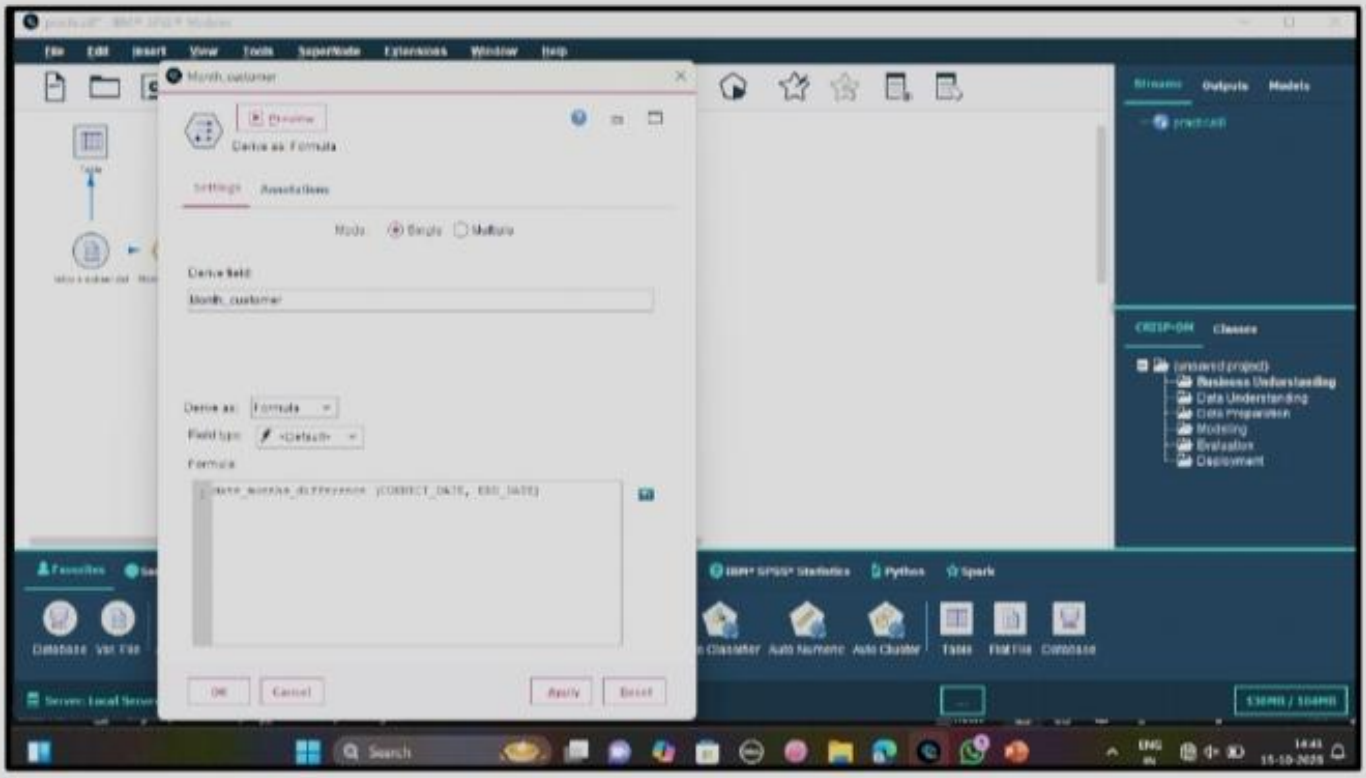
Add a **Derive** node and rename it **month_customer** to begin creating new variables based on the dataset.



Step 3:

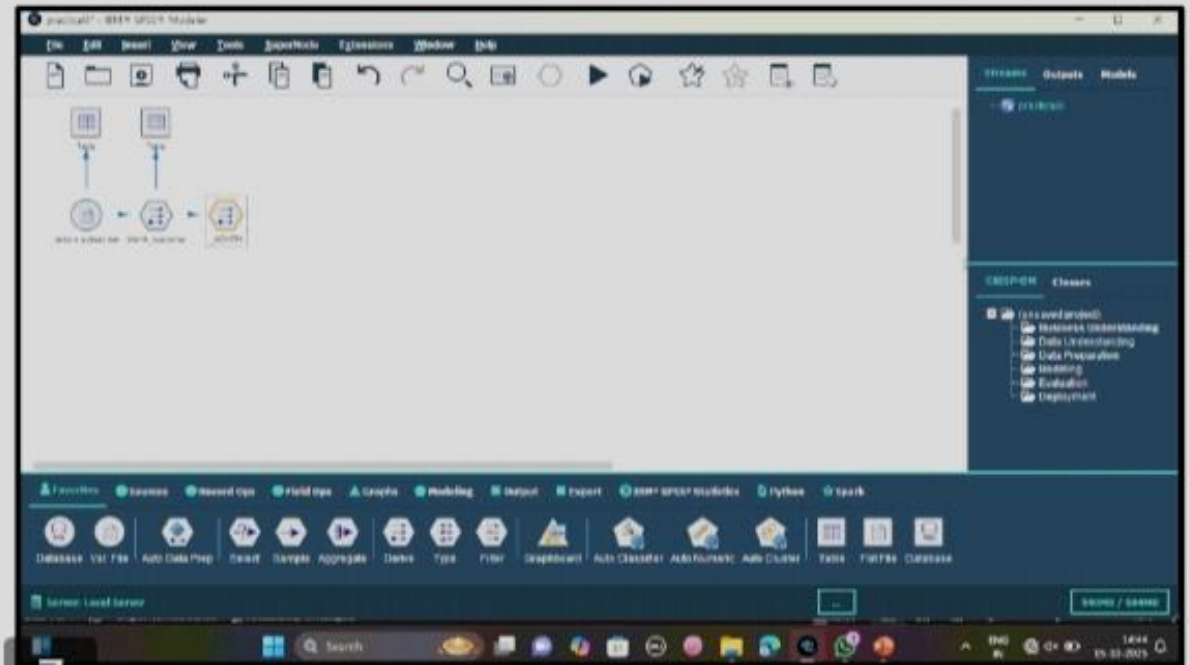
Use a **Formula-type Derive** to calculate how many months each customer stayed with the company.

Formula: **date_months_difference(CONNECT_DATE, END_DATE)**



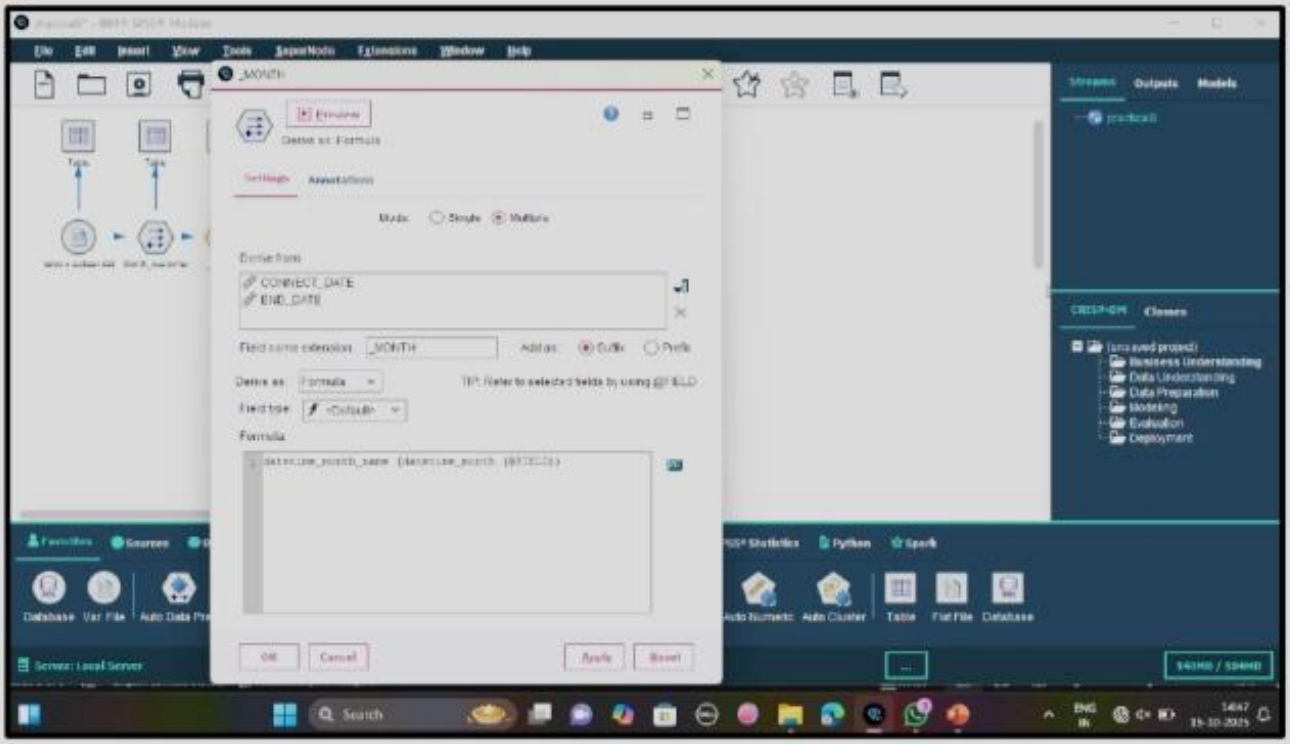
Step 4:

Insert another **Derive node** after the previous one to extract month names from the connection and disconnection dates.



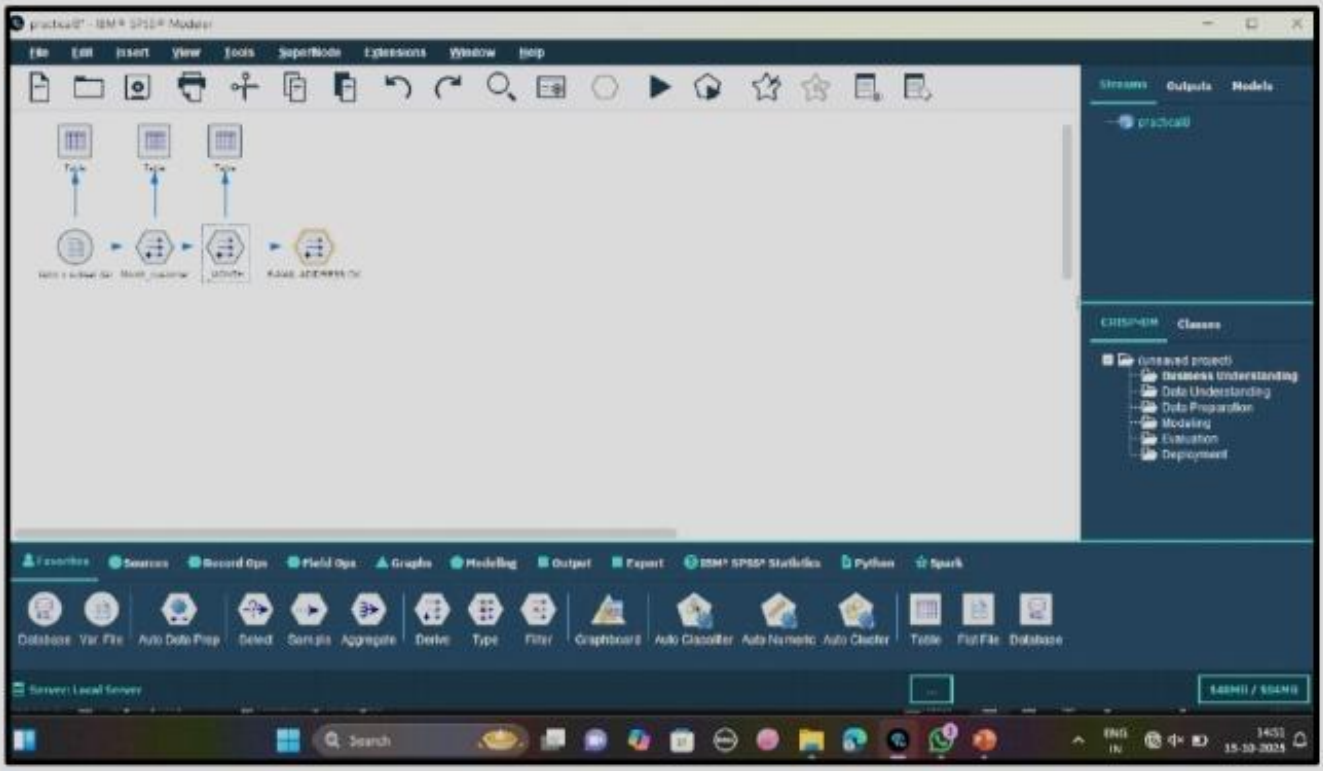
Step 5:

To identify the month names, apply the expression (using @field to avoid repeating operations): datetime_month_name(datetime_month(@FIELD)). Connect a **Table** node to view the output.



Step 6:

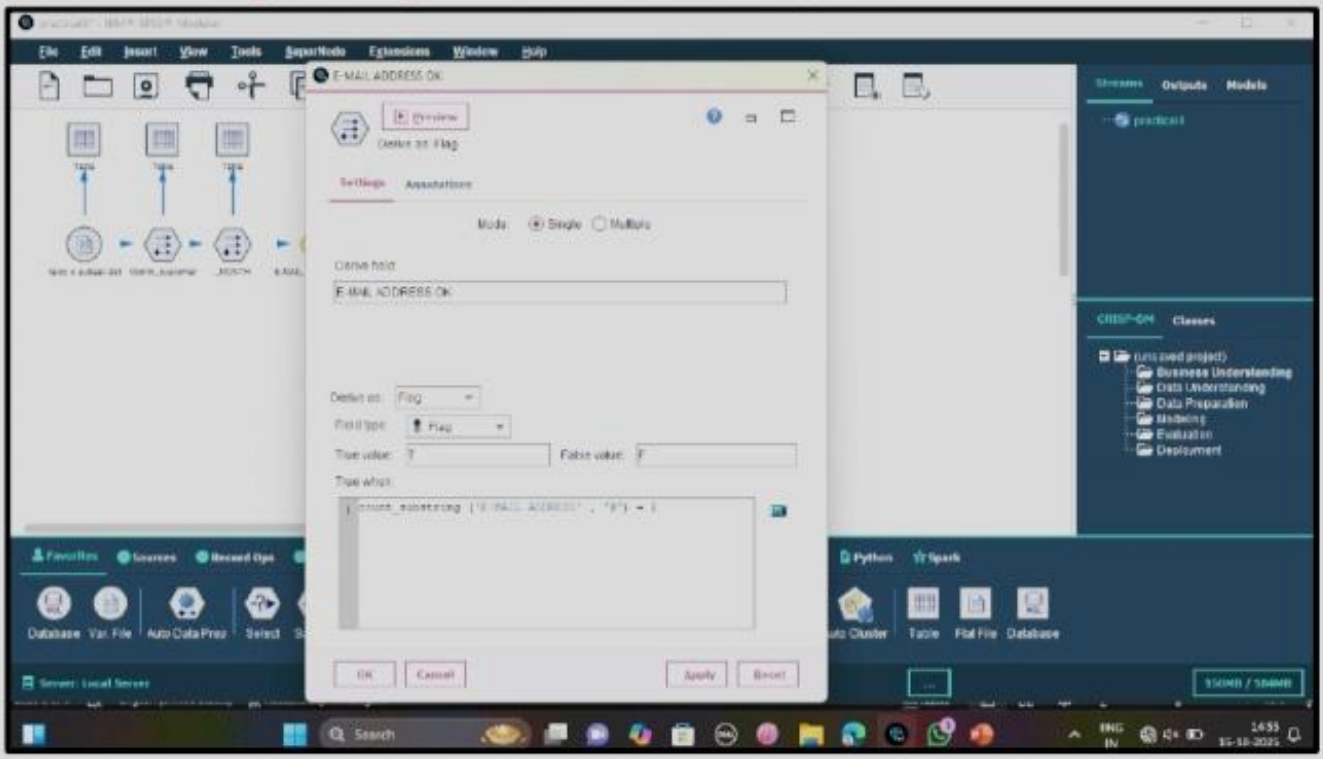
Add another **Derive** node to determine whether a customer has provided an email address.



Step 7:

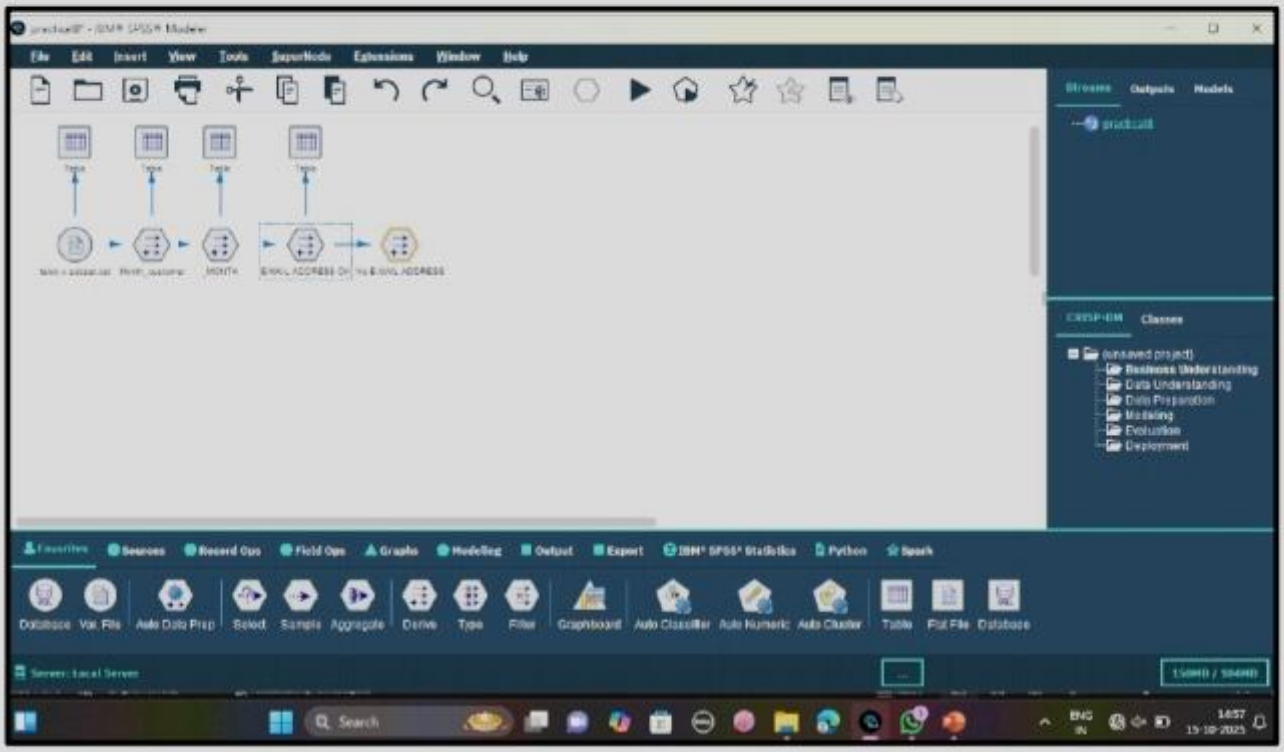
Use the **Flag derive type** to check for the presence of an '@' symbol in the email.

Formula: **count_substring('E-MAIL ADDRESS', "@") = 1**



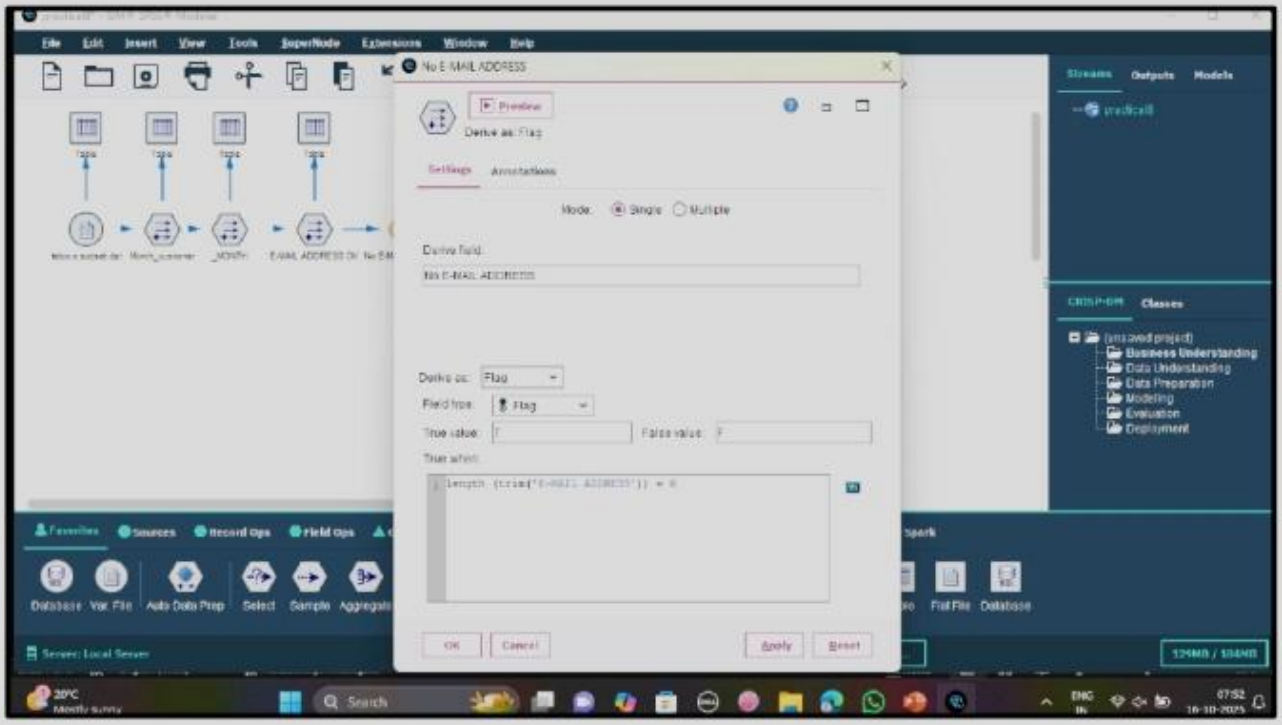
Step 8:

Insert another **Derive node** to identify customers with missing or blank email addresses.



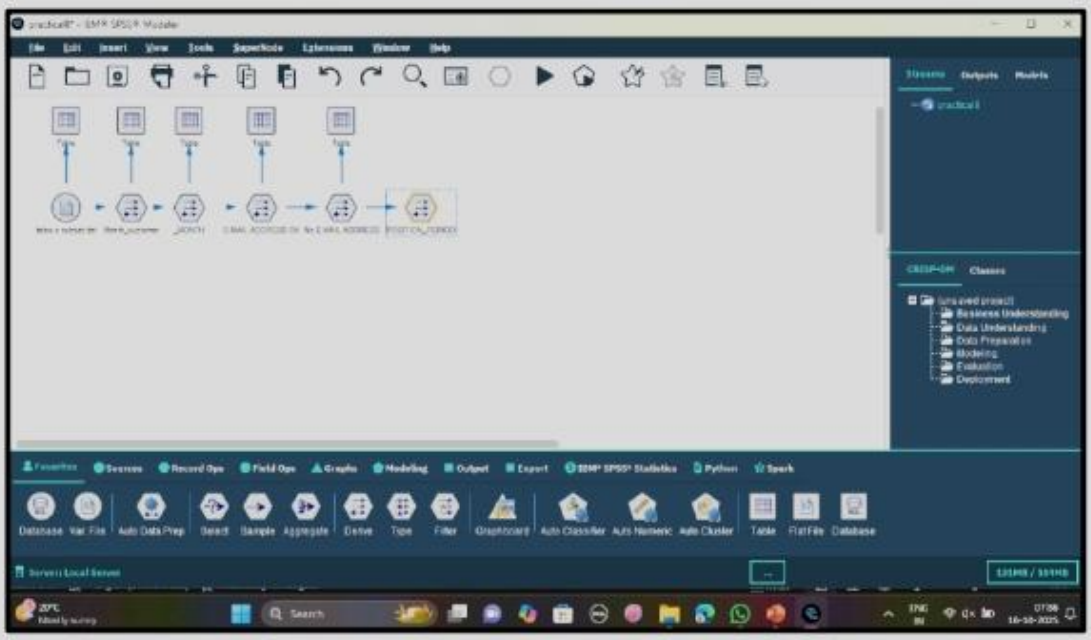
Step 9:

Use a Flag derive type and apply the string length check:



Step 10:

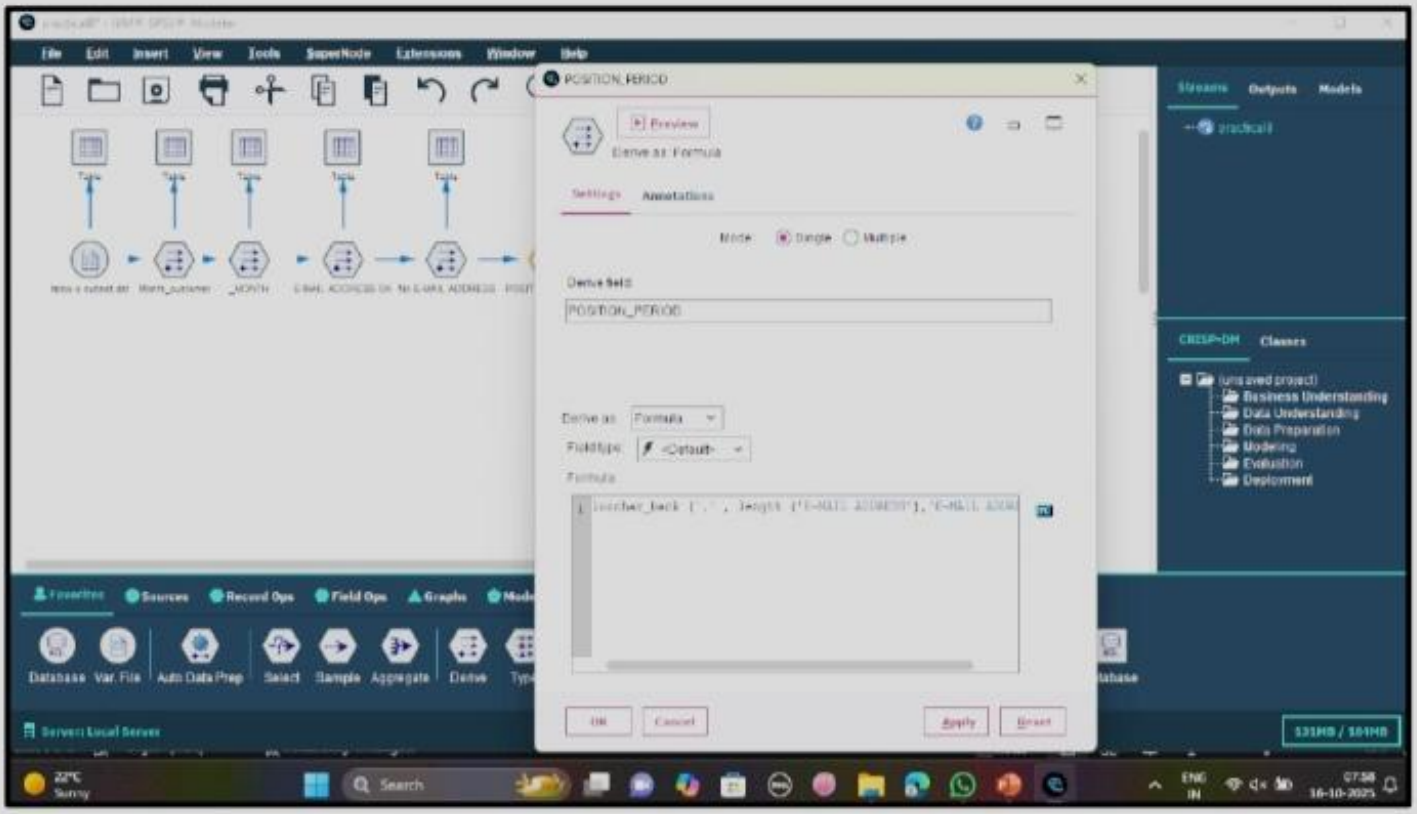
Add another **Derive node** and rename it position_period to locate the last dot (.) in the email string.



Step 11:

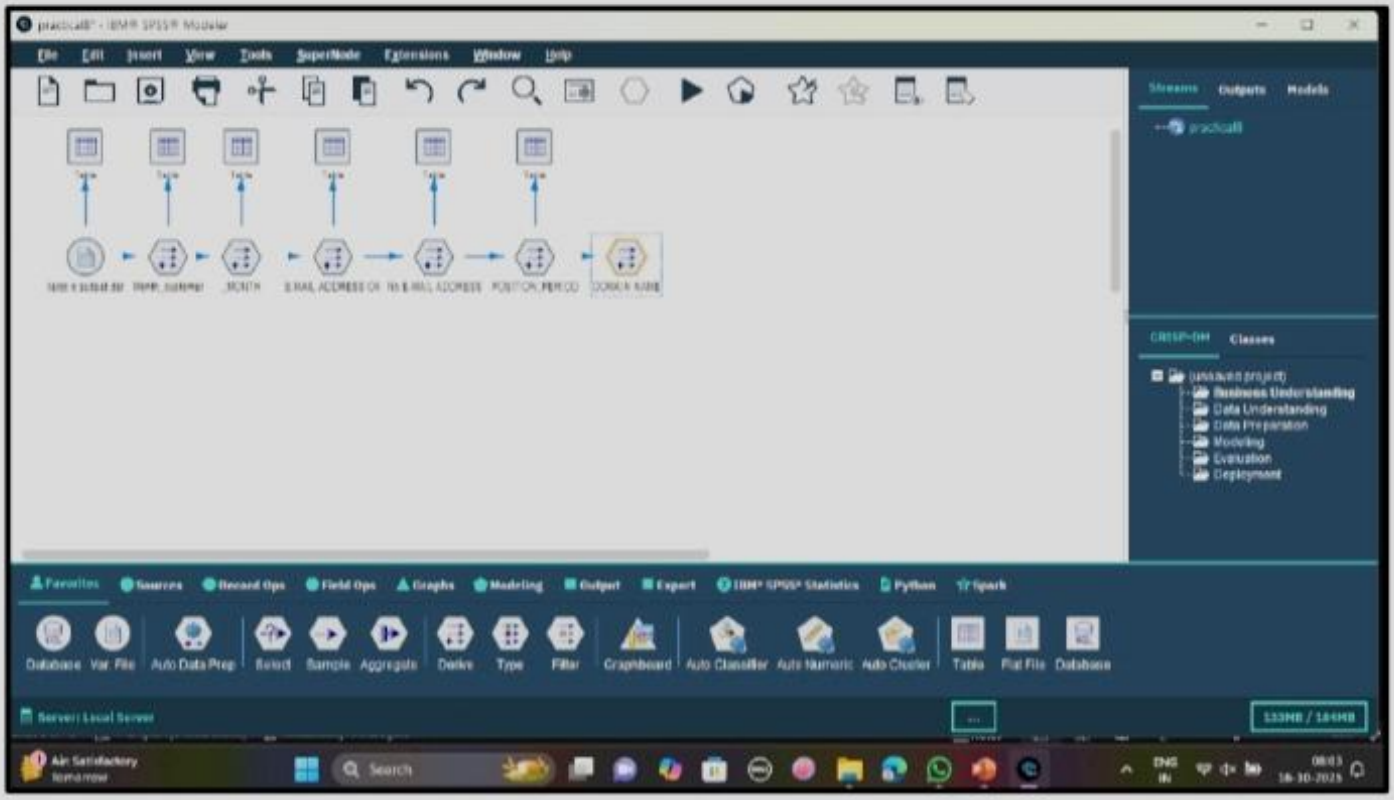
Apply a **Formula-type** derive to find the position:

locchar_back('.', length('E-MAIL ADDRESS'), 'E-MAIL ADDRESS')



Step 12:

Insert another **Derive** node named **domain_name** to extract text after the last dot.

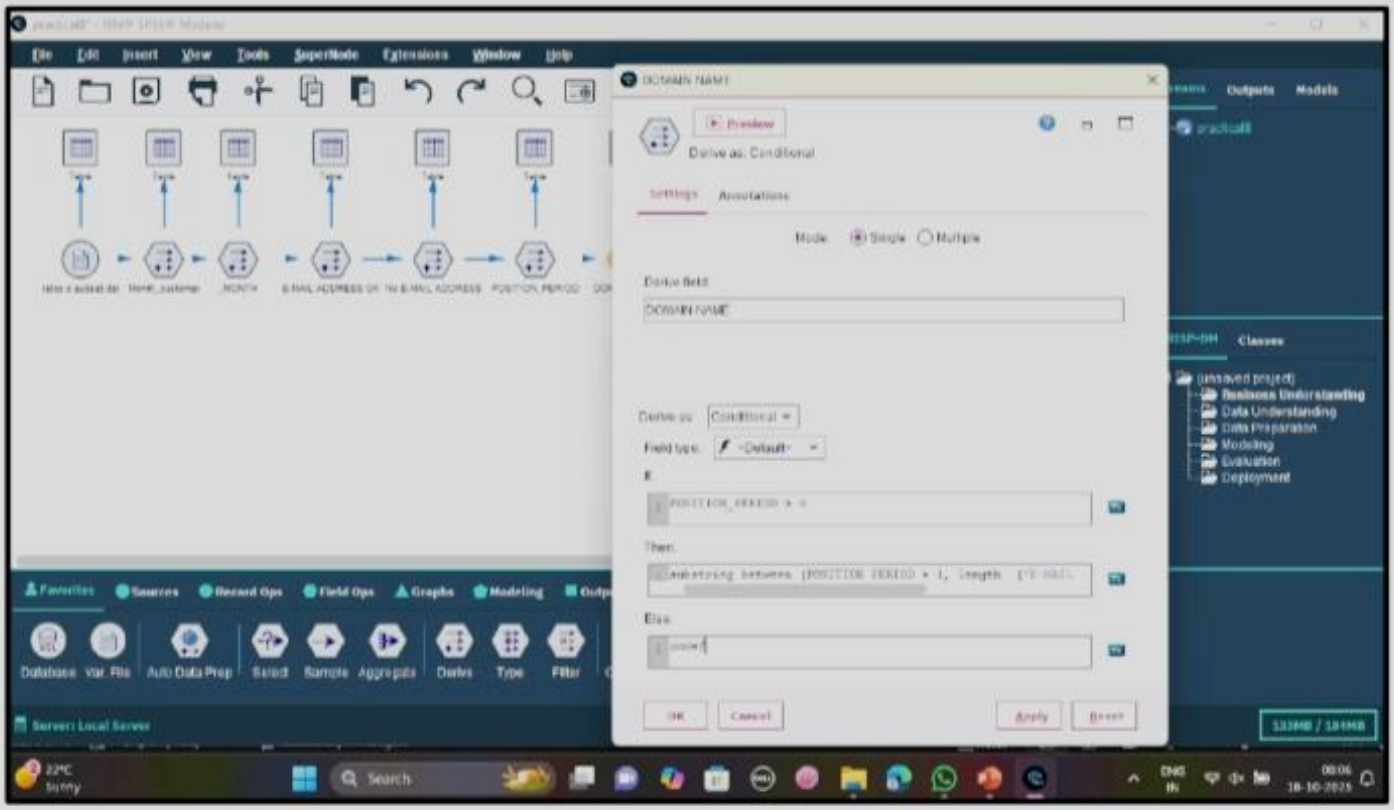


Step 13:

Use the **Conditional derive type** with the following logic:

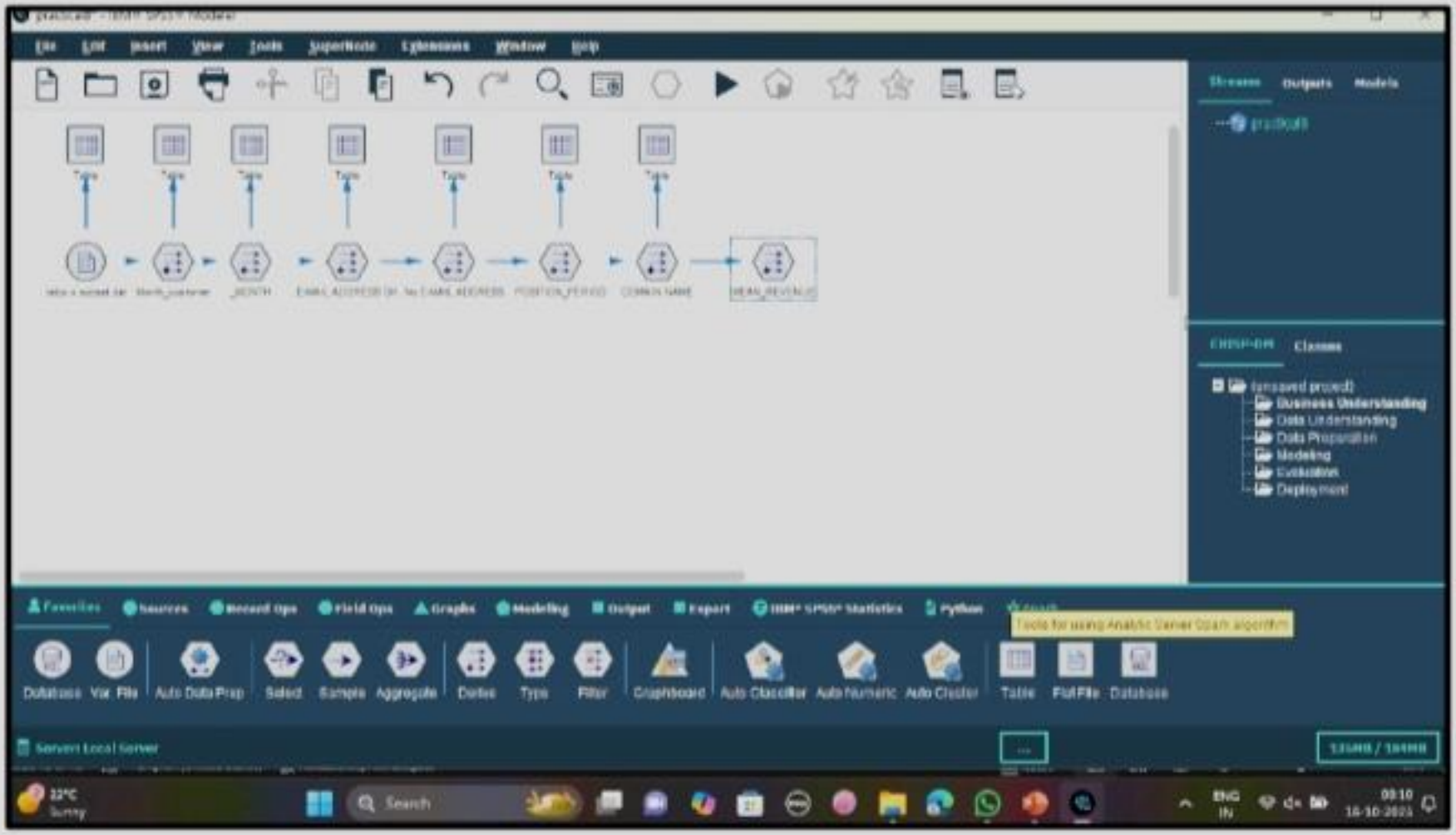
- If POSITION_PERIOD > 0
- Extract the substring after the dot
- Otherwise return **undef**

substring_between(POSITION_PERIOD + 1, length('E-MAIL ADDRESS'), 'E-MAIL ADDRESS')



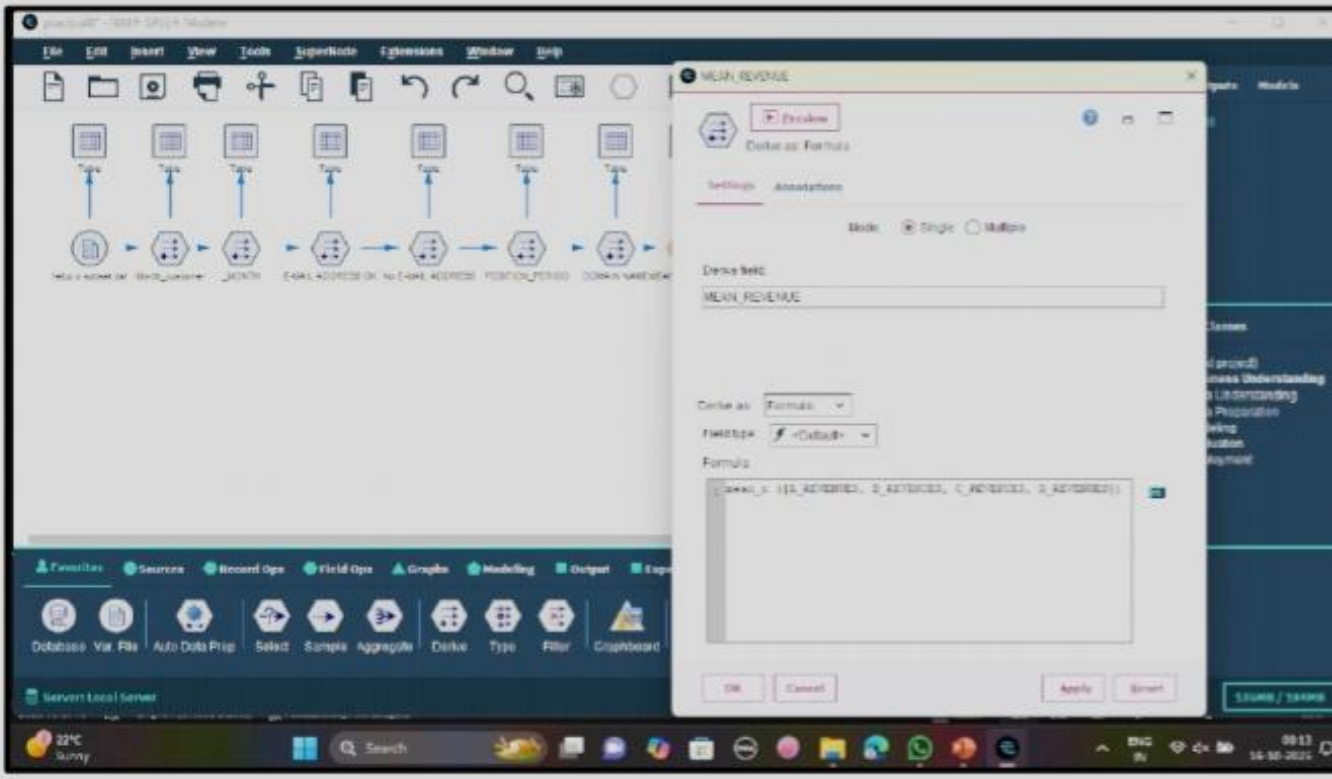
Step 14:

Add another **Derive** node named **mean_revenue** to calculate the average revenue.



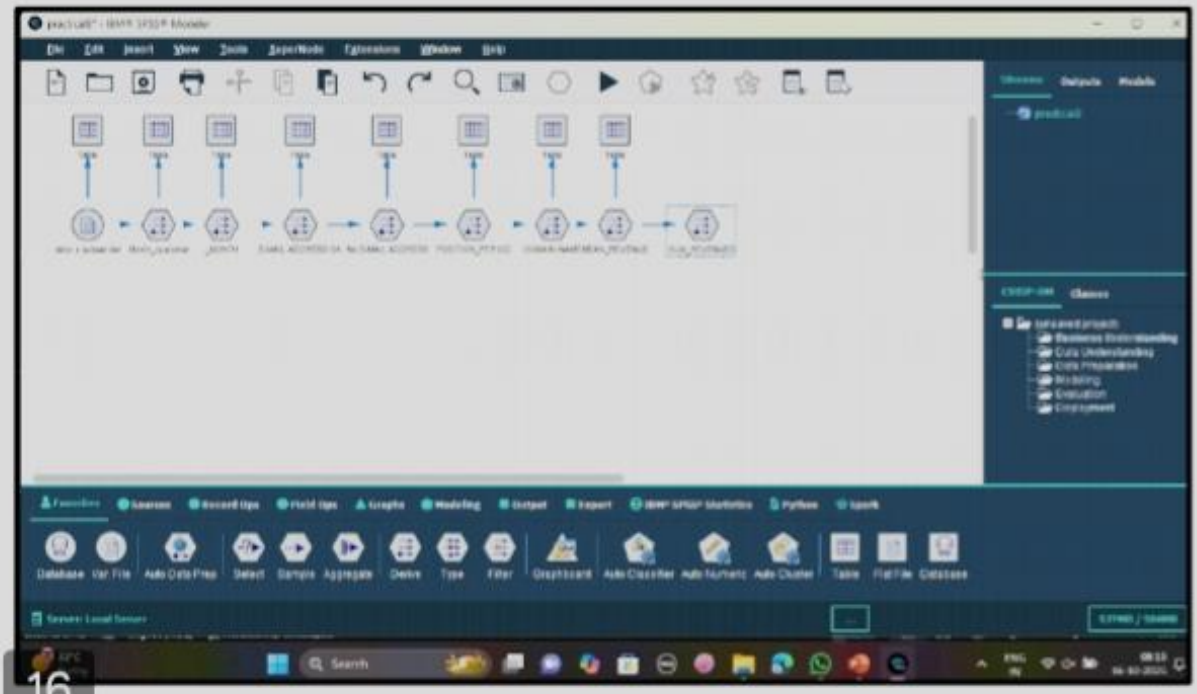
Step 15:

Use **Formula-type derive** to compute the mean of revenue fields: mean_n([A_REVENUES, B_REVENUES, C_REVENUES, D_REVENUES])



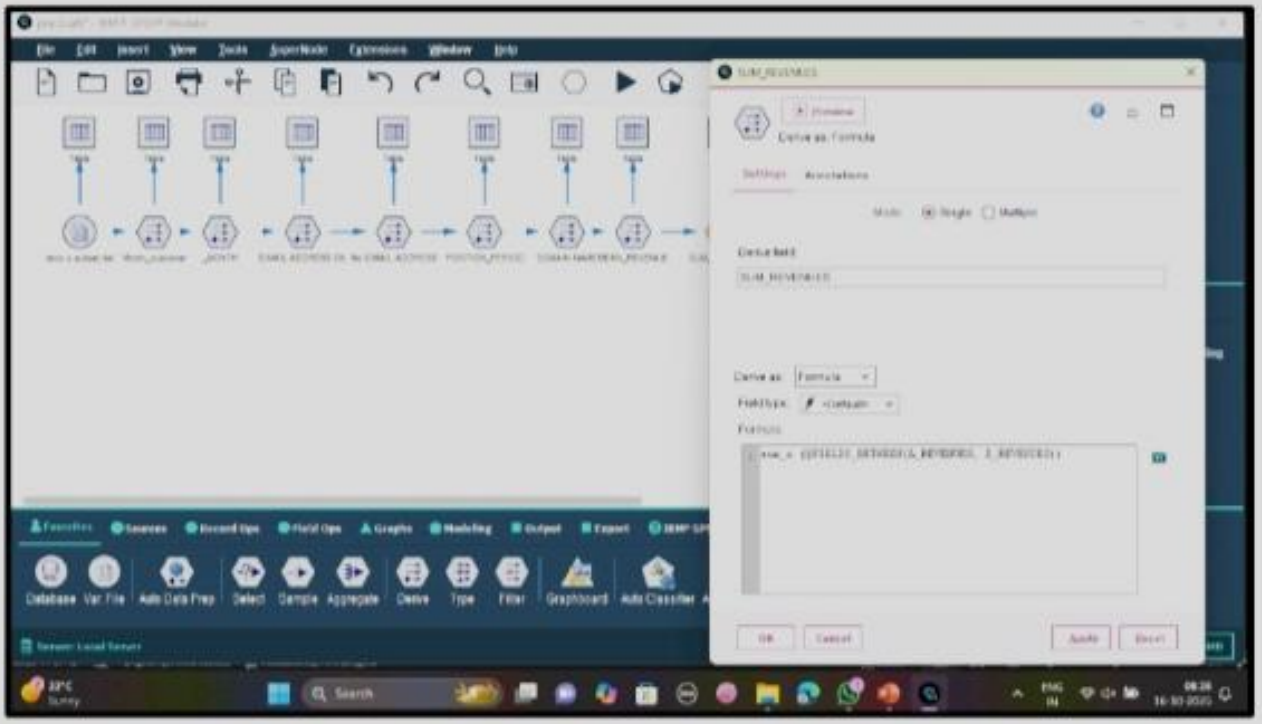
Step 16:

Insert another **Derive** node named **sum_revenue** to calculate the total revenue.



Step 17:

Apply sum formula using: sum_n(@FIELDS_BETWEEN(A_REVENUES, D_REVENUES))



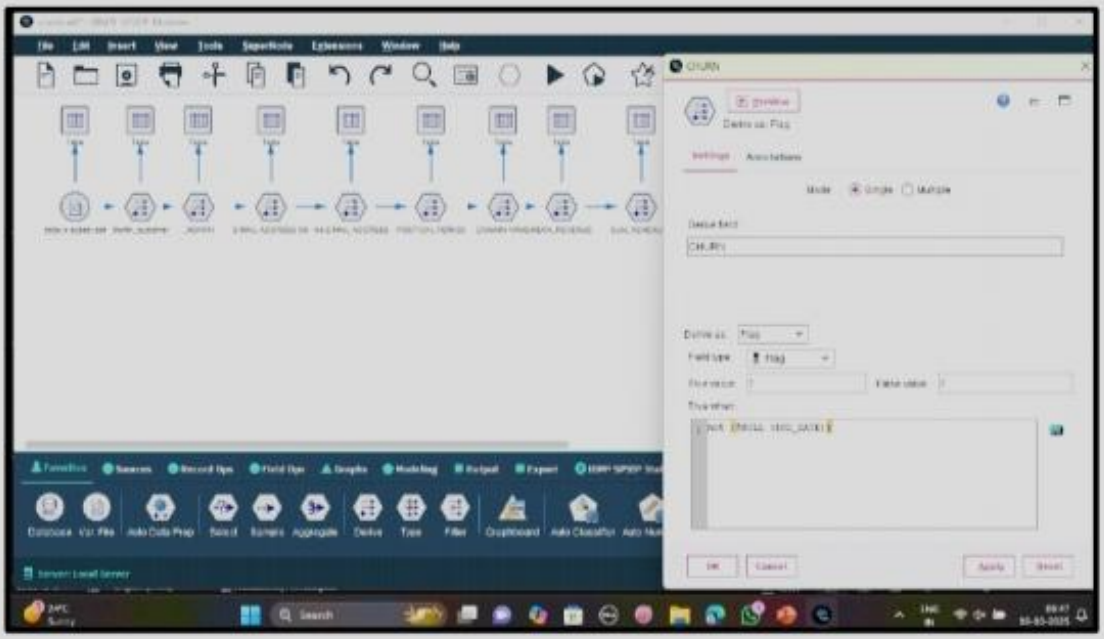
Step 18:

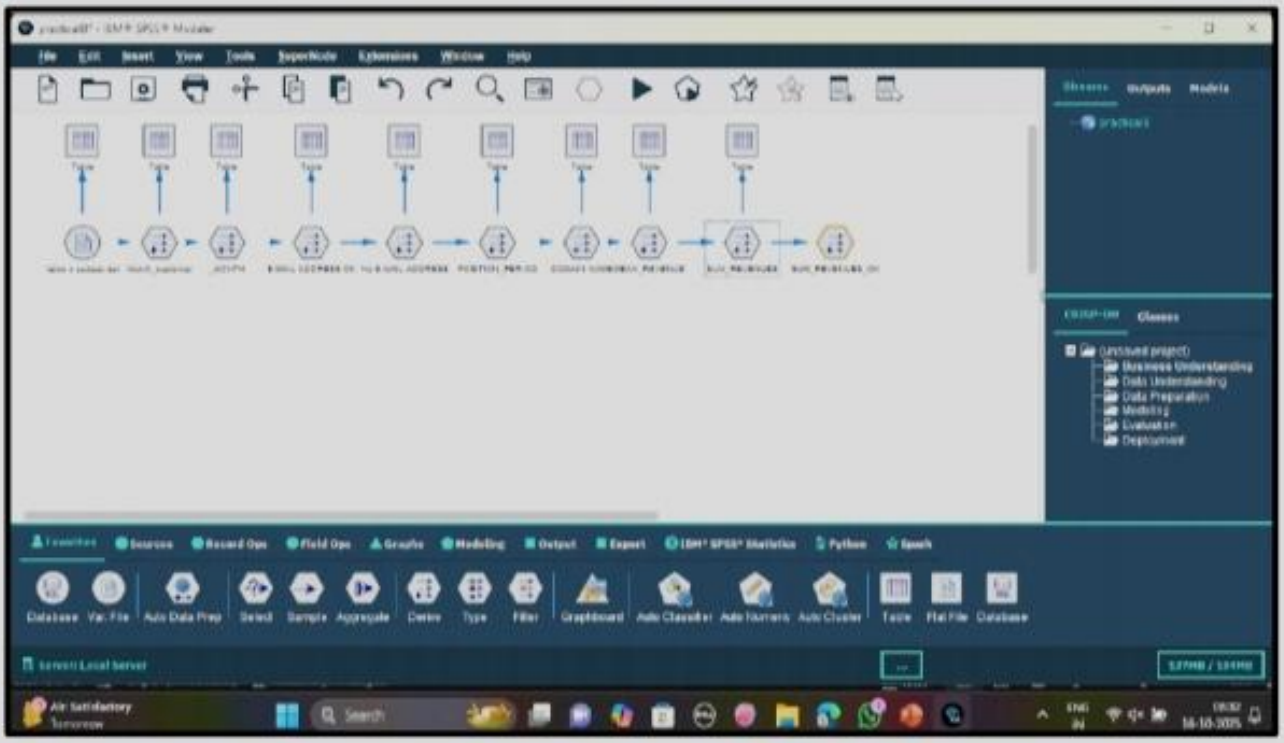
Add another **Derive** node and rename it **sum_revenue_ok**.

Step 21:

Use the **Flag derive type** to detect whether a customer has churned based on the **END_DATE** field, and attach a **Table** node to view the final dataset.

not(@NULL(END_DATE))

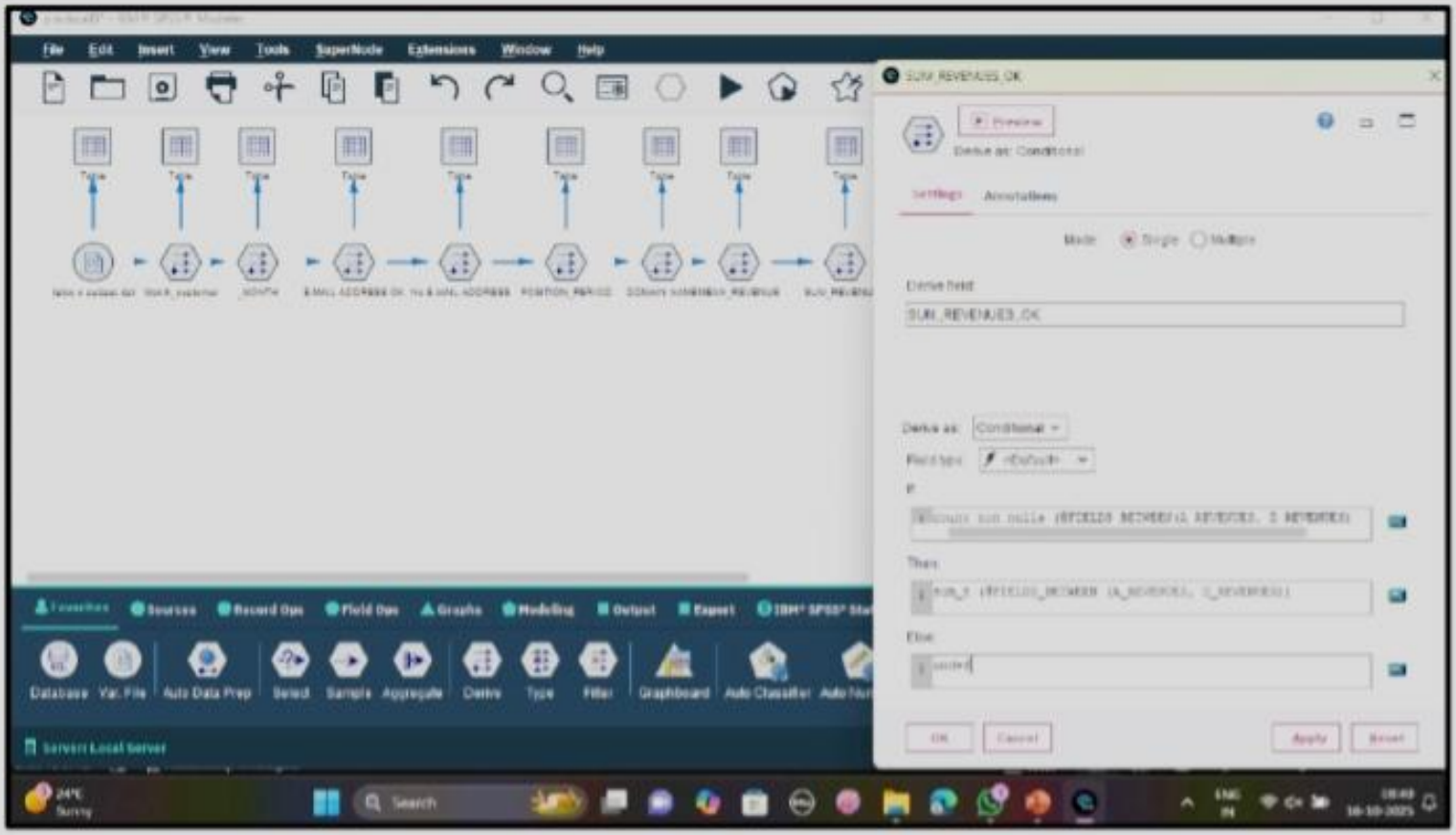




Step 19:

Use **Conditional derive** to return the sum only if values are not null, otherwise return undef:

- count_non_nulls(@FIELDS_BETWEEN(A_REVENUES, D_REVENUES)) > 0
- sum_n(@FIELDS_BETWEEN(A_REVENUES, D_REVENUES))
- undef



Step 20:

Insert a final **Derive** node and rename it **churn**.