Importing Required Libraries

```
import pandas as pd
import numpy as np
from datetime import datetime
import random
```

SettingUp Constants

```
alpha = 0.4
beta = 0.6
purchasing_power = 100.0
k = 7   # size of bundles
C_max = 100.0
theta = 1
gamma = 0.25
delta = 0.75
n_categories = 7  # max number of categories allowed per bundle
num_bundles = 4  # number of bundles
```

```
df = pd.read_csv('/content/sample_data_1000_C.csv')
df.head()
```

	CODEBELISTA	CODPRODUCTOSAP	DESCATEGORIA	DESMARCA	PRECIOOFERTA	FECHAPROCESO
0	44109905	200112294	MAQUILLAJE	CYZONE	9.50	2023-02-09
1	36949902	200086399	FRAGANCIAS	LBEL	19.95	2023-02-02
2	49968221	200089498	TRATAMIENTO CORPORAL	ESIKA	36.90	2023-02-10
3	46114531	210102388	BIJOUTERIE	ESIKA	59.90	2023-02-21
4	50368645	210100620	COMPLEMENTOS	CYZONE	74.90	2023-03-01

Data Pre-processing

```
#Renaming the columns
column_rename_map={
                "CODEBELISTA": "consultant_id",
                "CODPRODUCTOSAP": "product_id",
                "DESCATEGORIA": "category",
                "DESMARCA": "brand",
                "PRECIOOFERTA": "price",
                "FECHAPROCESO": "date"
            }
df.rename(columns=column_rename_map, inplace=True)
df.head()
```

	consultant_id	product_id	category	brand	price	date
0	44109905	200112294	MAQUILLAJE	CYZONE	9.50	2023-02-09
1	36949902	200086399	FRAGANCIAS	LBEL	19.95	2023-02-02
2	49968221	200089498	TRATAMIENTO CORPORAL	ESIKA	36.90	2023-02-10
3	46114531	210102388	BIJOUTERIE	ESIKA	59.90	2023-02-21
4	50368645	210100620	COMPLEMENTOS	CYZONE	74.90	2023-03-01

```
#Filling Null Values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36310 entries, 0 to 36309
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   consultant_id  36310 non-null  int64
```

```
 1   product_id     36310 non-null  int64
 2   category       36228 non-null  object
 3   brand          36310 non-null  object
 4   price          36310 non-null  float64
 5   date           36310 non-null  object
dtypes: float64(1), int64(2), object(3)
memory usage: 1.7+ MB
```

```
df['category'].fillna('OTHERS', inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36310 entries, 0 to 36309
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   consultant_id  36310 non-null  int64
 1   product_id     36310 non-null  int64
 2   category       36310 non-null  object
 3   brand          36310 non-null  object
 4   price          36310 non-null  float64
 5   date           36310 non-null  object
dtypes: float64(1), int64(2), object(3)
memory usage: 1.7+ MB
<ipython-input-29-d823663f0706>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['category'].fillna('OTHERS', inplace=True)
```

```
# Convert `date` column to datetime
df["date"] = pd.to_datetime(df["date"], errors="coerce")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36310 entries, 0 to 36309
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   consultant_id  36310 non-null  int64
 1   product_id     36310 non-null  int64
 2   category       36310 non-null  object
 3   brand          36310 non-null  object
 4   price          36310 non-null  float64
 5   date           36310 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(2), object(2)
memory usage: 1.7+ MB
```

```
# Perform daily aggregation
group_by_columns = ["consultant_id", "date"]
daily_agg = df.groupby(group_by_columns, as_index=False).agg({
    "product_id": "count",
    "price": "sum"
})

# Rename columns for clarity
daily_agg.rename(columns={
    "product_id": "frequency",
    "price": "daily_total_spent"
}, inplace=True)

daily_agg.head()
```

	consultant_id	date	frequency	daily_total_spent
0	3441296	2022-12-27	10	263.40
1	3441296	2023-02-07	8	273.30
2	3441296	2023-02-15	1	23.90
3	3441296	2023-02-28	7	261.45
4	3441296	2023-04-11	17	391.10

```
# Calculate IQR for daily_total_spent
Q1 = np.percentile(daily_agg["daily_total_spent"], 25)
Q3 = np.percentile(daily_agg["daily_total_spent"], 75)
IQR = Q3 - Q1

# Add average purchasing power (Q3) as a new column
daily_agg["average_purchasing_power"] = Q3



# Merge the daily aggregates back into the original DataFrame
df = pd.merge(df, daily_agg, on=group_by_columns, how="left")
df.head()
```

	consultant_id	product_id	category	brand	price	date	frequency	daily_total_spent	average_purchasing_power
0	44109905	200112294	MAQUILLAJE	CYZONE	9.50	2023-02-09	17	427.40	430.195
1	36949902	200086399	FRAGANCIAS	LBEL	19.95	2023-02-02	16	403.90	430.195
2	49968221	200089498	TRATAMIENTO CORPORAL	ESIKA	36.90	2023-02-10	21	406.30	430.195
3	46114591	210100000	BLOUTERIE	ESIKA	59.00	2023-02-	11	460.10	430.195

```
# Calculate recency in days
current_date = datetime.now()
df["recency"] = (current_date - df["date"]).dt.days
df.head()
```

	consultant_id	product_id	category	brand	price	date	frequency	daily_total_spent	average_purchasing_power	recency
0	44109905	200112294	MAQUILLAJE	CYZONE	9.50	2023-02-09	17	427.40	430.195	688
1	36949902	200086399	FRAGANCIAS	LBEL	19.95	2023-02-02	16	403.90	430.195	695
2	49968221	200089498	TRATAMIENTO CORPORAL	ESIKA	36.90	2023-02-10	21	406.30	430.195	687
3	46114591	210100000	BLOUTERIE	ESIKA	59.00	2023-	11	460.10	430.195	676

```
# Normalize frequency
freq_min = df["frequency"].min()
freq_max = df["frequency"].max()

if freq_min != freq_max:
    df["frequency_normalized"] = (df["frequency"] - freq_min) / (freq_max - freq_min)
else:
    df["frequency_normalized"] = 0.5



# Normalize recency
rec_min = df["recency"].min()
rec_max = df["recency"].max()

if rec_min != rec_max:
    df["recency_normalized"] = (df["recency"] - rec_min) / (rec_max - rec_min)
else:
    df["recency_normalized"] = 0.5


# Consultant-level metrics
df["total_spent"] = df.groupby("consultant_id")["price"].transform("sum")
df["purchase_frequency"] = df.groupby("consultant_id")["product_id"].transform("count")
df["unique_products"] = df.groupby("consultant_id")["product_id"].transform("nunique")


df.head()
```

	consultant_id	product_id	category	brand	price	date	frequency	daily_total_spent	average_purchasing_power	recency	fre
0	44109905	200112294	MAQUILLAJE	CYZONE	9.50	2023-02-09	17	427.40	430.195	688	
1	36949902	200086399	FRAGANCIAS	LBEL	19.95	2023-02-02	16	403.90	430.195	695	
2	49968221	200089498	TRATAMIENTO CORPORAL	ESIKA	36.90	2023-02-10	21	406.30	430.195	687	
3	46114531	210102388	BIJOUTERIE	ESIKA	59.90	2023-02-21	11	469.10	430.195	676	
4	50368645	210100620	COMPLEMENTOS	CYZONE	74.90	2023-03-01	5	202.55	430.195	668	

```
df.to_csv("preprocessed_data_1000_C.csv", index=False)
```

Combined Score for each row based on normalized frequency and recency

```
df = pd.read_csv('/content/preprocessed_data_1000_C.csv')

def normalize(value, min_value, max_value):
    if min_value == max_value:
        return 0
    return (value - min_value) / (max_value - min_value)

min_f = df['purchase_frequency'].min()
max_f = df['purchase_frequency'].max()
min_r = df['recency'].min()
max_r = df['recency'].max()

df['normalized_f'] = df['purchase_frequency'].apply(lambda x: normalize(x, min_f, max_f))
df['normalized_r'] = df['recency'].apply(lambda x: normalize(x, min_r, max_r))
df['combined_score'] = (alpha * df['normalized_f']) + (beta * df['normalized_r'])
df.head()
```

	consultant_id	product_id	category	brand	price	date	frequency	daily_total_spent	average_purchasing_power	recency	fre
0	44109905	200112294	MAQUILLAJE	CYZONE	9.50	2023-02-09	17	427.40	430.195	688	
1	36949902	200086399	FRAGANCIAS	LBEL	19.95	2023-02-02	16	403.90	430.195	695	
2	49968221	200089498	TRATAMIENTO CORPORAL	ESIKA	36.90	2023-02-10	21	406.30	430.195	687	
3	46114531	210102388	BIJOUTERIE	ESIKA	59.90	2023-02-21	11	469.10	430.195	676	
4	50368645	210100620	COMPLEMENTOS	CYZONE	74.90	2023-03-01	5	202.55	430.195	668	

```
def select_anchor_product(df):
        """
        Selects the product with the maximum 'combined_score'.
        """
        return df.loc[df['combined_score'].idxmax()]

def generate_candidates(bundle, df):
        """
        Returns products not already in the bundle.
        """
        bundle_product_ids = {p['product_id'] for p in bundle}
        return df[~df['product_id'].isin(bundle_product_ids)]

def category_score(product, bundle):
    """
    Returns 1 if product's category is in the current bundle, else 0.
    """
    return 1 if product['category'] in [b['category'] for b in bundle] else 0
```

```
def business_score(product):
    """
    Placeholder for any advanced logic; returns a random score for now.
    """
    return random.random()


def score_candidates(candidates, bundle):
        """
        Calculates the composite score for candidates.

        Parameters
        ----------
        candidates : pd.DataFrame
            The candidate products.
        bundle : list
            Current bundle (list of dicts or rows).

        Returns
        -------
        pd.DataFrame
            Updated candidates DataFrame with a 'score' column.
        """
        candidates = candidates.copy()
        candidates['category_score'] = candidates.apply(lambda p: category_score(p, bundle), axis=1)
        candidates['business_score'] = candidates.apply(business_score, axis=1)
        candidates['score'] = gamma * candidates['category_score'] + delta * candidates['business_score']
        return candidates

def build_bundle(df, anchor_product, purchasing_power, k, C_max, theta, n_categories):
        """
        Constructs the bundle starting with anchor_product.

        Parameters
        ----------
        df : pd.DataFrame
            DataFrame containing product information.
        anchor_product : pd.Series or dict
            The product chosen as the anchor.
        purchasing_power : float
            Unused in this example, but could factor into logic.
        k : int
            Max size of the bundle.
        C_max : float
            Maximum cost threshold for the bundle.
        theta : float
            Threshold multiplier for item price acceptance.
        n_categories : int
            Number of distinct categories allowed in the bundle.

        Returns
        -------
        list
            A list (bundle) containing the chosen products.
        """
        bundle = [anchor_product]
        current_total_cost = anchor_product['price']
        categories_in_bundle = {anchor_product['category']}

        # Exclude the anchor product from candidates
        candidates = df[df['product_id'] != anchor_product['product_id']]
        candidates = score_candidates(candidates, bundle)
        candidates = candidates[candidates['price'] <= theta * C_max].sort_values(by='score', ascending=False)

        for _, candidate in candidates.iterrows():
            if len(bundle) >= k:
                break
            if candidate['category'] in categories_in_bundle:
                # n_categories = 1 => do not add new categories
                if current_total_cost + candidate['price'] <= C_max:
                    bundle.append(candidate)
                    current_total_cost += candidate['price']
                    # We do not add a new category if n_categories=1
            else:
                # If your logic allows new category, incorporate that here
                pass
```

```
        return bundle


bundles = []
for consultant_id, group in df.groupby("consultant_id"):
            for bundle_index in range(num_bundles):
                anchor_product = select_anchor_product(group)
                bundle = build_bundle(
                    group,
                    anchor_product,
                    purchasing_power,  # from config
                    k,
                    C_max,
                    theta,
                    n_categories
                )
                unique_bundle_id = f"{consultant_id}_Bundle_{bundle_index+1}"
                for idx, product in enumerate(bundle):
                    product_copy = product.copy()
                    product_copy["consultant_id"] = consultant_id
                    product_copy["bundle_id"] = unique_bundle_id
                    product_copy["is_anchor"] = 1 if idx == 0 else 0
                    bundles.append(product_copy)


bundles
```

```
[consultant_id                       3441296
 product_id                         200095159
 category                    CUIDADO PERSONAL
 brand                                  ESIKA
 price                                   7.38
 date                              2022-12-27
 frequency                                 10
 daily_total_spent                      263.4
 average_purchasing_power             430.195
 recency                                  732
 frequency_normalized                0.157895
 recency_normalized                  0.979647
 total_spent                          7206.76
 purchase_frequency                       261
 unique_products                          194
 normalized_f                        0.343915
 normalized_r                        0.979647
 combined_score                      0.725354
 bundle_id                    3441296_Bundle_1
 is_anchor                                   1
 Name: 1793, dtype: object,
 consultant_id                       3441296
 product_id                         200108807
 category                    CUIDADO PERSONAL
 brand                                  ESIKA
 price                                   7.38
 date                              2022-12-27
 frequency                                 10
 daily_total_spent                      263.4
 average_purchasing_power             430.195
 recency                                  732
 frequency_normalized                0.157895
 recency_normalized                  0.979647
 total_spent                          7206.76
 purchase_frequency                       261
 unique_products                          194
 normalized_f                        0.343915
 normalized_r                        0.979647
 combined_score                      0.725354
 category_score                             1
 business_score                      0.921432
 score                               0.941074
 bundle_id                    3441296_Bundle_1
 is_anchor                                   0
 Name: 4744, dtype: object,
 consultant_id                       3441296
 product_id                         200095465
 category                    CUIDADO PERSONAL
 brand                                  ESIKA
 price                                    11.9
 date                              2023-06-08
 frequency                                 16
 daily_total_spent                      385.9
 average_purchasing_power             430.195
```

```
recency                                569
frequency_normalized            0.263158
recency_normalized               0.75848
```

```
data = pd.DataFrame(bundles)
data.head()
```

	consultant_id	product_id	category	brand	price	date	frequency	daily_total_spent	average_purchasing_power	recency	...	p
1793	3441296	200095159	CUIDADO PERSONAL	ESIKA	7.38	2022-12-27	10	263.40	430.195	732	...	
4744	3441296	200108807	CUIDADO PERSONAL	ESIKA	7.38	2022-12-27	10	263.40	430.195	732	...	
22714	3441296	200095465	CUIDADO PERSONAL	ESIKA	11.90	2023-06-08	16	385.90	430.195	569	...	
9107	3441296	200103025	CUIDADO PERSONAL	ESIKA	24.98	2023-08-15	11	156.66	430.195	501	...	
20479	3441296	200115451	CUIDADO PERSONAL	ESIKA	18.30	2024-03-20	7	183.50	430.195	283	...	

5 rows × 23 columns

```
data.to_csv('bundels_data_1000_C.csv', index=False)
```