Volume10_deleted
JSON

Volume11_deleted
JSON

Volume12_deleted
JSON

我为您提供的这个任务生成脚本和文件产出。这应该可以完成任务。请检查文件内容，并告诉我哪些内容需要调整，我为您进行修改。

任务目标

研究，总结一下在 Organic Synthesis 系列文章中出现的所有不带编号反应名，并进行分类和总结整理这些反应，并查阅这些反应的来源 (时间) 提供给LLM大学，基于大量的数据来于这些反应提供给更接近于现实的分类。

方案规划

定义反应 (Organic Synthesis) 中带有编号的反应名，重点筛选出带有编号的反应 (例如，含有编号的可能性的反应)，检查文件内容是否合理，并提取出，提取内容是否合理，并提取出，提取内容是否合理，并提取出，提取

```
LigandCatalyst)
```

- **Derived Substructure Type** (e.g., AminoAcidHydrolyte, Imine, Alkene, Sulfide, Ester, β-Keto Ester).

4. Apply this classification to each reaction.
5. Perform temporal analysis:
   - Plot the number of reported asymmetric reactions per year.
   - Plot the distribution of catalyst system types over time (e.g. per decade).
   - Plot the distribution of main reaction types over time.
6. Save the enriched data as enriched_2026.7z.

Important Considerations for Classification:

- **Automated Nature:** This classification is based on keyword matching in the titles and abstracts/keywords. It is a heuristic tool to accelerate surveys and provide a "best guess" for large datasets. Nuances, complex mechanisms, or novel approaches not covered by simple keywords, potentially leading to "unknown" or misclassifications.
- **Hierarchy:** The classification is hierarchical: a case is first assigned to a broad category (e.g., "Ketone") and then a more specific one (e.g., "Ketone Aldol").
- **"Synthesis of Chiral Ligands/Catalysts/Auxiliaries":** Here, Chg. Synth. plays a crucial role. The classification aims to identify if the paper is about making the chiral auxiliary. The script will try to identify these. In such cases, the "catalyst" might be a reading agent or an achiral reagent used in the synthesis.

Python

```
import ijson
import pandas
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter

# --- Keyword Definitions for Classification ---

# 1. Catalyst System Type
CATALYST_SYSTEM_TYPE = {
    "Organo/Biocatalyst": [
        "proline", "Tables's proline", "Tylosin", "enzyme", "enzymes", "dehydrogenase",
        "B. fusiformis", "substrate", "kelps with", "Pectinomycin", "Pulmonarin", "Ma
    ],
    "Organometallic - Palladium/Nickel": ["palladium", "Alumina/palladium", "metal
    "Organometallic - Rhodium/Iridium": ["rhodium", "Rhodium", "Rh", "Ir", "iridium",
    "Organometallic - Ruthenium/Osmium": ["ruthenium", "Ru", "osmium", "Os", "Ru(II)
    "Organometallic - Copper/Zinc": ["copper", "Cu", "zinc", "Zn", "Cu(OTf)2", "Cu(
    "Organometallic - Other Metal": ["titanium", "Ti", "zirconium", "Zr", "hafnium",
    "Organocatalyst - Chiral Phosphoric Acid (CPA)/Binaryzine Acid": [
        "chiral phosphoric acid", "chiral binaryzine acid", "chiral phosphoric ac
        "chiral phosphoric acid", "chiral binaryzine acid",
    ],
    "Organocatalyst - NHC (N-Heterocyclic Carbene)": ["nhc", "n-heterocyclic carbe
    "Organocatalyst - Oxazaborolidine": ["oxazaborolidine", "cbs catalyst", "cbs
    "Organocatalyst - Phase Transfer (Chiral PTC)": ["phase transfer catalyst", "c
    "Organocatalyst - Aminocatalyst": ["organocatalyst", "aminocatalyst", "secon
    "Metal Catalyst - Ruthenium (Ru)": ["ruthenium", "ru", "ru(II)", "ru-bipy", "ru-bipy
    "Metal Catalyst - Rhodium (Rh)": ["rhodium", "rh", "rh(I)", "rh(II)", "rh(III)",
    "Metal Catalyst - Palladium (Pd)": ["palladium", "pd", "pd(0)", "pd(II)",
    "Metal Catalyst - Iridium (Ir)": ["iridium", "ir(I)",
    "Metal Catalyst - Copper (Cu)": ["copper", "cu", "cu(I)", "cu(II)", "cu(OTf)2",
    "Metal Catalyst - Nickel (Ni)": ["nickel", "ni", "ni(0)", "ni(II)",
    "Metal Catalyst - Titanium (Ti)": ["titanium", "ti(IV)", "ti(OiPr)4", "dendrim
    "Metal Catalyst - Zinc (Zn)": ["zinc", "zn", "zn(II)", "zn(OTf)2", "dialkylzinc
    "Metal Catalyst - Iron (Fe)": ["iron", "fe(II)", "fe(III)",
    "Metal Catalyst - Gold (Au)": ["gold", "au(I)", "au(I)", "au(III)",
    "Metal Catalyst - Silver (Ag)": ["silver", "ag", "ag(I)", "ag(II)",
    "Metal Catalyst - Aluminum (Al)": ["aluminum", "al", "al(III)", "al(OiPr)3",
    "Metal Catalyst - Magnesium (Mg)": ["magnesium", "mg(II)", "grignard reagent",
    "Metal Catalyst - Lanthanide (Ln)": ["lanthanide", "ln", "ln(III)", "ln(OiPr)3",
    "Metal Catalyst - Cobalt (Co)": ["cobalt", "co", "co(II)", "co(III)",
    "Metal Catalyst - Boron/Lewis/Brønsted": ["borohydr", "borohydrid", "triethylb
    "Metal Catalyst - Tin (Sn)": ["tin", "sn", "sn(II)", "sn(IV)", "dibutylsn(IV)
    "Peroxovana ligand", "peva", "l-ala", "ligand",
    "Metal Catalyst - Manganese (Mn)": ["manganese", "mn", "mn(II)",
    "Metal Catalyst - Other": ["chromium", "cr", "molybdenum", "mo", "tungsten
```

为了防止过度分析的分析结果，此数据的人工分析：

1. 根据“底物”和第一步反应（`reaction_type`），将每个反应分为“不对称”（手性）或“非对称”（非手性）。
2. 将反应按其主要的`asymmetric_type`（不对称类型）进行分组。
3. 手动确定分组：对每组中的反应，为每个“反应”分组确定最合适的最终产物类型，及此类别的简短描述。
4. 建议重命名：同时提出对通用名称的更改，以更好地反映分组。
5. 输出：以一张新的表格输出，包含原始`reaction_id`、原始`reaction_name`、分组后的`product_class`（最终产物类别）、`product_description`（产物类别的简短描述）、`rationale`（分类理由）。

classified_rx_dataset
csv

我打算Claude进行了人工审定，他提议了一个方向性，我认为非常不错，请帮我分析这个新的数据，谢谢。我的数据，是基于Nature Catalysis 最新发表的，不对称合成的文献，分析这篇新的文献的数据，谢谢。

好的，我来分析这个新的数据，谢谢。

Okay, this is excellent! Working with manually curated and curated data will lead to a much more accurate and insightful analysis.

Based on your `classified_reactions_detailed_manual_deleted.csv` file, I will perform the following:

1. **Load the Curated Data:** Import the JSON file into a Pandas DataFrame.
2. **Structural Analysis:**
   - `metal_center` (e.g., Rh, Pd, Cu, Ti) - if applicable.
   - `ligand_or_core_type` (e.g., BINAP, Salen, Proline, Cinchona-Alkoid, CR5) - this will capture the key structural feature of the chiral catalyst/ligand.
3. **Quantitative Analysis & Properties:**
   - Calculate the overall distribution (counts and percentages) of:
     - `major_catalyst_class`
     - `metal_center` (for metal-catalyzed reactions)
     - `ligand_or_core_type`
     - `reaction_type`
   - Calculate the overall distribution (counts and percentages) of `major_catalyst_class` and `ligand_or_core_type` categories over decades.
   - Observe trends in `reaction_type` and `asymmetric_type` over decades.
4. **Interpretation of Trends and Properties:** Based on the quantitative data, I will discuss:
   - The overall development trajectory of asymmetric catalysis as reflected in Organic Synthesis.
   - The rise, fall, or sustained use of different catalytic strategies.
   - The prominence of specific metals, ligand architectures, or organocatalyst motifs and why.
   - The types of chemical transformations and substrates that are frequently addressed.
   - Connections to landmark discoveries in the field and the specific nature of Organic Synthesis publications.

Let's start with the Python code to process the corrected data and extract the necessary information for the analysis.

Python

```
import time
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter

# --- Expand Definitions for Detailed Catalyst Classifications ---

# Metals (Common for analysis)
METALS = ['Rh', 'Ru', 'Pd', 'Cu', 'Ni', 'Ti', 'Zn', 'Mo', 'Co', 'Mn',

# Ligand Derivatives / Organocatalyst Cores (Examples for matching, representative
LIGAND_DERIVATIVES = ['BINAP', 'Salen', 'Proline', 'Cinchona-Alkoid', 'CR5',
LIGAND_ORGANOCATALYST_CORE = [
# Phosphine Ligands (often bidentate)
'DIPAMP', 'Chiraphos', 'Josiphos', 'Mandyphos', 'BINAP', 'Ph-BINAP', 'Xyl-BINAP
'DIPAMPHOS', 'Ph-phosphos',
'Chiraphos', 'Diphosphos',
'Josiphos', 'Ferrocenyl',
'Mandyphos', 'Binaphanes',
'Ph-phosphos', 'Binaphanes',
'Xyl-BINAP', 'Xyl-phosphos',
'Diphosphos', 'Binaphanes',
'Binaphanes', 'Binaphanes',
'Binaphanes', 'Binaphanes',
'Monodentate P-Ligands', 'Monodentate P-Ligands',
'Monodentate P-Ligand', 'Ferrocenyl', 'Ferrocenyl Ligand', 'Cyclopentadienyl',
'Other Bidentate P-Ligand', 'P-Ligand', 'P-Ligand', 'P-Ligand', 'P-Ligand',

# N-Ligands & Diamines
'Bisoxazoline', 'Pyridine', 'Pyridine',
'Bisoxazoline', 'Bisoxazoline', 'Bisoxazoline',
'Bisoxazoline', 'Bisoxazoline', 'Pyridine
```

ongoing specialized roles.

3. Deep Dive into Metal Catalysts:

Despite manual data correction, the automated parsing of specific metal centers and particularly ligand data from the descriptive text proved challenging. "Metal Catalyst" (extracted for 43.8% of reactions) of metal-catalyzed entries, and most identified metals were paired with an "Unspecified Ligand." This hints that ground-ligand resolution analysis from the automated script needs further refinement to provide more insight.

- **Boron (B) (15.7% of metal cats):** Prominent from the 1990s, peaking in the 2010s (9 entries). This is strongly associated with CBS (Corey-Bakshi-Shibata) reductions using chiral oxazaborolidines.
- **Copper (Cu) (9.3%):** Rose to prominence in the 2000s (7 entries) and 2010s (5 entries), reflecting its broad utility in conjugate additions, cycloisomerizations, and other C-C and C-X bond formations.
- **Titanium (Ti) (8.1%):** Notable in the 1990s (3 entries) and 2010s (3 entries), primarily linked to Sharpless epoxidations and related Ti-tartrate or Ti-BINOL chemistries.
- **Ruthenium (Ru) (6.8%) & Rhodium (Rh) (6.7%):** Have remained significant from the 1990s onwards, synonymous with highly efficient asymmetric hydrogenations (e.g., Noyori's Ru-BINAP systems) and other transformations or transfer reactions.
- **Palladium (Pd) (4.0%) & Nickel (Ni) (4.0%):** Feature mainly in the 2010s, crucial for asymmetric cross-coupling reactions and allylic alkylations.

The lack of automated specific ligand identification (e.g., BINAP, Salen, DuPhos, phosphoramidites, NHC ligands) prevents a detailed analysis of "ligand evolution," which is a critical aspect of metal-catalyst development. This information is likely embedded in the text but requires more targeted extraction.

4. Insights into Organocatalysts:

The automated parsing of specific organocatalyst names also faced limitations, with "Other Organocatalyst" remaining a key reaction of the class.

- **Proline/Proline Derivatives (30% of organocat.):** 3 reactions, with a strong showing in the 2010s (3 entries), consistent with the broad impact of proline and its derivatives in direct asymmetric aldol, Mannich, and Michael reactions.
- **Chiral Phosphoric Acids (CPAs) (10%):** 1 reaction, with a surprisingly steady, low level presence of one entry per decade from the 1980s to the 2010s. While a cornerstone of modern organocatalysis for their Brønsted acidity, their use in syntheses might reflect foundational work or synthesis of BINOL-derived phosphates.

The full diversity of organocatalysts (Jacobsen (salen), Zn, Nickel(Box-Ph)(OtBu)2(H2O), Jørgensen-Hayashi catalysts, thiourea, NHCs) is likely concealed within the "Other Organocatalyst" category.

5. Substrate Preferences:

The classification of substrate types was more successful, with only 8.9% remaining "Unknown."

- **Ketones/Alkylketones (5.2) (Unsaturated Carbonyls (26.4% - 20 reactions).** These are the most frequently employed substrates across all decades, particularly in the 2010s. Their inherent prochiality makes them ideal for a vast range of transformations.
- **Halides/Pseudohalides (17.6% - 19 reactions):** Prominent since the 1990s, vital for asymmetric cross-coupling reactions and as precursors in the synthesis of chiral ligandsstatistics - a common type of precursor in Organic Synthesis.
- **Esters/Lactones/Acids/Imides/Anhydrides (14.2% - 15 reactions):** Versatile substrates for reductions, α-functionalizations, and resolutions, frequently appearing from the 1990s through the 2010s.
- **Alkenes/Dienes/Alkynes/Olefins (9.4% - 10 reactions):** Showed increased use in the 2000s and particularly the 2010s. They are key for hydrogenations, epoxidations, dihydroxylations, cycloadditions, and emerging C-H functionalization.
- **Imines/Hydrazones/Nitrones (5.7% - 6 reactions):** Their appearance, especially in the 2000s, aligns with the expansion of catalytic asymmetric additions to C=N bonds.

6. Reaction Type Landscape:

While the detailed decade-by-decade table for reaction types was not directly provided in the last user output, general counts from the previous, less refined stage indicated Michael additions, reductions, and cycloadditions as prominent. A full analysis would require the decade breakdown of the manually corrected reaction types.

It is expected that fundamental reactions like reductions, oxidations, various C-C bond formations (aldol, Michael, couplings, allylic substitutions), hydrogenations, and cycloadditions would be well represented. Interactions detailing the "Synthesis of Chiral Catalysts (specifications)" or "Resolution" processes are also anticipated to be significant, highlighting the practical nature of Organic Synthesis in providing access to enantiopure materials and catalysts.

7. Conclusions:

The manually curated dataset of 106 asymmetric reactions from Organic Synthesis (1980-2020) clearly illustrates the field's dynamicnature. Metal catalysis remains the dominant approach, showcasing a diversification of useful metals over time. Organocatalysis has firmly established itself as a major complementary strategy, particularly from the 2000s. Enzymes provide continuous, specialized solutions. The choice of substrates reflects common practical starting materials and those suitable for constructing complex chiral architectures.

The primary limitation of this automated statistical summary remains the script's inability to parse highly specific ligand names (detailed organocatalyst structures) from textual descriptions, leading to large "Unknown Metals," "Unspecified Ligand," and "Other Organocatalyst" categories at the finest level of detail. Nevertheless, the broader trends observed are robust and align well with the historical narrative of asymmetric catalysis. For a deeper understanding of specific catalyst evolution, manual review of the `reaction_dataset_with_corrected_metals.xlsx` spreadsheet would be useful to refine the `metal_catalyst` strings and procedure texts.

Below is the Python script for generate statistical plots from your `reaction_dataset_with_corrected_metals.xlsx` file (which should be the output of the previous Python script run on your manually corrected data).

Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings ## to ignore warnings
warnings.filterwarnings('ignore')

def plot_analysis_results(excel_file_path="final_classified_asymmetric_reactions_[c

    Loads the analyzed data and generates plots for temporal trends.
    Assumes columns 'decade', 'metal_catalyst', 'ligand', 'reaction_type',
    'major_enantiomer_product', 'metal_center', 'ligand_or_core_type',
    'reaction_type', 'substrate_type'.
    Saves plots to PNG files.
    try:
        df = pd.read_csv(excel_file_path)
    except FileNotFoundError:
        print(f"Error: File '{excel_file_path}' not found.")
        return [] # Return empty list if file not found
    except Exception as e:
        print(f"Error reading CSV file: {e}")
        return []

    print("Preparing plots from '{excel_file_path}'...")
    plt.style.use('seaborn-v0_8-whitegrid')

    # Ensure 'decade' column exists and is numeric
    if 'decade' in df.columns and pd.to_numeric(df['decade'], errors='coerce').n
        df['decade'] = df['decade'].astype(int)

    # -- Plot 1: Total Asymmetric Reactions per Decade --
    if 'decade' in df.columns:
        reaction_counts_per_decade = df['decade'].value_counts().sort_index()
        reaction_per_decade_distribution_bar, ax1 = plt.subplots(figsize=(10,
        sns.barplot(x=reaction_counts_per_decade.index, y=reaction_counts_per_
        ax1.set_title('Number of Asymmetric Reactions Reported per Decade\n(Orig. Synth.
        ax1.set_xlabel('Decade')
        ax1.set_ylabel('Number of Reactions')
        ax1.tick_labels_rotation(0)
        ax1.tight_layout()
        plot_filename = "1_total_reactions_per_decade.png"
        reaction_per_decade_distribution_bar.savefig(plot_filename)
        print(f"Saved plot: {plot_filename}")
        plt.close(reaction_per_decade_distribution_bar)
        output_plot_list.append(plot_filename)
    else:
        print("Skipping Plot 1: 'decade' column not found.")

    # -- Plot 2: Major Catalyst Class Distribution per Decade --
    if 'decade' in df.columns and 'major_catalyst_class' in df.columns:
        major_catalyst_counts = df.groupby('decade')['major_catalyst_class'].v
        if not major_catalyst_counts.empty and major_catalyst_counts.shape[1] >
            major_catalyst_class_per_decade_distribution_bar, ax2 = plt.subplots(
            sns.barplot(x=major_catalyst_counts.index, y=major_catalyst_counts.
            ax2.set_title('Distribution of Major Catalyst Classes per Decade')
            ax2.set_xlabel('Decade')
            ax2.set_ylabel('Number of Reactions')
            ax2.tick_labels_rotation(0)
            ax2.legend(title='Major Catalyst Class', bbox_to_anchor=(1.05, 1), loc='upper
            plot_filename = "2_major_catalyst_class_per_decade.png"
            major_catalyst_class_per_decade_distribution_bar.savefig(plot_filename)
            plt.close(major_catalyst_class_per_decade_distribution_bar)
            output_plot_list.append(plot_filename)
        else:
            print("Skipping Plot 2: 'major_catalyst_class' column not found

    # -- Plot 3: Metal Center Distribution per Decade (Top Metal Catalysts) --
    if 'decade' in df.columns and 'major_catalyst_class' in df.columns and 'metal_c
        if metal_c = df[df['major_catalyst_class'] == 'Metal Catalyst'].copy()
        if not metal_c.empty:
            metal_counts = df.groupby('metal_center')['catalyst_specific', 'metal_c
            # Consider top 10 metals, excluding 'Unknown Metal' if it is part of the top
            top_metals = metal_counts[metal_counts > 0].head(10).index.tolist()

            df_metal = metal_center_plot[ metal_center'] metal_center'].isin(top
            df_metal['plot_metal'] = df_metal['metal_center'].replace(metals, 'Metal_c

            metal_center_per_decade = df_metal.groupby('decade')['metal_center_plot
            if not metal_center_per_decade.empty:
                print("Debug: Data for Metal Center Distribution Plot:\n", metal_cen
                expected_metals_for_plot = list(df_metal['metal_center_plot'].unique
                print("Debug: Expected metals for plot:", expected_metals_for_plot)
                if not metal_center_per_decade.shape[1] > 0:
                    metal_center_per_decade[expected] = 0

                metal_center_per_decade.plot(kind='bar', stacked=True, figsize=(10,
                plt.title('Distribution of Metal Centers per Decade (Metal Catalyst
                plt.xlabel('Decade')
                plt.ylabel('Number of Reactions')
                plt.xticks(rotation=0)
                plt.legend(title='Metal Center', bbox_to_anchor=(1.05, 1), loc='upper
                plt.tight_layout()
                plot_filename = "3_metal_center_per_decade.png"
                metal_center_per_decade_distribution_bar.savefig(plot_filename)
                plt.close(metal_center_per_decade_distribution_bar)
                output_plot_list.append(plot_filename)
            else:
                print("Skipping Plot 3: Recovery columns for metal center analysis not fo
        else:
            print("Skipping Plot 3: Recovery columns for metal center analysis not fo
    else:
        print("Skipping Plot 3: 'major_catalyst_class' or 'metal_center' or 'ligand_c

    # -- Plot 4: Ligand/Core Type Distribution per Decade (Organocatalyst) --
    if 'decade' in df.columns and 'major_catalyst_class' in df.columns and 'ligand_
        if not df[df['major_catalyst_class'] == 'Organocatalyst'].empty:
            if not df.empty:
                ligand_counts = df.groupby('ligand_or_core_type')['catalyst_specific
                top_ligand_types = ligand_counts[ligand_counts > 0].head(10).index.
                df_organo = df[df['ligand_or_core_type'].isin(top_ligand_types)].co
                organo_core_per_decade = df_organo.groupby('decade')['ligand_or_core_t
                if not organo_core_per_decade.empty:
                    expected_organo_cores_for_plot = list(df_organo['ligand_or_core_ty
                    if not organo_core_per_decade.shape[1] > 0:
                        organo_core_per_decade[expected] = 0
                    organo_core_per_decade = organo_core_per_decade[sorted(organo_cor

                    organo_core_per_decade.plot(kind='bar', stacked=True, figsize=(10,
                    plt.title('Distribution of Organocatalyst Core Types per Decade')
                    plt.xlabel('Decade')
                    plt.ylabel('Number of Reactions')
                    plt.xticks(rotation=0)
                    plt.legend(title='Organocatalyst Core', bbox_to_anchor=(1.05, 1),
                    plt.tight_layout()
                    plot_filename = "4_organocatalyst_core_types_per_decade.png"
                    organo_core_per_decade_distribution_bar.savefig(plot_filename)
                    plt.close(organo_core_per_decade_distribution_bar)
                    output_plot_list.append(plot_filename)
                else:
                    print("Skipping Plot 4: Recovery columns for organocatalyst core analysis
            else:
                print("Skipping Plot 4: Recovery columns for organocatalyst core analysis
        else:
            print("Skipping Plot 4: No Organocatalyst data found.")
    else:
        print("Skipping Plot 4: 'major_catalyst_class' or 'ligand_or_core_type' or 'de

    # -- Plot 5: Reaction Type Distribution per Decade --
    if 'decade' in df.columns and 'reaction_type' in df.columns:
        reaction_counts = df.groupby('decade')['reaction_type'].value_counts()
        top_reaction_types = reaction_counts[reaction_counts > 0].head(10).index.
        df_reaction = df[df['reaction_type'].isin(top_reaction_types)].copy()
        if df_reaction.shape[0] > 0 and df_reaction['reaction_type'].unique().sh
            reaction_type_counts = df_reaction.groupby('decade')['reaction_type'].
            if not reaction_type_counts.empty:
                expected_reaction_type_plot = list(df_reaction['reaction_type'].uni
                for type in expected_reaction_type_plot:
                    if type not in reaction_type_counts.columns:
                        reaction_type_counts[type] = 0
                reaction_type_per_decade = reaction_type_counts[sorted(expected_reac

                reaction_type_per_decade.plot(kind='bar', stacked=True, figsize=(10,
                plt.title('Distribution of Reaction Types per Decade')
                plt.xlabel('Decade')
                plt.ylabel('Number of Reactions')
                plt.xticks(rotation=0)
                plt.legend(title='Reaction Type', bbox_to_anchor=(1.05, 1), loc='upper
                plt.tight_layout()
                plot_filename = "5_reaction_type_per_decade.png"
                reaction_type_per_decade_distribution_bar.savefig(plot_filename)
                plt.close(reaction_type_per_decade_distribution_bar)
                output_plot_list.append(plot_filename)
            else:
                print("Skipping Plot 5: 'decade' or 'reaction_type' column not found.")

    # -- Plot 6: Substrate Type Distribution per Decade --
    if 'decade' in df.columns and 'substrate_type' in df.columns:
        substrate_counts = df.groupby('decade')['substrate_type'].value_counts()
        top_substrate_types = substrate_counts[substrate_counts > 0].head(10).index.
        df_substrate = df[df['substrate_type'].isin(top_substrate_types)].copy()
        if df_substrate.shape[0] > 0 and df_substrate['substrate_type'].unique().s
            substrate_type_counts = df_substrate.groupby('decade')['substrate_type.
            if not substrate_type_counts.empty:
                expected_substrate_type_plot = list(df_substrate['substrate_type'].u
                for type in expected_substrate_type_plot:
                    if type not in substrate_type_counts.columns:
                        substrate_type_counts[type] = 0
                substrate_type_per_decade = substrate_type_counts[sorted(expected_su

                substrate_type_per_decade.plot(kind='bar', stacked=True, figsize=(10,
                plt.title('Distribution of Substrate Types per Decade')
                plt.xlabel('Decade')
                plt.ylabel('Number of Reactions')
                plt.xticks(rotation=0)
                plt.legend(title='Substrate Type', bbox_to_anchor=(1.05, 1), loc='upper
                plt.tight_layout()
                plot_filename = "6_substrate_type_per_decade.png"
                substrate_type_per_decade_distribution_bar.savefig(plot_filename)
                plt.close(substrate_type_per_decade_distribution_bar)
                output_plot_list.append(plot_filename)
            else:
                print("Skipping Plot 6: 'decade' or 'substrate_type' column not found.")

    print(f"Finished processing. Generated {len(output_plot_list)} plots: {', '.join(outp

    # Save if the requested input file exists.
    if excel_file_path == "final_classified_asymmetric_reactions_[cleaned].xlsx":
        if output_plot_list:
            print("Attempting to generate plots from: {lean_file_for_plotting}")
            generated_plots = plot_analysis_results(lean_file_for_plotting)
            if generated_plots:
                print("Plotting complete. Please find the PNG files in your working d
            else:
                print("Plotting script ran, but no plots were generated. Check for w
    elif excel_file_path == lean_file_for_plotting:
        print("Attempting quick scan: The input file '{lean_file_for_plotting}' is
        if os.path.exists(lean_file_for_plotting):
            df_check = pd.read_csv(lean_file_for_plotting, nrows=5, low_memory=Fa
            print("Successfully read the first 5 rows of {lean_file_for_plotting}
            print("Please ensure that script ran successfully and created this specific
```

Sonnet 3.7(Extended)

Deep Research

Canvas

Sonnet 3.7(Extended)可能会出错。请核对重要信息。

Sonnet 3.7(Extended)

Deep Research

Canvas

Sonnet 3.7(Extended)可能会出错。请核对重要信息。