Solution Approach: -

1. Data Loading and Inspection

- train = pd.read_csv("train.csv")
- test = pd.read_csv("test.csv")

- Inspect the structure of the dataset using train.head() to get an idea of what features are available.
- Check for missing values using train.isnull().sum() and handle them appropriately (either by imputation or removal).

2. Data Preprocessing

- **Handling Missing Values**:
  - For numerical columns like Item_Weight, use mean imputation to fill in the missing values.
  - For categorical columns like Outlet_Size, use the mode (most frequent value) to fill in the missing values.

- **Standardizing Categorical Features**:
  - In columns like Item_Fat_Content, there is inconsistent labels (like 'lf', 'low fat') Standardize these values using a mapping dictionary to 'Low Fat'

3. Feature and Target Separation

- We separate the features (X) and target (y) in the training set

4. Identifying Categorical Columns

- Identify which columns are categorical (i.e., object dtype) to inform CatBoost on how to handle them.

5. Train-Test Split

- Split the dataset into training and validation sets to assess the model's performance during training.

7. Model Training with CatBoostRegressor

- We train a model using **CatBoostRegressor**, which is a powerful gradient boosting algorithm that works well with both numerical and categorical features.

8. Hyperparameter Tuning

- Perform hyperparameter tuning using cross-validation (cv) to find the best parameters for the model.

9. Model Evaluation

- After training, evaluate the model's performance using metrics like **RMSE (Root Mean Squared Error)**. Print the best validation RMSE score during cross-validation.

10. Model Prediction

- Make predictions on the test set using the trained model.

11. Prepare Submission File

- Prepare the final submission file by combining the predicted sales with the corresponding Item_Identifier and Outlet_Identifier.

12. Conclusion

- We've built and evaluated a model for predicting Item_Outlet_Sales. The model uses **CatBoostRegressor**, which is particularly good at handling categorical data, and we've addressed issues like missing values and inconsistent categories. We've also validated the model using cross-validation to ensure it generalizes well to new data.