Project: Creating an AI-driven tool for analyzing single-case design data in behavior analytic research. The goal is to develop a system that can help clinicians select and perform the appropriate statistical tests for different design types, leveraging AI to guide this process.

1. Exploring the contents of the provided Excel file to identify the design types and the relevant data:

The Excel file contains four sheets, each representing a different single-case design type:

1. **ABA Reversal**
2. **Multiple Baseline**
3. **Alternating Treatment**
4. **Changing Criterion**

The datasets are organized as follows:

1. **ABA Reversal**: Contains columns for Baseline, Intervention, and Return to Baseline phases.
2. **Multiple Baseline**: Contains data for multiple participants (P1, P2, P3), each with Baseline and Intervention phases.
3. **Alternating Treatment**: Data is presented for two conditions, where each session alternates between conditions.
4. **Changing Criterion**: Contains a Baseline followed by multiple Criterion phases.

With this structure in mind, the tool can be designed to accept these types of inputs and guide the selection of appropriate statistical tests based on the design type.

For each design type, specific statistical tests can be recommended. For example:

- **ABA Reversal** might use tests that compare across phases, such as paired t-tests or non-parametric alternatives.
- **Multiple Baseline** might benefit from mixed-effects models or randomization tests.
- **Alternating Treatment** could use tests that compare across conditions, possibly using ANOVA or its non-parametric equivalent.
- **Changing Criterion** might require trend analysis or other time-series methods.

2. Description of the Simple Implementation of the First Step:

The initial implementation involves creating a basic framework for the tool that includes:

1. **Selection of Statistical Tests:** The code provides a mechanism to select appropriate statistical tests based on the design type input by the user. For instance, it identifies whether to use paired t-tests for ABA Reversal or logistic regression for Multiple Baseline.
2. **Data Processing and Analysis:** The tool processes the input data according to the design type and performs the selected statistical analysis. For example, it executes paired t-tests or Wilcoxon signed-rank tests for ABA Reversal data and logistic regression for Multiple Baseline data.
3. **Output Generation:** The implementation generates and presents the results of the analyses, such as test statistics and p-values, helping users interpret their data.

The implementation is designed to:

1. Load single-case design data from an Excel file.
2. Select appropriate statistical tests based on the design type.
3. Perform statistical analyses (e.g., paired t-tests, Wilcoxon signed-rank tests, logistic regression).
4. Output the results of these analyses.

Purpose of the Code: The code is designed to help analyze single-case design data in behavior analytic research by selecting appropriate statistical tests and performing analyses based on the design type provided by the user. It aims to provide initial results and guide the selection of statistical methods.

Code Breakdown:

1. **Data Loading:**
    - The code loads data from an Excel file containing sheets for different single-case design types.
    - It reads the `ABA Reversal` sheet into a DataFrame called `aba_reversal`.
    - It also reads the `Multiple Baseline` sheet into a DataFrame called `multiple_baseline_data`.
2. **Function Definitions:**
    - **`select_statistical_test(inputs):`**
        - This function takes user input specifying the design type and returns the name of the statistical test or method that is appropriate for that design type.
        - Example: For 'ABA Reversal', it suggests "Paired T-Test or Wilcoxon Signed-Rank Test".
    - **`input_processor(inputs):`**
        - This function is a placeholder that currently just returns the input data as-is. It is meant to process user inputs for further use in the analysis.
    - **`run_analysis(test_name, data):`**
        - This function performs statistical analysis based on the selected test name and data provided.

        - For `ABA Reversal`, it executes paired t-tests and Wilcoxon signed-rank tests to compare Baseline and Intervention phases.
        - For `Multiple Baseline`, it performs logistic regression analysis using data from multiple participants.
3. **Class Definition:**
    - **`CustomChain`:**
        - This class is designed to handle the process of selecting a statistical test and running the corresponding analysis.
        - It takes an input processor and a decision function, processes the inputs, and runs the chosen statistical test.
4. **Implementation and Results:**
    - **ABA Reversal Example:**
        - **Input Data:** The data includes columns for Baseline and Intervention phases.
        - **Analysis Performed:** Paired t-tests and Wilcoxon signed-rank tests are conducted to compare these phases.
        - **Results:**
            - Paired T-Test:
                - t-statistic: 1.86
                - p-value: 0.137
            - Wilcoxon Signed-Rank Test:
                - w-statistic: 2.00
                - p-value: 0.188
    - **Logistic Regression Example:**
        - **Input Data:** Data from the `Multiple Baseline` sheet, used to create a binary target variable for classification.
        - **Analysis Performed:** Logistic regression analysis to classify data based on baseline observations.
        - **Results:**
            - Predictions: Array of predicted classes (e.g., [1, 0, 0, 0])
            - Accuracy: 100% (1.0)

```
import pandas as pd

from scipy.stats import ttest_rel, wilcoxon

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score


# Load the Excel file containing hypothetical datasets

file_path = './SCD Hypothetical Datasets.xlsx'

excel_data = pd.ExcelFile(file_path)
```

```
# Load the ABA Reversal data from the relevant sheet
aba_reversal = excel_data.parse('ABA Reversal')

# Load the data for Logistic Regression from 'Multiple Baseline' sheet
multiple_baseline_data = excel_data.parse('Multiple Baseline')

# Function to select the appropriate statistical test based on design type
def select_statistical_test(inputs):
    design_type = inputs.get('design_type')

    if design_type == 'ABA Reversal':
        return "Paired T-Test or Wilcoxon Signed-Rank Test"
    elif design_type == 'Multiple Baseline':
        return "Logistic Regression"
    elif design_type == 'Alternating Treatment':
        return "ANOVA or Equivalent"
    elif design_type == 'Changing Criterion':
        return "Trend Analysis"
    else:
        return None

# Function to process the input data
def input_processor(inputs):
    return inputs

# Class definition for the CustomChain
class CustomChain:
    def __init__(self, input_processor, decision_function):
        self.input_processor = input_processor
```

```
        self.decision_function = decision_function

    def run(self, inputs):
        processed_input = self.input_processor(inputs)
        result = self.decision_function(processed_input)
        return result

# Function to run the selected analysis
def run_analysis(test_name, data):
    if test_name == "Paired T-Test or Wilcoxon Signed-Rank Test":
        baseline = data['Baseline']
        intervention = data['Intervention']

        # Align the data to ensure equal length
        min_length = min(len(baseline), len(intervention))
        aligned_baseline = baseline[:min_length]
        aligned_intervention = intervention[:min_length]

        # Perform a paired t-test
        t_stat, p_value_t = ttest_rel(aligned_baseline, aligned_intervention,
nan_policy='omit')

        # Perform a Wilcoxon signed-rank test
        w_stat, p_value_w = wilcoxon(aligned_baseline, aligned_intervention,
zero_method='wilcox', correction=True)

        return {
            't_stat': t_stat,
            'p_value_t': p_value_t,
            'w_stat': w_stat,
```

```
            'p_value_w': p_value_w
        }
    elif test_name == "Logistic Regression":
        # Assuming we are using 'Multiple Baseline' data for logistic
regression
        X = data[['P1 - Baseline', 'P2 - Baseline', 'P3 -
Baseline']].fillna(0)
        y = (X.mean(axis=1) > X.mean(axis=1).median()).astype(int)  #
Creating a binary target variable for demo purposes

        X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
        model = LogisticRegression()
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)
        accuracy = accuracy_score(y_test, predictions)
        return {'predictions': predictions, 'accuracy': accuracy}
    else:
        print(f"Test {test_name} not recognized.")
        return None

# Example of using the CustomChain with ABA Reversal design
aba_data = {
    'Baseline': aba_reversal['Baseline'].dropna(),
    'Intervention': aba_reversal['Intervention'].dropna()
}

user_input_aba = {
    'design_type': 'ABA Reversal',
    'number_of_phases': 3,
    'number_of_observations': len(aba_data['Baseline']),
```

```
    'desired_comparison': 'Compare Baseline and Intervention phases'
}

chain_aba = CustomChain(input_processor=input_processor,
decision_function=select_statistical_test)
result_aba = chain_aba.run(user_input_aba)
analysis_results_aba = run_analysis(result_aba, aba_data)

# Display the analysis results for ABA Reversal
print("ABA Reversal Analysis Results:")
print(analysis_results_aba)

# Prepare inputs for Logistic Regression using the 'Multiple Baseline' sheet
user_input_logistic = {
    'design_type': 'Multiple Baseline',
    'number_of_phases': 2,
    'number_of_observations': len(multiple_baseline_data),
    'desired_comparison': 'Classify based on baseline data'
}

# Instantiate a new CustomChain class for Logistic Regression
chain_logistic = CustomChain(input_processor=input_processor,
decision_function=select_statistical_test)

# Run the chain for Logistic Regression
result_logistic = chain_logistic.run(user_input_logistic)

# Run the analysis for Logistic Regression
analysis_results_logistic = run_analysis(result_logistic,
multiple_baseline_data)
```

```
# Display the analysis results for Logistic Regression

print("Logistic Regression Analysis Results:")

print(analysis_results_logistic)
```

```
ABA Reversal Analysis Results:
{'t_stat': 1.856558243265828, 'p_value_t': 0.13694512334578732, 'w_stat':
2.0, 'p_value_w': 0.1875}
Logistic Regression Analysis Results:
{'predictions': array([1, 0, 0, 0]), 'accuracy': 1.0}
```

3. Next Steps for Further Development:

1. **Expand Statistical Methods:** Include additional statistical tests and methods relevant to all design types, such as ANOVA for Alternating Treatment and trend analysis for Changing Criterion.
2. **Develop User Interface:** Create a user-friendly interface for inputting data and design details, ensuring accessibility and ease of use.
3. **Integrate Existing Tools:** Explore integration with existing statistical tools or platforms (e.g., R, Shiny) to extend the tool's capabilities.
4. **Implement Data Validation:** Incorporate robust data validation and preprocessing steps to ensure the input data is suitable for analysis.
5. **Enhance Outputs and Visualization:** Add features for detailed and interpretable outputs, including visualizations and comprehensive reports.
6. **Conduct User Testing:** Gather feedback from behavior analysts to refine the tool's functionality and ensure it meets user needs.