In [2]:

```
# Applied for: Administrative Data Analyst - Req. #581696
# Data sets

import pandas as pd
coverage_df = pd.read_excel(r"D:\Downloads\GLOBAL_DATAFLOW_2018-2022.xlsx")
status_df = pd.read_excel(r"D:\Downloads\On-track and off-track countries.xlsx")
population_df = pd.read_excel(r"D:\Downloads\WPP2022_GEN_F01_DEMOGRAPHIC_INDICATORS_COMPACT
```

In [16]:

```
# 1. Working With Data from UNICEF Global Data Repository

# Filter for 2022 and desired indicators

coverage_filtered = coverage_df[
    (coverage_df['TIME_PERIOD'] == 2022) &
    (coverage_df['Indicator'].isin([
        'Antenatal care 4+ visits - percentage of women (aged 15-49 years) attended at leas
        'Skilled birth attendant - percentage of deliveries attended by skilled health pers
    ]))
]

# Pivot so that ANC4 and SBA are in separate columns
coverage_pivot = coverage_filtered.pivot_table(
    index='Geographic area',
    columns='Indicator',
    values='OBS_VALUE'
).reset_index()

# Renaming columns for simplicity
coverage_pivot.rename(columns={
    'Geographic area': 'Country',
    'Antenatal care 4+ visits - percentage of women (aged 15-49 years) attended at least fo
    'Skilled birth attendant - percentage of deliveries attended by skilled health personne
}, inplace=True)
```

In [17]:

```
# 2. Working with on-track and off-track countries data

# Select and rename relevant columns
country_status = status_df[['OfficialName', 'Status.U5MR']].copy()
country_status.rename(columns={
    'OfficialName': 'Country',
    'Status.U5MR': 'Track_Status'
}, inplace=True)

# Optional: strip whitespaces from country names
country_status['Country'] = country_status['Country'].str.strip()
```

In [18]:

```
print(country_status.head())
print(country_status['Track_Status'].unique())
```

```
       Country         Track_Status
0  Afghanistan  Acceleration Needed
1       Angola  Acceleration Needed
2     Anguilla             Achieved
3      Albania             Achieved
4      Andorra             Achieved
['Acceleration Needed' 'Achieved' 'On Track']
```

In [19]:

```
# Normalize to lowercase for easier comparison
country_status['Track_Status'] = country_status['Track_Status'].str.lower().str.strip()

# Map to simplified binary classification
country_status['Track_Status'] = country_status['Track_Status'].map({
    'achieved': 'On-track',
    'on track': 'On-track',
    'acceleration needed': 'Off-track'
})

# Confirm it worked
print(country_status['Track_Status'].value_counts())
```

```
On-track     141
Off-track     59
Name: Track_Status, dtype: int64
```

In [20]:

```
# 3. Working with Population Data: UN World Population Prospects, 2022

# Subset the relevant columns
subset_df = population_df[[
    'Region, subregion, country or area *',
    'Year',
    'Under-Five Mortality (deaths under age 5 per 1,000 live births)'
]]

# renaming columns for clarity and ease
subset_df = subset_df.rename(columns={
    'Region, subregion, country or area *': 'Country',
    'Under-Five Mortality (deaths under age 5 per 1,000 live births)': 'U5MR'
})

# Check the first few rows
subset_df.head()
```

Out[20]:

	Country	Year	U5MR
0	WORLD	1950.0	224.01
1	WORLD	1951.0	219.119
2	WORLD	1952.0	212.198
3	WORLD	1953.0	206.944
4	WORLD	1954.0	202.18

In [46]:

```
# Prep for merging all the 3 datasets to create a suitable dataframe for further analysis

# Step 1: Filter population_df for projected births in 2022
population_2021 = population_df[
    population_df['Year'] == 2021
][['Region, subregion, country or area *', 'Year', 'Births (thousands)']].copy()

# Rename for consistency
population_2021.rename(columns={
    'Region, subregion, country or area *': 'Country',
    'Births (thousands)': 'Projected_Births'
}, inplace=True)

# Optional: Clean country names
population_2021['Country'] = population_2021['Country'].str.strip()
```

In [47]:

```
# Step 2: Merge all 3 datasets on Country
merged_df = (
    coverage_pivot
    .merge(country_status, on='Country', how='inner')
    .merge(population_2021, on='Country', how='inner')
)

# Drop any rows with missing values in ANC4, SBA, or Projected_Births
merged_df = merged_df.dropna(subset=['ANC4', 'SBA', 'Projected_Births'])
```

In [48]:

```
# Step 3: Define function to compute weighted average

def weighted_avg(df, value_column, weight_column='Projected_Births'):
    weights = df[weight_column]
    values = df[value_column]

    # Remove rows where weights are zero or NaN
    valid = (weights > 0) & weights.notna() & values.notna()
    weights = weights[valid]
    values = values[valid]

    if weights.sum() == 0:
        return 0  # or return 0, depending on how you want to handle it

    return (values * weights).sum() / weights.sum()
```

In [49]:

```
# Step 4: Group by Track_Status and compute population-weighted averages
results = merged_df.groupby('Track_Status').apply(
    lambda group: pd.Series({
        'Weighted_ANC4': weighted_avg(group, 'ANC4'),
        'Weighted_SBA': weighted_avg(group, 'SBA')
    })
)

# Display the results
print(results)
```

```
              Weighted_ANC4  Weighted_SBA
Track_Status
Off-track         71.006587     87.399835
On-track          62.829343     82.245623
```

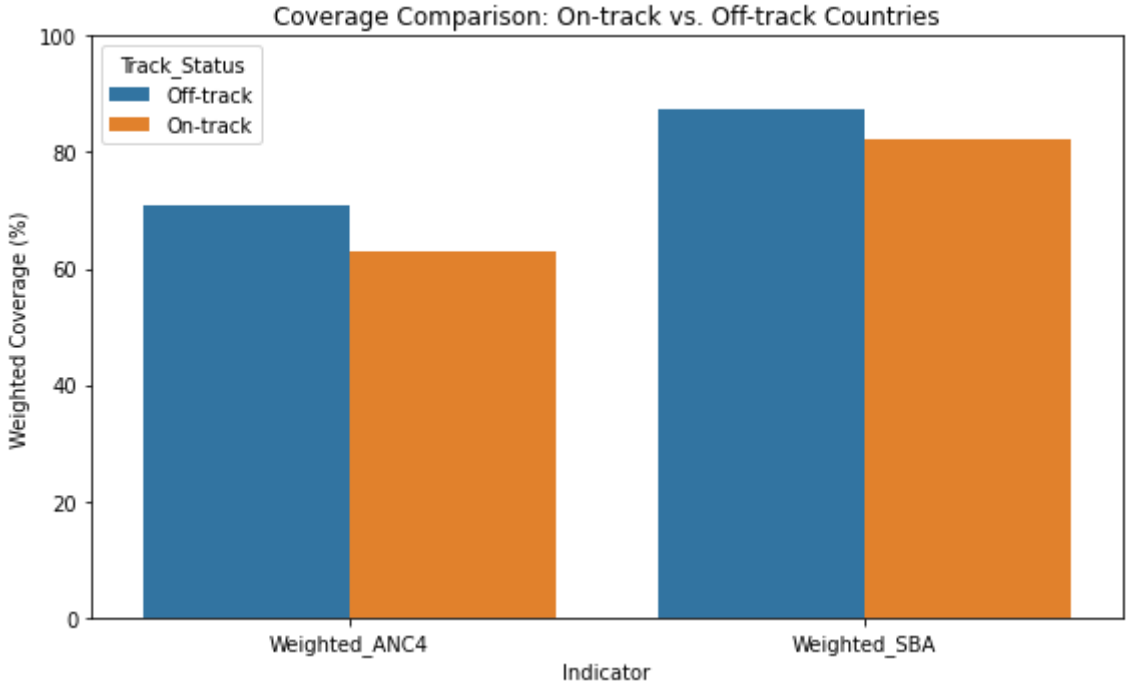
In [58]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Calculate weighted averages by Track_Status
coverage_df = merged_df.groupby('Track_Status').apply(
    lambda g: pd.Series({
        'Weighted_ANC4': (g['ANC4'] * g['Projected_Births']).sum() / g['Projected_Births'].
        'Weighted_SBA':  (g['SBA'] * g['Projected_Births']).sum() / g['Projected_Births'].s
    })
).reset_index()

# Step 2: Melt to long format
melted_df = coverage_df.melt(
    id_vars='Track_Status',
    value_vars=['Weighted_ANC4', 'Weighted_SBA'],
    var_name='Indicator',
    value_name='Coverage'
)

# Step 3: Plot
plt.figure(figsize=(8, 5))
sns.barplot(data=melted_df, x='Indicator', y='Coverage', hue='Track_Status')
plt.title('Coverage Comparison: On-track vs. Off-track Countries')
plt.ylabel('Weighted Coverage (%)')
plt.ylim(0, 100)
plt.tight_layout()
plt.show()
```



In [60]:

```
# Reporting
# Step 1

# Save the figure as an image
plt.figure(figsize=(8, 5))
sns.barplot(data=melted_df, x='Indicator', y='Coverage', hue='Track_Status')
plt.title('Coverage Comparison: On-track vs. Off-track Countries')
plt.ylabel('Weighted Coverage (%)')
plt.ylim(0, 100)
plt.tight_layout()
plt.savefig(r"C:\Users\lenovo\Documents\coverage_comparison.png")
plt.close()
```

In [61]:

```
interpretation = """
The bar plot compares the weighted coverage of ANC4 and SBA indicators between on-track and
On-track countries show slightly lower coverage than off-track countries for both indicator
that countries initially off-track may have received more focused intervention efforts. How
assumes accurate and consistent reporting of coverage and projected birth data. Any gaps in
may affect the reliability of the comparison.
"""
```

In [65]:

```
html_content = f"""
<html>
<head><title>Coverage Comparison Report</title></head>
<body>
    <h1>Coverage Comparison Report</h1>
    <h2>Visualization</h2>
    <img src="C:\\Users\\lenovo\\Documents\\coverage_comparison.png" alt="Coverage Comparis

    <h2>Interpretation</h2>
    <p>{interpretation}</p>
</body>
</html>
"""

# Save as HTML
with open(r"C:\Users\lenovo\Documents\coverage_report.html", "w") as f:
    f.write(html_content)
```

In [ ]: