In [373…
```
# Importing and configuring essential data libraries for analysis.
# These settings allow us to view all columns and a large number of rows for b
import pandas as pd
import numpy as np

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 200)
```

IMPORTING DATASETS AND CLEANING

In [374…
```
# Importing dataset 1 (Employer Plans) from CSV
df1 = pd.read_csv("/content/Employer Plans post-COVID WFH - Sheet1.csv")

# Cleaning column names: removing spaces, lowering case, replacing spaces with
df1.columns = df1.columns.str.strip().str.lower().str.replace(" ", "_")

# Keeping only rows where 'date' contains digits, then converting date column
df1 = df1[df1['date'].astype(str).str.contains(r"\d")]
df1['date'] = pd.to_datetime(df1['date'], errors='coerce')

# Identifying numeric and categorical columns separately
num_cols = df1.select_dtypes(include=[np.number]).columns
cat_cols = df1.select_dtypes(include=['object']).columns

# Removing rows that have all numeric values missing and dropping duplicates
df1 = df1.dropna(subset=num_cols, how='all').drop_duplicates()

# Filling missing numeric values with the median; filling categorical values w
df1[num_cols] = df1[num_cols].fillna(df1[num_cols].median())
df1[cat_cols] = df1[cat_cols].fillna("Unknown")

#Displaying the cleaned dataset
df1.head()
```

Out[374…

	date	wfh_days_postcovid_planmad	wfh_days_postcovid_plan_emad	wfh
1	2020-07-01	1.06	Unknown	
2	2020-08-01	1.06	1.58	
3	2020-09-01	1.09	1.56	
7	2021-01-01	1.06	1.58	
8	2021-02-01	1.12	1.69	

In [375…
```
# Loading Dataset 2 and cleaning column names + date format
df2 = pd.read_csv("/content/Full Remote-Hybrid-Full Onsite - Sheet1.csv")

df2.columns = df2.columns.str.strip().str.lower().str.replace(" ", "_")
df2 = df2[df2['date'].astype(str).str.contains(r"\d")]
df2['date'] = pd.to_datetime(df2['date'], errors='coerce')
```

```python
# Handling missing values and removing duplicates
num_cols = df2.select_dtypes(include=[np.number]).columns
cat_cols = df2.select_dtypes(include=['object']).columns

df2 = df2.dropna(subset=num_cols, how='all').drop_duplicates()

# Filling missing numeric + categorical values
df2[num_cols] = df2[num_cols].fillna(df2[num_cols].median())
df2[cat_cols] = df2[cat_cols].fillna("Unknown")

df2.head()
```

Out[375…

	date	full_onsite_curr	hybrid_curr	full_remote_curr	full_onsite_curr_e
0	2021-11-01	54.4	30.4	15.3	30.9
1	2021-12-01	53.4	32.6	14.0	29.1
2	2022-01-01	56.8	25.5	17.8	32.1
3	2022-02-01	59.5	22.8	17.7	31.1
4	2022-03-01	57.3	27.2	15.5	30.5

In [376…

```python
# Loading Dataset 3 and standardizing column names + fixing date format
df3 = pd.read_csv("/content/LEGACY WFH series - Sheet1.csv")

df3.columns = df3.columns.str.strip().str.lower().str.replace(" ", "_")
df3 = df3[df3['date'].astype(str).str.contains(r"\d")]
df3['date'] = pd.to_datetime(df3['date'], errors='coerce')

# Removing rows with all missing numeric values and duplicates
num_cols = df3.select_dtypes(include=[np.number]).columns
cat_cols = df3.select_dtypes(include=['object']).columns

df3 = df3.dropna(subset=num_cols, how='all').drop_duplicates()

# Filling missing values for numeric and categorical columns
df3[num_cols] = df3[num_cols].fillna(df3[num_cols].median())
df3[cat_cols] = df3[cat_cols].fillna("Unknown")

df3.head()
```

Out[376…

	date	atus_wfhdays	wfhcovid	wfh_days_postcovid	notes	license
0	2020-03-01	4.8	42.3	31.8	Pre-COVID value is authors' estimate using dat...	Copyright 2024 by Jose Maria Barrero, Nicholas...
1	2020-05-01	4.8	61.5	31.8	Unknown	Unknown
2	2020-07-01	4.8	51.0	31.8	Unknown	Unknown
3	2020-08-01	4.8	48.3	31.8	Unknown	Unknown
4	2020-09-01	4.8	44.3	31.8	Unknown	Unknown

In [377…

```
df5 = pd.read_csv("/content/WFH 1965 - present - Sheet1.csv")

df5.columns = df5.columns.str.strip().str.lower().str.replace(" ", "_")
df5 = df5[df5['date'].astype(str).str.contains(r"\d")]
df5['date'] = pd.to_datetime(df5['date'], errors='coerce')

num_cols = df5.select_dtypes(include=[np.number]).columns
cat_cols = df5.select_dtypes(include=['object']).columns

df5 = df5.dropna(subset=num_cols, how='all').drop_duplicates()

df5[num_cols] = df5[num_cols].fillna(df5[num_cols].median())
df5[cat_cols] = df5[cat_cols].fillna("Unknown")

df5.head()
```

Out[377…

	date	wfh_share	source_historical_series	fullremote_hist	source_fullrer
0	1965-01-01	0.4	AHTUS	3.0	
1	1975-01-01	0.6	AHTUS	3.0	
2	1993-01-01	2.5	AHTUS	3.0	
3	1995-01-01	2.2	AHTUS	3.0	
4	1998-01-01	2.9	AHTUS	3.0	

In [378…

```
df6 = pd.read_csv("/content/WFH by city - Sheet1.csv")
```

```
df6.columns = df6.columns.str.strip().str.lower().str.replace(" ", "_")
df6 = df6[df6['date'].astype(str).str.contains(r"\d")]
df6['date'] = pd.to_datetime(df6['date'], errors='coerce')

num_cols = df6.select_dtypes(include=[np.number]).columns
cat_cols = df6.select_dtypes(include=['object']).columns

df6 = df6.dropna(subset=num_cols, how='all').drop_duplicates()

df6[num_cols] = df6[num_cols].fillna(df6[num_cols].median())
df6[cat_cols] = df6[cat_cols].fillna("Unknown")

df6.head()
```

Out[378…

	date	wfhcovid_series_top10_ma6_	wfhcovid_series_11to50_ma6_	wfhco
0	2020-10-01	51.1	44.0	
1	2020-11-01	42.2	37.1	
2	2020-12-01	41.7	35.8	
3	2021-01-01	40.1	34.8	
4	2021-02-01	40.6	35.4	

In [379…

```
df7 = pd.read_csv("/content/WFH Rates by Industry - Sheet1.csv")

df7.columns = df7.columns.str.strip().str.lower().str.replace(" ", "_")
df7 = df7[df7['date'].astype(str).str.contains(r"\d")]
df7['date'] = pd.to_datetime(df7['date'], errors='coerce')

num_cols = df7.select_dtypes(include=[np.number]).columns
cat_cols = df7.select_dtypes(include=['object']).columns

df7 = df7.dropna(subset=num_cols, how='all').drop_duplicates()

df7[num_cols] = df7[num_cols].fillna(df7[num_cols].median())
df7[cat_cols] = df7[cat_cols].fillna("Unknown")

df7.head()
```

Out[379…

	date	wfh_fracmat_arts_entertain	wfh_fracmat_education	wfh_fracmat_f
0	2022-01-01	34.9	24.4	
1	2023-01-01	35.7	21.7	
2	2024-01-01	32.5	19.5	
3	2025-01-01	33.6	18.9	

In [380…

```python
df8 = pd.read_csv("/content/WFH Rates for Women & Men - Sheet1.csv")

df8.columns = df8.columns.str.strip().str.lower().str.replace(" ", "_")
df8 = df8[df8['date'].astype(str).str.contains(r"\d")]
df8['date'] = pd.to_datetime(df8['date'], errors='coerce')

num_cols = df8.select_dtypes(include=[np.number]).columns
cat_cols = df8.select_dtypes(include=['object']).columns

df8 = df8.dropna(subset=num_cols, how='all').drop_duplicates()

df8[num_cols] = df8[num_cols].fillna(df8[num_cols].median())
df8[cat_cols] = df8[cat_cols].fillna("Unknown")

df8.head()
```

Out[380…

	date	wfhcovid_fracmat_women	wfhcovid_fracmat_men	license	citat
0	2020-05-01	64.66	59.58	Copyright 2025 by Jose Maria Barrero, Nicholas...	W u w ple (Barr
1	2020-07-01	51.69	50.59	Unknown	Unkn
2	2020-08-01	50.32	46.96	Unknown	Unkn
3	2020-09-01	44.41	44.33	Unknown	Unkn
4	2020-10-01	47.20	40.82	Unknown	Unkn

In [381…

```
df9 = pd.read_csv("/content/WFH_WFO_dataset.csv")

df9.columns = df9.columns.str.strip().str.lower().str.replace(" ", "_")
df9 = df9.drop_duplicates()

num_cols = df9.select_dtypes(include=[np.number]).columns
cat_cols = df9.select_dtypes(include=['object']).columns

df9[num_cols] = df9[num_cols].fillna(df9[num_cols].median())
df9[cat_cols] = df9[cat_cols].fillna("Unknown")

df9.head()
```

Out[381…

	id	name	age	occupation	gender	same_ofiice_home_location	kids	rm_s
0	1	Bhavana	45	Tutor	Female	Yes	Yes	
1	2	Harry	24	Tutor	Male	No	No	
2	3	Banditaa	53	HR	Female	Yes	Yes	
3	4	Neetha	26	Engineer	Female	Yes	No	
4	5	Ram	26	Recruiter	Male	Yes	No	

In [382…

```
df10 = pd.read_csv("/content/WFHtimeseries_monthly dataset  - Sheet1.csv")

df10.columns = df10.columns.str.strip().str.lower().str.replace(" ", "_")
df10 = df10[df10['date'].astype(str).str.contains(r"\d")]
df10['date'] = pd.to_datetime(df10['date'], errors='coerce')

num_cols = df10.select_dtypes(include=[np.number]).columns
cat_cols = df10.select_dtypes(include=['object']).columns

df10 = df10.dropna(subset=num_cols, how='all').drop_duplicates()
```

```
df10[num_cols] = df10[num_cols].fillna(df10[num_cols].median())
df10[cat_cols] = df10[cat_cols].fillna("Unknown")

df10.head()
```

Out[382…

	date	wfhcovid_matquestion	wfhcovid_frac_hps	notes	license	ci
0	2020-03-01	7.2	29.1	Pre-COVID value is Authors' estimate using dat...	Unknown	Un
1	2020-05-01	61.6	29.1	Unknown	Copyright 2025 by Jose Maria Barrero, Nicholas...	Ba
2	2020-06-01	56.4	29.1	The SWAA June 2020 estimate is averages the Ma...	Unknown	Un
3	2020-07-01	51.2	29.1	Unknown	Unknown	Un
4	2020-08-01	48.4	29.1	Unknown	Unknown	Un

In [383…

```
df11 = pd.read_csv("/content/Work Arrangements by Industry - Sheet1.csv")

df11.columns = df11.columns.str.strip().str.lower().str.replace(" ", "_")
df11 = df11[df11['date'].astype(str).str.contains(r"\d")]
df11['date'] = pd.to_datetime(df11['date'], errors='coerce')

num_cols = df11.select_dtypes(include=[np.number]).columns
cat_cols = df11.select_dtypes(include=['object']).columns

df11 = df11.dropna(subset=num_cols, how='all').drop_duplicates()

df11[num_cols] = df11[num_cols].fillna(df11[num_cols].median())
df11[cat_cols] = df11[cat_cols].fillna("Unknown")

df11.head()
```

Out[383…

	date	full_onsite_arts_entertain	full_onsite_education	full_onsite_financ
2	2022-01-01	31.4	61.3	
3	2022-02-01	41.0	61.9	
4	2022-03-01	36.8	64.8	
5	2022-04-01	38.6	64.1	
6	2022-05-01	43.5	58.4	

In [384…

```
df12 = pd.read_csv("/content/Workday-weighted WFH series - Sheet1.csv")

df12.columns = df12.columns.str.strip().str.lower().str.replace(" ", "_")
df12 = df12[df12['date'].astype(str).str.contains(r"\d")]
df12['date'] = pd.to_datetime(df12['date'], errors='coerce')

num_cols = df12.select_dtypes(include=[np.number]).columns
cat_cols = df12.select_dtypes(include=['object']).columns

df12 = df12.dropna(subset=num_cols, how='all').drop_duplicates()

df12[num_cols] = df12[num_cols].fillna(df12[num_cols].median())
df12[cat_cols] = df12[cat_cols].fillna("Unknown")

df12.head()
```

Out[384…

	date	wfhcovid_matquestion_daywt	notes	license	citation
0	2020-05-01	61.6	Unknown	Copyright 2025 by Jose Maria Barrero, Nicholas...	When using this work, please cite: Barrero, Jo...
1	2020-06-01	56.4	The SWAA June 2020 estimate is averages the Ma...	Unknown	Unknown
2	2020-07-01	51.2	Unknown	Unknown	Unknown
3	2020-08-01	48.4	Unknown	Unknown	Unknown
4	2020-09-01	44.4	Unknown	Unknown	Unknown

In [385…

```
df13 = pd.read_csv("/content/Worker Desires post-COVID WFH - Sheet1.csv")

df13.columns = df13.columns.str.strip().str.lower().str.replace(" ", "_")
df13 = df13[df13['date'].astype(str).str.contains(r"\d")]
df13['date'] = pd.to_datetime(df13['date'], errors='coerce')
```

```
num_cols = df13.select_dtypes(include=[np.number]).columns
cat_cols = df13.select_dtypes(include=['object']).columns

df13 = df13.dropna(subset=num_cols, how='all').drop_duplicates()

df13[num_cols] = df13[num_cols].fillna(df13[num_cols].median())
df13[cat_cols] = df13[cat_cols].fillna("Unknown")

df13.head()
```

Out[385…

	date	wfh_days_postcovid_desmad	wfh_days_postcovid_des_emad	wfhc
0	2020-05-01	2.09	Unknown	
1	2020-07-01	2.10	Unknown	
2	2020-08-01	2.17	2.58	
3	2020-09-01	2.20	2.55	
7	2021-01-01	2.41	2.96	

Imports for Analysis & Plot Style

In [386…
```
# A1 — Analysis & visualization setup

import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use("default")
sns.set_theme(style="whitegrid")

def format_plot(title, xlabel, ylabel):
    plt.title(title, fontsize=14, pad=10)
    plt.xlabel(xlabel, fontsize=12)
    plt.ylabel(ylabel, fontsize=12)
    plt.xticks(rotation=45)
    plt.tight_layout()
```

Long-run WFH trend (1965–Present)

In [387…
```
# Creating a line chart using Plotly to visualize the long-run WFH share from
# The code maps year on the x-axis and WFH percentage on the y-axis, with labe
# This helps convert the cleaned dataset into an interactive visualization for
import plotly.express as px
```

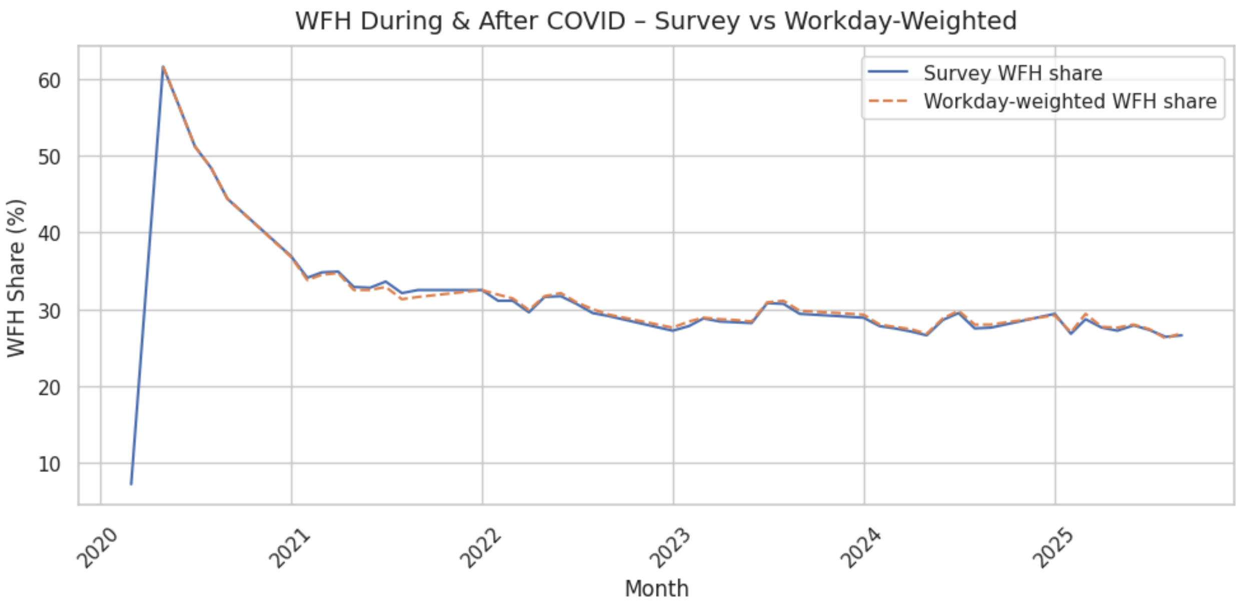
```python
fig_wfh_longrun = px.line(
    df5,
    x="date",
    y="wfh_share",
    title="Long-Run Work-From-Home Share (1965–Present)",
    labels={
        "date": "Year",
        "wfh_share": "WFH Share (%)"
    }
)

fig_wfh_longrun.show()
```

The chart shows that Work-From-Home rates remained extremely low for several decades until the COVID-19 pandemic in 2020, where WFH adoption spiked dramatically. After the peak, WFH levels gradually declined but stabilized at a much higher level than the pre-pandemic era. This highlights a major structural shift in workplace behavior over time.

Monthly WFH during & after COVID

In [388…

```
# Plotting monthly WFH trends during and after COVID using survey data (df10)
# and workday-weighted data (df12) to compare measurement methods.
# This code overlays both series on one chart to show how closely survey respo

fig, ax = plt.subplots(figsize=(10, 5))

ax.plot(df10["date"], df10["wfhcovid_matquestion"], label="Survey WFH share")
ax.plot(df12["date"], df12["wfhcovid_matquestion_daywt"],
        label="Workday-weighted WFH share", linestyle="--")

format_plot(
    "WFH During & After COVID – Survey vs Workday-Weighted",
    "Month",
    "WFH Share (%)"
)
ax.legend()
plt.show()
```

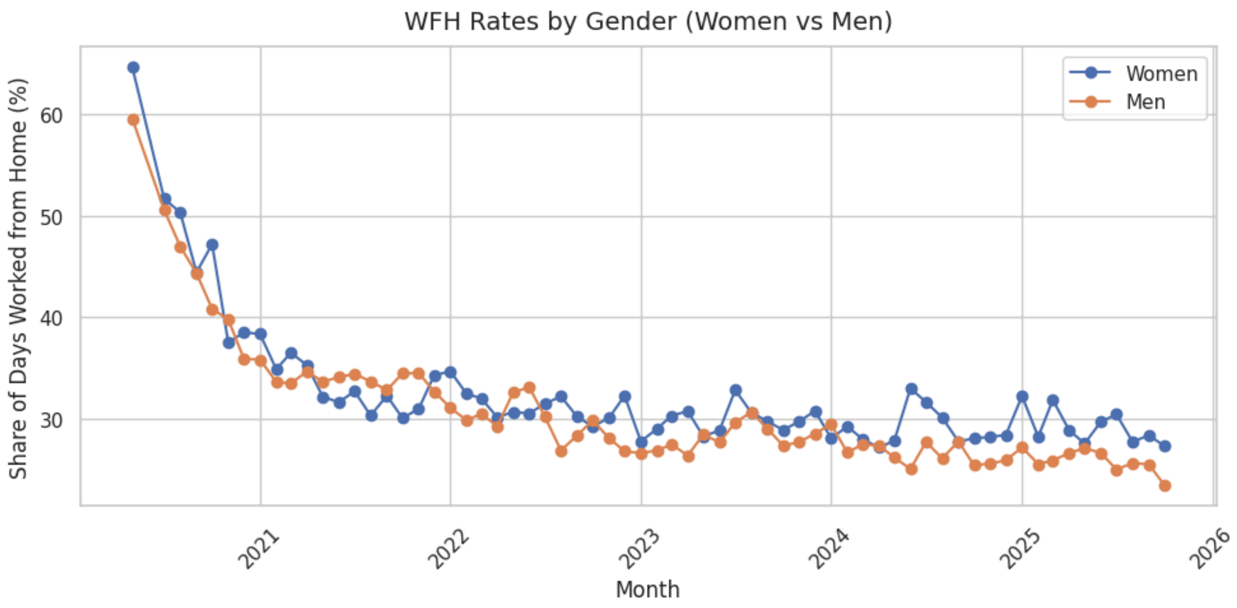


The graph shows that WFH levels peaked sharply during early COVID, then declined and stabilized around 27–30% afterward. Both the survey-based and workday-weighted measures follow almost the same pattern, indicating that self-reported WFH rates are reliable. The alignment of both lines suggests consistent long-term behavior in how employees and employers report remote work.

WFH by Gender

In [389…

```
# Plotting WFH rates for women and men over time using monthly data from df8.
# This code overlays the two gender series to compare how their remote work sh
# It helps visualize gender-based differences in WFH participation during and

fig, ax = plt.subplots(figsize=(10, 5))

ax.plot(df8["date"], df8["wfhcovid_fracmat_women"], label="Women", marker="o")
ax.plot(df8["date"], df8["wfhcovid_fracmat_men"], label="Men", marker="o")

format_plot(
    "WFH Rates by Gender (Women vs Men)",
    "Month",
    "Share of Days Worked from Home (%)"
)
ax.legend()
plt.show()
```



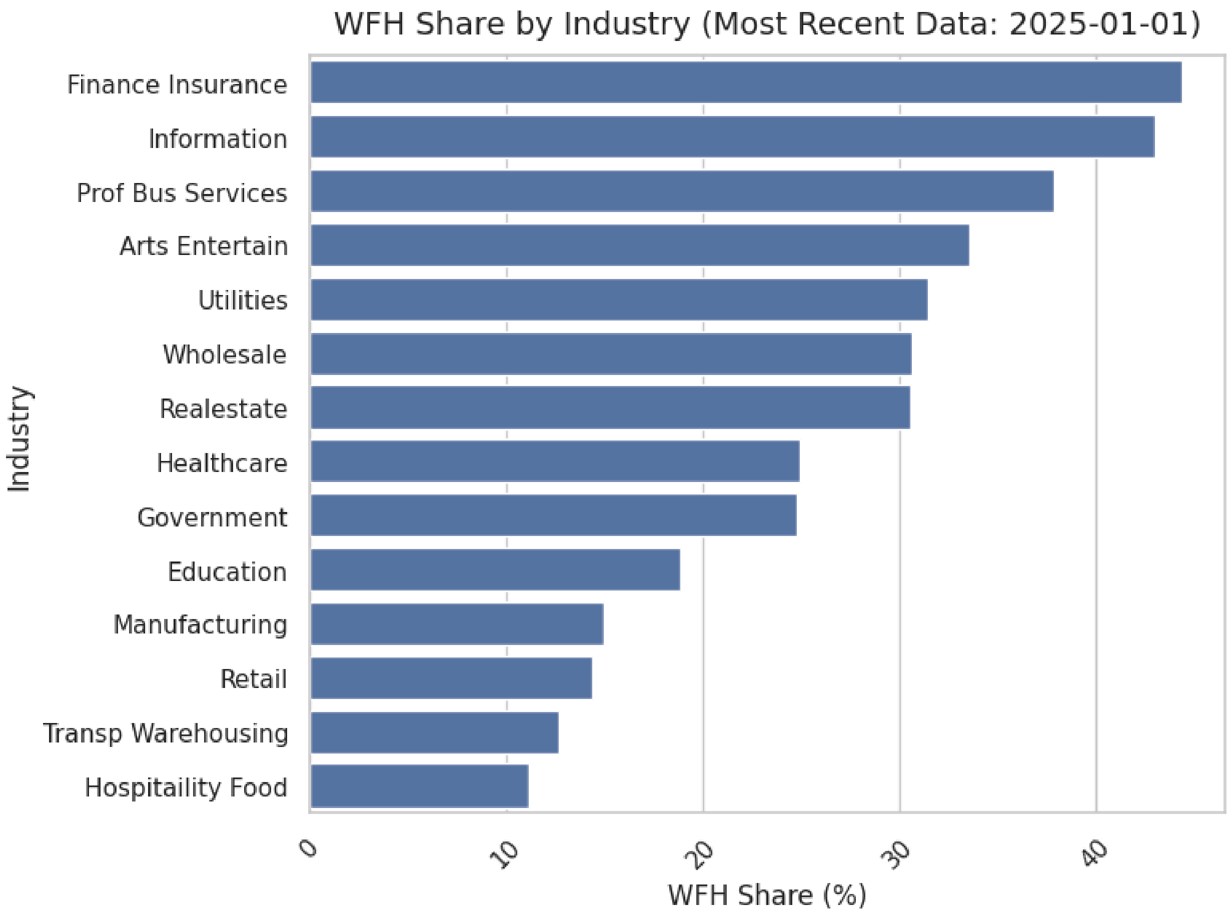
The graph shows that women consistently work from home slightly more than men throughout the observed period. Both groups show a sharp decline after the pandemic peak, stabilizing in the 25–35% range. The persistent gap suggests that women may have stronger preference or need for remote work due to structural or household factors.

WFH Rates by Industry

In [390…

```
# Converting the industry WFH dataset into a long format to make it easier to
# Selecting only the most recent date to show a clear cross-sectional view of
# Sorting industries by WFH share and creating a horizontal bar chart for bett


# Melt to long format for easier plotting
industry_cols = [c for c in df7.columns if c.startswith("wfh_fracmat_")]
id_vars = ["date"]
df7_long = df7[id_vars + industry_cols].melt(
    id_vars="date", var_name="industry", value_name="wfh_share"
)

# Clean industry names
df7_long["industry"] = df7_long["industry"].str.replace("wfh_fracmat_", "").st
```

```python
# Use latest date only for a cross-sectional comparison
latest_date = df7_long["date"].max()
df7_latest = df7_long[df7_long["date"] == latest_date].copy()

# Sort industries by WFH share
df7_latest = df7_latest.sort_values("wfh_share", ascending=False)

fig, ax = plt.subplots(figsize=(8, 6))
sns.barplot(data=df7_latest, x="wfh_share", y="industry", ax=ax)

format_plot(
    f"WFH Share by Industry (Most Recent Data: {latest_date.date()})",
    "WFH Share (%)",
    "Industry"
)
plt.show()

df7_latest.head()
```



Out[390…

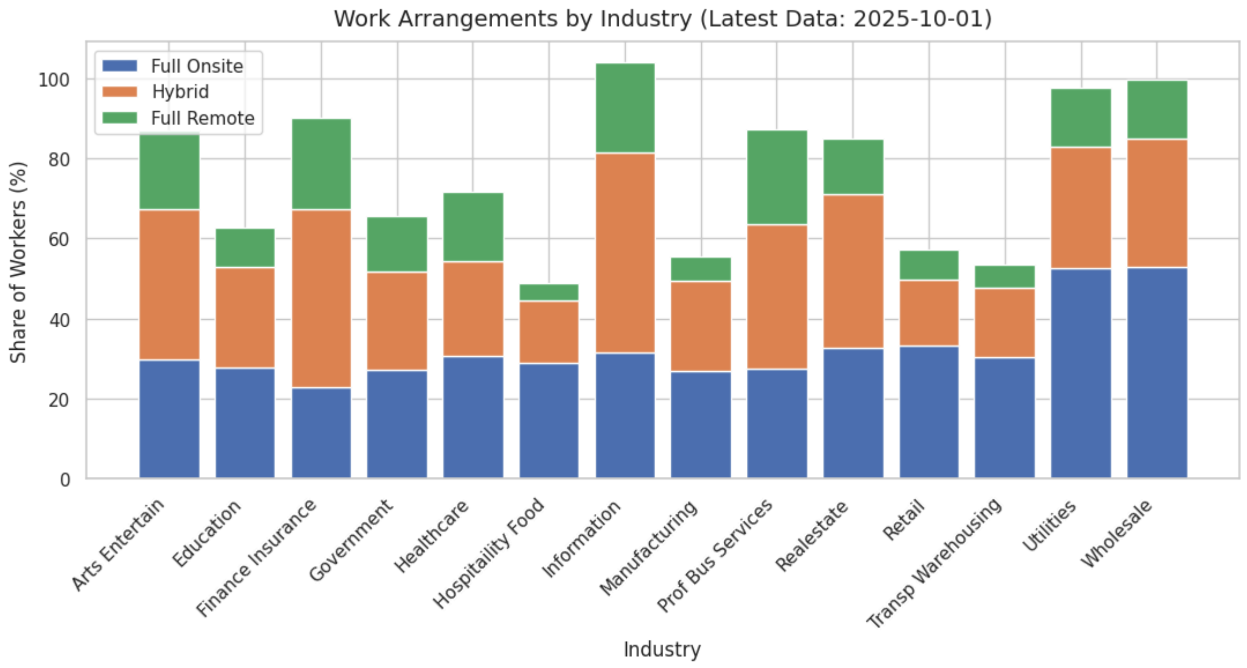
	date	industry	wfh_share
11	2025-01-01	Finance Insurance	44.40
27	2025-01-01	Information	43.08
35	2025-01-01	Prof Bus Services	37.93
3	2025-01-01	Arts Entertain	33.60
51	2025-01-01	Utilities	31.51

This chart compares WFH adoption across industries using the most recent available data. Finance, Information, and Professional Services show the highest remote work levels, while Hospitality, Retail, and Manufacturing remain the lowest. The ranking highlights how WFH suitability strongly depends on the nature of work within each industry.

Work Arrangements by Industry (Onsite vs Hybrid vs Remote)

In [391…

```
# Extracting the latest available date and preparing onsite, hybrid, and remot
# Converting column names into readable industry labels and plotting a stacked
# This visualization shows how industries differ in their mix of onsite, hybri

# Pick latest date
latest_date_wa = df11["date"].max()
wa_latest = df11[df11["date"] == latest_date_wa].copy()

# Columns for full onsite / hybrid / full remote — get patterns
onsite_cols  = [c for c in wa_latest.columns if c.startswith("full_onsite_")]
hybrid_cols  = [c for c in wa_latest.columns if c.startswith("hybrid_")]
remote_cols  = [c for c in wa_latest.columns if c.startswith("full_remote_")]

industries = [c.replace("full_onsite_", "").replace("_", " ").title()
              for c in onsite_cols]

onsite_vals = wa_latest[onsite_cols].iloc[0].values
hybrid_vals = wa_latest[hybrid_cols].iloc[0].values
remote_vals = wa_latest[remote_cols].iloc[0].values

x = np.arange(len(industries))

fig, ax = plt.subplots(figsize=(11, 6))

ax.bar(x, onsite_vals, label="Full Onsite")
ax.bar(x, hybrid_vals, bottom=onsite_vals, label="Hybrid")
ax.bar(x, remote_vals, bottom=onsite_vals + hybrid_vals, label="Full Remote")

ax.set_xticks(x)
ax.set_xticklabels(industries, rotation=45, ha="right")

format_plot(
    f"Work Arrangements by Industry (Latest Data: {latest_date_wa.date()})",
    "Industry",
    "Share of Workers (%)"
)
ax.legend()
plt.show()
```



This chart compares how different industries split their workforce across onsite, hybrid, and fully remote arrangements. Knowledge-based sectors like Information and Professional Services show higher hybrid or remote shares, while industries such as Retail, Manufacturing, and Hospitality remain predominantly onsite. The distribution clearly highlights which industries have the flexibility for remote work and which rely heavily on physical presence.

Divergence chart: hybrid vs full onsite by industry (latest)

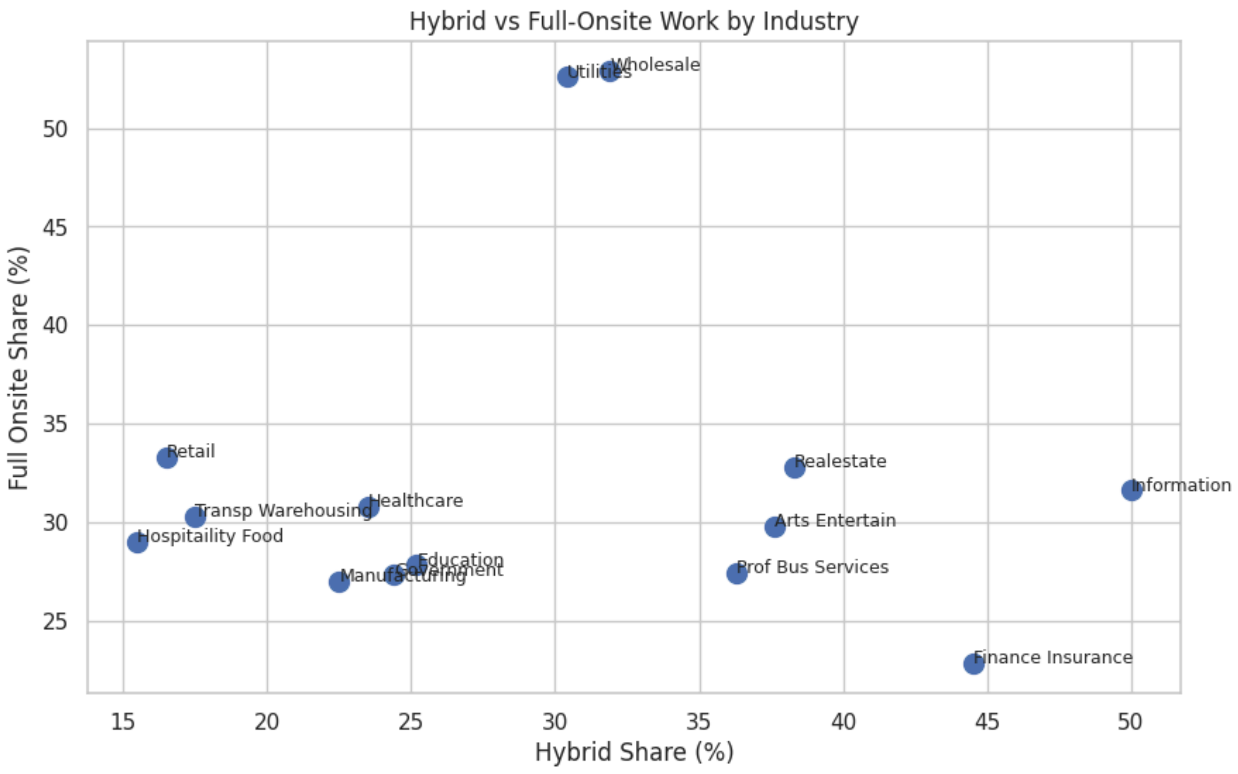
In [392…
```
# Plotting a divergence scatter chart to compare hybrid vs full onsite work sh
# Each industry is positioned based on its hybrid percentage (x-axis) and onsi
# This visualization helps identify which industries lean more toward hybrid m


plt.figure(figsize=(10, 6))

plt.scatter(
    wa_latest[hybrid_cols].iloc[0],
    wa_latest[onsite_cols].iloc[0],
    s=100
)

for i, industry in enumerate(industries):
    plt.text(
        wa_latest[hybrid_cols].iloc[0][i],
        wa_latest[onsite_cols].iloc[0][i],
        industry,
        fontsize=9
    )

plt.title("Hybrid vs Full-Onsite Work by Industry")
plt.xlabel("Hybrid Share (%)")
plt.ylabel("Full Onsite Share (%)")
plt.grid(True)
plt.show()
```

```
/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`
```

```
/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`
```

```
/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`

/tmp/ipython-input-4031870399.py:16: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`
```

```
/tmp/ipython-input-4031870399.py:17: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`
```



This scatter plot shows how industries vary in their balance between hybrid and full onsite work. Sectors like Information and Finance have high hybrid shares and low onsite shares, indicating greater remote flexibility. In contrast, industries such as Utilities, Wholesale, and Retail have high onsite shares, showing limited adoption of hybrid work models.

Industry Cluster Analysis

In [393…

```
# Applying K-Means clustering on the latest industry WFH percentages to group
# Creating three clusters (Low, Medium, High WFH) and assigning descriptive la
# Plotting a scatter chart to visualize how industries naturally group based o

from sklearn.cluster import KMeans

# Prepare data for clustering
industry_features = df7_latest[['wfh_share']].values
kmeans = KMeans(n_clusters=3, random_state=42, n_init='auto')
df7_latest.loc[:, 'cluster'] = kmeans.fit_predict(industry_features)
df7_latest.loc[:, 'cluster_name'] = df7_latest['cluster'].map({
    0: 'Low WFH (<20%)',
    1: 'Medium WFH (20-35%)',
    2: 'High WFH (>35%)'
})

px.scatter(df7_latest, x='industry', y='wfh_share', color='cluster_name',
           title='Industry Clusters by WFH Adoption').show()
```

The cluster plot shows clear separation between industries with high, medium, and low WFH adoption. Knowledge-driven industries like Finance, Information, and Professional Services fall into the High WFH cluster, while sectors requiring physical presence, such as Retail, Manufacturing, and Hospitality, cluster at the low end. This reveals how the nature of work strongly influences an industry's ability to support remote work.

Trend of Remote/Hybrid/Onsite Work Arrangements by Industry

In [394…
```
industries_to_plot = ['arts_entertain', 'finance_insurance', 'information']

# Prepare a list of all relevant columns for these industries
all_industry_cols = ['date']
```

```
for industry in industries_to_plot:
    all_industry_cols.append(f'full_onsite_{industry}')
    all_industry_cols.append(f'hybrid_{industry}')
    all_industry_cols.append(f'full_remote_{industry}')

# Create a dataframe with only the relevant columns
df11_trends = df11[all_industry_cols].copy()

# Rename columns for clarity in plotting
rename_cols = {}
for col in df11_trends.columns:
    if 'full_onsite_' in col:
        rename_cols[col] = col.replace('full_onsite_', '') + '_full_onsite'
    elif 'hybrid_' in col:
        rename_cols[col] = col.replace('hybrid_', '') + '_hybrid'
    elif 'full_remote_' in col:
        rename_cols[col] = col.replace('full_remote_', '') + '_full_remote'
df11_trends = df11_trends.rename(columns=rename_cols)

df11_trends.head()
```

Out[394…

	date	arts_entertain_full_onsite	arts_entertain_hybrid	arts_entertain_ful
2	2022-01-01	31.4	44.6	
3	2022-02-01	41.0	35.3	
4	2022-03-01	36.8	39.7	
5	2022-04-01	38.6	37.7	
6	2022-05-01	43.5	37.0	

In [395…

```
# Looping through selected industries to create individual stacked-area charts
# The code reshapes each industry's data into long format, cleans labels, and
# This helps visualize the dynamic shift between onsite, hybrid, and remote wo

import plotly.express as px

for industry in industries_to_plot:
    # Select columns for the current industry
    industry_cols = [
        'date',
        f'{industry}_full_onsite',
        f'{industry}_hybrid',
        f'{industry}_full_remote'
    ]

    # Create a temporary DataFrame for the current industry
    temp_df = df11_trends[industry_cols].copy()

    # Melt to long format
    temp_df_long = pd.melt(
        temp_df,
        id_vars=['date'],
        var_name='Work Arrangement',
        value_name='Share (%)'
    )

    # Clean 'Work Arrangement' names for display
    temp_df_long['Work Arrangement'] = temp_df_long['Work Arrangement'].str.re

    # Create stacked area chart
    fig = px.area(
        temp_df_long,
        x='date',
        y='Share (%)',
        color='Work Arrangement',
        title=f'Work Arrangement Trends for {industry.replace("_", " ").title(
        labels={
            'date': 'Date',
            'Share (%)': 'Share of Workers (%)'
        },
        line_group='Work Arrangement'
    )
    fig.update_layout(hovermode="x unified")
    fig.show()
```

Micro-Level Preferences: WFH_WFO Survey

```
In [396…
# Calculating the percentage distribution of the target variable (0 = Onsite,
# This gives a quick overview of how many respondents prefer WFH versus onsite


df9["target"].value_counts(normalize=True) * 100
```

```
Out[396…
          proportion
target
     0      59.42029
     1      40.57971
```

dtype: float64

The distribution shows that about 59% of respondents prefer onsite work (target = 0), while about 41% prefer work-from-home (target = 1). This indicates that onsite preference is slightly higher in the dataset, setting the baseline before exploring what influences WFH preference.

In [397…

```
# Computing crosstabs to examine how satisfaction factors relate to the WFH pr
# Each crosstab shows row-wise percentages comparing satisfaction responses wi

satisfaction_cols = [
    "rm_save_money", "rm_quality_time", "rm_better_sleep",
    "rm_better_work_life_balance"
]

for col in satisfaction_cols:
    crosstab = pd.crosstab(df9[col], df9["target"], normalize="index") * 100
    print(f"\n {col} vs Target (row %)**")
    print(crosstab.round(1))
```

```
 rm_save_money vs Target (row %)**
target            0     1
rm_save_money
No             58.8  41.2
Yes            59.6  40.4

 rm_quality_time vs Target (row %)**
target              0     1
rm_quality_time
No               66.7  33.3
Yes              55.6  44.4

 rm_better_sleep vs Target (row %)**
target              0     1
rm_better_sleep
No               76.3  23.7
Yes              38.7  61.3

 rm_better_work_life_balance vs Target (row %)**
target                          0     1
rm_better_work_life_balance
1                            90.0  10.0
2                            69.2  30.8
3                            63.2  36.8
4                            55.6  44.4
5                            11.1  88.9
```

These crosstabs help identify which satisfaction factors are associated with a stronger preference for WFH. By comparing row percentages, we can see how responses like saving money, better sleep, or improved work-life balance differ between WFH and onsite groups.

Summary Statistics for All Datasets

In [398…

```
# Summary statistics for all datasets
```

```
datasets = {
    "df1_plans": df1,
    "df5_longrun": df5,
    "df6_city": df6,
    "df7_industry": df7,
    "df8_gender": df8,
    "df10_monthly": df10,
    "df11_arrangements": df11,
    "df12_dayweighted": df12,
    "df13_desires": df13
}

for name, df in datasets.items():
    print(f"\n SUMMARY STATISTICS: {name} \n")
    display(df.describe(include="all"))
```

```
SUMMARY STATISTICS: df1_plans
```

	date	wfh_days_postcovid_planmad	wfh_days_postcovid_plar
count	65	65.000000	
unique	NaN	NaN	
top	NaN	NaN	
freq	NaN	NaN	
mean	2023-08-13 16:14:46.153846272	8.696615	
min	2020-07-01 00:00:00	1.060000	
25%	2022-05-01 00:00:00	1.440000	
50%	2024-02-01 00:00:00	1.500000	
75%	2025-01-01 00:00:00	26.600000	
max	2025-09-01 00:00:00	29.500000	
std	NaN	11.876448	

```
SUMMARY STATISTICS: df5_longrun
```

	date	wfh_share	source_historical_series	fullremote_hist	source_fu
count	72	72.000000	72	72.000000	
unique	NaN	NaN	3	NaN	
top	NaN	NaN	SWAA	NaN	
freq	NaN	NaN	50	NaN	
mean	2017-07-29 15:40:00	23.600000	NaN	3.012500	
min	1965-01-01 00:00:00	0.400000	NaN	1.800000	
25%	2015-10-01 18:00:00	6.400000	NaN	3.000000	
50%	2021-11-01 00:00:00	27.850000	NaN	3.000000	
75%	2023-10-01 12:00:00	31.100000	NaN	3.000000	
max	2025-09-01 00:00:00	61.600000	NaN	4.700000	
std	NaN	14.302664	NaN	0.392872	

```
SUMMARY STATISTICS: df6_city
```

	date	wfhcovid_series_top10_ma6_	wfhcovid_series_11to50_
count	65	65.000000	65.00
unique	NaN	NaN	
top	NaN	NaN	
freq	NaN	NaN	
mean	2023-05-23 07:00:55.384615424	34.361538	29.90
min	2020-10-01 00:00:00	26.200000	27.00
25%	2022-02-01 00:00:00	31.100000	28.10
50%	2023-06-01 00:00:00	33.600000	29.20
75%	2024-10-01 00:00:00	37.700000	30.40
max	2025-10-01 00:00:00	51.100000	44.00
std	NaN	4.278752	2.89

```
SUMMARY STATISTICS: df7_industry
```

	date	wfh_fracmat_arts_entertain	wfh_fracmat_education	wfh_fracm
count	4	4.00000	4.000000	
unique	NaN	NaN	NaN	
top	NaN	NaN	NaN	
freq	NaN	NaN	NaN	
mean	2023-07-02 18:00:00	34.17500	21.125000	
min	2022-01-01 00:00:00	32.50000	18.900000	
25%	2022-10-01 18:00:00	33.32500	19.350000	
50%	2023-07-02 12:00:00	34.25000	20.600000	
75%	2024-04-01 12:00:00	35.10000	22.375000	
max	2025-01-01 00:00:00	35.70000	24.400000	
std	NaN	1.41274	2.493157	

```
SUMMARY STATISTICS: df8_gender
```

	date	wfhcovid_fracmat_women	wfhcovid_fracmat_men	lic
count	65	65.000000	65.000000	
unique	NaN	NaN	NaN	
top	NaN	NaN	NaN	Unk
freq	NaN	NaN	NaN	
mean	2023-01-30 10:42:27.692307712	32.388615	30.710000	
min	2020-05-01 00:00:00	27.130000	23.420000	
25%	2021-10-01 00:00:00	28.860000	26.630000	
50%	2023-02-01 00:00:00	30.490000	28.420000	
75%	2024-06-01 00:00:00	32.430000	33.460000	
max	2025-10-01 00:00:00	64.660000	59.580000	
std	NaN	6.490025	6.434925	

```
SUMMARY STATISTICS: df10_monthly
```

	date	wfhcovid_matquestion	wfhcovid_frac_hps	notes	license
count	51	51.000000	51.000000	51	51
unique	NaN	NaN	NaN	4	2
top	NaN	NaN	NaN	Unknown	Unknown
freq	NaN	NaN	NaN	48	50
mean	2022-12-29 00:00:00	31.554902	29.203922	NaN	NaN
min	2020-03-01 00:00:00	7.200000	28.000000	NaN	NaN
25%	2021-07-16 12:00:00	27.700000	29.100000	NaN	NaN
50%	2023-02-01 00:00:00	29.400000	29.100000	NaN	NaN
75%	2024-05-16 12:00:00	32.500000	29.100000	NaN	NaN
max	2025-09-01 00:00:00	61.600000	31.900000	NaN	NaN
std	NaN	8.231022	0.627363	NaN	NaN

```
SUMMARY STATISTICS: df11_arrangements
```

	date	full_onsite_arts_entertain	full_onsite_education	full_o
count	52	52.000000	52.000000	
unique	NaN	NaN	NaN	
top	NaN	NaN	NaN	
freq	NaN	NaN	NaN	
mean	2024-04-27 07:50:46.153846272	38.528846	55.788462	
min	2022-01-01 00:00:00	26.200000	27.700000	
25%	2023-04-23 12:00:00	31.275000	47.150000	
50%	2024-09-16 00:00:00	41.000000	64.200000	
75%	2025-05-08 18:00:00	44.000000	67.100000	
max	2025-10-01 00:00:00	51.100000	71.500000	
std	NaN	7.297551	16.574995	

```
SUMMARY STATISTICS: df12_dayweighted
```

	date	wfhcovid_matquestion_daywt	notes	license	citation
count	50	50.000000	50	50	50
unique	NaN	NaN	3	2	2
top	NaN	NaN	Unknown	Unknown	Unknown
freq	NaN	NaN	48	49	49
mean	2023-01-18 15:50:24	32.158000	NaN	NaN	NaN
min	2020-05-01 00:00:00	26.200000	NaN	NaN	NaN
25%	2021-08-08 18:00:00	28.000000	NaN	NaN	NaN
50%	2023-02-15 00:00:00	29.800000	NaN	NaN	NaN
75%	2024-05-24 06:00:00	32.400000	NaN	NaN	NaN
max	2025-09-01 00:00:00	61.600000	NaN	NaN	NaN
std	NaN	7.445807	NaN	NaN	NaN

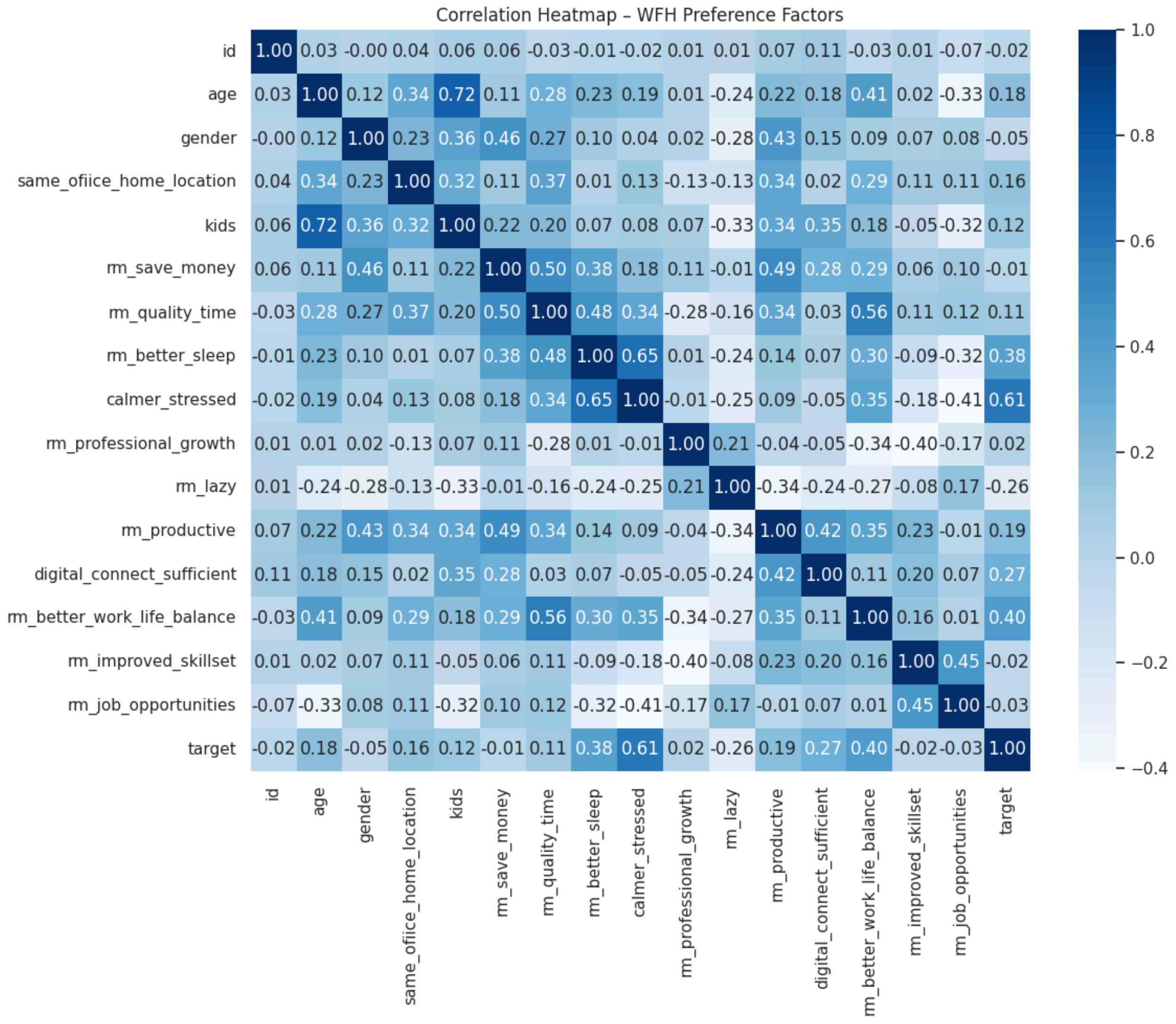
```
SUMMARY STATISTICS: df13_desires
```

	date	wfh_days_postcovid_desmad	wfh_days_postcovid_des_emad	w
count	66	66.000000	66	
unique	NaN	NaN	28	
top	NaN	NaN	SWAA	
freq	NaN	NaN	18	
mean	2023-07-26 12:00:00	9.259697	NaN	
min	2020-05-01 00:00:00	2.090000	NaN	
25%	2022-04-08 12:00:00	2.292500	NaN	
50%	2024-02-01 00:00:00	2.405000	NaN	
75%	2025-01-01 00:00:00	26.550000	NaN	
max	2025-09-01 00:00:00	29.500000	NaN	
std	NaN	11.403354	NaN	

In [399…

```
# Correlation heatmap for WFH preference survey

import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

df9_numeric = df9.copy()

# Map binary 'Yes'/'No' to 1/0 for relevant columns
binary_map = {'Yes': 1, 'No': 0}
for col in ['same_ofiice_home_location', 'kids', 'rm_save_money', 'rm_quality_
            'rm_better_sleep', 'digital_connect_sufficient', 'rm_job_opportuni
    if col in df9_numeric.columns:
        df9_numeric[col] = df9_numeric[col].map(binary_map)

# Map 'Female'/'Male' to 1/0 for 'gender'
gender_map = {'Female': 1, 'Male': 0}
if 'gender' in df9_numeric.columns:
    df9_numeric['gender'] = df9_numeric['gender'].map(gender_map)

# Map 'CALMER'/'STRESSED' to 1/0 for 'calmer_stressed'
calmer_stressed_map = {'CALMER': 1, 'STRESSED': 0}
if 'calmer_stressed' in df9_numeric.columns:
    df9_numeric['calmer_stressed'] = df9_numeric['calmer_stressed'].map(calmer

# Select only numeric columns for correlation calculation
```

```
numeric_cols_for_corr = df9_numeric.select_dtypes(include=[np.number]).columns
corr = df9_numeric[numeric_cols_for_corr].corr()

plt.figure(figsize=(12, 10))
sns.heatmap(corr, cmap="Blues", annot=True, fmt=".2f")
plt.title("Correlation Heatmap – WFH Preference Factors")
plt.tight_layout()
plt.show()
```



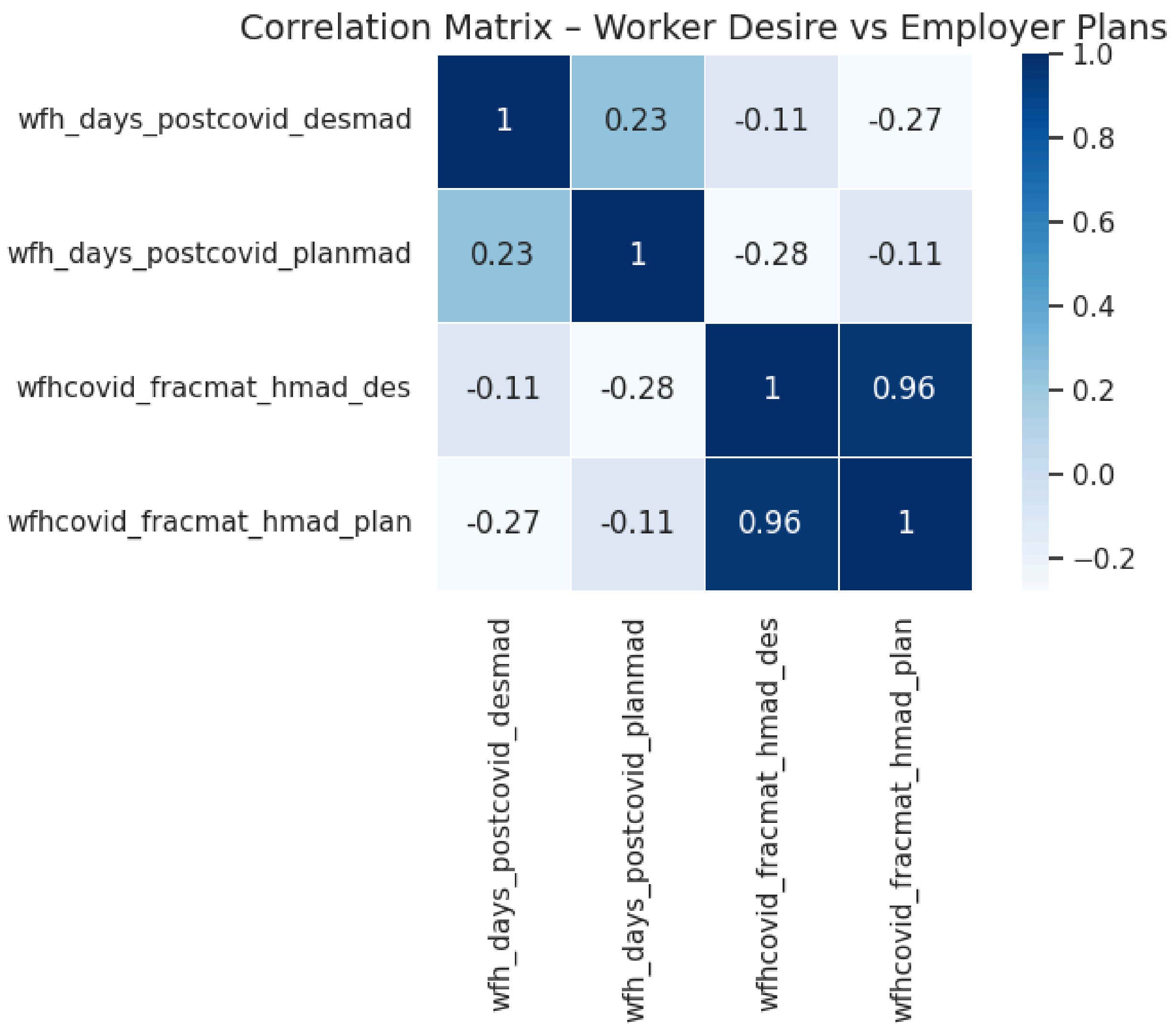
	id	age	gender	same_ofiice_home_location	kids	rm_save_money	rm_quality_time	rm_better_sleep	calmer_stressed	rm_professional_growth	rm_lazy	rm_productive	digital_connect_sufficient	rm_better_work_life_balance	rm_improved_skillset	rm_job_opportunities	target
id	1.00	0.03	-0.00	0.04	0.06	0.06	-0.03	-0.01	-0.02	0.01	0.01	0.07	0.11	-0.03	0.01	-0.07	-0.02
age	0.03	1.00	0.12	0.34	0.72	0.11	0.28	0.23	0.19	0.01	-0.24	0.22	0.18	0.41	0.02	-0.33	0.18
gender	-0.00	0.12	1.00	0.23	0.36	0.46	0.27	0.10	0.04	0.02	-0.28	0.43	0.15	0.09	0.07	0.08	-0.05
same_ofiice_home_location	0.04	0.34	0.23	1.00	0.32	0.11	0.37	0.01	0.13	-0.13	-0.13	0.34	0.02	0.29	0.11	0.11	0.16
kids	0.06	0.72	0.36	0.32	1.00	0.22	0.20	0.07	0.08	0.07	-0.33	0.34	0.35	0.18	-0.05	-0.32	0.12
rm_save_money	0.06	0.11	0.46	0.11	0.22	1.00	0.50	0.38	0.18	0.11	-0.01	0.49	0.28	0.29	0.06	0.10	-0.01
rm_quality_time	-0.03	0.28	0.27	0.37	0.20	0.50	1.00	0.48	0.34	-0.28	-0.16	0.34	0.03	0.56	0.11	0.12	0.11
rm_better_sleep	-0.01	0.23	0.10	0.01	0.07	0.38	0.48	1.00	0.65	0.01	-0.24	0.14	0.07	0.30	-0.09	-0.32	0.38
calmer_stressed	-0.02	0.19	0.04	0.13	0.08	0.18	0.34	0.65	1.00	-0.01	-0.25	0.09	-0.05	0.35	-0.18	-0.41	0.61
rm_professional_growth	0.01	0.01	0.02	-0.13	0.07	0.11	-0.28	0.01	-0.01	1.00	0.21	-0.04	-0.05	-0.34	-0.40	-0.17	0.02
rm_lazy	0.01	-0.24	-0.28	-0.13	-0.33	-0.01	-0.16	-0.24	-0.25	0.21	1.00	-0.34	-0.24	-0.27	-0.08	0.17	-0.26
rm_productive	0.07	0.22	0.43	0.34	0.34	0.49	0.34	0.14	0.09	-0.04	-0.34	1.00	0.42	0.35	0.23	-0.01	0.19
digital_connect_sufficient	0.11	0.18	0.15	0.02	0.35	0.28	0.03	0.07	-0.05	-0.05	-0.24	0.42	1.00	0.11	0.20	0.07	0.27
rm_better_work_life_balance	-0.03	0.41	0.09	0.29	0.18	0.29	0.56	0.30	0.35	-0.34	-0.27	0.35	0.11	1.00	0.16	0.01	0.40
rm_improved_skillset	0.01	0.02	0.07	0.11	-0.05	0.06	0.11	-0.09	-0.18	-0.40	-0.08	0.23	0.20	0.16	1.00	0.45	-0.02
rm_job_opportunities	-0.07	-0.33	0.08	0.11	-0.32	0.10	0.12	-0.32	-0.41	-0.17	0.17	-0.01	0.07	0.01	0.45	1.00	-0.03
target	-0.02	0.18	-0.05	0.16	0.12	-0.01	0.11	0.38	0.61	0.02	-0.26	0.19	0.27	0.40	-0.02	-0.03	1.00

The heatmap shows mostly weak to moderate correlations among the WFH preference factors. There are **no strong correlations (nothing above ~0.70)** except for a few internal links between related satisfaction variables (e.g., quality time, better sleep, calmer/stressed). WFH preference (‘target’) has **moderate positive correlations** with factors like better sleep, better work–life balance, and feeling calmer, but demographic variables such as age and gender show **very weak or no meaningful correlation** with WFH preference. Overall, the chart suggests that personal well-being and lifestyle improvements drive WFH interest much more than demographic characteristics.

Correlation Heatmap

In [400…

```
# Merging worker desire and employer plan datasets on date to compare aligned
# Selecting the key variables for desired vs planned WFH days and WFH shares,
# Plotting a clean heatmap to visually compare how closely worker preferences


merged = df13.merge(df1, on="date", suffixes=("_des", "_plan"), how="inner")

cols = [
    "wfh_days_postcovid_desmad",
    "wfh_days_postcovid_planmad",
    "wfhcovid_fracmat_hmad_des",
    "wfhcovid_fracmat_hmad_plan"
]

corr = merged[cols].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap="Blues", linewidths=0.5, square=True)
plt.title("Correlation Matrix – Worker Desire vs Employer Plans", fontsize=14)
plt.tight_layout()
plt.show()
```



The heatmap shows that worker-desired and employer-planned WFH days have only weak positive correlation (~0.23), suggesting workers and employers often differ in how many days should be remote after COVID.

However, the strongest correlation in the matrix is between the WFH share variables
(wfhcovid_fracmat_hmad_des vs wfhcovid_fracmat_hmad_plan = 0.96), indicating that worker and employer estimates of *overall WFH share* are almost identical.

This means: while workers and employers may disagree on exact weekly remote days,
their expectations about the *percentage of total work done remotely* are nearly the same — a strong alignment.

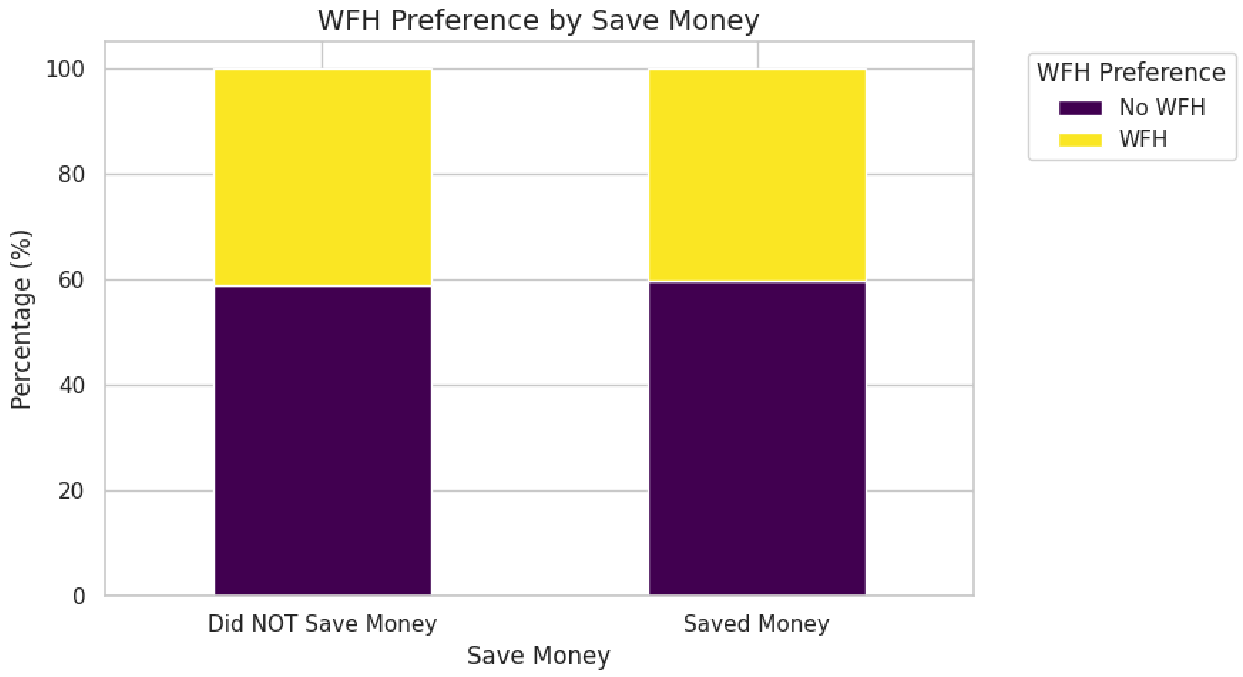
Remote/Hybrid/Onsite Trends Over Time

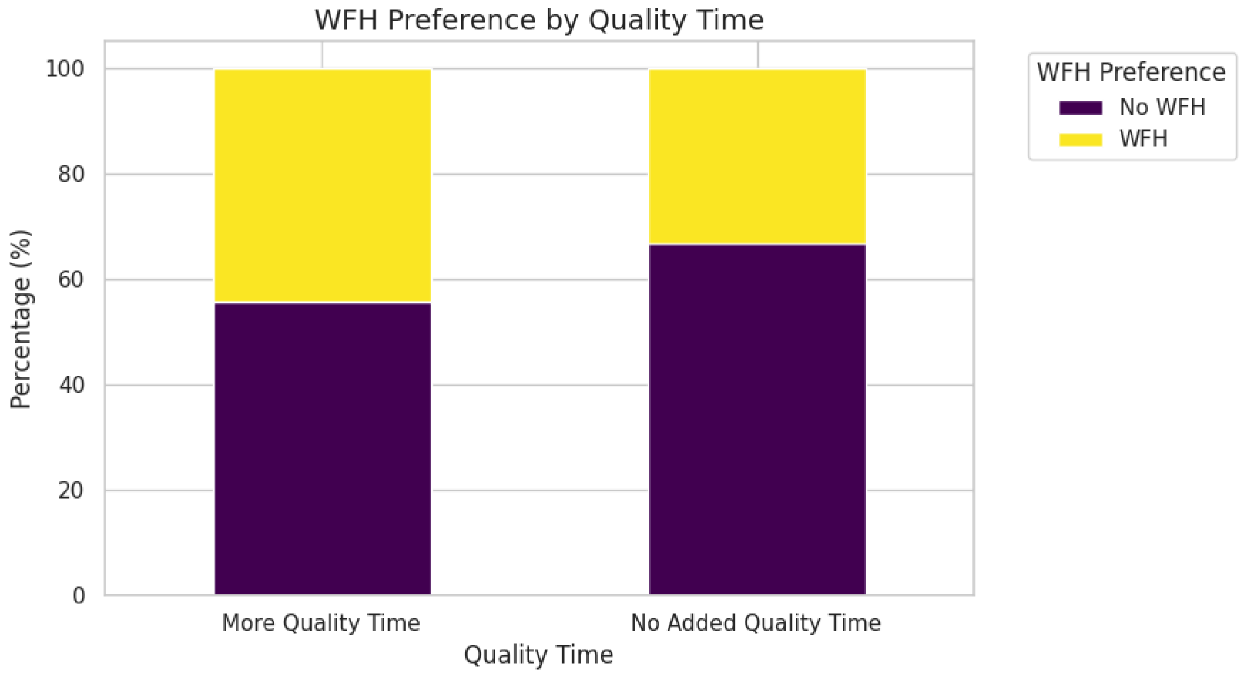
In [401…
```
# Plotting time-series trends for full remote, hybrid, and full onsite work us
# Each work arrangement is added as a stacked area (fill='tonexty') to clearly
# This visualization helps compare how different work modes shift month-to-mon

import plotly.graph_objects as go

fig = go.Figure()
fig.add_trace(go.Scatter(x=df2['date'], y=df2['full_remote_curr'],
                         name='Full Remote', fill='tonexty'))
fig.add_trace(go.Scatter(x=df2['date'], y=df2['hybrid_curr'],
                         name='Hybrid', fill='tonexty'))
fig.add_trace(go.Scatter(x=df2['date'], y=df2['full_onsite_curr'],
                         name='Full Onsite', fill='tonexty'))
fig.update_layout(title='Work Arrangement Trends Over Time')
fig.show()
```

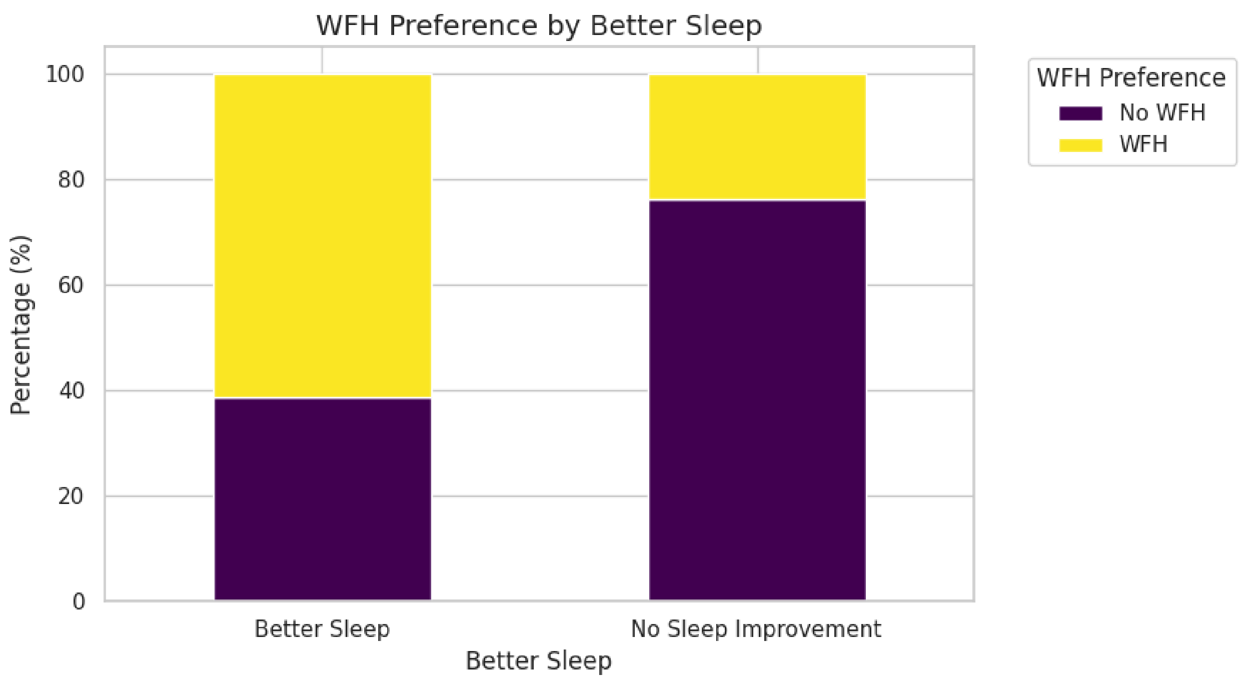
The chart shows that full onsite work consistently remains the dominant arrangement over time, gradually increasing from around 55% to over 60%. Hybrid work stays fairly stable between 25%–30%, showing moderate fluctuation but no major upward trend. Full remote work slowly declines across the period, suggesting that employers may be shifting toward more onsite expectations while retaining hybrid as a steady middle ground.

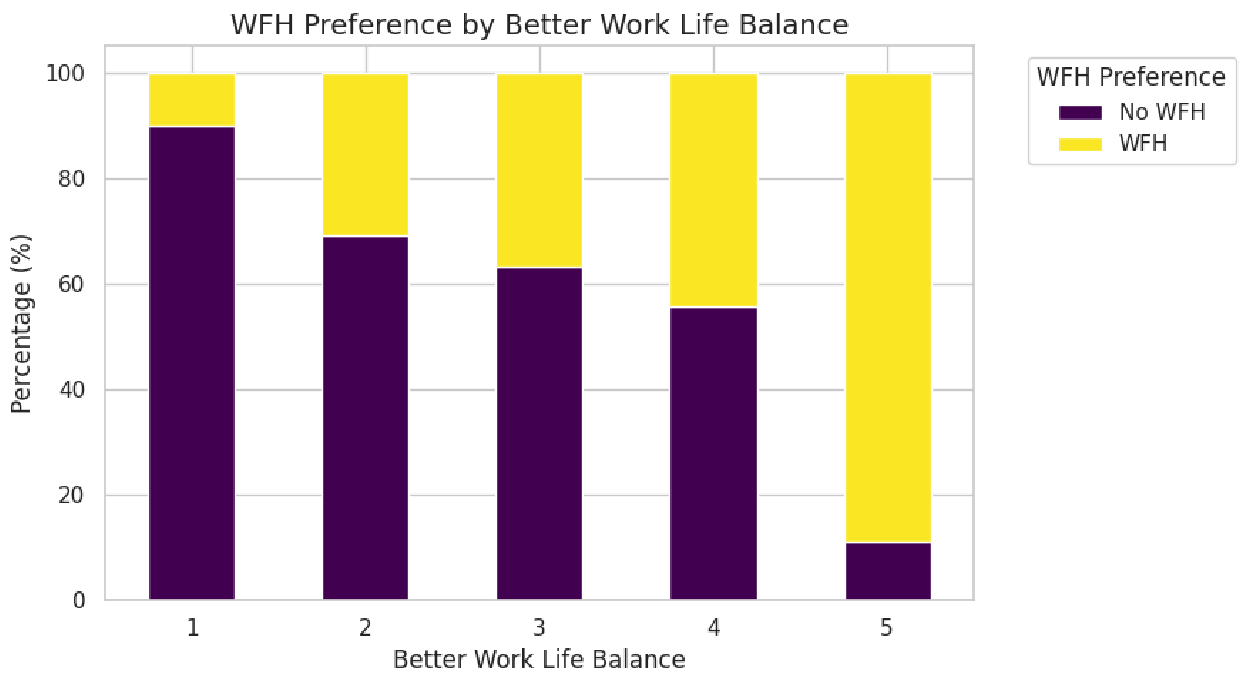
WFH PREFERENCES

```
In [402…
# Cleaning the satisfaction columns by mapping Yes/No responses into clearer l
# and converting the target variable into readable categories (WFH / No WFH).
# For each satisfaction factor, generating a stacked bar chart to compare how
# between respondents who experienced the benefit vs those who did not.


# Mapping for clear labels
label_map = {
    "rm_save_money": {"Yes": "Saved Money", "No": "Did NOT Save Money"},
    "rm_quality_time": {"Yes": "More Quality Time", "No": "No Added Quality Ti
    "rm_better_sleep": {"Yes": "Better Sleep", "No": "No Sleep Improvement"},
}

# Apply mappings
for col, mapping in label_map.items():
    df9[col] = df9[col].replace(mapping)

# Convert target as before
df9["target_label"] = df9["target"].map({0: "No WFH", 1: "WFH"})

satisfaction_cols = [
    "rm_save_money", "rm_quality_time", "rm_better_sleep",
    "rm_better_work_life_balance"
]

for col in satisfaction_cols:
    crosstab = pd.crosstab(df9[col], df9["target_label"], normalize="index") *

    ax = crosstab.plot(kind='bar', stacked=True, figsize=(9, 5), cmap='viridis

    plt.title(f"WFH Preference by {col.replace('rm_', '').replace('_', ' ').ti
    plt.xlabel(col.replace('rm_', '').replace('_', ' ').title(), fontsize=12)
    plt.ylabel("Percentage (%)", fontsize=12)
    plt.xticks(rotation=0)
    plt.legend(title="WFH Preference", bbox_to_anchor=(1.05, 1), loc="upper le

    plt.tight_layout()
    plt.show()
```


WFH Preference by Save Money
Percentage (%)
0
20
40
60
80
100
Did NOT Save Money
Saved Money
Save Money
WFH Preference
No WFH
WFH


WFH Preference by Quality Time
Percentage (%)
0
20
40
60
80
100
More Quality Time
No Added Quality Time
Quality Time
WFH Preference
No WFH
WFH

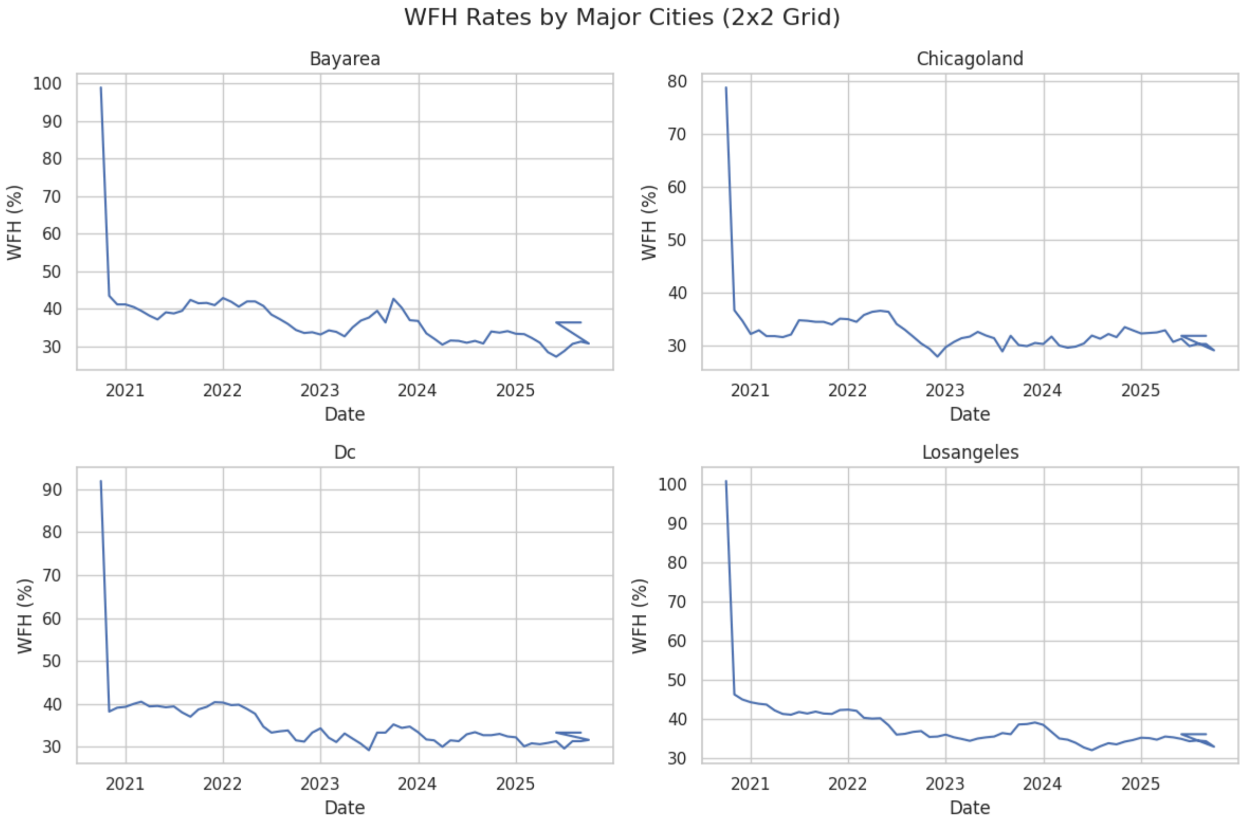




These stacked bar charts show how different satisfaction-related factors influence WFH preference.
Respondents who reported benefits such as saving money, more quality time, or better sleep consistently show a higher share of WFH preference than those who did not experience these improvements.
The strongest pattern appears in work–life balance: as work–life balance ratings increase from 1 to 5, the proportion of people preferring WFH rises sharply.
Overall, the trends indicate that WFH preference is strongly driven by lifestyle advantages and personal well-being rather than demographic factors.

Multi-city comparison (4-panel)

```
In [403…
# Creating a 2x2 panel of line charts to compare WFH trends across four major
# Looping through each city column to plot its WFH percentage over time, with
# This grid layout makes it easy to visually compare how remote work patterns


cities = [
    "wfhcovid_series_ma6_bayarea",
    "wfhcovid_series_ma6_chicagoland",
    "wfhcovid_series_ma6_dc",
    "wfhcovid_series_ma6_losangeles"
]

fig, axes = plt.subplots(2, 2, figsize=(12, 8))

for ax, col in zip(axes.flatten(), cities):
    ax.plot(df6["date"], df6[col])
    ax.set_title(col.replace("wfhcovid_series_ma6_", "").title())
    ax.set_xlabel("Date")
    ax.set_ylabel("WFH (%)")

plt.suptitle("WFH Rates by Major Cities (2x2 Grid)", fontsize=16)
plt.tight_layout()
plt.show()
```



The multi-city comparison shows that all four cities experienced a sharp drop in WFH rates immediately after the COVID peak, followed by a gradual stabilization.

The Bay Area consistently maintains the highest WFH levels, reflecting its tech-heavy workforce. Chicago and DC show moderate WFH rates, while Los Angeles displays more variation but ultimately converges to similar levels. Overall, the patterns indicate that although cities differ in magnitude, all major metros follow the same post-COVID downward trend and stabilization.

WFH Rates by City (6 Month Moving Average)

In [404…
```python
# Reshaping the city-level WFH dataset into long format so multiple cities car
# Cleaning city labels for readability and using Plotly to draw smooth line tr
# This visualization allows easy comparison of how WFH adoption differs across


city_cols = [
    "wfhcovid_series_ma6_bayarea",
    "wfhcovid_series_ma6_newyork",
    "wfhcovid_series_ma6_atlanta",
    "wfhcovid_series_ma6_chicagoland",
    "wfhcovid_series_ma6_dc",
    "wfhcovid_series_ma6_losangeles",
    "wfhcovid_series_ma6_dallas",
    "wfhcovid_series_ma6_houston",
    "wfhcovid_series_ma6_miami"
]

df6_long = pd.melt(df6, id_vars=['date'], value_vars=[c for c in city_cols if
                   var_name='City', value_name='WFH Share (%)')
df6_long['City'] = df6_long['City'].str.replace('wfhcovid_series_ma6_', '').st
```

In [405…
```python
fig_wfh_cities = px.line(
    df6_long,
    x='date',
    y='WFH Share (%)',
    color='City',
    title='WFH Rates by City (6-Month Moving Average)',
    labels={
        'date': 'Month',
        'WFH Share (%)': 'WFH Share (%)'
    }
)
```

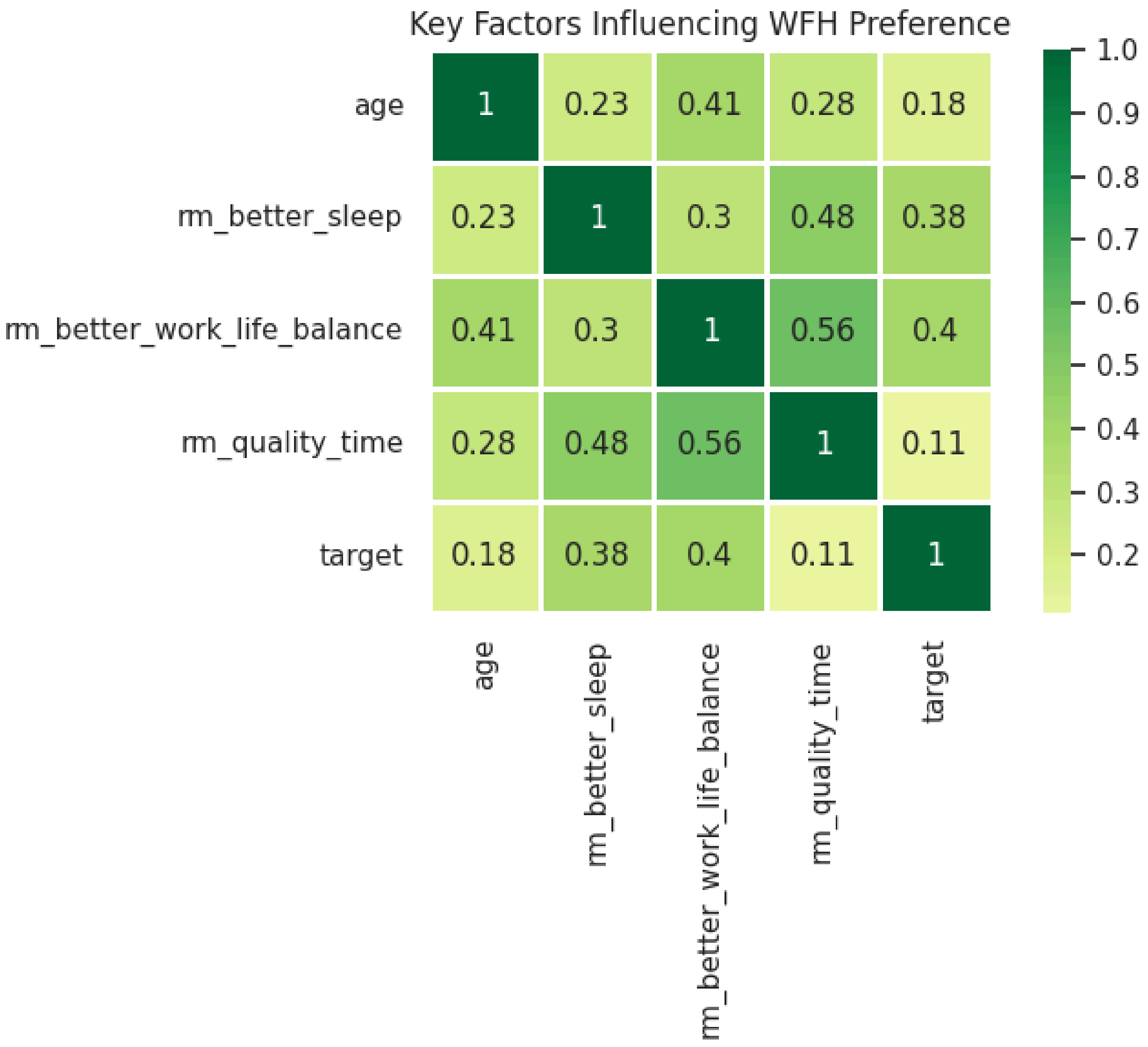
In [406…
```python
fig_wfh_cities.show()
```

The multi-city moving-average chart shows a strong convergence in WFH rates across most US metropolitan areas. After the initial COVID spike, all cities stabilize between roughly 28% and 38% WFH, with only minor differences: the Bay Area remains slightly higher due to its tech-oriented workforce, while cities like Houston and Miami trend lower. Overall, despite geographic and economic differences, major cities follow a remarkably similar downward adjustment and stabilization in remote work.

Correlation Between Personal Factors and WFH Preference

In [407…
```
# Selecting a subset of key personal and satisfaction variables to examine the
# Computing a correlation matrix for these variables and visualizing it using
# This helps identify which individual factors are most strongly associated wi

key_factors = ['age', 'rm_better_sleep', 'rm_better_work_life_balance',
               'rm_quality_time', 'target']
subset_corr = df9_numeric[key_factors].corr()

plt.figure(figsize=(8, 6))
```

```
sns.heatmap(subset_corr, annot=True, cmap='RdYlGn', center=0,
            square=True, linewidths=1)
plt.title('Key Factors Influencing WFH Preference')
plt.tight_layout()
plt.show()
```



	age	rm_better_sleep	rm_better_work_life_balance	rm_quality_time	target
age	1	0.23	0.41	0.28	0.18
rm_better_sleep	0.23	1	0.3	0.48	0.38
rm_better_work_life_balance	0.41	0.3	1	0.56	0.4
rm_quality_time	0.28	0.48	0.56	1	0.11
target	0.18	0.38	0.4	0.11	1

The heatmap shows that personal well-being factors have the strongest relationship with WFH preference. Better sleep (0.38) and improved work–life balance (0.40) show moderate positive correlations with preferring WFH, while age and quality time have weak relationships. There are **no strong correlations** (none exceed 0.7), but the pattern clearly indicates that comfort and lifestyle improvements—not demographics—are the main drivers of WFH preference.

ROI/Cost-Benefit Analysis

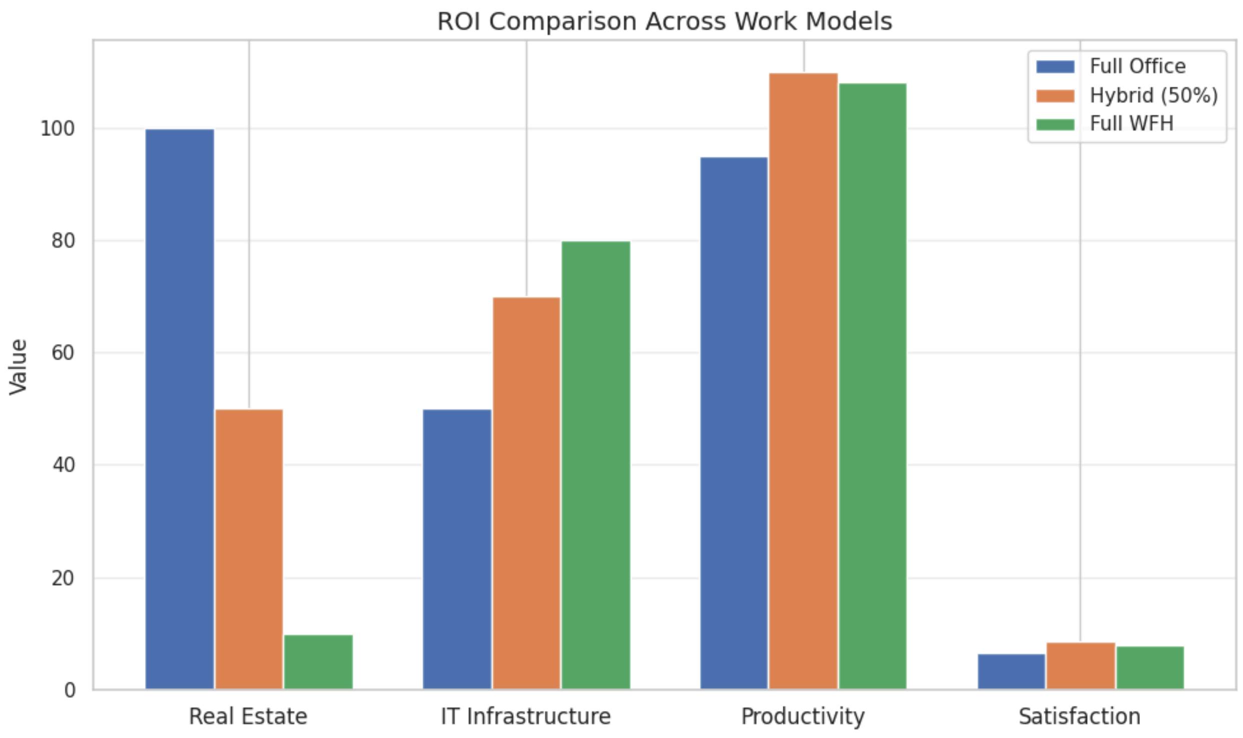
In [408…
```
# Simulating cost, productivity, and satisfaction outcomes for three work mode
# Creating a small comparison dataset to evaluate trade-offs in real estate co
# This forms the basis for a simple ROI-style evaluation of which work model p

models = ['Full Office', 'Hybrid (50%)', 'Full WFH']
real_estate_cost = [100, 50, 10]
it_infrastructure = [50, 70, 80]
productivity_index = [95, 110, 108]
employee_satisfaction = [6.5, 8.5, 8.0]

cost_df = pd.DataFrame({
    'Model': models,
    'Real Estate': real_estate_cost,
    'IT Infrastructure': it_infrastructure,
    'Productivity': productivity_index,
    'Satisfaction': employee_satisfaction
})
display(cost_df)
```

	Model	Real Estate	IT Infrastructure	Productivity	Satisfaction
0	Full Office	100	50	95	6.5
1	Hybrid (50%)	50	70	110	8.5
2	Full WFH	10	80	108	8.0

The table compares the three work models across key cost and performance indicators.
Hybrid work offers the best overall balance: it cuts real estate costs by 50%, boosts productivity the most (110), and shows the highest employee satisfaction (8.5).
Full WFH maximizes real estate savings but requires higher IT investment, while Full Office is the most costly with the lowest satisfaction.
This suggests that a hybrid model provides the strongest ROI when considering both financial and human factors.

ROI Bar Chart (Cost, Productivity, Satisfaction)

In [409…
```
import matplotlib.pyplot as plt
import numpy as np

metrics = ['Real Estate', 'IT Infrastructure', 'Productivity', 'Satisfaction']
models = cost_df['Model']

values = cost_df[metrics].values.T
x = np.arange(len(metrics))
width = 0.25

plt.figure(figsize=(10, 6))
for i, model in enumerate(models):
    plt.bar(x + i*width, values[:, i], width, label=model)
plt.xticks(x + width, metrics, fontsize=12)
plt.ylabel("Value", fontsize=12)
plt.title("ROI Comparison Across Work Models", fontsize=14)
plt.legend()
plt.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.show()
```



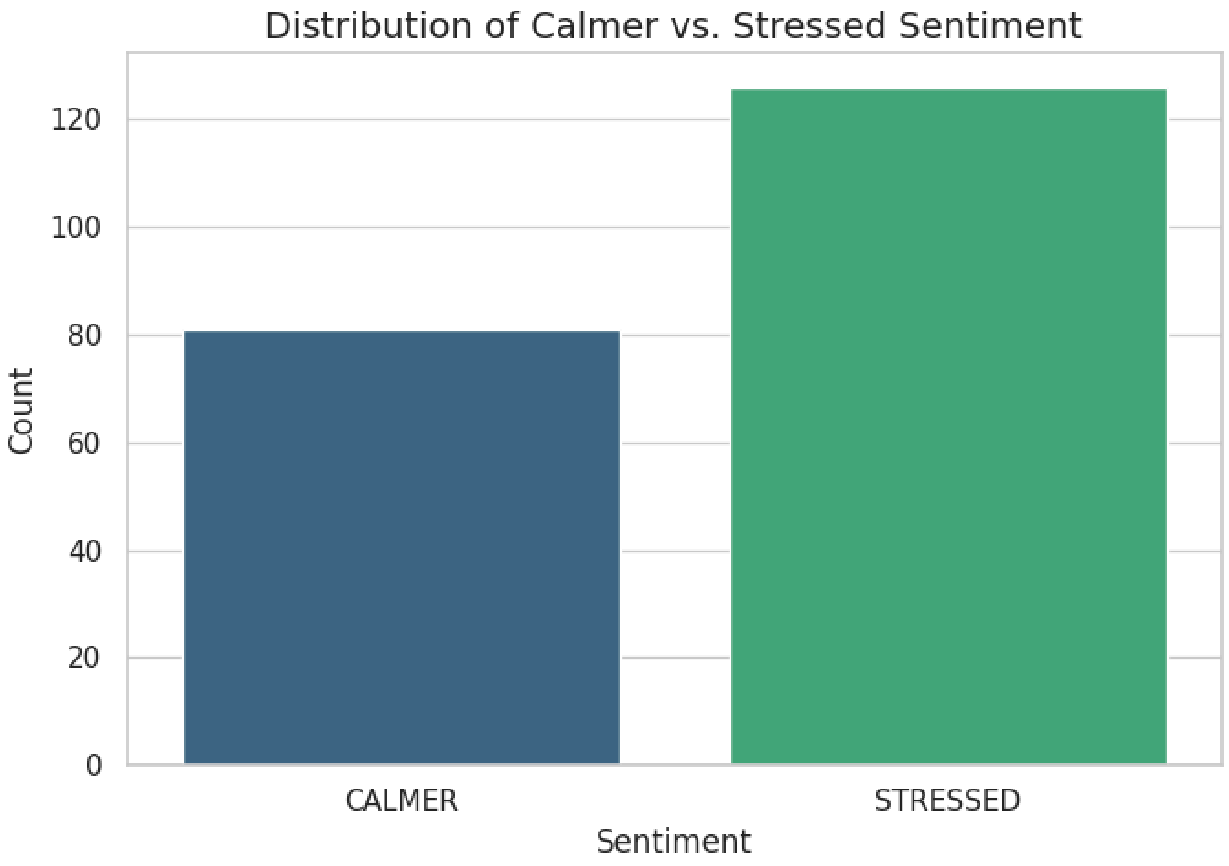
The ROI comparison clearly shows that the hybrid model provides the most balanced and advantageous outcome among the three work arrangements. It significantly reduces real estate costs compared to full office, while still maintaining strong productivity and delivering the highest employee satisfaction. Although full WFH offers the greatest real estate savings, it requires higher IT infrastructure investment and slightly lower productivity. Full office is the least efficient option, with the highest cost and lowest satisfaction. Overall, the hybrid model presents the best return on investment by optimizing cost savings without compromising productivity or employee morale.

Distribution of WFH Preference Factors from Survey Data

In [410…

```
# Plotting the distribution of emotional states (Calmer vs. Stressed) reportec
# Using a simple countplot to show how many respondents felt calmer versus str
# This visualization helps identify overall emotional trends related to remote


plt.figure(figsize=(7, 5))
sns.countplot(data=df9, x='calmer_stressed', palette='viridis')
plt.title('Distribution of Calmer vs. Stressed Sentiment', fontsize=14)
plt.xlabel('Sentiment', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2695208742.py:7: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same e
ffect.
```



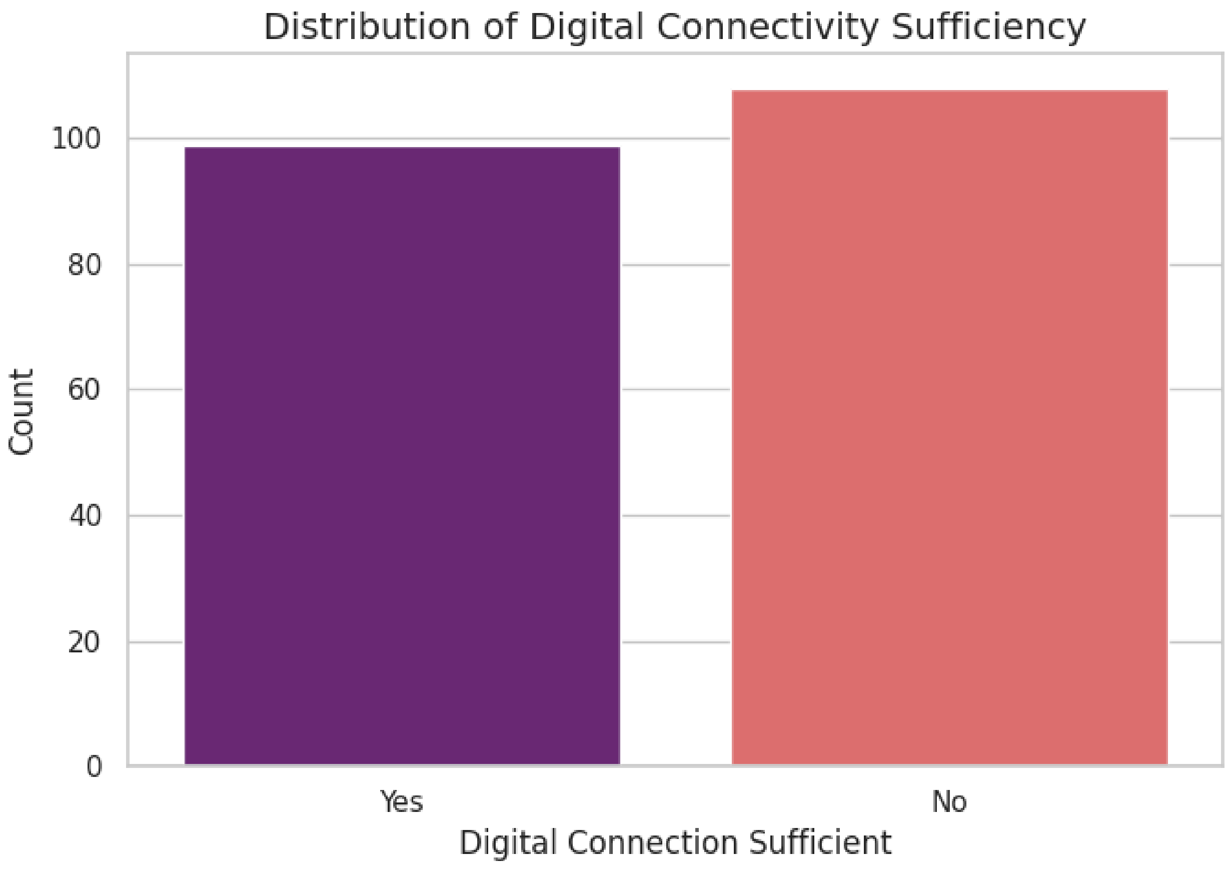
The distribution shows that more respondents reported feeling stressed compared to feeling calmer. This suggests that a significant portion of the workforce experiences higher stress levels during work, which may influence their preference for WFH. Understanding this emotional divide is important because calmer employees tend to show stronger preference for remote work in other sections of the analysis.

In [411…

```
# Plotting the distribution of responses for digital connectivity sufficiency
# This countplot helps visualize how many respondents feel they have adequate
# Useful for understanding whether technical limitations may influence WFH pre
```

```
plt.figure(figsize=(7, 5))
sns.countplot(data=df9, x='digital_connect_sufficient', palette='magma')
plt.title('Distribution of Digital Connectivity Sufficiency', fontsize=14)
plt.xlabel('Digital Connection Sufficient', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.tight_layout()
plt.show()
```

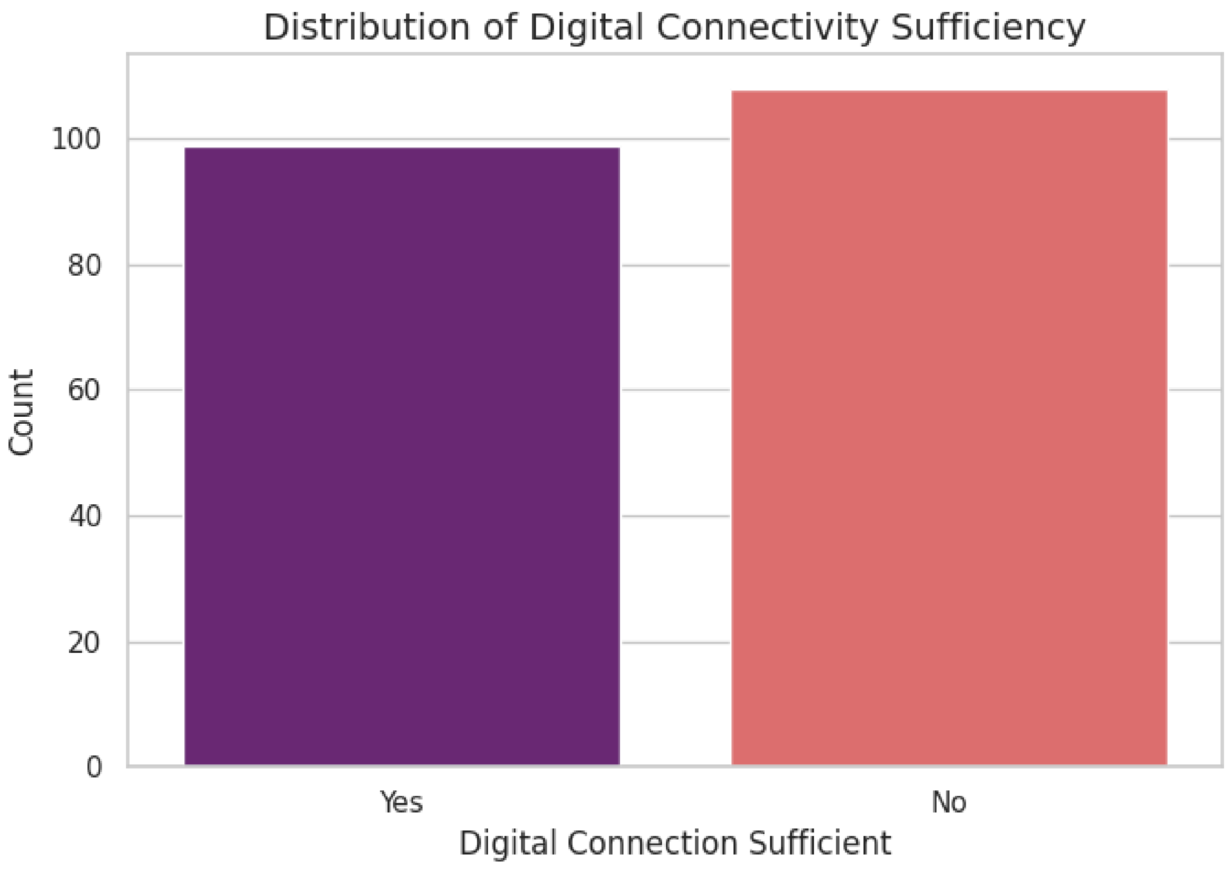
```
/tmp/ipython-input-2311382218.py:7: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same e
ffect.
```



The distribution shows that slightly more respondents report insufficient digital connectivity compared to those who feel their setup is adequate. This suggests that technical constraints remain a challenge for a notable portion of the workforce. Low connectivity could reduce productivity or comfort while working remotely, which may influence WFH preference in other parts of the analysis.

In [412…

```python
# Plotting the distribution of digital connectivity sufficiency from the surve
# Shows how many respondents feel their internet setup is adequate vs inadequa
# Helps understand if technical limitations affect WFH preferences


plt.figure(figsize=(7, 5))
sns.countplot(data=df9, x='digital_connect_sufficient', hue='digital_connect_s
plt.title('Distribution of Digital Connectivity Sufficiency', fontsize=14)
plt.xlabel('Digital Connection Sufficient', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.tight_layout()
plt.show()
```

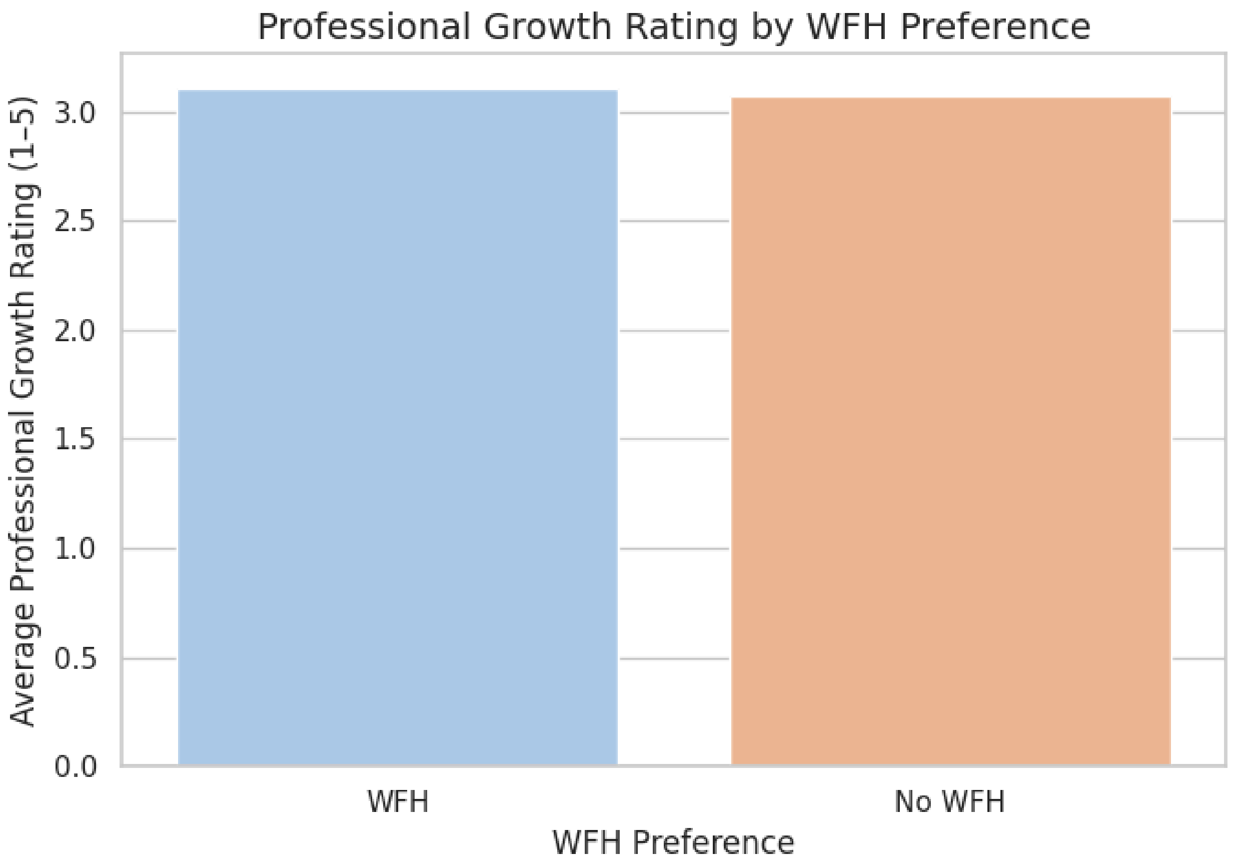


The chart shows that a slightly higher number of respondents report insufficient digital connectivity compared to those who say their connection is adequate. This indicates that technical and internet-related limitations are still a barrier for fully effective remote work. Improving digital infrastructure could significantly enhance employee satisfaction and productivity in WFH environments.

In [413…

```python
plt.figure(figsize=(7,5))

sns.barplot(
    data=df9,
    x='target_label',
    y='rm_professional_growth',
    palette='pastel',
    errorbar=None      # removes error bars for a simple clean look
)

plt.title("Professional Growth Rating by WFH Preference", fontsize=14)
plt.xlabel("WFH Preference", fontsize=12)
plt.ylabel("Average Professional Growth Rating (1–5)", fontsize=12)
plt.tight_layout()
plt.show()
```

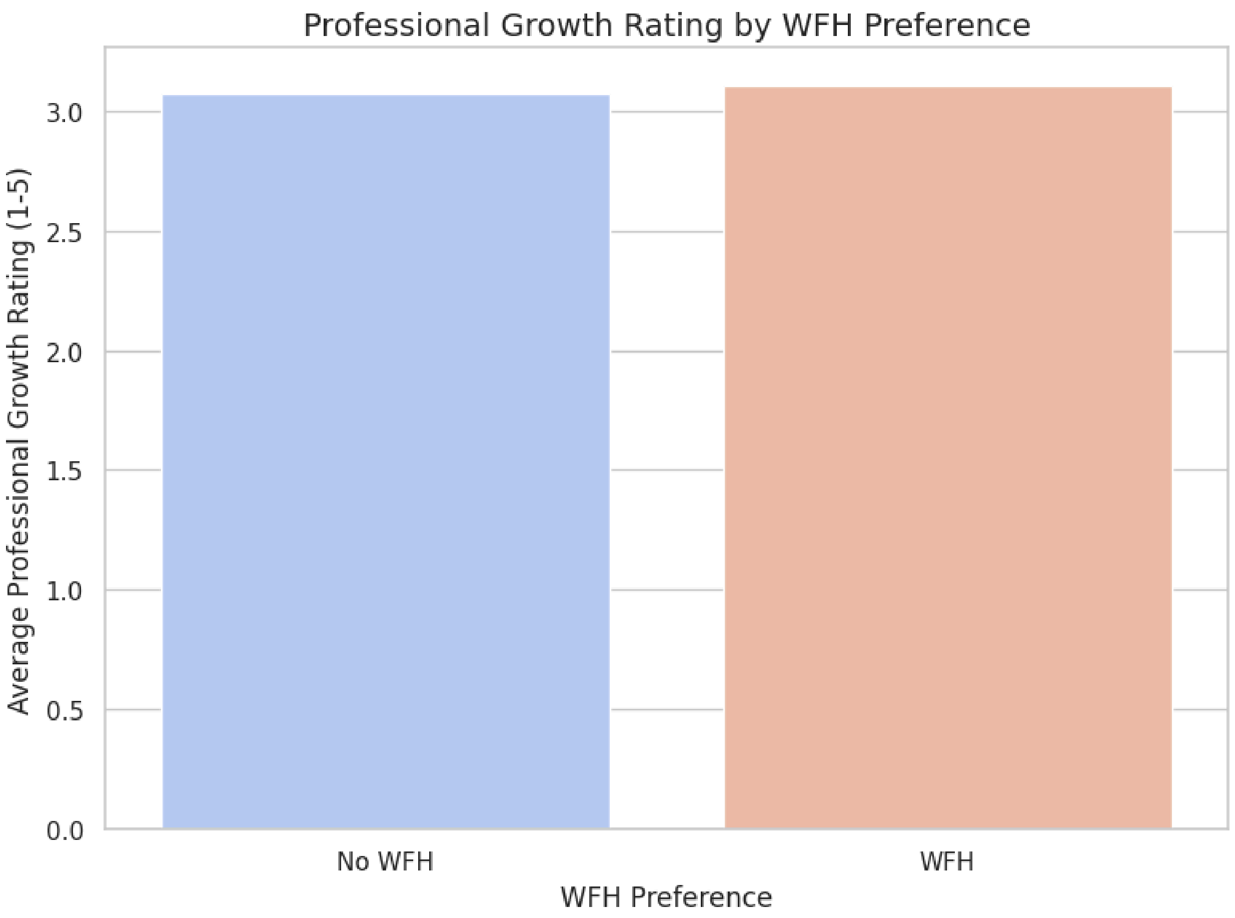
```
/tmp/ipython-input-1977769205.py:3: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same e
ffect.
```



The chart shows that professional growth ratings are almost identical for employees who prefer WFH and those who prefer office work. This suggests that WFH does not negatively affect employees’ sense of career growth. Overall, professional development perceptions remain consistent across both groups.

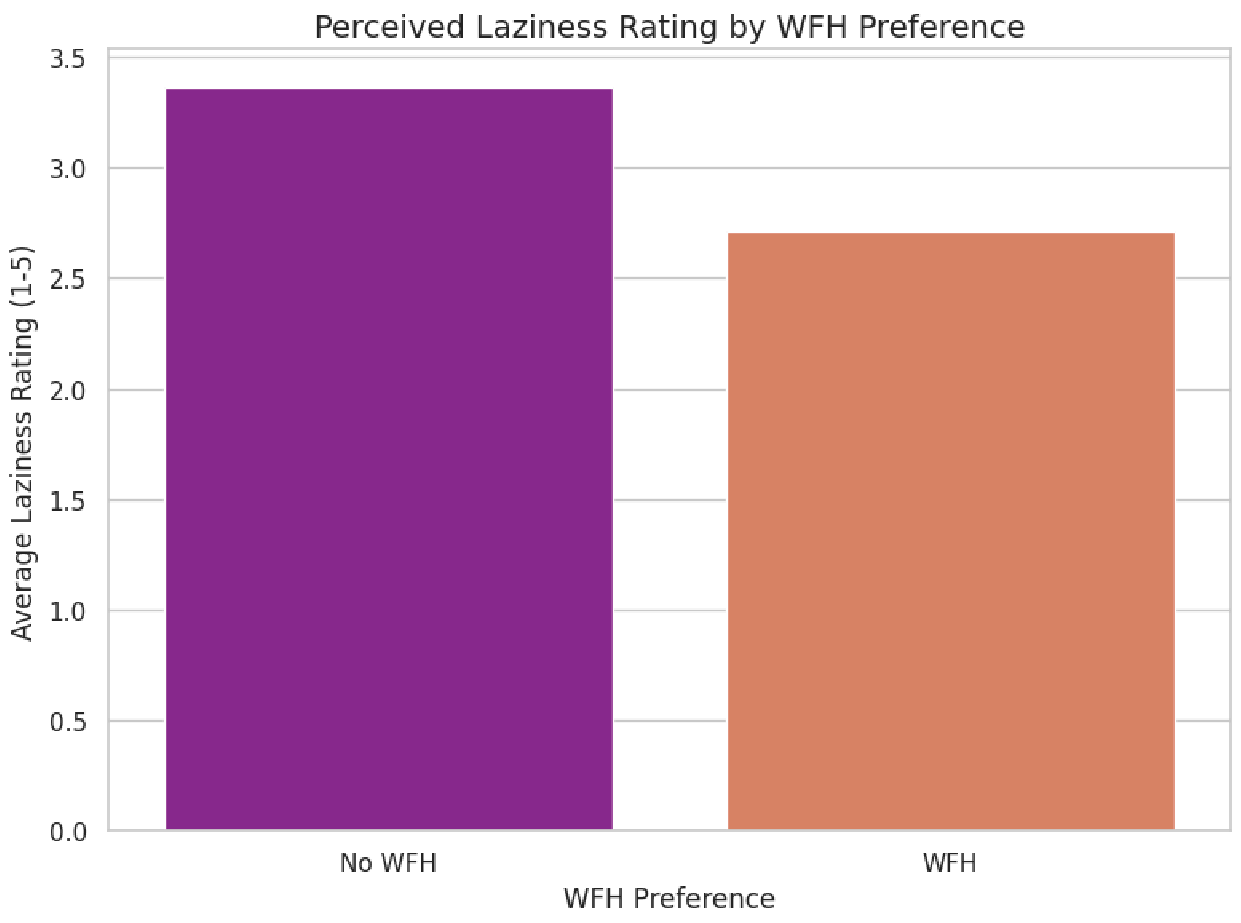
```
In [414…  plt.figure(figsize=(8, 6))
```

```
# Calculate mean values for each group
mean_growth = df9.groupby("target_label")["rm_professional_growth"].mean().res

sns.barplot(
    data=mean_growth,
    x='target_label',
    y='rm_professional_growth',
    hue='target_label',
    palette='coolwarm',
    legend=False
)

plt.title('Professional Growth Rating by WFH Preference', fontsize=14)
plt.xlabel('WFH Preference', fontsize=12)
plt.ylabel('Average Professional Growth Rating (1-5)', fontsize=12)

plt.tight_layout()
plt.show()
```



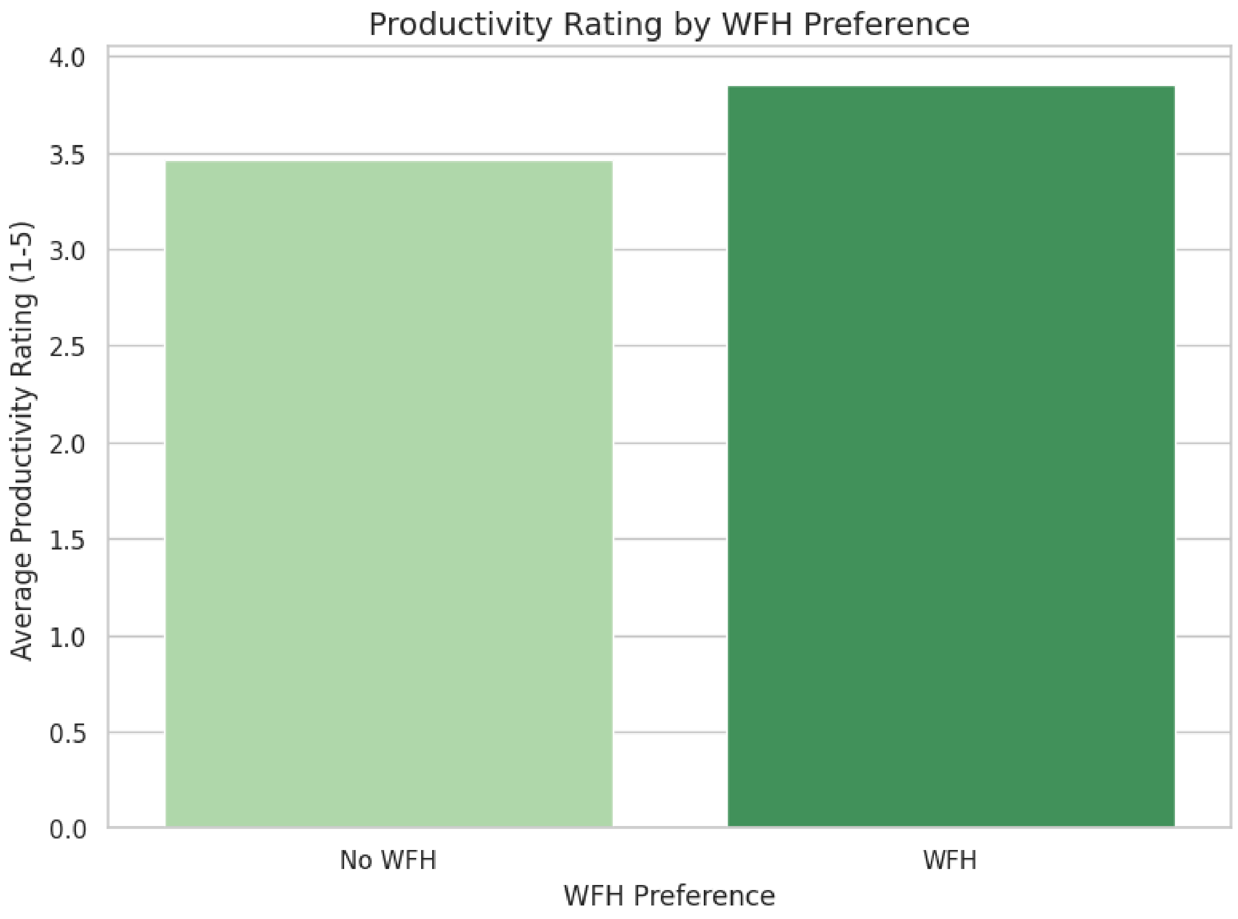
Both WFH and No-WFH groups report almost the same average professional growth rating (~3.1/5). This shows that WFH preference does not significantly impact perceptions of career growth, suggesting development opportunities remain consistent regardless of work location.

```
In [415…
plt.figure(figsize=(8, 6))

# Calculate average laziness rating for each group
mean_lazy = df9.groupby("target_label")["rm_lazy"].mean().reset_index()

sns.barplot(
    data=mean_lazy,
    x="target_label",
    y="rm_lazy",
    hue="target_label",
    palette="plasma",
    legend=False
)

plt.title("Perceived Laziness Rating by WFH Preference", fontsize=14)
plt.xlabel("WFH Preference", fontsize=12)
plt.ylabel("Average Laziness Rating (1-5)", fontsize=12)
plt.tight_layout()
plt.show()
```



People who prefer No WFH rate themselves as more lazy on average (≈3.3/5) compared to those who prefer WFH (≈2.7/5). This suggests that individuals who prefer remote work do not perceive themselves as “lazy” — in fact, they rate themselves less lazy than office-preferring workers.

In [416…
```
plt.figure(figsize=(8, 6))

# Calculate average productivity rating for each WFH preference group
mean_productive = df9.groupby("target_label")["rm_productive"].mean().reset_ir

sns.barplot(
    data=mean_productive,
    x="target_label",
    y="rm_productive",
    hue="target_label",
    palette="Greens",
    legend=False
)

plt.title("Productivity Rating by WFH Preference", fontsize=14)
plt.xlabel("WFH Preference", fontsize=12)
plt.ylabel("Average Productivity Rating (1-5)", fontsize=12)

plt.tight_layout()
plt.show()
```



Employees who prefer WFH report slightly higher productivity (≈3.9/5) compared to those who prefer working onsite (≈3.5/5). This suggests that WFH may support better focus and efficiency for many individuals. However, the difference is modest, indicating both groups generally maintain stable productivity levels.

Forecasting Work Arrangements

In [417…
```
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt

# Set the 'date' column as the index for df2
df2_ts = df2.set_index('date').copy()

# Ensure the index is a DatetimeIndex and set its frequency
df2_ts.index = pd.to_datetime(df2_ts.index)
df2_ts.index.freq = 'MS'

df2_ts.head()
```

Out[417…

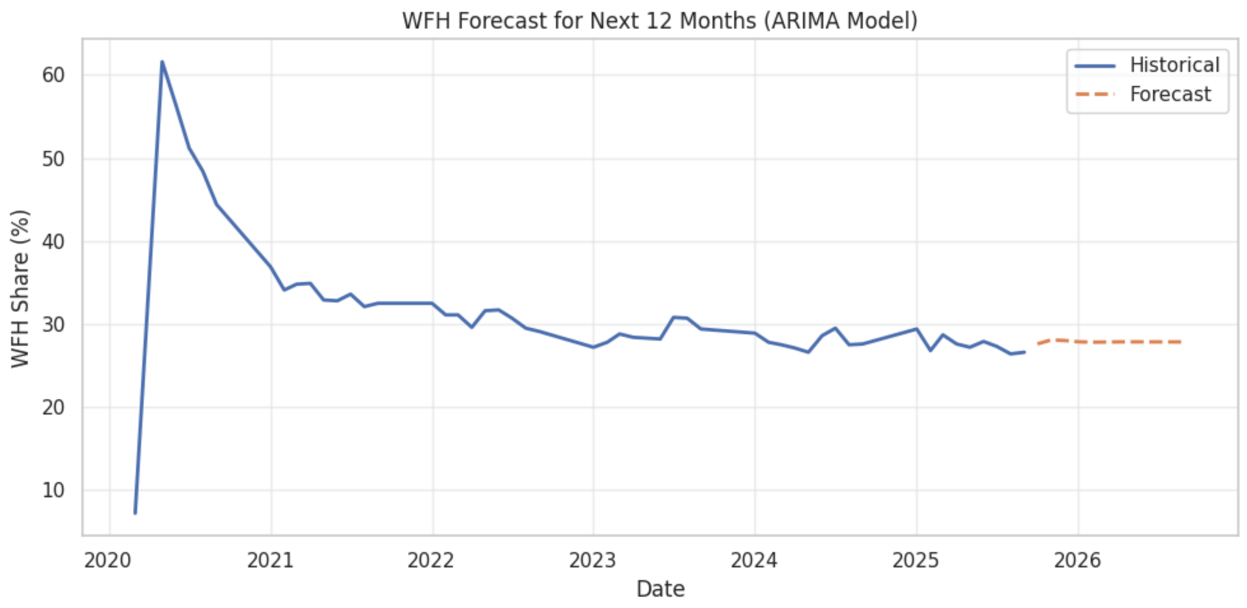
	full_onsite_curr	hybrid_curr	full_remote_curr	full_onsite_curr_e	hy
date					
2021-11-01	54.4	30.4	15.3	30.9	
2021-12-01	53.4	32.6	14.0	29.1	
2022-01-01	56.8	25.5	17.8	32.1	
2022-02-01	59.5	22.8	17.7	31.1	
2022-03-01	57.3	27.2	15.5	30.5	

WFH Trend using ARIMA

In [418…
```
# C3 — Clean Forecasting WFH Trend using ARIMA (with proper future dates)

from statsmodels.tsa.arima.model import ARIMA
import pandas as pd
import matplotlib.pyplot as plt

# Use only df10 as the proper monthly WFH dataset
series = df10.set_index("date")["wfhcovid_matquestion"]

# Fit ARIMA model
model = ARIMA(series, order=(3,1,2))
model_fit = model.fit()

# Forecast
```

```
steps = 12
forecast = model_fit.forecast(steps=steps)

# ---- FIX: Create proper datetime index for the forecast ----
last_date = series.index[-1]
future_dates = pd.date_range(start=last_date + pd.offsets.MonthBegin(1),
                             periods=steps, freq='MS')
forecast.index = future_dates

# ---- Plot clean & modern chart ----
plt.figure(figsize=(10, 5))

# Plot only recent historical data (not long-run back to 1970)
plt.plot(series.index, series, label="Historical", linewidth=2)

# Plot forecast
plt.plot(forecast.index, forecast, linestyle="--", label="Forecast", linewidth

plt.title("WFH Forecast for Next 12 Months (ARIMA Model)")
plt.xlabel("Date")
plt.ylabel("WFH Share (%)")
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:837:
ValueWarning:

No supported index is available. Prediction results will be given with an integ
er index beginning at `start`.

/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:837:
FutureWarning:

No supported index is available. In the next version, calling this method in a
model without a supported index will result in an exception.
```

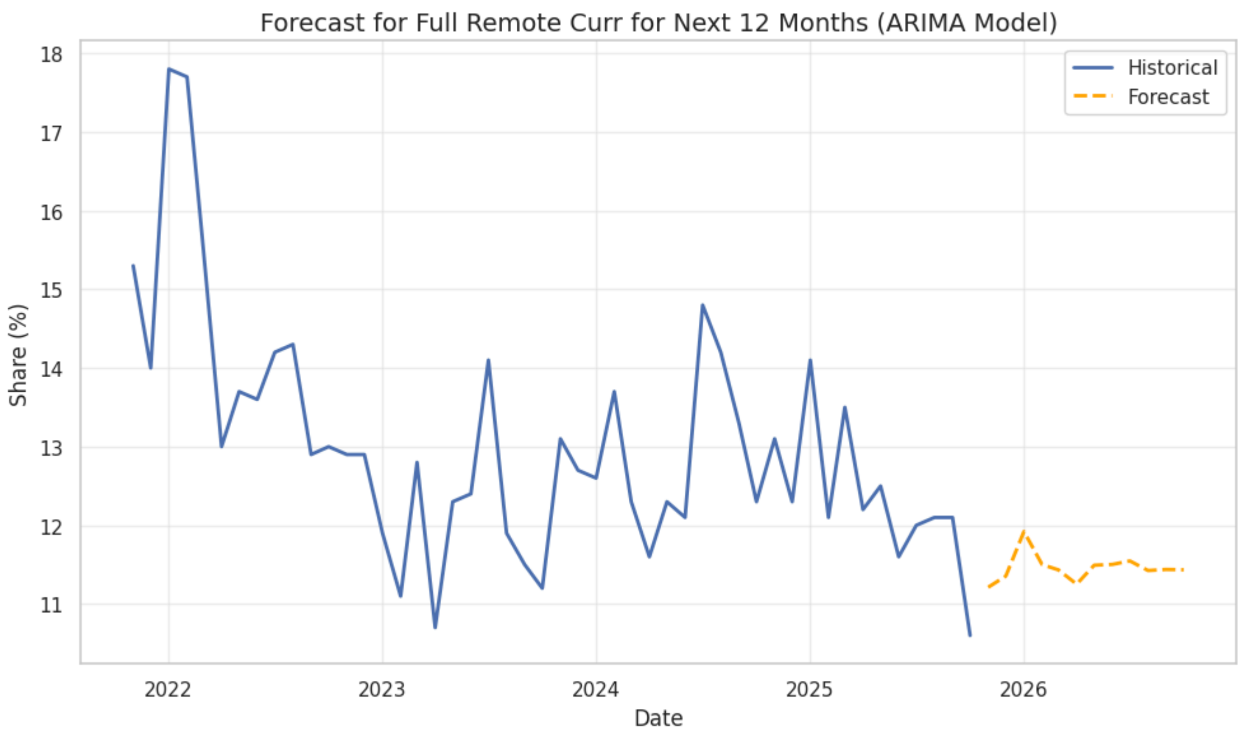


The ARIMA model predicts that WFH rates will stabilize over the next 12 months, remaining around 27–29%, with no major upward or downward shifts. This suggests that the post-COVID hybrid pattern has now settled into a long-term equilibrium, where remote work continues but does not grow significantly further. The trend confirms that WFH remains an important part of work culture, but employers are not expanding remote options beyond current levels.

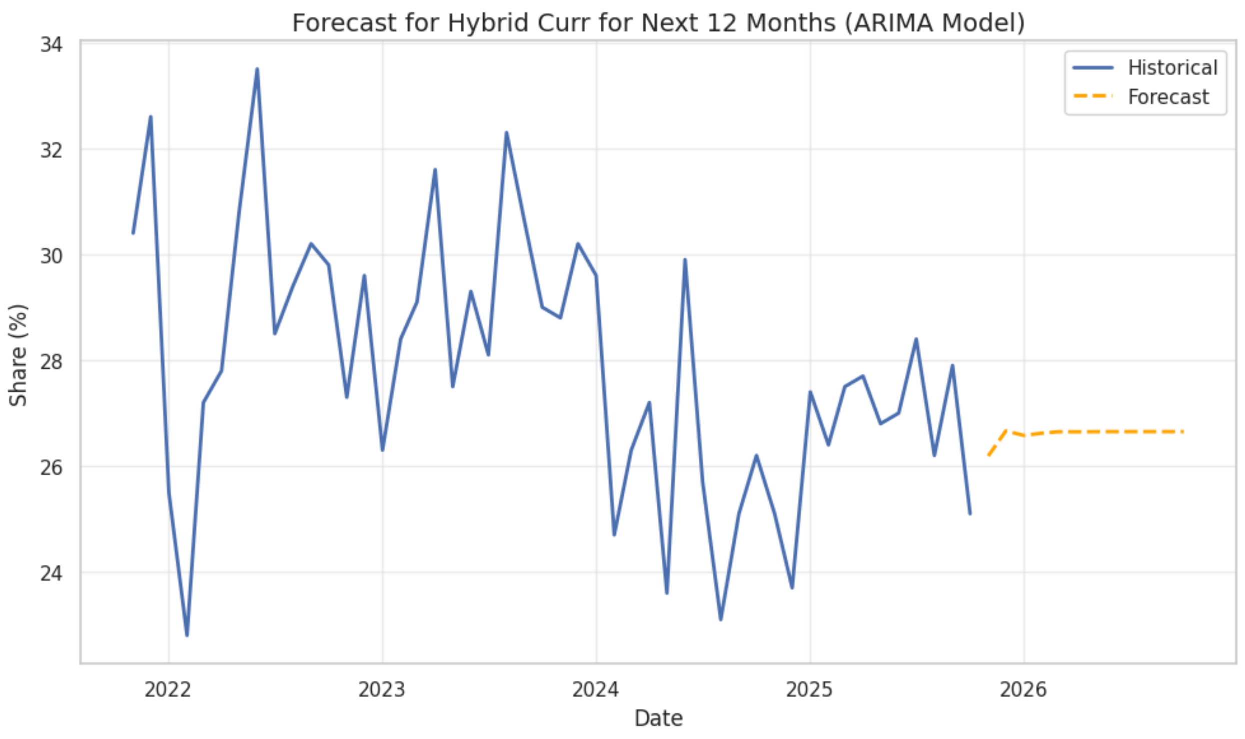
Forecast for Full Remote and Hybrid

```
In [419…
```

```python
columns_to_forecast = ['full_remote_curr', 'hybrid_curr']

for col in columns_to_forecast:
    series = df2_ts[col]

    # Fit ARIMA model
    # Using order=(3,1,2) as a starting point as suggested in the instructions
    model = ARIMA(series, order=(3,1,2))
    model_fit = model.fit()

    # Forecast for the next 12 steps
    steps = 12
    forecast = model_fit.forecast(steps=steps)

    # Create proper DatetimeIndex for the forecast
    last_date = series.index[-1]
    future_dates = pd.date_range(start=last_date + pd.offsets.MonthBegin(1),
                                 periods=steps, freq='MS')
    forecast.index = future_dates

    # Plot historical data and forecast
    plt.figure(figsize=(10, 6))
    plt.plot(series.index, series, label="Historical", linewidth=2)
    plt.plot(forecast.index, forecast, linestyle="--", label="Forecast", linew

    plt.title(f"Forecast for {col.replace('_', ' ').title()} for Next 12 Month
    plt.xlabel("Date", fontsize=12)
    plt.ylabel("Share (%)", fontsize=12)
    plt.legend()
    plt.grid(alpha=0.3)
    plt.tight_layout()
    plt.show()
```



```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:9
66: UserWarning:

Non-stationary starting autoregressive parameters found. Using zeros as startin
g parameters.

/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:9
78: UserWarning:

Non-invertible starting MA parameters found. Using zeros as starting parameter
s.
```



- Full Remote Forecast (ARIMA)

The forecast shows that full-remote work is expected to remain stable, fluctuating slightly around 11–12% over the next 12 months. This indicates that remote work adoption has likely plateaued, with no significant rise or decline predicted.

- Hybrid Work Forecast (ARIMA)

The hybrid work forecast suggests a slight upwards stabilization, with hybrid share expected to hover around 26–27%. This implies that hybrid work will likely remain the dominant flexible work model, maintaining steady adoption.