```
import pandas as pd
from pandas import DataFrame
import gspread
from google.oauth2.service_account import Credentials
from google.oauth2 import service_account
from pandas_gbq import to_gbq
```

Important Functions

For production use only (example)

```
def get_google_sheet_as_df(sheet_id: str, sheet_name: str, creds_path: str) -> pd.DataFrame:
    """
    Retrieve data from a Google Sheet and return it as a Pandas DataFrame.

    :param sheet_id: The Google Sheet ID (found in the URL after /d/ and before /edit)
    :param sheet_name: The name of the sheet (tab) inside the document
    :param creds_path: Path to your service account credentials JSON file
    :return: Pandas DataFrame with the sheet data
    """
    scopes = ["https://www.googleapis.com/auth/spreadsheets.readonly"]
    credentials = Credentials.from_service_account_file(creds_path, scopes=scopes)
    client = gspread.authorize(credentials)

    sheet = client.open_by_key(sheet_id).worksheet(sheet_name)
    data = sheet.get_all_records()

    return pd.DataFrame(data)

def load_to_bigquery(df: DataFrame, table_name: str, credentials_path: str, if_exists: str = 'replace'):
    """
    Uploads a DataFrame to Google BigQuery using a service account credential file.

    Args:
        df (DataFrame): The pandas DataFrame to upload.
        table_name (str): Table name in the format 'project_id.dataset.table'.
        credentials_path (str): Path to the service account JSON credentials file.
        if_exists (str): What to do if the table exists: 'fail', 'replace', or 'append'. Default is 'replace'.

    Raises:
        ValueError: If the DataFrame is empty.
        Exception: If the upload fails.
    """
    if df.empty:
        raise ValueError("The DataFrame is empty. Nothing will be uploaded to BigQuery.")

    try:
        credentials = service_account.Credentials.from_service_account_file(credentials_path)
        to_gbq(df, table_name, credentials=credentials, if_exists=if_exists)
        print(f"✅ Data successfully uploaded to {table_name}.")
    except Exception as e:
        print(f"❌ Failed to upload data to BigQuery: {e}")
```

Extraction

📁 Data Setup Instructions

The script expects a `data/` folder located in the **root of the GitHub repository**:
🔗 https://github.com/JoaoPGOS/leanlayer-study-case.git

This folder should contain the following CSV files:

- `deals_meta.csv`
- `deals_snapshot.csv`
- `owners.csv`
- `targets.csv`

✅ Structure Example

```
leanlayer-study-case/
├── data/
```

```
│   ├── deals_meta.csv
│   ├── deals_snapshot.csv
│   ├── owners.csv
│   └── targets.csv
├── etl.py
```

```
# In a production environment, it should use the function get_google_sheet_as_df using a google service account to extract this data
deals_meta_df = pd.read_csv('data/deals_meta.csv')
deals_snapshot_df = pd.read_csv('data/deals_snapshot.csv')
owners_df = pd.read_csv('data/owners.csv')
targets_df = pd.read_csv('data/targets.csv')
```

```
print(f"Log type: info\n\nDeals Meta\n{deals_meta_df.head()}\nDeals Snapshot\n{deals_snapshot_df.head()}\nOwners\n{owners_df.head()}\nTa
```

```
Log type: info

Deals Meta
   deal_id owner_id created_date  close_date     deal_stage forecast_category  \
0  deal_31  owner_1   2024-12-21  2025-06-02    Closed Lost           Closed
1  deal_34  owner_1   2024-05-29  2024-11-11    Prospecting         Pipeline
2  deal_65  owner_1   2024-08-26  2025-02-04    Prospecting         Pipeline
3  deal_78  owner_1   2024-02-21  2024-03-03     Closed Won           Closed
4  deal_92  owner_1   2024-06-08  2024-11-24  Qualification         Pipeline

  record_type          deal_type deal_order_type account_industry  \
0     Channel  Existing Business          Upsell             Tech
1      Direct  Existing Business      Cross-Sell           Retail
2      Direct  Existing Business      Cross-Sell       Healthcare
3     Channel       New Business      New Client          Finance
4     Channel       New Business      New Client       Healthcare

  account_region account_size deal_source  deal_amount
0          LATAM          SMB    Outbound        22127
1           EMEA          SMB     Partner        23224
2          LATAM          SMB    Outbound        17680
3           EMEA          SMB    Outbound        53629
4          LATAM          SMB     Inbound        23492
Deals Snapshot
   deal_id snapshot_date        stage forecast_category  amount  close_date  \
0  deal_31    2024-12-23  Prospecting          Pipeline   25164  2025-07-02
1  deal_31    2024-12-30  Prospecting          Pipeline   17950  2025-04-02
2  deal_31    2025-01-06  Prospecting          Pipeline   24985  2025-05-02
3  deal_31    2025-01-13  Prospecting          Pipeline   18770  2025-07-02
4  deal_31    2025-01-20  Prospecting          Pipeline   25485  2025-08-02

  owner_id
0  owner_1
1  owner_1
2  owner_1
3  owner_1
4  owner_1
Owners
  owner_id              name                    email role_start_date  \
0  owner_1  Christopher Moore   danielledunn@gmail.com      2023-02-12
1  owner_2      Amber Jenkins        afoster@yahoo.com      2023-06-21
2  owner_3         Mark Lopez       halleric@yahoo.com      2023-09-22
3  owner_4       Kristen Mann          mhunt@james.org      2022-12-26
4  owner_5     Dawn Alexander  kelliparker@coleman.com      2023-07-27

  role_end_date              role         manager segment
0           NaN  Account Executive     Traci Price     SMB
1           NaN  Account Executive  Victoria Brown     SMB
2           NaN  Account Executive    Stacey Mason     SMB
3           NaN  Account Executive   Teresa Vaughn     SMB
4           NaN  Account Executive   Laura Sanchez     SMB
Targets
   quarter owner_id segment  target_amount
0  2024 Q1  owner_1     SMB          42000
1  2024 Q2  owner_1     SMB          48300
2  2024 Q3  owner_1     SMB          55545
3  2024 Q4  owner_1     SMB          63877
4  2025 Q1  owner_1     SMB          73458
```

Static analysis

Breakdown of Lost Deals in Q1 2025

This code processes deals that were **Closed Lost** in **Q1 2025** and analyzes their progression:

1. **Parse Close Dates**
   - Ensures `close_date` is in datetime format for filtering.

2. **Filter Lost Deals in Q1**
    - Defines the Q1 2025 range (`2025-01-01` to `2025-03-31`)
    - Filters deals where `deal_stage == "Closed Lost"` and `close_date` is within Q1
    - Extracts the unique `deal_id` s of these lost deals
3. **Filter Snapshots for Those Deals**
    - Filters the `deals_snapshot_df` to include only rows related to the lost deals
    - Converts `snapshot_date` to datetime and sorts snapshots chronologically by `deal_id` and `snapshot_date`
4. **Extract First and Last Snapshot Info**
    - Groups by `deal_id` and retrieves:
        - First and last **stage**
        - First and last **forecast category**
        - Final **owner ID**
        - Final **deal amount**
5. **Output**
    - Displays a DataFrame summarizing the lifecycle of each lost deal, useful for analyzing pipeline flow and potential misclassification.

```
#>>>>>>>>> Filter deals Closed Lost in Q1 <<<<<<<<<<
deals_meta_df['close_date'] = pd.to_datetime(deals_meta_df['close_date'])

# Define Q1 range (adjust year if needed)
start_q1 = "2025-01-01"
end_q1 = "2025-03-31"

lost_deals_q1 = deals_meta_df[
    (deals_meta_df['close_date'].between(start_q1, end_q1)) &
    (deals_meta_df['deal_stage'] == "Closed Lost")
]['deal_id'].unique()

print(f"Total lost deals in Q1: {len(lost_deals_q1)}")

#>>>>>>>>> Filter snapshots for those deals <<<<<<<<<<
lost_snapshots = deals_snapshot_df[deals_snapshot_df['deal_id'].isin(lost_deals_q1)].copy()

lost_snapshots['snapshot_date'] = pd.to_datetime(lost_snapshots['snapshot_date'])
lost_snapshots = lost_snapshots.sort_values(by=['deal_id', 'snapshot_date'])

# >>>>>>>>> Get first & last snapshot for each deal <<<<<<<<<<
lost_deals_df = (
    lost_snapshots.groupby('deal_id')
    .agg(
        old_stage=('stage', 'first'),
        last_stage=('stage', 'last'),
        old_forecast_category=('forecast_category', 'first'),
        last_forecast_category=('forecast_category', 'last'),
        owner_id=('owner_id', 'last'),
        amount=('amount', 'last')
    )
    .reset_index()
)

# ---- 4) Result ----
print(lost_deals_df)
```

```
Total lost deals in Q1: 17
      deal_id    old_stage   last_stage old_forecast_category  \
0   deal_114  Prospecting  Closed Lost              Pipeline
1   deal_147  Prospecting  Closed Lost              Pipeline
2   deal_271  Prospecting  Closed Lost              Pipeline
3   deal_296  Prospecting  Closed Lost              Pipeline
4   deal_299  Prospecting  Closed Lost              Pipeline
5   deal_314  Prospecting  Closed Lost              Pipeline
6   deal_401  Prospecting  Closed Lost              Pipeline
7   deal_410  Prospecting  Closed Lost              Pipeline
8    deal_47  Prospecting  Closed Lost              Pipeline
9   deal_487  Prospecting  Closed Lost              Pipeline
10  deal_514  Prospecting  Closed Lost              Pipeline
11  deal_556  Prospecting  Closed Lost              Pipeline
12  deal_575  Prospecting  Closed Lost              Pipeline
13  deal_604  Prospecting  Closed Lost              Pipeline
14  deal_674  Prospecting  Closed Lost              Pipeline
15  deal_676  Prospecting  Closed Lost              Pipeline
16   deal_96  Prospecting  Closed Lost              Pipeline
```

	last_forecast_category	owner_id	amount
0	Closed	owner_11	239748
1	Closed	owner_1	30694
2	Closed	owner_7	58571
3	Closed	owner_13	54979
4	Closed	owner_8	257671
5	Closed	owner_3	31818
6	Closed	owner_9	48684
7	Closed	owner_7	27056
8	Closed	owner_5	14266
9	Closed	owner_9	59781
10	Closed	owner_3	29119
11	Closed	owner_5	19764
12	Closed	owner_3	18281
13	Closed	owner_9	43355
14	Closed	owner_5	36002
15	Closed	owner_3	25931
16	Closed	owner_5	38931

Analysis of Deals Closed as Lost in Q1 2025

This script identifies and analyzes deals that were marked as "**Closed Lost**" during **Q1 2025**:

1. **Filter by Close Date & Stage**
   - Select deals with `close_date` between **2025-01-01** and **2025-03-31**
   - Only include deals where `deal_stage` is "**Closed Lost**"
2. **Snapshot Filtering**
   - Filter `deals_snapshot_df` to include only rows where `deal_id` is in the list of lost deals
3. **Chronological Ordering**
   - Sort the snapshots by `deal_id` and `snapshot_date` to ensure proper sequence
4. **First vs. Last Snapshot Comparison**
   - For each deal, extract:
     - `old_stage` and `last_stage`: to track stage movement
     - `old_forecast_category` and `last_forecast_category`: to assess forecasting consistency
     - Final `owner_id` and `amount`: for owner and deal size context

This analysis helps understand the evolution of lost deals and identify potential issues such as forecast misclassification or unexpected deal movement.

```
def compute_stage_metrics(deals_snapshot_df: pd.DataFrame, stall_days_threshold: int = 30):
    """
    Given a full deals snapshot dataframe, adds columns:
    - 'stage_change': True when stage differs from previous for same deal
    - 'stage_start_date': first date stage started (forward filled)
    - 'days_in_stage': days spent in current stage
    - 'stalled': True if days_in_stage >= threshold
    Returns the full dataframe with these columns.
    """
    df = deals_snapshot_df.copy()
    df['snapshot_date'] = pd.to_datetime(df['snapshot_date'])
    df = df.sort_values(by=['deal_id', 'snapshot_date'])

    # Detect stage changes
    df['stage_change'] = df.groupby('deal_id')['stage'].transform(lambda x: x != x.shift())

    # Compute stage start date (ffill where stage_change is True)
    df['stage_start_date'] = df.groupby('deal_id')['snapshot_date'].transform(
        lambda dates: dates.where(df['stage_change']).ffill()
    )

    # Calculate days in current stage
    df['days_in_stage'] = (df['snapshot_date'] - df['stage_start_date']).dt.days

    # Mark stalled deals
    df['stalled'] = df['days_in_stage'] >= stall_days_threshold

    return df

def get_stalled_result_df(deals_snapshot_df: pd.DataFrame, stall_days_threshold: int = 30):
    """
    Returns a reduced dataframe with only latest snapshot per deal,
    filtered by deals whose snapshot_date is in the current year.
    Includes only selected columns plus stalled flag.
```

```
    """
    df = compute_stage_metrics(deals_snapshot_df, stall_days_threshold)

    # Get current year
    today = pd.Timestamp.today()
    current_year = today.year

    # Get latest snapshot per deal
    latest_snapshots = df.sort_values(by='snapshot_date').groupby('deal_id').tail(1)

    # Filter to only deals in current year based on snapshot_date
    filtered = latest_snapshots[
        (latest_snapshots['snapshot_date'].dt.year == current_year)
    ]

    # Select only relevant columns for AI analysis
    columns = ['deal_id', 'stage', 'days_in_stage', 'owner_id', 'forecast_category', 'amount', 'stalled']
    return filtered[columns]


# full_deals_df has all snapshots + metrics
full_deals_df = compute_stage_metrics(deals_snapshot_df, stall_days_threshold=30)

# stalled_result_df filtered by current quarter, latest snapshot per deal (for AI input)
stalled_result_df = get_stalled_result_df(deals_snapshot_df, stall_days_threshold=30)

print(stalled_result_df.head())
full_deals_df["lost_deals_q1"] = full_deals_df["deal_id"].isin(lost_deals_df["deal_id"]).map({True: "Yes", False: "No"})


print(full_deals_df.loc[
          (full_deals_df["lost_deals_q1"] == "Yes")
      ])
```

```
        deal_id          stage  days_in_stage  owner_id forecast_category  \
4541  deal_374    Prospecting            126   owner_4          Pipeline
6894  deal_358     Closed Won              0   owner_7            Closed
6671  deal_113       Proposal              0   owner_7         Best Case
1704  deal_583    Negotiation              0  owner_11            Commit
1741  deal_601  Qualification              0  owner_11          Pipeline

      amount  stalled
4541   21761     True
6894   53559    False
6671   23953    False
1704   47192    False
1741  135126    False
       deal_id snapshot_date          stage forecast_category  amount  \
1318  deal_114    2024-10-21    Prospecting          Pipeline  239748
1319  deal_114    2024-10-28    Prospecting          Pipeline  239748
1320  deal_114    2024-11-04    Prospecting          Pipeline  239748
1321  deal_114    2024-11-11  Qualification          Pipeline  239748
1322  deal_114    2024-11-18  Qualification          Pipeline  239748
...        ...           ...            ...               ...     ...
5050   deal_96    2025-02-17       Proposal         Best Case   38931
5051   deal_96    2025-02-24       Proposal         Best Case   38931
5052   deal_96    2025-03-03    Negotiation         Best Case   38931
5053   deal_96    2025-03-10    Closed Lost            Closed   38931
5054   deal_96    2025-03-17    Closed Lost            Closed   38931

      close_date  owner_id  stage_change stage_start_date  days_in_stage  \
1318  2025-02-28  owner_11          True       2024-10-21              0
1319  2025-02-28  owner_11         False       2024-10-21              7
1320  2024-11-30  owner_11         False       2024-10-21             14
1321  2024-11-30  owner_11          True       2024-11-11              0
1322  2024-12-30  owner_11         False       2024-11-11              7
...          ...       ...           ...              ...            ...
5050  2025-04-18   owner_5          True       2025-02-17              0
5051  2025-01-18   owner_5         False       2025-02-17              7
5052  2025-02-18   owner_5          True       2025-03-03              0
5053  2025-01-18   owner_5          True       2025-03-10              0
5054  2025-03-18   owner_5         False       2025-03-10              7

      stalled lost_deals_q1
1318    False           Yes
1319    False           Yes
1320    False           Yes
1321    False           Yes
1322    False           Yes
...       ...           ...
5050    False           Yes
5051    False           Yes
5052    False           Yes
5053    False           Yes
5054    False           Yes

[244 rows x 12 columns]
```

Static Analysis

Build Pacing Table for Next Quarter

This function estimates sales performance per owner for the **next quarter**, based on current pipeline status and static probability weights.

1. Active Owners

- Filters `owners_df` to include only those whose role is active or ends **after** the start of the next quarter.

2. Prepare Deals

- Converts `snapshot_date` to datetime.
- Reclassifies `'Discovery'` stage to `'Prospecting'`.
- Filters deals with snapshots in the **current year**.

3. Completion Rate

- Calculates completion rate per owner = `Closed Won` deals ÷ total deals (clipped between 0 and 1).

4. Static Weights

- Defines **static probabilities** for each stage and forecast category:
  - `stage_prob`: e.g., `'Proposal'` = 0.7, `'Closed Won'` = 1.0
  - `forecast_weight`: e.g., `'Commit'` = 0.9, `'Pipeline'` = 0.3

5. Filter Valid Open Deals

- Removes deals in any `'Closed'` stage.
- Excludes stalled deals (those stuck in a stage longer than `stall_days_threshold`).

6. Apply Weights & Estimate Value

- Gets the **latest snapshot** for each open deal.
- Applies:
  - `stage_prob`
  - `forecast_weight`
  - `owner completion_rate`
- Computes `adjusted_prob = stage_prob * forecast_weight * completion_rate`
- Estimates value = `amount * adjusted_prob`

7. Aggregate by Owner

- Sums estimated pipeline value and open deal count per owner.

8. Merge with Targets

- Merges owner data with target data for the **next quarter**.
- Computes:
  - `pacing = estimated_value / target_amount`
  - `gap = estimated_value - target_amount`

9. Predictive Classification

- Classifies whether the owner is likely to hit their target using simple rules:
  - `"Yes"`: if estimated ≥ target
  - `"Probably Yes"`: if pacing ≥ 0.5 and has open deals
  - `"Probably No"`: if pacing < 0.5 and has open deals
  - `"No"`: no open deals

Returns:

A table with:

- Owner ID, Name, Segment
- Target and Estimated Values
- Pacing, Gap
- Open Deals Count
- Classification of likelihood to hit the target

```
import pandas as pd
from pandas.tseries.offsets import QuarterEnd

def build_pacing_table(owners_df, full_deals_df, targets_df, stall_days_threshold=30):
    today = pd.Timestamp.today()
    next_quarter = (today.month - 1) // 3 + 2
    next_year = today.year + (1 if next_quarter == 5 else 0)
    if next_quarter == 5:
        next_quarter = 1

    # --- 1) Active owners ---
    owners_active = owners_df[
        (owners_df['role_end_date'].isna()) |
        (pd.to_datetime(owners_df['role_end_date'], errors='coerce') >= pd.Timestamp(f"{next_year}-01-01"))
    ]

    # --- 2) Prepare deals ---
    df = full_deals_df.copy()
    df['snapshot_date'] = pd.to_datetime(df['snapshot_date'], errors='coerce')
    df['stage'] = df['stage'].replace({'Discovery': 'Prospecting'})
    current_year = today.year
    deals_this_year = df[df['snapshot_date'].dt.year == current_year]

    # --- 3) Owner completion rate ---
    total_deals = df.groupby("owner_id")["deal_id"].nunique()
    closed_deals = df[df['stage'].str.contains("Closed Won", case=False, na=False)] \
                      .groupby("owner_id")["deal_id"].nunique()
    completion_rate = (closed_deals / total_deals).fillna(0).clip(0, 1)

    # --- 4) STATIC stage & forecast weights ---
    stage_prob = {
        'Proposal': 0.7,
        'Negotiation': 0.8,
        'Contract Sent': 0.9,
        'Qualification': 0.4,
        'Prospecting': 0.3,
        'Closed Won': 1.0,
        'Closed Lost': 0.0
    }
    forecast_weight = {
        'Commit': 0.9,
        'Best Case': 0.7,
        'Pipeline': 0.3,
        'Closed': 1.0
    }

    # --- 5) Filter valid open deals ---
    open_deals = deals_this_year[
        ~deals_this_year['stage'].str.contains("Closed", case=False, na=False)
    ]
    filtered_deals = open_deals[
        ~((open_deals['stalled'] == True) & (open_deals['days_in_stage'] > stall_days_threshold))
    ]

    # Latest snapshot per deal
    latest_deals = (
        filtered_deals
        .sort_values('snapshot_date')
        .groupby('deal_id')
        .tail(1)
        .set_index('deal_id')
    )

    # --- 6) Apply static probabilities ---
    latest_deals['stage_prob'] = latest_deals['stage'].map(stage_prob).fillna(0.2)
    latest_deals['forecast_weight'] = latest_deals['forecast_category'].map(forecast_weight).fillna(0.1)
    latest_deals['completion_rate'] = latest_deals['owner_id'].map(completion_rate).fillna(0)

    latest_deals['adjusted_prob'] = (
        latest_deals['stage_prob'] * latest_deals['forecast_weight'] * latest_deals['completion_rate']
    )
    latest_deals['estimated_value'] = latest_deals['amount'] * latest_deals['adjusted_prob']

    # --- 7) Aggregate by owner ---
    expected_pipeline = latest_deals.groupby('owner_id')['estimated_value'].sum().reset_index()
    open_deals_count = latest_deals.groupby('owner_id').size().reset_index(name='open_deals_count')

    # Targets for the next quarter
    target_quarter_str = f"{next_year} Q{next_quarter}"
    targets_next_q = targets_df[targets_df['quarter'] == target_quarter_str]

    # --- 8) Merge everything ---
    pacing_df = (
```

```
    owners_active
    .merge(targets_next_q[['owner_id', 'target_amount']], on='owner_id', how='left')
    .merge(expected_pipeline, on='owner_id', how='left')
    .merge(open_deals_count, on='owner_id', how='left')
)

pacing_df['estimated_value'] = pacing_df['estimated_value'].fillna(0)
pacing_df['open_deals_count'] = pacing_df['open_deals_count'].fillna(0).astype(int)
pacing_df['pacing'] = pacing_df['estimated_value'] / pacing_df['target_amount']
pacing_df['gap'] = pacing_df['estimated_value'] - pacing_df['target_amount']

# --- 9) Simplified classification (Yes, Probably Yes, Probably No, No) ---
end_of_quarter = today + QuarterEnd(0)
days_remaining_in_quarter = (end_of_quarter - today).days

def classify_hit(row):
    if row["estimated_value"] >= row["target_amount"]:
        return "Yes"
    # Heuristic based only on pacing & deal count
    if row["pacing"] >= 0.5 and row["open_deals_count"] > 0:
        return "Probably Yes"
    if row["pacing"] < 0.5 and row["open_deals_count"] > 0:
        return "Probably No"
    return "No"

pacing_df["will_hit_target"] = pacing_df.apply(classify_hit, axis=1)

return pacing_df[['owner_id', 'name', 'segment', 'target_amount',
                  'estimated_value', 'pacing', 'gap',
                  'open_deals_count', 'will_hit_target']]


pacing_df = build_pacing_table(owners_df, full_deals_df, targets_df)
pacing_df["pacing"] = pacing_df["pacing"].round(2).astype(float)

pacing_df
```

	owner_id	name	segment	target_amount	estimated_value	pacing	gap	open_deals_count	will_hit_target
0	owner_1	Christopher Moore	SMB	111721	30266.732075	0.27	-81454.267925	25	Probably No
1	owner_2	Amber Jenkins	SMB	111721	79201.428980	0.71	-32519.571020	24	Probably Yes
2	owner_3	Mark Lopez	SMB	111721	47120.025455	0.42	-64600.974545	16	Probably No
3	owner_4	Kristen Mann	SMB	111721	54769.281538	0.49	-56951.718462	22	Probably No
4	owner_5	Dawn Alexander	SMB	111721	92643.665068	0.83	-19077.334932	34	Probably Yes
5	owner_6	Kelly Lin	Mid-Market	138321	109736.456923	0.79	-28584.543077	21	Probably Yes
6	owner_7	Dr. Kimberly Brennan	Mid-Market	138321	86818.062857	0.63	-51502.937143	19	Probably Yes
7	owner_8	Allen Gilmore	Mid-Market	138321	70248.582326	0.51	-68072.417674	18	Probably Yes
8	owner_9	Barbara Robinson	Mid-Market	138321	87188.497200	0.63	-51132.502800	25	Probably Yes
9	owner_10	Anthony Jackson	Enterprise	156941	31806.725385	0.20	-125134.274615	14	Probably No
10	owner_11	Michael Williams	Enterprise	156941	55410.930789	0.35	-101530.069211	13	Probably No

Clean

Final Pacing Data Preparation & Formatting

This section finalizes the `pacing_df` and adds helpful formatting and ordering for deal stage progression analysis.

1. Merge Owner Names into Full Deals

- Merges `pacing_df[['owner_id', 'name']]` into `full_deals_df` to include owner names.

2. Stage Ordering for Analysis

- Defines a custom sort order for deal stages using the `stage_order` dictionary:
  - e.g., `'Prospecting' = 'a'`, `'Closed Lost' = 'f'`
- Adds a new column `order` to `full_deals_df` based on this mapping.

3. Round Numeric Columns

- Rounds numeric columns (`estimated_value`, `gap`, `target_amount`, `pacing`) in `pacing_df` to **2 decimal places** and casts them as `float`.

4. Format Columns for Display

- Formats the main numeric columns to **two decimal places with comma as decimal separator** (e.g., `1.23` → `"1,23"`) for user-friendly display:
  - Columns: `"pacing"`, `"target_amount"`, `"estimated_value"`, `"gap"`

This ensures the final pacing table is both accurate and presentation-ready.

```
full_deals_df = full_deals_df.merge(
    pacing_df[["owner_id", "name"]],
    on="owner_id",
    how="left"
)

stage_order = {
    "Prospecting": "a",
    "Qualification": "b",
    "Proposal": "c",
    "Negotiation": "d",
    "Closed Won": "e",
    "Closed Lost": "f"
}

full_deals_df["order"] = full_deals_df["stage"].map(stage_order)


pacing_df[["estimated_value", "gap", "target_amount", "pacing"]] = pacing_df[["estimated_value", "gap", "target_amount", "pacing"]].rou

cols_to_format = ["pacing", "target_amount", "estimated_value", "gap"]
for col in cols_to_format:
    pacing_df[col] = pacing_df[col].apply(lambda x: f"{x:.2f}".replace(".", ",") if isinstance(x, float) else x)
pacing_df
```

	owner_id	name	segment	target_amount	estimated_value	pacing	gap	open_deals_count	will_hit_target
0	owner_1	Christopher Moore	SMB	111721,00	30266,73	0,27	-81454,27	25	Probably No
1	owner_2	Amber Jenkins	SMB	111721,00	79201,43	0,71	-32519,57	24	Probably Yes
2	owner_3	Mark Lopez	SMB	111721,00	47120,03	0,42	-64600,97	16	Probably No
3	owner_4	Kristen Mann	SMB	111721,00	54769,28	0,49	-56951,72	22	Probably No
4	owner_5	Dawn Alexander	SMB	111721,00	92643,67	0,83	-19077,33	34	Probably Yes
5	owner_6	Kelly Lin	Mid-Market	138321,00	109736,46	0,79	-28584,54	21	Probably Yes
6	owner_7	Dr. Kimberly Brennan	Mid-Market	138321,00	86818,06	0,63	-51502,94	19	Probably Yes
7	owner_8	Allen Gilmore	Mid-Market	138321,00	70248,58	0,51	-68072,42	18	Probably Yes
8	owner_9	Barbara Robinson	Mid-Market	138321,00	87188,50	0,63	-51132,50	25	Probably Yes
9	owner_10	Anthony Jackson	Enterprise	156941,00	31806,73	0,20	-125134,27	14	Probably No
10	owner_11	Michael Williams	Enterprise	156941,00	55410,93	0,35	-101530,07	13	Probably No
11	owner_12	Amanda Hutchinson	Strategic	170241,00	146908,22	0,86	-23332,78	24	Probably Yes
12	owner_13	Eileen Nunez	Strategic	170241,00	180273,04	1,06	10032,04	24	Yes

Load

```
full_deals_df.to_csv("full_deals.csv", index=False)
pacing_df.to_csv("pacing.csv", index=False)
```