XYZ TECH is a Neo vehicle insurance provider based in major Indian city. The dataset contains insurance claims filed by customers during April 2025, including details like claim amounts, premiums collected, payment status, and rejection reasons (if any).

Below is columns description:

Column Name	Description
`CLAIM_ID`	Unique identifier for each insurance claim.
`CLAIM_DATE`	Date on which the claim was filed.
`CUSTOMER_ID`	Unique ID assigned to the customer filing the claim.
`CLAIM_AMOUNT`	Total amount claimed by the customer (in INR).
`PREMIUM_COLLECTED`	Amount of premium collected from the customer for the policy (in INR).
`PAID_AMOUNT`	Amount actually paid to the customer (in INR).
`CITY`	City where the customer resides (limited to 4 major Indian cities).
`REJECTION_REMARKS`	Short reason for claim rejection if `PAID_AMOUNT` is zero, else left blank.

1. You are provided with a dataset in CSV format, where the columns represent various data attributes. Your task is to write clean, reusable, and well-structured Python code to automate pre-process of the data. The pre-processing tasks involve cleaning, handling missing, invalid, null values and data types.
   Key Requirements:
   - Do not use any external libraries like pandas, numpy, or csv. You are only allowed to use basic Python for this task.
   - Your function will take csv file as input parameter and return cleaned data.

2. XYZ TECH is considering shutting down operations in one of the four cities — Pune, Kolkata, Ranchi, or Guwahati — to optimize costs. As a data scientist, analyze the April 2025 claims data and recommend which city should be considered for closure based on data-driven insights.

3. You are provided with a Python function complex_rejection_classifier that is designed to classify rejection remarks into predefined. The function is intended to be applied to the REJECTION_REMARKS column of a DataFrame, and generate a new column REJECTION_CLASS based on the classification results.
   However, the function contains several errors that need to be identified and corrected. Your task is to debug and fix the errors in the given function, and then apply the corrected function to the REJECTION_REMARKS column of the DataFrame. The final output should be a new column REJECTION_CLASS with the corrected rejection classifications.
   (
   df['REJECTION_CLASS'] = df['REJECTION_REMARKS'].apply(lambda remark: complex_rejection_classifier(remark) if pd.notna(remark) else 'No Remark'
)

~~Note:~~

~~Please provide solution in Notebook PDF format only.~~