1. Team Member's Details:

- **Group Name:** Data Innovators

- **Name 1:** Stephanie Dawsonn-Andoh
  - **Email:** stephandoh@gmail.com
  - **Country:** Ghana
  - **Specialization:** Data Analytics

- **Name 2:** Sreedhar Rongala
  - **Email:** rongalasreedhar@gmail.com
  - **Country:** India
  - **Specialization:** Data Analytics

- **Name 3:** Brittney Smith
  - **Email:** brittneysintership@gmail.com
  - **Country:** USA
  - **Specialization:** Data Analytics

---

2. Problem Description:

XYZ Bank is aiming to enhance its marketing campaign by delivering personalized Christmas offers to its customers. Instead of rolling out a generic offer for everyone, they want to target specific customer segments with relevant offers. To solve this problem efficiently, XYZ Bank approached ABC Analytics Company to help with customer segmentation. The bank's requirement is to group customers into no more than 5 segments, as more segments would be inefficient for their marketing efforts.

---

3. Data Understanding:

Dataset Context

1. **Customer Demographics and Attributes**
   - The dataset includes a unique identifier for each customer (ncodpers), their country of residence (pais_residencia), gender (sexo), and age. This information helps understand the geographic distribution and demographic characteristics of the customer base.
2. **Customer Relationship and Status**
   - It captures the employment status of customers (ind_empleado), which provides insight into their financial stability. The dataset also includes an indicator for new customers (ind_nuevo), the length of their relationship with the bank (antiguedad), and the type of relationship they have with the bank (indrel). Additionally, it tracks the customer's relationship status at the beginning of the month (indrel_1mes) and the nature of that relationship (tiprel_1mes).

3. **Financial Products and Services**
    ○ This section outlines various financial indicators, such as whether the customer is active (ind_actividad_cliente), their gross income (renta), and whether they hold specific products like savings accounts, current accounts, mortgages, and credit cards. Each product has a corresponding indicator that shows if the customer has that type of account or service.
4. **Temporal Aspects**
    ○ The dataset records key dates, including the date the customer became the primary holder of a contract (fecha_alta) and the last date they were considered a primary customer (ult_fec_cli_1t). These temporal aspects are essential for understanding customer engagement over time.

---

4. Data Types in the dataset:

Categorical Data:

- The dataset contains various categorical variables that help classify customers and their attributes. Examples include:
    - Customer Demographics: pais_residencia (Country of residence), sexo (Gender), ind_empleado (Employment status), and canal_entrada (Entry channel).
    - Relationship Status: indrel (Customer relationship type), indrel_1mes (Relationship type at the beginning of the month), and tiprel_1mes (Type of relationship).

Numerical Data:

- Numerical variables are essential for quantitative analysis and can be further categorized into discrete and continuous data:
    - Discrete Data:
        - age (Customer's age) and ind_nuevo (New customer index) are examples of discrete numerical data.
    - Continuous Data:
        - renta (Gross income) and antiguedad (Seniority in months) are examples of continuous data that can provide insights into customer financial capacities and durations of relationships with the bank.

Date/Time Data:

- The dataset includes date-related variables that track customer interactions with the bank:
    - fecha_alta (Date of becoming the primary holder of a contract) and ult_fec_cli_1t (Last date as a primary customer) are crucial for analyzing customer engagement over time.

---

5. Challenges and Solutions:

Language Barrier:

- **Problem:** The dataset contained column names in Spanish, creating a barrier to understanding.
- **Solution:** Translated all column names from Spanish to English to facilitate better comprehension of the data.

High Percentage of Missing Data:

- **Problem:** The columns ult_fec_cli_1t (latest date as primary customer) and conyuemp (spouse employee indicator) had more than 90% missing data, rendering them ineffective for analysis.
- **Solution:** Removed these columns from the dataset.

Irrelevant Column:

- **Problem:** The column Unnamed: 0 did not provide useful information.
- **Solution:** Dropped this column to streamline the dataset.

Redundant Geographic Information:

- **Problem:** The columns cod_prov (province code) and nomprov (province name) provided redundant information since country data was more pertinent.
- **Solution:** Removed these columns to focus on more relevant geographic data.

Recency Overlap:

- **Problem:** The fecha_alta (signup date) column was not necessary for analyzing customer recency, as antiguedad (seniority) could serve that purpose.
- **Solution:** Dropped the fecha_alta column.

Missing Values in Categorical Columns:

- **Problem:** Categorical columns such as sexo (gender) and pais_residencia (country of residence) contained missing values.
- **Solution:** Replaced missing values with the most frequent values (mode) in those columns.

Missing Values in Numerical Columns:

- **Problem:** The columns renta (income), ind_nomina_ult1 (payroll product indicator), and ind_nom_pens_ult1 (pension nomination product indicator) had missing values.
- **Solution:** Replaced missing values with median values for these numerical columns to minimize the impact of outliers.

Data Type Adjustments:

- **Problem:** Several columns had inappropriate data types, such as float where integer was more suitable.

- **Solution:** Changed the data types from float to int for several columns, including ind_empleado (employee indicator) and indresi (resident indicator), among others.

Date Format Correction:

- **Problem:** The fecha_dato (data date) column was in an object format instead of a date format.
- **Solution:** Converted fecha_dato to a date data type for accurate temporal analysis.

Outliers in Age Column:

- **Problem:** The age column had a maximum age of 116 and several outliers above the upper bound of 92.
- **Solution:** Capped all outliers in the age column at the upper bound of 92 to ensure realistic values.

Negative Values in Seniority:

- **Problem:** The antiguedad (seniority) column had a minimum value of -999999, which was nonsensical.
- **Solution:** Capped outliers in this column to a minimum sensible value of 0 after investigation.

The screenshots below show the implementation of the problems and solutions

```python
# Getting a list of seniority values below 33 and sorting them in ascending order
seniority_below_33_list = df_renamed[df_renamed['seniority'] < 33]['seniority'].tolist()
seniority_below_33_list_sorted = sorted(seniority_below_33_list)

# Displaying the sorted list
print(seniority_below_33_list_sorted)
```

[156] ✓ 0.0s

```
[-999999, -999999, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
```

```python
# Replacing -999999 with 0 in the 'seniority' column
df_renamed['seniority'] = df_renamed['seniority'].replace(-999999, 0)

# Verifying if the replacement was successful
print(df_renamed['seniority'].value_counts())
```

[157] ✓ 0.0s

```
seniority
21    19801
12    18894
10    17046
33    15886
45    14611
      ...
4        38
0        37
2        31
1        26
3        22
Name: count, Length: 247, dtype: int64
```

```python
# Identify outliers in the age column
outliers = df_renamed[(df_renamed['age'] < lower_bound) | (df_renamed['age'] > upper_bound)]

# Display the outliers
print(outliers[['customer_id', 'age']])  # Include other relevant columns if necessary
```

✓ 0.0s

```
        customer_id  age
1917        1049693   95
1924        1049682   96
3680        1053264   96
4370        1054486   95
5063        1044349   92
...             ...  ...
733834       875227  103
802843       132059   91
855489        73077   93
857304        73564  104
901269       389204   91

[4084 rows x 2 columns]
```

```python
# Cap ages above 92 to 92
df_renamed.loc[df_renamed['age'] > 92, 'age'] = 92
```

✓ 0.0s

Observation

- 2 and 116 in the age column seems weirds, check for outliers
- -9999999 in seniority seems weirds, investigate

```python
# Get unique ages below 27
unique_ages_below_27 = df_renamed[df_renamed['age'] < 27]['age'].unique()

# Convert to a list if needed
unique_ages_below_27_list = unique_ages_below_27.tolist()

# Display the unique ages
print(unique_ages_below_27_list)
```

✓ 0.0s

```
[23, 22, 24, 25, 26, 15, 12, 8, 6, 10, 9, 16, 11, 17, 14, 19, 13, 20, 7, 21, 18, 4, 5, 3, 2]
```

```python
# Get unique ages below 27
unique_ages_above_90 = df_renamed[df_renamed['age'] > 90]['age'].unique()

# Convert to a list if needed
unique_ages_above_90_list = unique_ages_above_90.tolist()

# Display the unique ages
print(unique_ages_above_90_list)
```

✓ 0.0s

```
[95, 96, 92, 93, 91, 94, 99, 98, 97, 100, 101, 106, 103, 102, 104, 111, 107, 109, 105, 112, 115, 110, 116, 108, 113]
```

```python
# Get a DataFrame with duplicate customer IDs
duplicate_customers = df_renamed[df_renamed.duplicated(subset='customer_id', keep=False)]

# Group by customer_id and aggregate the signup_dates
signup_date_check = duplicate_customers.groupby('customer_id')['signup_date'].agg(
    unique_dates='nunique',  # Count unique signup_dates
    all_dates=lambda x: list(x.unique())  # List all unique signup_dates
).reset_index()

# Filter for customer_ids with more than one unique signup_date
inconsistent_dates = signup_date_check[signup_date_check['unique_dates'] > 1]

# Display the results
if inconsistent_dates.empty:
    print("All duplicate instances have the same signup_date.")
else:
    print("Inconsistent signup_dates found for the following customer_ids:")
    print(inconsistent_dates[['customer_id', 'all_dates']])
```

✓ 14.2s

```
All duplicate instances have the same signup_date.
```

```python
# Create a new DataFrame with unique customer IDs
df_renamed = df_renamed.drop_duplicates(subset='customer_id')

# Now you can get the distinct count again
distinct_customer_count = df_renamed['customer_id'].nunique()
print(f'Distinct Customer Count after removing duplicates: {distinct_customer_count}')
```

✓ 0.1s

```python
df_renamed['new_customer_indicator'] = df_renamed['new_customer_indicator'].astype(int)
df_renamed['primary_relationship_status'] = df_renamed['primary_relationship_status'].astype(int)
df_renamed['customer_type_at_month_start'] = df_renamed['customer_type_at_month_start'].astype(int)
df_renamed['address_type'] = df_renamed['address_type'].astype(int)
df_renamed['customer_activity_index'] = df_renamed['customer_activity_index'].astype(int)
df_renamed['pension_nomination_product_indicator'] = df_renamed['pension_nomination_product_indicator'].astype(int)
df_renamed['direct_debit_product_indicator'] = df_renamed['direct_debit_product_indicator'].astype(int)
df_renamed['payroll_product_indicator'] = df_renamed['payroll_product_indicator'].astype(int)
```

✓ 0.0s

```python
df_renamed['new_customer_indicator'] = pd.to_numeric(df_renamed['new_customer_indicator'], errors='coerce')
```

✓ 0.0s

```python
# Convert numeric columns stored as objects to integers or floats
df_renamed['age'] = pd.to_numeric(df_renamed['age'], errors='coerce')
df_renamed['seniority'] = pd.to_numeric(df_renamed['seniority'], errors='coerce')
df_renamed['customer_activity_index'] = pd.to_numeric(df_renamed['customer_activity_index'], errors='coerce')
```

✓ 0.6s

```python
# Convert categorical columns to appropriate numeric types
df_renamed['new_customer_indicator'] = pd.to_numeric(df_renamed['new_customer_indicator'], errors='coerce')
df_renamed['customer_type_at_month_start'] = pd.to_numeric(df_renamed['customer_type_at_month_start'], errors='coerce')
```

✓ 0.0s

```python
df_renamed['data_date'] = pd.to_datetime(df_renamed['data_date'], errors='coerce')
```

```python
# Imputing with most frequent value
df_renamed['employee_indicator'].fillna(df_renamed['employee_indicator'].mode()[0], inplace=True)
df_renamed['customer_relation_type'].fillna(df_renamed['customer_relation_type'].mode()[0], inplace=True)
df_renamed['resident_indicator'].fillna(df_renamed['resident_indicator'].mode()[0], inplace=True)
df_renamed['foreigner_indicator'].fillna(df_renamed['foreigner_indicator'].mode()[0], inplace=True)
df_renamed['entry_channel'].fillna(df_renamed['entry_channel'].mode()[0], inplace=True)
df_renamed['deceased_indicator'].fillna(df_renamed['deceased_indicator'].mode()[0], inplace=True)
```

```python
# Imputing numeric values with median
df_renamed['income'].fillna(df_renamed['income'].median(), inplace=True)
df_renamed['payroll_product_indicator'].fillna(0, inplace=True)
df_renamed['pension_nomination_product_indicator'].fillna(0, inplace=True)
```

✓ 0.0s

```python
# Imputing with most frequent value
df_renamed['gender'].fillna(df_renamed['gender'].mode()[0], inplace=True)
df_renamed['country_of_residence'].fillna(df_renamed['country_of_residence'].mode()[0], inplace=True)
```

✓ [illegible]

```python
# Dropping unnecessary columns
df_renamed.drop(['province_code', 'province_name'], axis=1, inplace=True)
```

✓ 0.1s

```python
df_renamed.isnull().sum()
```

✓ [illegible]

```python
    "ind_nomina_ult1": "payroll_product_indicator",
    "ind_nom_pens_ult1": "pension_nomination_product_indicator",
    "ind_recibo_ult1": "direct_debit_product_indicator"
}

# Applying the renaming to the dataframe
df_renamed = df.rename(columns=column_mapping)

# Preview the renamed dataframe
df_renamed.head()
```

[120] ✓ 0.1s

	unnamed_index	data_date	customer_id	employee_indicator	country_of_residence	gender	age	signup_date	new_customer_indicator	seniority	primary_relationship_status	last_date_primary_cust
0	0	1/28/2015	1375586	N	ES	H	35	1/12/2015	0.00	6	1.00	
1	1	1/28/2015	1050611	N	ES	V	23	8/10/2012	0.00	35	1.00	
2	2	1/28/2015	1050612	N	ES	V	23	8/10/2012	0.00	35	1.00	
3	3	1/28/2015	1050613	N	ES	H	22	8/10/2012	0.00	35	1.00	
4	4	1/28/2015	1050614	N	ES	V	23	8/10/2012	0.00	35	1.00	

```python
df_renamed.shape
```

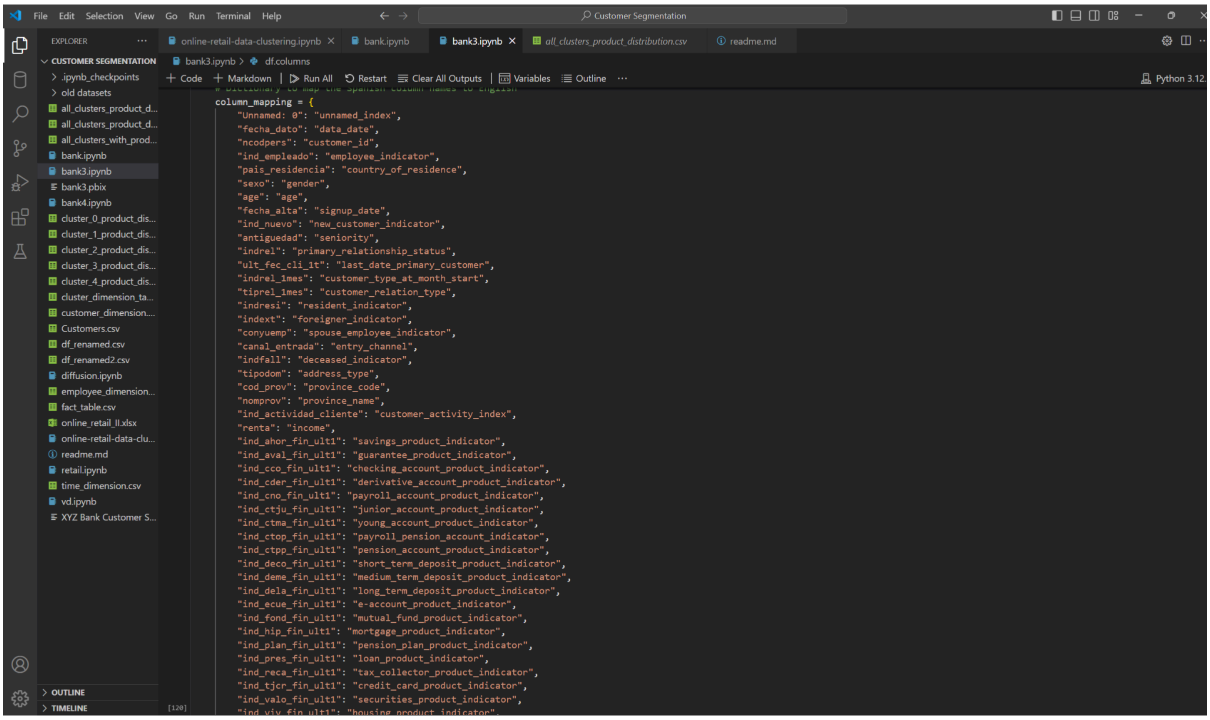
[121] ✓ 0.0s

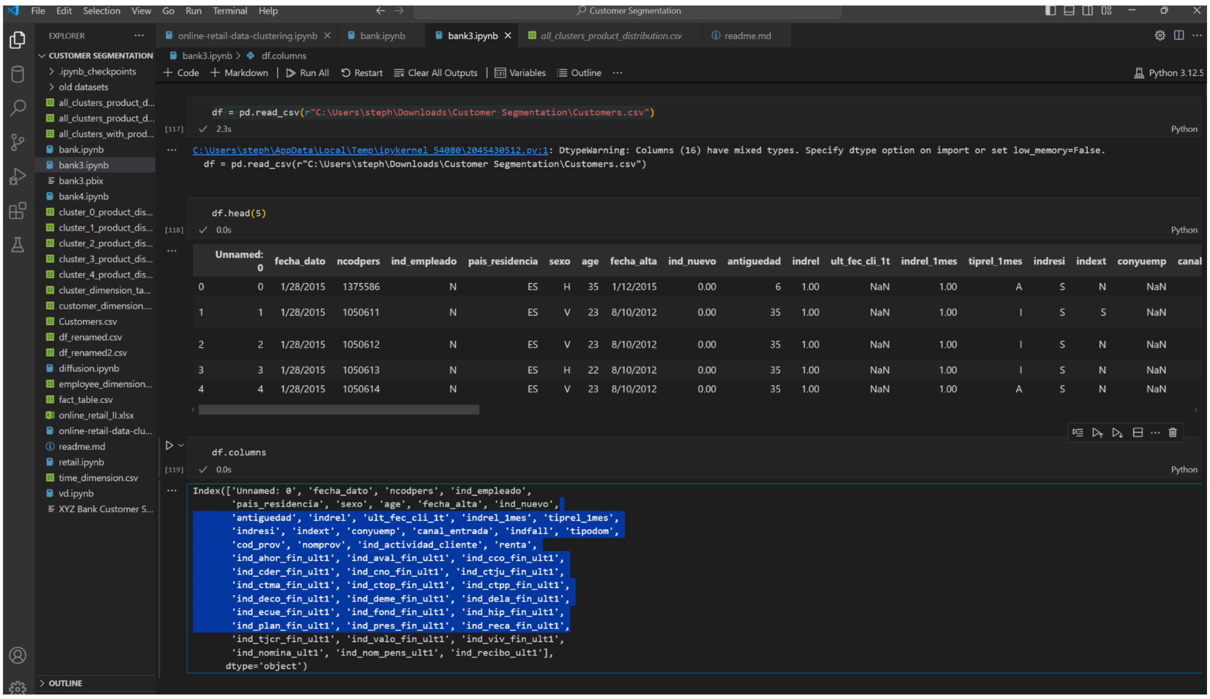
```
(1000000, 48)
```

```python
df_renamed.isnull().sum()
```

[122] ✓ 0.4s

```
unnamed_index                          0
data_date                              0
customer_id                            0
employee_indicator                 10782
country_of_residence               10782
gender                             10786
age                                    0
signup_date                        10782
new_customer_indicator             10782
seniority                              0
primary_relationship_status        10782
last_date_primary_customer        998899
```





Data storage location:

https://github.com/stephandoh/Data-Glacier-Internship/tree/main/Week_9