Federal_Circuit_Annual_Report_Generator

May 30, 2025

1 Federal Circuit Annual Report Generator

This Jupyter Notebook processes and analyzes decisions from the U.S. Court of Appeals for the Federal Circuit. It generates tables and visualizations for inclusion in the court's annual statistical report, focusing on categories such as disposition types, origins of appeals, and writs of mandamus.

The notebook reads from a tab-delimited master dataset and filters it by document year, document type, and other metadata. Processed outputs include cross-tabulations, bar and pie charts, and summaries written to a formatted Excel workbook.

Author: Jason Rantanen
Assistance: This notebook was developed with the assistance of ChatGPT (OpenAI).

[1]:
```
from datetime import datetime
from io import BytesIO
from openpyxl import load_workbook
from openpyxl import load_workbook, Workbook
from openpyxl.drawing.image import Image as OpenpyxlImage
from openpyxl.styles import Font
from openpyxl.utils.dataframe import dataframe_to_rows
from IPython.display import display, Markdown
import matplotlib.pyplot as plt
import os
import pandas as pd
```

[2]:
```
#
#  Cell 1: Load dataset
#

filename = "2025-05-30 CAFC documents annual export.tab"
df = pd.read_csv(filename, sep="\t", dtype=str)
df.head()
```

[2]:

	caseName	appealNumber	origin	\
0	ARNOLD V. FEDERAL DEPOSIT INSURANCE CORP.	2007-3304	MSPB	
1	CORPUZ V. OPM	2006-3177	MSPB	
2	MELLICK v. INTERIOR [OPINION](pdf)	23-1733	MSPB	
3	Geraldine H. WANG, Petitioner, v. DEPARTMENT O…	NaN	NaN	

```
4      MUNCHKIN, INC. v. LUV N' CARE, LTD. [OPINION]      19-1454    DCT

  PrecedentialStatus                                          CAFC_URL  \
0    Nonprecedential  http://www.cafc.uscourts.gov/sites/default/fil…
1    Nonprecedential  http://www.cafc.uscourts.gov/sites/default/fil…
2    Nonprecedential  https://cafc.uscourts.gov/opinions-orders/23-1…
3                NaN                                                NaN
4       Precedential  http://www.cafc.uscourts.gov/sites/default/fil…

                                FileName  docDate uniqueID  \
0                             07-3304.pdf  5/12/08    24328
1                             06-3177.pdf   8/9/06    25172
2  23-1733.OPINION.4-17-2024_2303360.pdf  4/17/24    42992
3                                    NaN  8/29/06    39847
4   19-1454.OPINION.6-8-2020_1600475.pdf   6/8/20    27123

             NewFileName                                          CloudLink  \
0  02339CAFCDocument.pdf  https://storage.googleapis.com/cafc_compendium…
1  01538CAFCDocument.pdf  https://storage.googleapis.com/cafc_compendium…
2  51014CAFCDocument.pdf  https://storage.googleapis.com/cafc_compendium…
3                    NaN                                                NaN
4  18528CAFCDocument.pdf  https://storage.googleapis.com/cafc_compendium…

   … Issue_Definiteness Issue_ObvTypDblPatenting Issue_ClaimConstruction  \
0  …                NaN                      NaN                     NaN
1  …                NaN                      NaN                     NaN
2  …                 No                       No                      No
3  …                NaN                      NaN                     NaN
4  …                 No                       No                      No

  Appellant_Type_Primary Appellant_Type_Cross Misc_Docket_Appeal_Type  \
0                    NaN                 NaN                     NaN
1                    NaN                 NaN                     NaN
2                  Other                 NaN                     NaN
3                    NaN                 NaN                     NaN
4         Patent asserter                  NaN                     NaN

     DataSource Dissent Concurrence Misc_Docket
0  CAFC Website      No          No         NaN
1  CAFC Website      No          No         NaN
2  CAFC Website     NaN         NaN       False
3           CAP     NaN         NaN         NaN
4  CAFC Website      No          No         NaN

[5 rows x 52 columns]
```

[3]:
```
#
#   Cell 2: Define reusable utility functions
#


def filter_base_data(df):
    filtered = df[
        (df['docYear'].astype(float) > 2009) &
        (df['docYear'].astype(float) <= 2024) &
        (df['docType'].isin(['Opinion', 'Rule 36'])) &
        (df['Replaced'] != 'Yes')
    ].copy()
    filtered['docYear'] = filtered['docYear'].astype(int)
    return filtered


def simplify_origin(origin):
    if origin in ['PTO', 'DCT', 'MSPB', 'CAVC', 'CFC', 'CIT']:
        return origin
    elif pd.isna(origin):
        return 'Other'
    else:
        return 'Other'


def generate_chart(df, title, stacked=True, percentage=False, image_name=None):
    fig, ax = plt.subplots(figsize=(12, 6))
    if percentage:
        df = df.div(df.sum(axis=1), axis=0) * 100
    df.plot(
        kind='bar',
        stacked=stacked,
        ax=ax,
        edgecolor='black',
        linewidth=0.6
    )
    ax.set_title(title, fontsize=14, fontweight='bold')
    ax.set_xlabel("Year", fontsize=12)
    ax.set_ylabel("% of Appeals" if percentage else "Number of Appeals",␣
 ↪fontsize=12)
    plt.xticks(rotation=45)
    ax.legend(title=df.columns.name if df.columns.name else "",␣
 ↪bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.show()

    img_data = BytesIO()
```

```
    fig.savefig(img_data, format='png', dpi=300)

    if image_name:
        fig.savefig(f"images/{image_name}", format="png", dpi=300)

    img_data.seek(0)
    plt.close(fig)
    return img_data


def write_table_to_excel(worksheet, title, filter_note, table, start_row,␣
 ↪bold_format):
    worksheet.write(start_row, 0, title, bold_format)
    worksheet.write(start_row + 1, 0, filter_note)
    for r_idx, row in enumerate(dataframe_to_rows(table, index=True,␣
 ↪header=True)):
        for c_idx, val in enumerate(row):
            worksheet.write(start_row + 2 + r_idx, c_idx, val)
    return start_row + 2 + table.shape[0] + 3

def write_excel_table_and_chart(ws, title, note, table, start_row, bold_font):
    ws.cell(row=start_row, column=1, value=title).font = bold_font
    ws.cell(row=start_row + 1, column=1, value=note)
    for r_idx, row_data in enumerate(dataframe_to_rows(table.round(1),␣
 ↪index=True, header=True)):
        for c_idx, val in enumerate(row_data):
            ws.cell(row=start_row + 2 + r_idx, column=1 + c_idx, value=val)
    return start_row + 2 + len(table.index) + 4
```

[4]:
```
#
#   Cell 3: Origins table and stacked chart
#

filtered_df = filter_base_data(df)

# Generate origin tables
origin_table = pd.crosstab(index=filtered_df['docYear'],␣
 ↪columns=filtered_df['origin'], dropna=False)
origin_table['Total'] = origin_table.sum(axis=1)

filtered_df['origin_simplified'] = filtered_df['origin'].apply(simplify_origin)
simplified_table = pd.crosstab(index=filtered_df['docYear'],␣
 ↪columns=filtered_df['origin_simplified'], dropna=False)
simplified_table['Total'] = simplified_table.sum(axis=1)
simplified_table_sorted = simplified_table.sort_index()

# Display tables in notebook
```

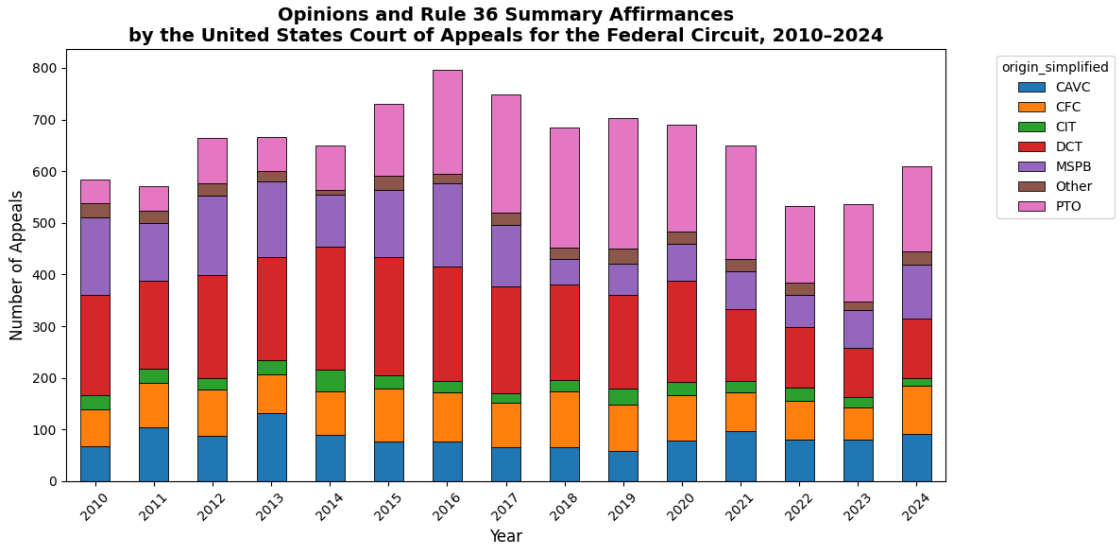
```
display(origin_table)
display(simplified_table_sorted)

# Generate chart
chart_data = simplified_table_sorted.drop(columns='Total')
img_data = generate_chart(
    chart_data,
    "Opinions and Rule 36 Summary Affirmances\nby the United States Court of␣
 ↪Appeals for the Federal Circuit, 2010-2024",
    stacked=True,
    image_name="chart_1.png"
)

# Write to Excel
today_str = datetime.today().strftime('%Y-%m-%d')
excel_filename = f"{today_str} Annual Report.xlsx"

with pd.ExcelWriter(excel_filename, engine='xlsxwriter') as writer:
    workbook = writer.book
    worksheet_origins = workbook.add_worksheet('Origins')
    writer.sheets['Origins'] = worksheet_origins
    bold = workbook.add_format({'bold': True})
    note = "Includes documents where 2010  docYear  2024, docType is 'Opinion'␣
 ↪or 'Rule 36', and Replaced  'Yes'."

    row = write_table_to_excel(worksheet_origins, "Table 1. Full Origin Table␣
 ↪by Year", note, origin_table, 0, bold)
    row = write_table_to_excel(worksheet_origins, "Table 2. Simplified Origin␣
 ↪Table by Year", note, simplified_table_sorted, row, bold)
    worksheet_origins.insert_image(row, 0, 'images/chart_2.png', {'image_data':␣
 ↪img_data})

print(f"  Origins sheet written to: {excel_filename}")
```

origin	BCA	CAVC	CFC	CIT	DCT	DCT-PTO	DOJ	DVA	ITC	MISC	MSPB	OCBD	\
docYear													
2010	10	68	71	28	194	0	0	0	10	6	149	2	
2011	8	104	86	27	170	0	0	0	8	7	113	0	
2012	10	87	91	22	198	1	0	1	7	5	154	0	
2013	8	131	76	27	200	0	0	1	10	0	146	0	
2014	2	90	84	41	238	0	1	0	7	0	101	0	
2015	15	76	104	25	229	0	2	1	9	0	129	1	
2016	9	76	95	22	223	0	2	4	2	1	160	1	
2017	9	66	85	19	207	0	0	3	11	0	119	1	
2018	9	66	107	23	184	0	2	1	5	0	50	5	
2019	10	59	89	32	180	3	1	2	12	0	60	3	
2020	13	78	88	26	196	0	2	2	6	0	72	0	

2021	14	96	76	21	140	0	1	1	8	0	73	0
2022	13	81	75	25	117	0	1	4	6	0	63	0
2023	7	80	62	21	95	0	0	1	6	0	73	1
2024	16	92	92	16	115	0	0	1	6	1	104	1

origin	Other	PTO	Total
docYear			
2010	0	45	583
2011	0	48	571
2012	1	88	665
2013	2	66	667
2014	0	86	650
2015	0	139	730
2016	0	201	796
2017	0	229	749
2018	0	233	685
2019	0	251	702
2020	0	207	690
2021	0	219	649
2022	0	147	532
2023	1	190	537
2024	0	165	609

origin_simplified	CAVC	CFC	CIT	DCT	MSPB	Other	PTO	Total
docYear								
2010	68	71	28	194	149	28	45	583
2011	104	86	27	170	113	23	48	571
2012	87	91	22	198	154	25	88	665
2013	131	76	27	200	146	21	66	667
2014	90	84	41	238	101	10	86	650
2015	76	104	25	229	129	28	139	730
2016	76	95	22	223	160	19	201	796
2017	66	85	19	207	119	24	229	749
2018	66	107	23	184	50	22	233	685
2019	59	89	32	180	60	31	251	702
2020	78	88	26	196	72	23	207	690
2021	96	76	21	140	73	24	219	649
2022	81	75	25	117	63	24	147	532
2023	80	62	21	95	73	16	190	537
2024	92	92	16	115	104	25	165	609




```
Origins sheet written to: 2025-05-30 Annual Report.xlsx
```

[5]:
```
#
#   Cell 4: docType tables and grouped charts
#

base_df = filter_base_data(df)

# Break out by origin
dct_df = base_df[base_df['origin'] == 'DCT'].copy()
pto_df = base_df[base_df['origin'] == 'PTO'].copy()

for subset in [dct_df, pto_df]:
    subset['docYear'] = subset['docYear'].astype(int)

# Tables
dct_table = pd.crosstab(index=dct_df['docYear'], columns=dct_df['docType'],␣
 ↪dropna=False)
pto_table = pd.crosstab(index=pto_df['docYear'], columns=pto_df['docType'],␣
 ↪dropna=False)
all_table = pd.crosstab(index=base_df['docYear'], columns=base_df['docType'],␣
 ↪dropna=False)

# Add totals
dct_table['Total'] = dct_table.sum(axis=1)
pto_table['Total'] = pto_table.sum(axis=1)
all_table['Total'] = all_table.sum(axis=1)
```

```
# Display tables
display(Markdown("**Appeals from District Courts**"))
display(dct_table)
display(Markdown("**Appeals from USPTO**"))
display(pto_table)
display(Markdown("**Appeals from All Origins**"))
display(all_table)

# Chart generator
def chart_to_openpyxl_image(df, title, image_path):
    img_buffer = BytesIO()
    fig, ax = plt.subplots(figsize=(7, 3))
    df.plot(kind='bar', stacked=False, ax=ax, edgecolor='black', linewidth=0.6)
    ax.set_title(title, fontsize=12, fontweight='bold')
    ax.set_xlabel("Year", fontsize=10)
    ax.set_ylabel("Number of Appeals", fontsize=10)
    plt.xticks(rotation=45)
    ax.legend(title=df.columns.name if df.columns.name else "",␣
 ↪bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.savefig(img_buffer, format='png', dpi=125)
    plt.savefig(image_path, format='png', dpi=300)
    plt.show()
    plt.close(fig)
    img_buffer.seek(0)
    return OpenpyxlImage(img_buffer)

# Generate chart images
dct_img = chart_to_openpyxl_image(
    dct_table.drop(columns='Total'),
    "Merits Decision Type\nAppeals from District Courts (2010-2024)",
    "images/chart_3_dct_docType.png"
)
pto_img = chart_to_openpyxl_image(
    pto_table.drop(columns='Total'),
    "Merits Decision Type\nAppeals from District Courts (2010-2024)",
    "images/chart_4_pto_docType.png"
)
all_img = chart_to_openpyxl_image(
    all_table.drop(columns='Total'),
    "Merits Decision Type\nAppeals from All Origins (2010-2024)",
    "images/chart_5_all_docType.png"
)

# Write to Excel
wb = load_workbook(excel_filename)
ws = wb.create_sheet('docType')
```

```
bold = Font(bold=True)
note = "Includes documents where 2010  docYear  2024, docType is 'Opinion' or␣
 ↪'Rule 36', and Replaced  'Yes'."

row_cursor = 1
row_cursor = write_excel_table_and_chart(ws, "Table 3. Appeals from District␣
 ↪Courts", note, dct_table, row_cursor, bold)
dct_img.anchor = f"J2"
ws.add_image(dct_img)

row_cursor = write_excel_table_and_chart(ws, "Table 4. Appeals from USPTO",␣
 ↪note, pto_table, row_cursor, bold)
pto_img.anchor = f"J{row_cursor}"
ws.add_image(pto_img)

row_cursor = write_excel_table_and_chart(ws, "Table 5. Appeals from All␣
 ↪Origins", note, all_table, row_cursor, bold)
all_img.anchor = f"J{row_cursor}"
ws.add_image(all_img)

wb.save(excel_filename)
print(f"  docType sheet written to: {excel_filename}")
```

Appeals from District Courts

docType	Opinion	Rule 36	Total
docYear			
2010	138	56	194
2011	113	57	170
2012	129	69	198
2013	135	65	200
2014	157	81	238
2015	151	78	229
2016	141	82	223
2017	142	65	207
2018	111	73	184
2019	119	61	180
2020	145	51	196
2021	101	39	140
2022	93	24	117
2023	81	14	95
2024	97	18	115

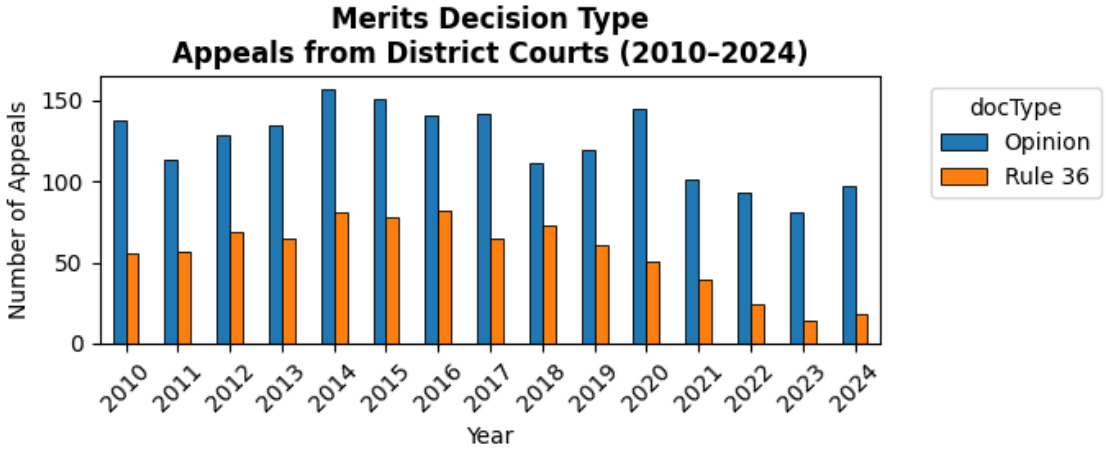
Appeals from USPTO

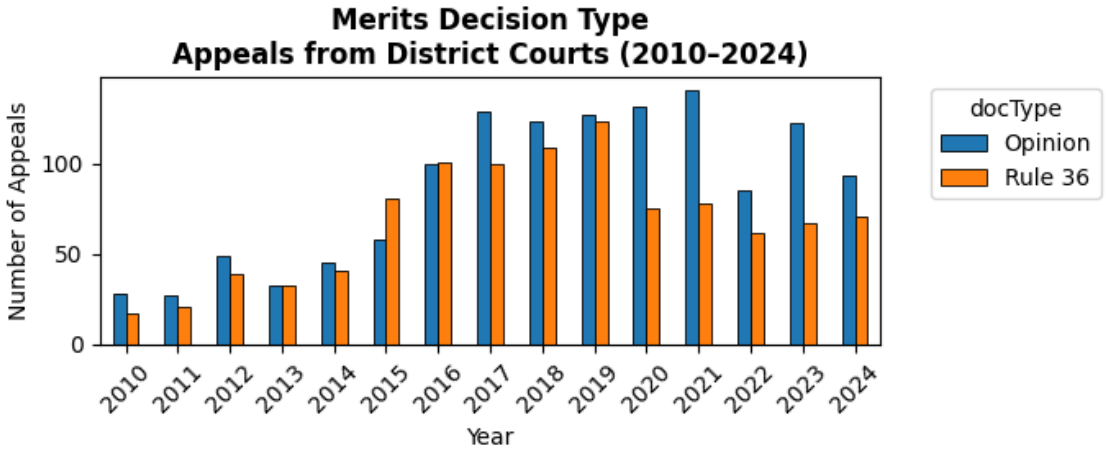
docType	Opinion	Rule 36	Total
docYear			
2010	28	17	45
2011	27	21	48

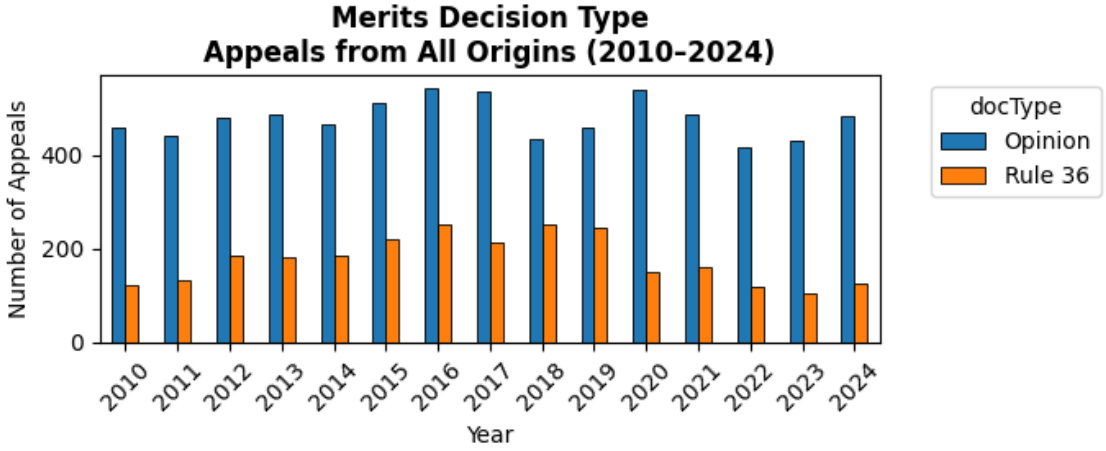
2012	49	39	88
2013	33	33	66
2014	45	41	86
2015	58	81	139
2016	100	101	201
2017	129	100	229
2018	124	109	233
2019	127	124	251
2020	132	75	207
2021	141	78	219
2022	85	62	147
2023	123	67	190
2024	94	71	165

Appeals from All Origins

docType	Opinion	Rule 36	Total
docYear			
2010	460	123	583
2011	440	131	571
2012	481	184	665
2013	486	181	667
2014	465	185	650
2015	511	219	730
2016	543	253	796
2017	534	215	749
2018	434	251	685
2019	458	244	702
2020	539	151	690
2021	488	161	649
2022	415	117	532
2023	432	105	537
2024	482	127	609








```
docType sheet written to: 2025-05-30 Annual Report.xlsx
```

[6]:
```
#
#   Cell 5: Precedential Status by Origin
#

# Create or reuse Excel file path
excel_filename = f"{datetime.today().strftime('%Y-%m-%d')} Annual Report.xlsx"

# Apply base filter for docType='Opinion' only
status_filter = (
    (df['docType'] == 'Opinion') &
    (df['Replaced'] != 'Yes') &
```

```
    (df['docYear'].astype(float) >= 2010) &
    (df['docYear'].astype(float) <= 2024)
)

# Subset by origin
status_dct_df = df[status_filter & (df['origin'] == 'DCT')].copy()
status_pto_df = df[status_filter & (df['origin'] == 'PTO')].copy()
status_all_df = df[status_filter].copy()

# Crosstabs
status_dct_table = pd.crosstab(index=status_dct_df['docYear'],␣
 ↪columns=status_dct_df['PrecedentialStatus'], dropna=False)
status_pto_table = pd.crosstab(index=status_pto_df['docYear'],␣
 ↪columns=status_pto_df['PrecedentialStatus'], dropna=False)
status_all_table = pd.crosstab(index=status_all_df['docYear'],␣
 ↪columns=status_all_df['PrecedentialStatus'], dropna=False)

# Display tables
display(Markdown("**Precedential Status: DCT**"))
display(status_dct_table)
display(Markdown("**Precedential Status: PTO**"))
display(status_pto_table)
display(Markdown("**Precedential Status: All Origins**"))
display(status_all_table)

# Match Cell 4: Image-saving chart function
def chart_to_openpyxl_image(df, title, image_path):
    img_buffer = BytesIO()
    fig, ax = plt.subplots(figsize=(7, 3))  # Smaller figure size
    df.plot(kind='bar', stacked=False, ax=ax, edgecolor='black', linewidth=0.6)
    ax.set_title(title, fontsize=12, fontweight='bold')
    ax.set_xlabel("Year", fontsize=10)
    ax.set_ylabel("Number of Appeals", fontsize=10)
    plt.xticks(rotation=45)
    ax.legend(title=df.columns.name if df.columns.name else "",␣
 ↪bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.savefig(img_buffer, format='png', dpi=125)
    plt.savefig(image_path, format='png', dpi=300)
    plt.show()
    plt.close(fig)
    img_buffer.seek(0)
    return OpenpyxlImage(img_buffer)

# Generate and save charts
chart_dct = chart_to_openpyxl_image(status_dct_table, "Precedential␣
 ↪Status\nDistrict Court Appeals (2010-2024)", "images/chart_5_dct_status.png")
```

```
chart_pto = chart_to_openpyxl_image(status_pto_table, "Precedential␣
 ↪Status\nUSPTO Appeals (2010-2024)", "images/chart_6_pto_status.png")
chart_all = chart_to_openpyxl_image(status_all_table, "Precedential Status\nAll␣
 ↪Origins (2010-2024)", "images/chart_7_all_status.png")

# Write to Excel
with pd.ExcelWriter(excel_filename, engine='openpyxl', mode='a') as writer:
    workbook = writer.book
    worksheet_status = workbook.create_sheet('Precedential Status')
    bold_font = Font(bold=True)
    row_cursor = 1

    status_sections = [
        ("Table 5. Appeals from District Courts", status_dct_table, "origin =␣
 ↪'DCT'", chart_dct),
        ("Table 6. Appeals from USPTO", status_pto_table, "origin = 'PTO'",␣
 ↪chart_pto),
        ("Table 7. Appeals from All Origins", status_all_table, "no origin␣
 ↪filter", chart_all)
    ]

    status_filter_note = (
        "Includes documents where 2010  docYear  2024, docType = 'Opinion',␣
 ↪Replaced  'Yes', and {origin_desc}."
    )

    for title, table, origin_desc, chart_img in status_sections:
        worksheet_status.cell(row=row_cursor, column=1, value=title).font =␣
 ↪bold_font
        row_cursor += 1
        worksheet_status.cell(row=row_cursor, column=1,␣
 ↪value=status_filter_note.format(origin_desc=origin_desc))
        row_cursor += 1

        start_row = row_cursor
        max_col_width = table.shape[1] + 3

        for r in dataframe_to_rows(table, index=True, header=True):
            worksheet_status.append(r)
            row_cursor += 1

        chart_img.anchor = f"{chr(65 + max_col_width)}{start_row}"
        worksheet_status.add_image(chart_img)

        row_cursor += 10
```

```
print(f"  Precedential Status sheet written to: {excel_filename}")
```

Precedential Status: DCT

PrecedentialStatus	Nonprecedential	Precedential
docYear		
2010	38	100
2011	42	71
2012	32	97
2013	49	86
2014	54	103
2015	50	101
2016	52	89
2017	70	72
2018	46	65
2019	52	67
2020	77	68
2021	54	47
2022	41	52
2023	47	34
2024	47	50

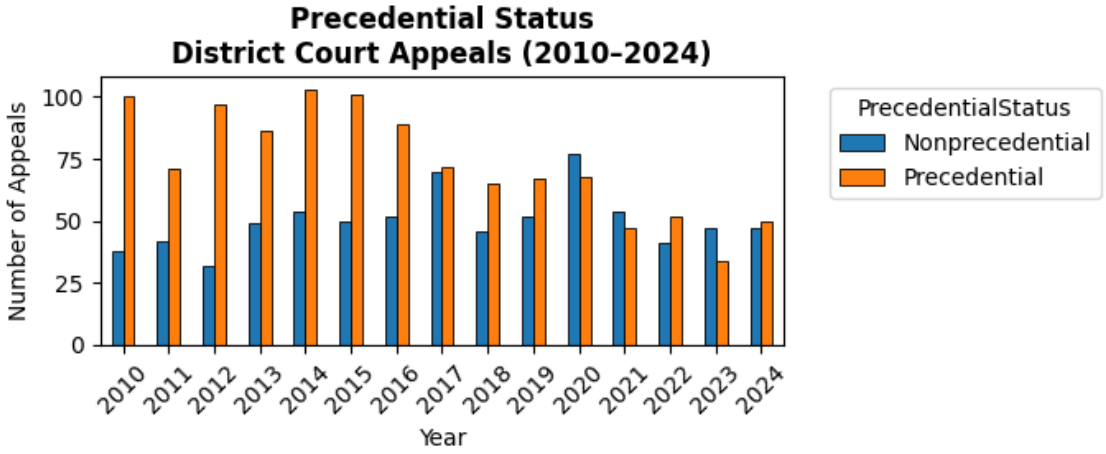
Precedential Status: PTO

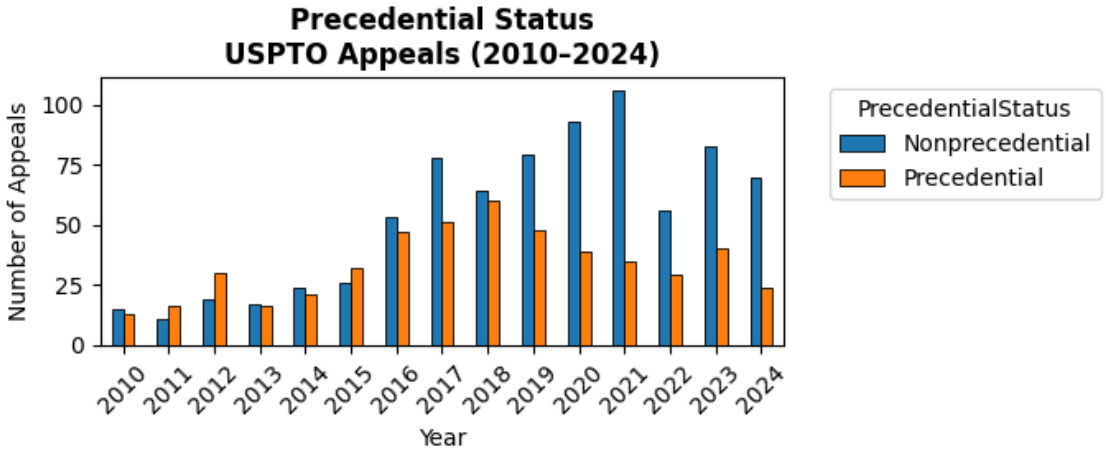
PrecedentialStatus	Nonprecedential	Precedential
docYear		
2010	15	13
2011	11	16
2012	19	30
2013	17	16
2014	24	21
2015	26	32
2016	53	47
2017	78	51
2018	64	60
2019	79	48
2020	93	39
2021	106	35
2022	56	29
2023	83	40
2024	70	24

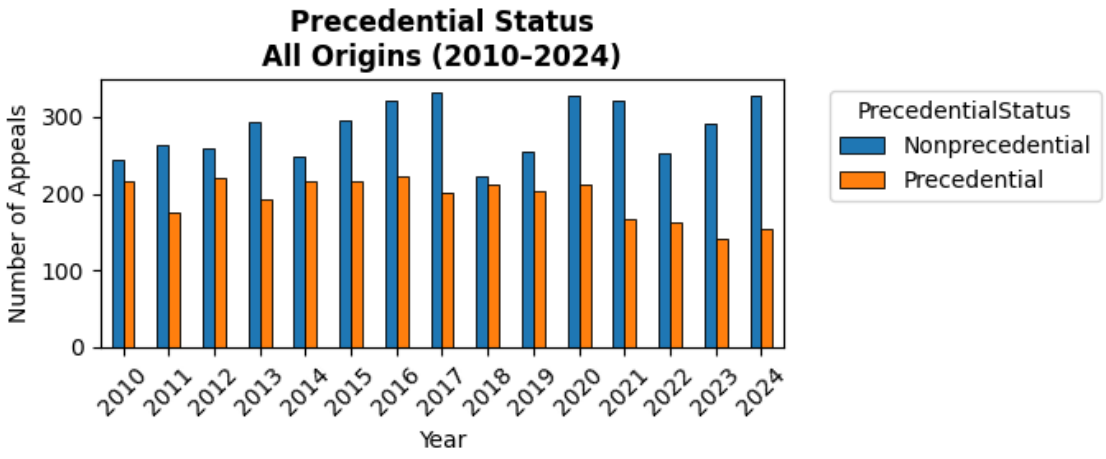
Precedential Status: All Origins

PrecedentialStatus	Nonprecedential	Precedential
docYear		
2010	244	216
2011	264	176
2012	260	221
2013	293	193

2014	248	217
2015	295	216
2016	321	222
2017	332	202
2018	222	212
2019	255	203
2020	327	212
2021	321	167
2022	253	162
2023	290	142
2024	328	154








```
Precedential Status sheet written to: 2025-05-30 Annual Report.xlsx
```

[7]:
```
#
#   Cell 6: Merits Dispositions by Origin (Percentages)
#

valid_doc_types = ['Opinion', 'Rule 36']
valid_dispositions = ['Affirmed', 'Affirmed-in-part', 'Reversed', 'Vacated']

base_df = filter_base_data(df)
base_df = base_df[
    base_df['docType'].isin(valid_doc_types) &
    base_df['DispGeneral'].isin(valid_dispositions)
].copy()

dct_df = base_df[base_df['origin'] == 'DCT'].copy()
pto_df = base_df[base_df['origin'] == 'PTO'].copy()

# Crosstab by docYear × DispGeneral
dct_disp_table = pd.crosstab(index=dct_df['docYear'],␣
 ↪columns=dct_df['DispGeneral'], dropna=False)
pto_disp_table = pd.crosstab(index=pto_df['docYear'],␣
 ↪columns=pto_df['DispGeneral'], dropna=False)

# Normalize rows to percentages
dct_disp_pct = dct_disp_table.div(dct_disp_table.sum(axis=1), axis=0) * 100
pto_disp_pct = pto_disp_table.div(pto_disp_table.sum(axis=1), axis=0) * 100

# Add combined column: Affirmed in full or in part
dct_disp_pct['Affirmed in full or in part'] = (
    dct_disp_pct.get('Affirmed', 0) + dct_disp_pct.get('Affirmed-in-part', 0)
```

```
)
pto_disp_pct['Affirmed in full or in part'] = (
    pto_disp_pct.get('Affirmed', 0) + pto_disp_pct.get('Affirmed-in-part', 0)
)

# Ensure combined column is last
def move_column_to_end(df, column_name):
    cols = [c for c in df.columns if c != column_name] + [column_name]
    return df[cols]

dct_disp_pct = move_column_to_end(dct_disp_pct, 'Affirmed in full or in part')
pto_disp_pct = move_column_to_end(pto_disp_pct, 'Affirmed in full or in part')

# Display tables
display(Markdown("**Disposition % by Year: DCT**"))
display(dct_disp_pct.round(1))
display(Markdown("**Disposition % by Year: PTO**"))
display(pto_disp_pct.round(1))

# Use same charting logic as Cell 4
def chart_to_openpyxl_image(df, title, image_path):
    img_buffer = BytesIO()
    fig, ax = plt.subplots(figsize=(7, 3))
    df.plot(kind='bar', stacked=True, ax=ax, edgecolor='black', linewidth=0.6)
    ax.set_title(title, fontsize=12, fontweight='bold')
    ax.set_xlabel("Year", fontsize=10)
    ax.set_ylabel("Percentage of Appeals", fontsize=10)
    plt.xticks(rotation=45)
    ax.legend(title="General Disposition", bbox_to_anchor=(1.05, 1), loc='upper␣
 ↪left')  #  Updated legend title
    plt.tight_layout()
    plt.savefig(img_buffer, format='png', dpi=125)
    plt.savefig(image_path, format='png', dpi=300)
    plt.show()
    plt.close(fig)
    img_buffer.seek(0)
    return OpenpyxlImage(img_buffer)

# Save charts to 'images/' and prepare for Excel
dct_img = chart_to_openpyxl_image(
    dct_disp_pct.drop(columns=['Affirmed in full or in part']),
    "Disposition Outcomes by Year\nDistrict Court Appeals (Percent)",
    "images/chart_8_dct_dispositions.png"
)
pto_img = chart_to_openpyxl_image(
    pto_disp_pct.drop(columns=['Affirmed in full or in part']),
    "Disposition Outcomes by Year\nUSPTO Appeals (Percent)",
```

```
    "images/chart_9_pto_dispositions.png"
)

# Write to Excel
excel_filename = f"{datetime.today().strftime('%Y-%m-%d')} Annual Report.xlsx"
wb = load_workbook(excel_filename)
ws = wb.create_sheet('Dispositions')
bold = Font(bold=True)
note = "Includes documents where 2010  docYear  2024, docType is 'Opinion' or␣
 ↪'Rule 36', Replaced  'Yes', and {origin_desc}. Limited to dispositions:␣
 ↪Affirmed, Affirmed-in-part, Reversed, Vacated."

row_cursor = 1

def write_excel_table_and_chart(ws, title, note, table, start_row):
    ws.cell(row=start_row, column=1, value=title).font = bold
    ws.cell(row=start_row + 1, column=1, value=note)
    for r_idx, row_data in enumerate(dataframe_to_rows(table.round(1),␣
 ↪index=True, header=True)):
        for c_idx, val in enumerate(row_data):
            ws.cell(row=start_row + 2 + r_idx, column=1 + c_idx, value=val)
    return start_row + 2 + len(table.index) + 4

row_cursor = write_excel_table_and_chart(ws, "Table 8. Dispositions from␣
 ↪District Courts", note.format(origin_desc="origin = 'DCT'"), dct_disp_pct,␣
 ↪row_cursor)
row_cursor = write_excel_table_and_chart(ws, "Table 9. Dispositions from␣
 ↪USPTO", note.format(origin_desc="origin = 'PTO'"), pto_disp_pct, row_cursor)

# Add charts to Excel
dct_img.anchor = f"J2"
ws.add_image(dct_img)
pto_img.anchor = f"J{row_cursor}"
ws.add_image(pto_img)

wb.save(excel_filename)
print(f"  Dispositions sheet with % stacked charts saved to: {excel_filename}")
```

Disposition % by Year: DCT

DispGeneral	Affirmed	Affirmed-in-part	Reversed	Vacated	\
docYear					
2010	63.0	17.7	13.0	6.2	
2011	59.5	23.2	11.9	5.4	
2012	66.5	18.8	8.6	6.1	
2013	59.1	20.2	16.6	4.1	
2014	67.9	16.2	7.7	8.1	
2015	65.5	14.6	12.4	7.5	

2016	76.4	11.8	6.4	5.5
2017	75.7	9.9	9.4	5.0
2018	72.0	13.7	8.2	6.0
2019	68.6	13.1	9.1	9.1
2020	71.8	12.8	8.2	7.2
2021	79.7	10.5	6.0	3.8
2022	56.8	18.0	17.1	8.1
2023	74.2	11.8	7.5	6.5
2024	64.5	11.8	14.5	9.1

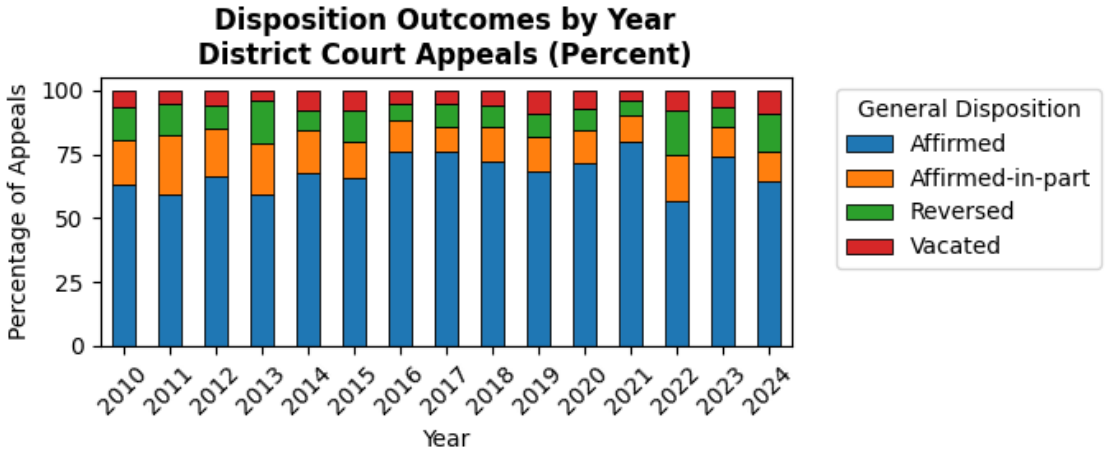
DispGeneral	Affirmed in full or in part
docYear	
2010	80.7
2011	82.7
2012	85.3
2013	79.3
2014	84.2
2015	80.1
2016	88.2
2017	85.6
2018	85.7
2019	81.7
2020	84.6
2021	90.2
2022	74.8
2023	86.0
2024	76.4

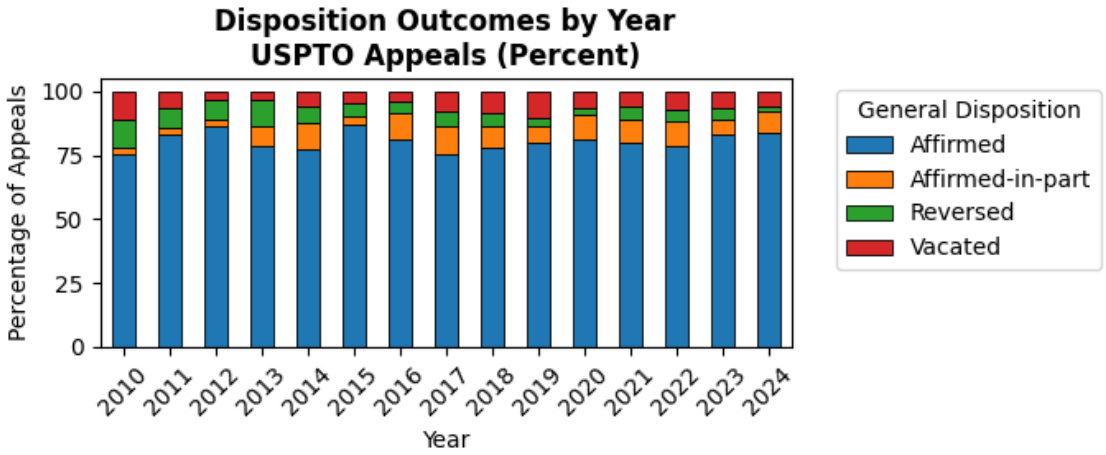
Disposition % by Year: PTO

DispGeneral	Affirmed	Affirmed-in-part	Reversed	Vacated	\
docYear					
2010	75.6	2.2	11.1	11.1	
2011	83.3	2.1	8.3	6.2	
2012	86.4	2.3	8.0	3.4	
2013	78.5	7.7	10.8	3.1	
2014	77.1	10.8	6.0	6.0	
2015	86.8	3.7	5.1	4.4	
2016	80.9	10.3	4.6	4.1	
2017	75.6	10.7	5.8	8.0	
2018	77.9	8.7	4.8	8.7	
2019	79.7	6.9	2.8	10.6	
2020	81.4	9.3	2.9	6.4	
2021	79.7	9.0	5.2	6.1	
2022	78.3	9.8	4.9	7.0	
2023	83.3	5.4	4.8	6.5	
2024	83.8	8.1	2.5	5.6	

DispGeneral	Affirmed in full or in part

docYear	
2010	77.8
2011	85.4
2012	88.6
2013	86.2
2014	88.0
2015	90.4
2016	91.2
2017	86.2
2018	86.6
2019	86.6
2020	90.7
2021	88.7
2022	88.1
2023	88.7
2024	91.9





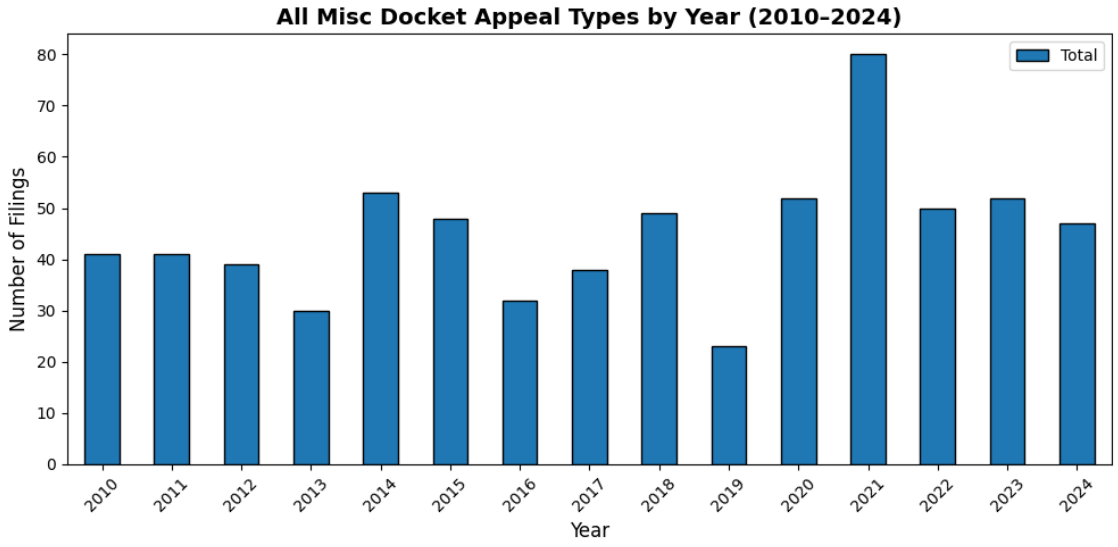
```
 Dispositions sheet with % stacked charts saved to: 2025-05-30 Annual
Report.xlsx
```

[8]:
```
#
#   Cell 7: All Misc Docket Appeal Types by Year (Excl. NaN, Add Total)
#

# Clean docYear and docType
df['docYear'] = pd.to_numeric(df['docYear'], errors='coerce')
df['docType'] = df['docType'].str.strip()
filtered_df = df[
    (df['docYear'] > 2009) &
    (df['docYear'] <= 2024) &
    (df['docType'] != "Errata")
].copy()
filtered_df['docYear'] = filtered_df['docYear'].astype(int)

# Crosstab table, drop NaN columns, add Totals column
misc_tab_cleaned = pd.crosstab(
    index=filtered_df['docYear'],
    columns=filtered_df['Misc_Docket_Appeal_Type'],
    dropna=True
)
misc_tab_cleaned['Total'] = misc_tab_cleaned.sum(axis=1)

# Show in notebook
display(Markdown("**All Misc Docket Appeal Types by Year (Cleaned)**"))
display(misc_tab_cleaned)

# Use consistent charting and image export function
def chart_to_openpyxl_image(df, title, image_path):
    img_buffer = BytesIO()
    fig, ax = plt.subplots(figsize=(10, 5))
    df.plot(kind='bar', ax=ax, edgecolor='black')
    ax.set_title(title, fontsize=14, fontweight='bold')
    ax.set_xlabel("Year", fontsize=12)
    ax.set_ylabel("Number of Filings", fontsize=12)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig(img_buffer, format='png', dpi=125)
    plt.savefig(image_path, format='png', dpi=300)
    plt.show()
    plt.close(fig)
    img_buffer.seek(0)
    return OpenpyxlImage(img_buffer)
```

```
# Save image and create openpyxl image
chart_img = chart_to_openpyxl_image(
    misc_tab_cleaned.drop(columns="Total").sum(axis=1).to_frame(name="Total"),
    "All Misc Docket Appeal Types by Year (2010-2024)",
    "images/chart_10_misc_types.png"
)

# Excel export setup
excel_filename = f"{datetime.today().strftime('%Y-%m-%d')} Annual Report.xlsx"
if os.path.exists(excel_filename):
    wb = load_workbook(excel_filename)
else:
    wb = Workbook()
    wb.remove(wb.active)

ws = wb.create_sheet('All Misc Appeal Types')
bold = Font(bold=True)

note = "Includes documents where 2010  docYear  2024 and docType  'Errata'.␣
 ↪NaN appeal types excluded. Includes 'Total' column."

def write_excel_table_and_chart(ws, title, note, table, start_row):
    ws.cell(row=start_row, column=1, value=title).font = bold
    ws.cell(row=start_row + 1, column=1, value=note)
    for r_idx, row_data in enumerate(dataframe_to_rows(table.round(1),␣
 ↪index=True, header=True)):
        for c_idx, val in enumerate(row_data):
            ws.cell(row=start_row + 2 + r_idx, column=1 + c_idx, value=val)
    return start_row + 2 + len(table.index) + 4

# Write to Excel
row_cursor = 1
row_cursor = write_excel_table_and_chart(
    ws,
    "Table 10. All Misc Docket Appeal Types by Year (Excl. NaN, With Total)",
    note,
    misc_tab_cleaned,
    row_cursor
)

# Add image to Excel
chart_img.anchor = f"J2"
ws.add_image(chart_img)
wb.save(excel_filename)

print(f"  Excel worksheet written to: {excel_filename}")
```

All Misc Docket Appeal Types by Year (Cleaned)

Misc_Docket_Appeal_Type	Other	Petition for permission to appeal	Writ of Mandamus	Total
docYear				
2010	2	6	33	41
2011	3	4	34	41
2012	1	3	35	39
2013	1	3	26	30
2014	0	2	51	53
2015	1	2	45	48
2016	0	5	27	32
2017	0	3	35	38
2018	1	7	41	49
2019	1	1	21	23
2020	0	8	44	52
2021	0	6	74	80
2022	1	4	45	50
2023	0	2	50	52
2024	1	4	42	47



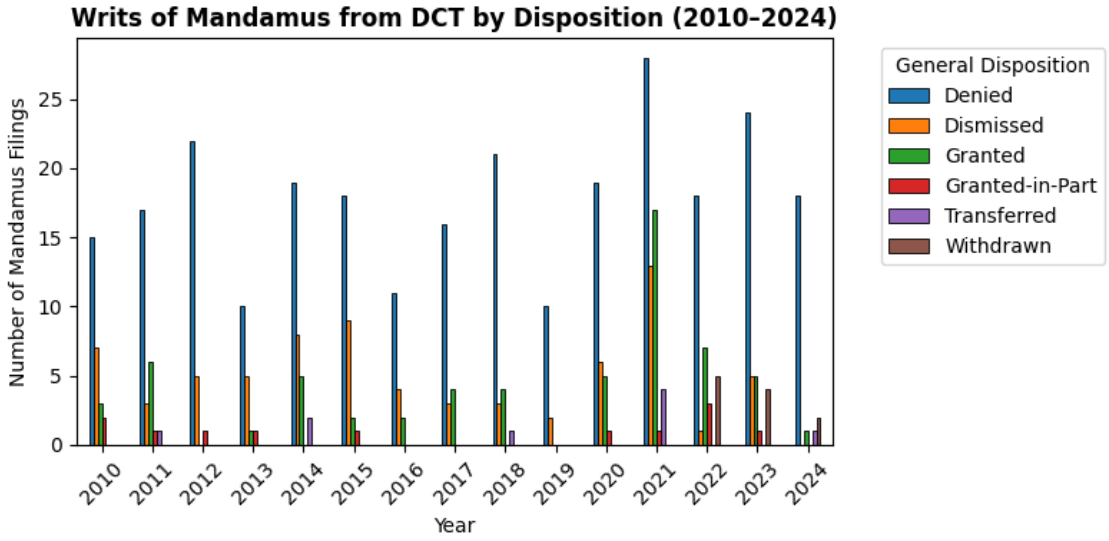

```
Excel worksheet written to: 2025-05-30 Annual Report.xlsx
```

[9]:
```
#
#   Cell 8: DCT Writs of Mandamus by Disposition (2010-2024)
#

# Filter dataset for DCT Mandamus filings
mandamus_df = df[
    (df['docYear'].astype(float) > 2009) &
    (df['docYear'].astype(float) <= 2024) &
    (df['docType'] != "Errata") &
    (df['Misc_Docket_Appeal_Type'] == "Writ of Mandamus") &
    (df['origin'] == "DCT")
].copy()
mandamus_df['docYear'] = mandamus_df['docYear'].astype(int)

# Crosstab by docYear × DispGeneral
mandamus_disp_table = pd.crosstab(index=mandamus_df['docYear'],␣
 ↪columns=mandamus_df['DispGeneral'], dropna=False)

# Display table inline
print("  DCT Writs of Mandamus by Disposition")
display(mandamus_disp_table.round(1))

# Use standard export pattern
def chart_to_openpyxl_image(df, title, image_path):
    img_buffer = BytesIO()
    fig, ax = plt.subplots(figsize=(8, 4))
    df.plot(kind='bar', stacked=False, ax=ax, edgecolor='black', linewidth=0.6)
```

```
    ax.set_title(title, fontsize=12, fontweight='bold')
    ax.set_xlabel("Year", fontsize=10)
    ax.set_ylabel("Number of Mandamus Filings", fontsize=10)
    plt.xticks(rotation=45)
    ax.legend(title="General Disposition", bbox_to_anchor=(1.05, 1), loc='upper␣
 ↪left')
    plt.tight_layout()
    plt.savefig(img_buffer, format='png', dpi=125)
    plt.savefig(image_path, format='png', dpi=300)
    plt.show()
    plt.close(fig)
    img_buffer.seek(0)
    return OpenpyxlImage(img_buffer)

# Save chart and create image
mandamus_img = chart_to_openpyxl_image(
    mandamus_disp_table,
    "Writs of Mandamus from DCT by Disposition (2010-2024)",
    "images/chart_11_dct_mandamus.png"
)

# Append to Excel
excel_filename = f"{datetime.today().strftime('%Y-%m-%d')} Annual Report.xlsx"
wb = load_workbook(excel_filename)
ws = wb.create_sheet("Mandamus DCT Disps")
bold = Font(bold=True)

note = (
    "Includes documents where 2010  docYear  2024, docType  'Errata', "
    "Misc_Docket_Appeal_Type = 'Writ of Mandamus', and origin = 'DCT'."
)

def write_excel_table_and_chart(ws, title, note, table, start_row):
    ws.cell(row=start_row, column=1, value=title).font = bold
    ws.cell(row=start_row + 1, column=1, value=note)
    for r_idx, row_data in enumerate(dataframe_to_rows(table.round(1),␣
 ↪index=True, header=True)):
        for c_idx, val in enumerate(row_data):
            ws.cell(row=start_row + 2 + r_idx, column=1 + c_idx, value=val)
    return start_row + 2 + len(table.index) + 4

# Write table and image
row_cursor = 1
row_cursor = write_excel_table_and_chart(
    ws,
    "Table 11. Dispositions for DCT Writs of Mandamus by Year",
    note,
```

```
    mandamus_disp_table,
    row_cursor
)
mandamus_img.anchor = f"J2"
ws.add_image(mandamus_img)

wb.save(excel_filename)
print(f"  Mandamus DCT sheet written to: {excel_filename}")
```

```
  DCT Writs of Mandamus by Disposition

DispGeneral  Denied  Dismissed  Granted  Granted-in-Part  Transferred  \
docYear
2010              15          7        3                2            0
2011              17          3        6                1            1
2012              22          5        0                1            0
2013              10          5        1                1            0
2014              19          8        5                0            2
2015              18          9        2                1            0
2016              11          4        2                0            0
2017              16          3        4                0            0
2018              21          3        4                0            1
2019              10          2        0                0            0
2020              19          6        5                1            0
2021              28         13       17                1            4
2022              18          1        7                3            0
2023              24          5        5                1            0
2024              18          0        1                0            1

DispGeneral  Withdrawn
docYear
2010                 0
2011                 0
2012                 0
2013                 0
2014                 0
2015                 0
2016                 0
2017                 0
2018                 0
2019                 0
2020                 0
2021                 0
2022                 5
2023                 4
2024                 2
```




```
Mandamus DCT sheet written to: 2025-05-30 Annual Report.xlsx
```

[10]:

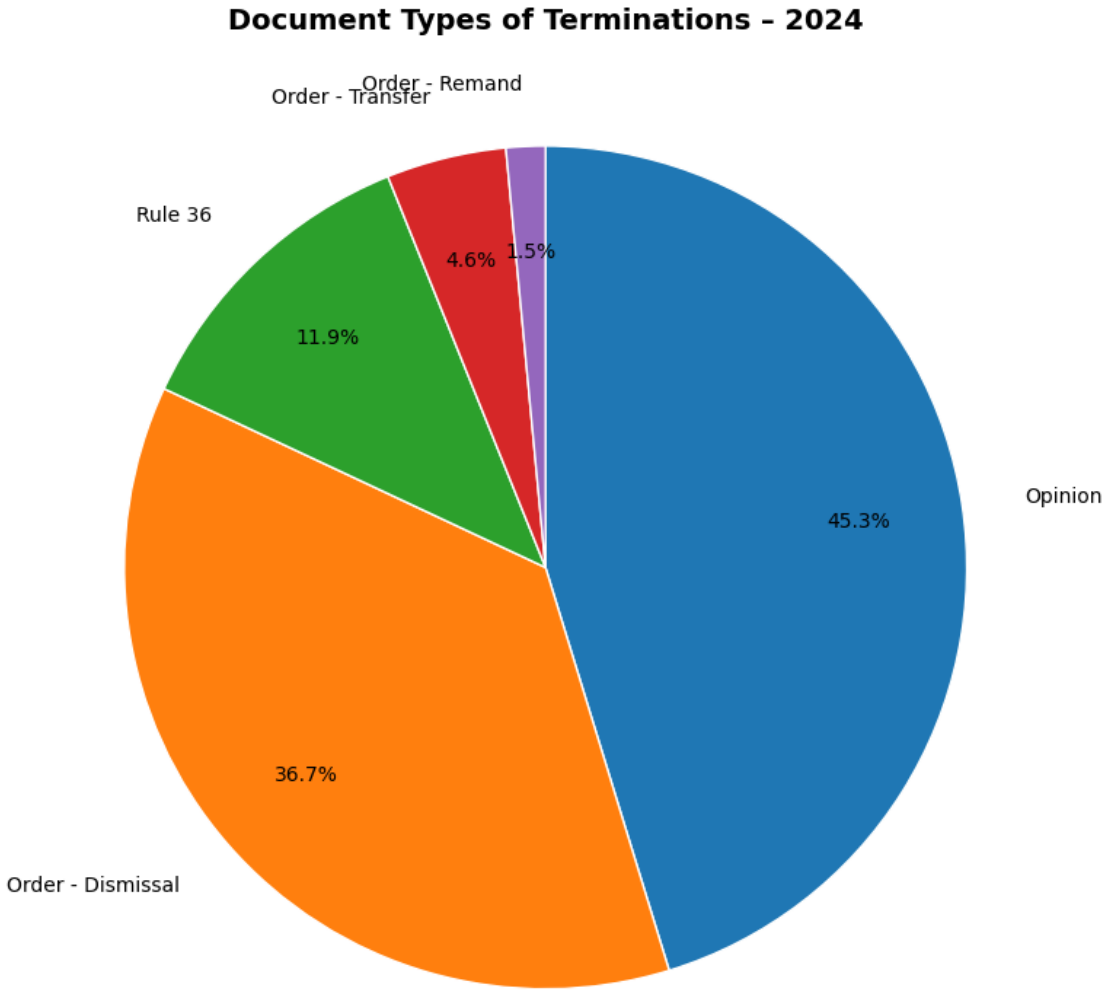
```
#
#   Cell 9: Termination Types by docType and Disposition for 2024
#

# Filter records for 2024 and specified docTypes, excluding replaced documents
df['docYear'] = pd.to_numeric(df['docYear'], errors='coerce')
termination_df = df[
    (df['docYear'] == 2024) &
    (df['Replaced'] != "Yes") &
    (df['Misc_Docket'] == "False") &
    (df['docType'].isin([
        "Opinion",
        "Order - Dismissal",
        "Order - Remand",
        "Order - Transfer",
        "Rule 36"
    ]))
].copy()

# Table: DispGeneral × docType
termination_table = pd.crosstab(index=termination_df['DispGeneral'],␣
 ↪columns=termination_df['docType'], dropna=False)

# Display table
print("  Termination Table by Disposition and docType for 2024")
display(termination_table.round(1))
```

```python
# Pie chart of docType with standardized export logic
def chart_to_openpyxl_image_for_pie(series, title, image_path):
    img_buffer = BytesIO()
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.pie(
        series,
        labels=series.index,
        autopct='%1.1f%%',
        startangle=90,
        counterclock=False,
        wedgeprops={'edgecolor': 'white', 'linewidth': 1},
        labeldistance=1.15,
        pctdistance=0.75
    )
    ax.set_title(title, fontsize=14, fontweight='bold')
    plt.tight_layout()
    plt.savefig(img_buffer, format='png', dpi=125)
    plt.savefig(image_path, format='png', dpi=300)
    plt.show()
    plt.close(fig)
    img_buffer.seek(0)
    return OpenpyxlImage(img_buffer)

doc_type_counts = termination_df['docType'].value_counts().
 ↪sort_values(ascending=False)
doc_type_img = chart_to_openpyxl_image_for_pie(
    doc_type_counts,
    "Document Types of Terminations - 2024",
    "images/chart_12_doc_types_2024.png"
)

# Write to Excel workbook
excel_filename = f"{datetime.today().strftime('%Y-%m-%d')} Annual Report.xlsx"
wb = load_workbook(excel_filename)
ws = wb.create_sheet("Terminations 2024")
bold = Font(bold=True)

note = (
    "Includes documents from 2024 where Replaced  'Yes' and "
    "docType is one of: Opinion, Rule 36, Order - Dismissal, "
    "Order - Remand, Order - Transfer."
)

def write_openpyxl_table(ws, title, note, table, start_row):
    ws.cell(row=start_row, column=1, value=title).font = bold
    ws.cell(row=start_row + 1, column=1, value=note)
```

```
    for r_idx, row_data in enumerate(dataframe_to_rows(table.round(1),␣
 ↪index=True, header=True)):
        for c_idx, val in enumerate(row_data):
            ws.cell(row=start_row + 2 + r_idx, column=1 + c_idx, value=val)
    return start_row + 2 + len(table.index) + 4

# Write table and chart to Excel
row_cursor = 1
row_cursor = write_openpyxl_table(
    ws,
    "Table 12. Disposition by Document Type - 2024",
    note,
    termination_table,
    row_cursor
)

doc_type_img.anchor = f"J2"
ws.add_image(doc_type_img)

wb.save(excel_filename)
print(f"  Terminations 2024 worksheet written to: {excel_filename}")
```

```
 Termination Table by Disposition and docType for 2024
docType           Opinion  Order - Dismissal  Order - Remand  \
DispGeneral
Affirmed              308                  0               0
Affirmed-in-part       44                  0               0
Dismissed              50                391               0
Granted                 2                  0               1
Other                   7                  0               0
Remanded                0                  0              14
Reversed               36                  0               0
Transferred             0                  0               0
Vacated                35                  0               0
NaN                     0                  0               1

docType           Order - Transfer  Rule 36
DispGeneral
Affirmed                         0      127
Affirmed-in-part                 0        0
Dismissed                        0        0
Granted                          0        0
Other                            0        0
Remanded                         0        0
Reversed                         0        0
Transferred                     49        0
Vacated                          0        0
```

```
NaN                                   0          0
```




```
Terminations 2024 worksheet written to: 2025-05-30 Annual Report.xlsx
```

[ ]: