To build a logistic regression model based on these insights, let's focus on segmenting by customer behavior patterns (e.g., shipping mode preferences, seasonal ordering habits, and regional purchasing trends). Since these variables have shown significant associations, they could serve as predictors in our logistic regression model.

Steps to Build the Logistic Regression Model

1. **Define the Target Variable**:
   - Let's assume our goal is to predict the likelihood of a particular *customer segment* buying during a specific *Order Month*. So, our target variable will be a binary outcome: whether a customer belongs to a particular segment based on other predictors (e.g., Order Month, Ship Mode, Region).
2. **Select Predictor Variables**:
   - Based on the chi-squared test results, we'll use:
     - **Ship Mode**
     - **Order Month**
     - **Region**
   - These predictors are categorical, so we'll encode them using dummy variables.
3. **Prepare the Data**:
   - Create dummy variables for the categorical predictors (`Ship Mode` , `Order Month` , and `Region`).
   - Split the data into training and testing sets to evaluate the model.
4. **Build and Train the Logistic Regression Model**:
   - Fit a logistic regression model using `Ship Mode` , `Order Month` , and `Region` as predictors.
5. **Evaluate the Model**:
   - Use metrics like accuracy, precision, recall, and AUC-ROC to assess model performance.

Here's an example code implementation in Python:

python Copy code

```
import pandas as pd from sklearn.model_selection import train_test_split from
sklearn.linear_model import LogisticRegression from sklearn.metrics import
classification_report, roc_auc_score, confusion_matrix import statsmodels.api as sm #
Assuming data is your DataFrame and contains 'Segment', 'Ship Mode', 'Order Month', and
'Region' # Step 1: Define Target Variable data['Target_Segment'] = (data['Segment'] ==
'desired_segment').astype(int) # Adjust 'desired_segment' as needed # Step 2: Select
Predictor Variables and Create Dummy Variables X = pd.get_dummies(data[['Ship Mode',
'Order Month', 'Region']], drop_first=True) y = data['Target_Segment'] # Step 3: Split
the Data X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42) # Step 4: Initialize and Train the Model log_reg = LogisticRegression()
```

```
log_reg.fit(X_train, y_train) # Step 5: Model Evaluation y_pred = log_reg.predict(X_test)
y_pred_proba = log_reg.predict_proba(X_test)[:,1] print("Classification Report:\n",
classification_report(y_test, y_pred)) print("ROC AUC Score:", roc_auc_score(y_test,
y_pred_proba)) print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred)) # For
detailed statistics, use statsmodels X_sm = sm.add_constant(X_train) log_reg_sm =
sm.Logit(y_train, X_sm).fit() print(log_reg_sm.summary())
```

Explanation of the Process:

- **Defining the Target Variable**: Here, `Target_Segment` represents whether a customer belongs to a specific segment, which you can adjust according to your business objective.
- **Using Dummy Variables**: Dummy encoding converts categorical variables (Ship Mode, Order Month, and Region) into a format suitable for logistic regression.
- **Training and Evaluation**: The logistic regression model is trained on the predictors and target variable, and its performance is evaluated on test data.

Insights from Chi-Squared to Logistic Regression

The chi-squared results revealed that our predictors (`Ship Mode` , `Order Month` , `Region`) are significantly associated with customer segments and could impact their purchasing behaviors. This association suggests that including these predictors in our logistic regression model could help improve the accuracy of predicting customer segments based on these preferences.