Starting code

In [1]:
```
# the usual imports.
import pprint as pp
import json
import plotly.express as px
import plotly.graph_objects as go
```

In [2]:
```
# this function just loads the data from files, there is no need to understand how it does it.
def load_json_file_named(file_name):
    try:
        loaded_data = []
        file_location = f"{file_name}"
        with open(file_location, 'r') as file: # or f"data/{file_name}" depending on your files
            loaded_data =  json.load(file)
    except OSError as e:
        print(f"Error. Does the file exist in this folder? {file_location}\n\n {e}")
    return loaded_data

boards_info = load_json_file_named('../data/nhs_boards_25.json')
covid_days = load_json_file_named('../data/nhs_covid_days_25.json')

# check if data loaded:
print(f"Boards: {len(boards_info)}")
print(f"Days: {len(covid_days)}")
```

```
Boards: 14
Days: 749
```

End of starting code!

Report 1:

Report 1 code:

In [3]:
```
# Load and process ONLY 2020 COVID death data by board
def calculate_total_deaths_per_board(covid_data):
    death_totals = {}
    for day in covid_data:
        if day["day"]["year"] == 2020:                # limit to 2020
            for board_name, board_info in day["individual_boards"].items():
                deaths = board_info["new_died"]
                death_totals[board_name] = death_totals.get(board_name, 0) + deaths
    return death_totals

# Assign it
death_totals = calculate_total_deaths_per_board(covid_days)

# Assertions to validate output
assert isinstance(death_totals, dict), "death_totals should be a dictionary"
assert len(death_totals) >= 14, "Expected at least 14 NHS boards"
assert all(isinstance(value, int) and value >= 0 for value in death_totals.values()), "All death counts shoul

print("2020 death totals tests passed")
```

```
2020 death totals tests passed
```

In [4]:
```
# Extract board population and 2020 budget
def extract_budget_and_population(board_data):
    summary = {}
    for board in board_data:
        name = board["board_name"]
        summary[name] = {
            "population": board["population"]["patients"],
            "budget_2020": board["budget_year"]["20"]
        }
    return summary

budget_info = extract_budget_and_population(boards_info)

# Assert: Each board should have population and 2020 budget data
assert isinstance(budget_info, dict), "budget_info should be a dictionary"
assert len(budget_info) >= 14, "Expected at least 14 NHS boards in budget info"
for board, data in budget_info.items():
    assert "population" in data and "budget_2020" in data, f"Missing data for {board}"
    assert data["population"] > 0, f"Population must be positive for {board}"
    assert data["budget_2020"] > 0, f"Budget must be positive for {board}"

print("tests passed")
```

```
tests passed
```

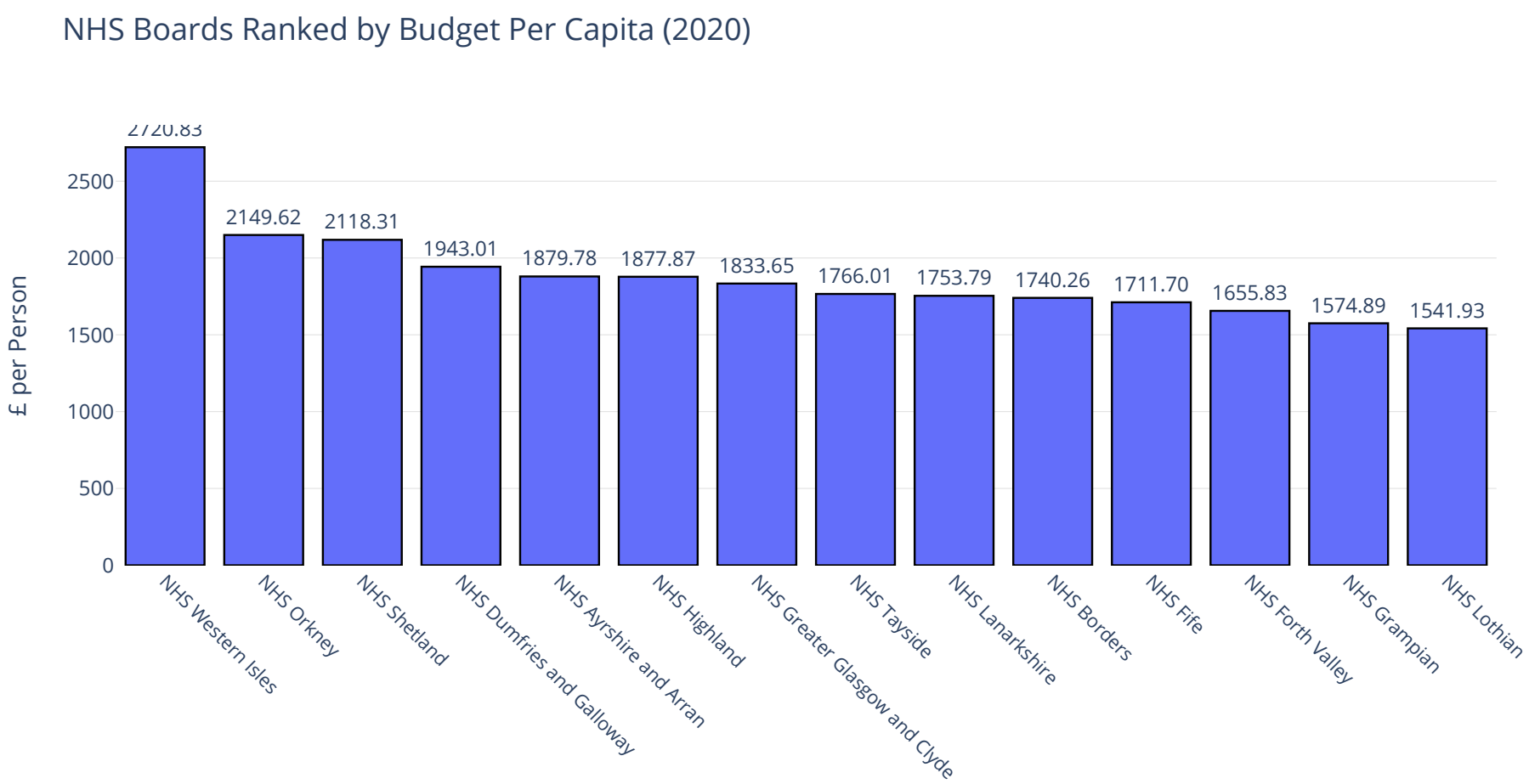
In [5]:
```
# Combine datasets: Calculate deaths per 100k and budget per capita
def create_combined_analysis(death_data, board_summary):
    combined = []
    for board, total_deaths in death_data.items():
        if board in board_summary:
            pop = board_summary[board]["population"]
            budget = board_summary[board]["budget_2020"]

            combined.append({
                "board": board,
                "population": pop,
                "budget_2020": budget,
                "total_deaths": total_deaths,
                "deaths_per_100k": round((total_deaths / pop) * 100000, 2),
                "budget_per_capita": round(budget / pop, 2)
            })
    return combined

combined_budget_data = create_combined_analysis(death_totals, budget_info)

# Assert: All calculated values should be present and logical
assert isinstance(combined_budget_data, list), "combined_budget_data should be a list"
assert len(combined_budget_data) > 0, "Combined dataset should not be empty"
for item in combined_budget_data:
    assert "deaths_per_100k" in item and "budget_per_capita" in item, "Missing calculated metrics"
    assert item["deaths_per_100k"] >= 0, "Deaths per 100k should be non-negative"
    assert item["budget_per_capita"] > 0, "Budget per capita should be positive"

print("tests passed")
```

```
tests passed
```

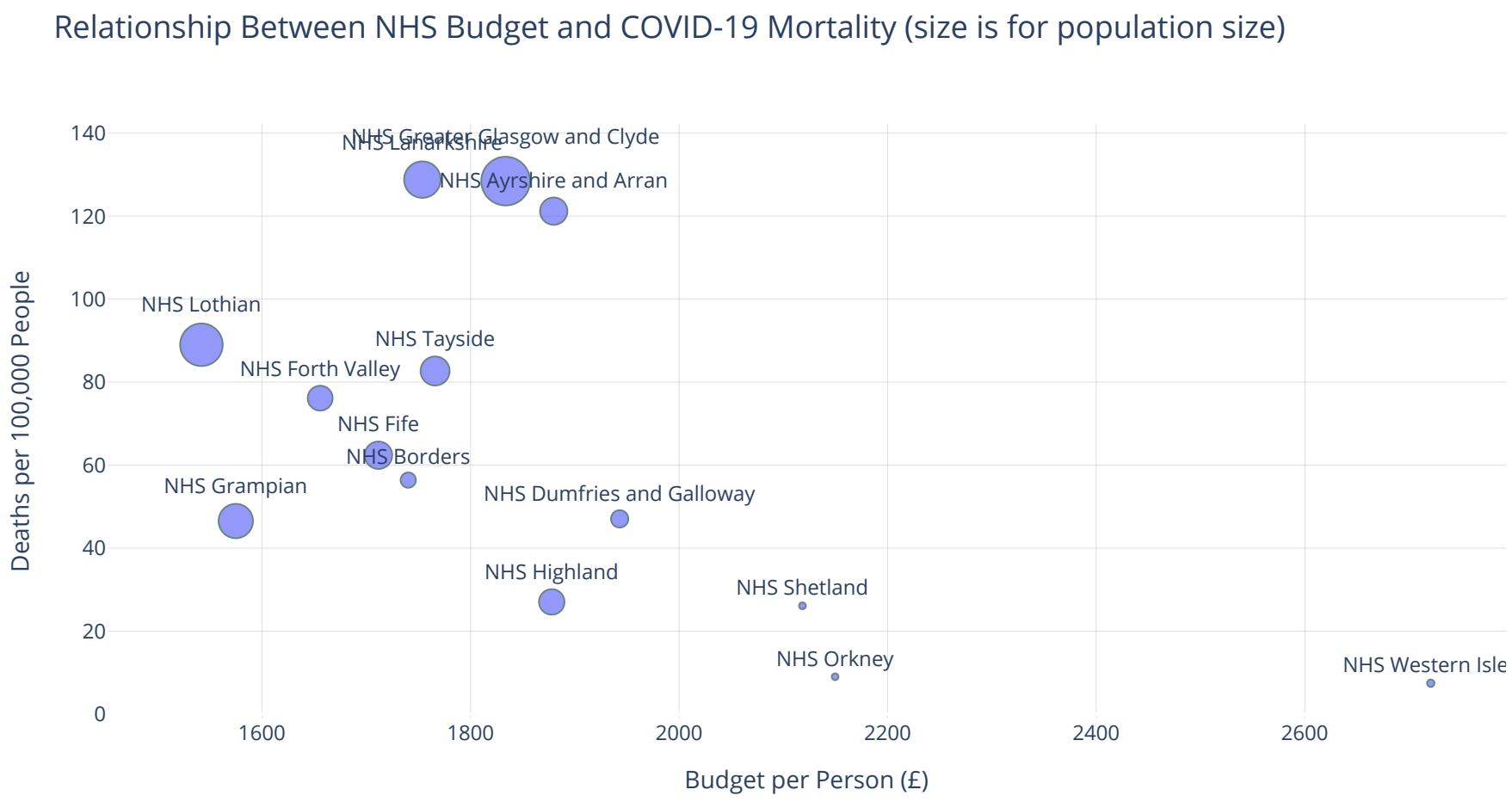
Report 1 visualisation:

In [6]:

```
# Sort data by budget per capita (descending)

sorted_budget = sorted(combined_budget_data, key=lambda x: x["budget_per_capita"], reverse=True)

# Create bar chart

fig1 = px.bar(
    sorted_budget,
    x="board",
    y="budget_per_capita",

    title="NHS Boards Ranked by Budget Per Capita (2020)",
    labels={
        "budget_per_capita": "Budget per Person (£)",
        "board": "NHS Board"
    },
    text="budget_per_capita",
    hover_data=["population", "budget_2020", "total_deaths"]  # Add rich hover info
)

# Update text and layout styling

fig1.update_traces(
    texttemplate='%{text:.2f}',
    textposition='outside',
    marker_line_color='black',
    marker_line_width=1.2
)

fig1.update_layout(
    title_font_size= 18,
    xaxis_tickangle= 45,
    xaxis_tickfont= dict(size= 11),
    yaxis=dict(title="£ per Person", gridcolor="lightgrey"),
    plot_bgcolor="white",
    uniformtext_minsize= 8,
    uniformtext_mode='hide',
    showlegend=False
)

fig1.show()
```



In [7]:
```
fig2 = px.scatter(
    combined_budget_data,
    x="budget_per_capita",
    y="deaths_per_100k",
    text="board",                                        # Show board name on the chart

    size="population",                                   # Optional: represent population size
    hover_data=["population", "budget_2020", "total_deaths"],
    title="Relationship Between NHS Budget and COVID-19 Mortality (size is for population size)",
    labels={
        "budget_per_capita": "Budget per Person (£)",
        "deaths_per_100k": "Deaths per 100,000 People",
        "board": "NHS Board"
    }
)

# Improve label appearance and marker styling

fig2.update_traces(
    textposition='top center',
    marker=dict(line=dict(width=1, color='DarkSlateGrey'))
)

# Refine the layout for clarity and presentation

fig2.update_layout(
    title_font_size=18,
    xaxis=dict(title="Budget per Person (£)", gridcolor="lightgrey"),
    yaxis=dict(title="Deaths per 100,000 People", gridcolor="lightgrey"),
    plot_bgcolor="white",
    showlegend=False
)

fig2.show()
```



Report 1 mini-report:

Increased funding of NHS Scotland is a measurement by which policymakers sometimes intend to improve performance to face sudden healthcare services demand as in the case of global COVID-19 (1).

In this analysis, we display the association between NHS Scotland boards' budgets per capita and their response performance to early pandemic measured by mortality rates per 100000 people. Data was derived from `nhs_boards_25.jason` to extract the population size and total budget of each board to identify boards with the highest funding (1). From `nhs_covid_days_25.json` , total COVID-19-related deaths were then calculated per 100000 persons as an indicator of efficiency (2).

Although there is a semi-pattern suggesting that boards with higher budgets witnessed less death rates, outliers do exist suggesting variability. Some densely populated boards such as NHS Greater Glasgow and Clyde had lower budget-per-capita figures but higher absolute death counts. However, when normalised by population, the death rates did not show a simple inverse relationship with funding levels (1).

Report 2:

Report 2 code:

In [8]:
```
# Function: Summarise COVID cases per weekday

def get_weekday_case_summary(covid_data):
    total_cases = {}
    weekday_counts = {}

    for day in covid_data:
        weekday = day["day"]["weekday"]
        new_cases = day["whole_scotland"]["new_positive"]

        total_cases[weekday] = total_cases.get(weekday, 0) + new_cases
        weekday_counts[weekday] = weekday_counts.get(weekday, 0) + 1

    return total_cases, weekday_counts


# Run summary function

weekday_totals, weekday_counts = get_weekday_case_summary(covid_days)

#  Assert: Ensure all weekdays are covered and valid

assert isinstance(weekday_totals, dict), "weekday_totals should be a dictionary"
assert isinstance(weekday_counts, dict), "weekday_counts should be a dictionary"
assert len(weekday_totals) == 7, "Expected 7 weekdays"
assert all(day in weekday_counts for day in ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturda
print("tests passed")
```

```
tests passed
```

In [9]:
```
# Calculate average cases for each weekday

weekday_averages = []
for day in weekday_totals:
    avg = weekday_totals[day] / weekday_counts[day]
    weekday_averages.append({"weekday": day, "avg_cases": round(avg, 2)})
```

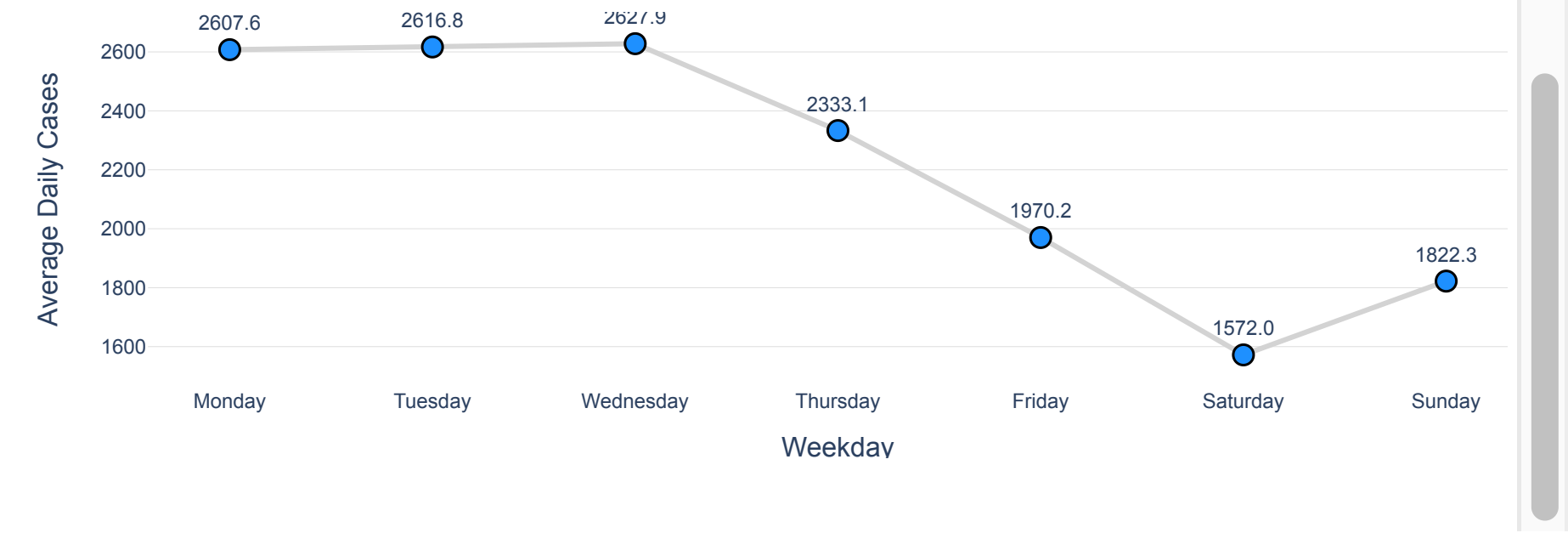
In [10]:
```
# Sort by weekday order

weekday_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
weekday_averages_sorted = sorted(weekday_averages,
                                 key=lambda
                                 x: weekday_order.index(x["weekday"]))
```

In [11]:
```
# Prepare data
weekdays = [entry["weekday"]
            for entry in weekday_averages_sorted]
avg_cases = [entry["avg_cases"]
             for entry in weekday_averages_sorted]
```

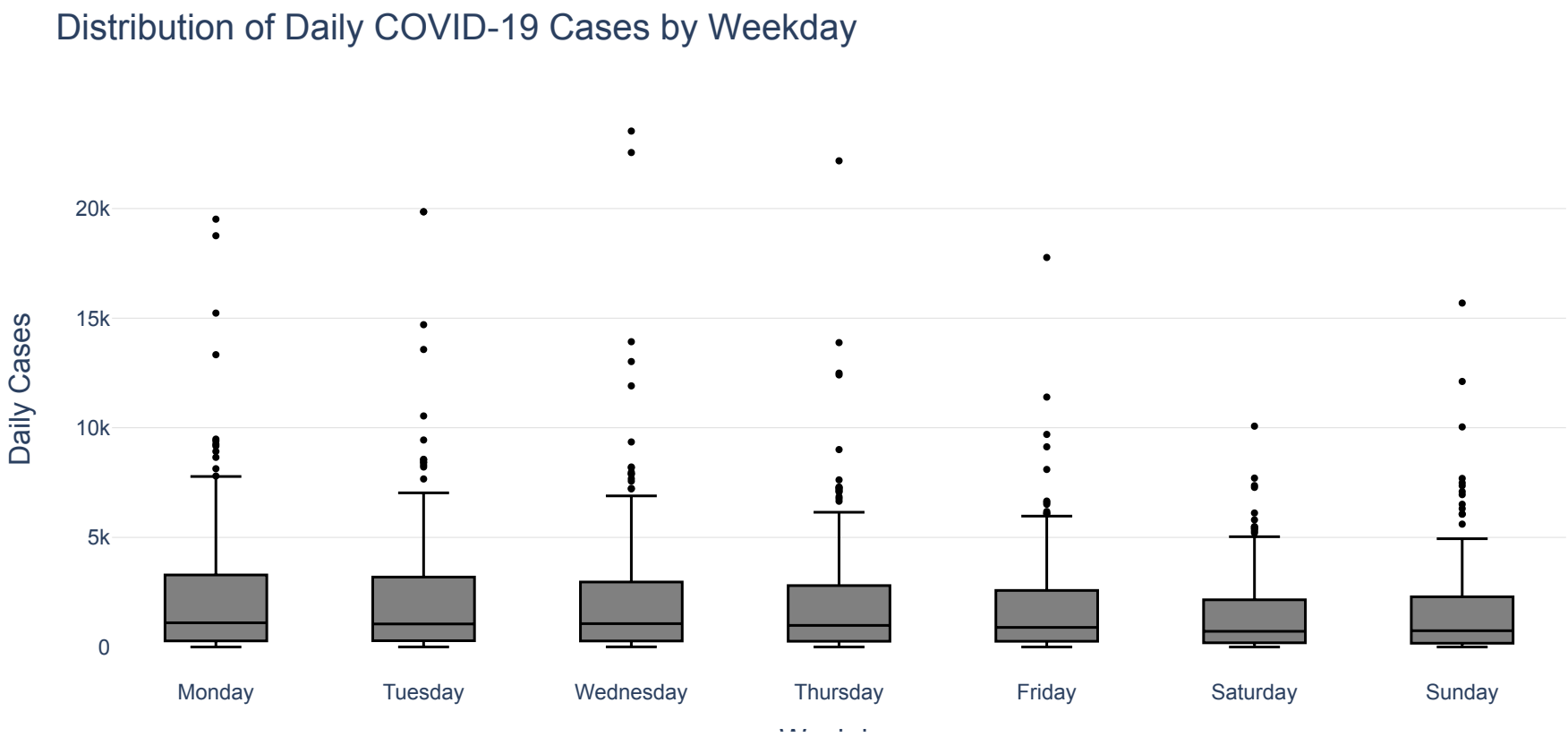
In [12]:
```
# Build list of daily cases by weekday (for box plot )

cases_by_day = []
for day in covid_days:
    weekday = day["day"]["weekday"]
    new_cases = day["whole_scotland"]["new_positive"]
    cases_by_day.append({"weekday": weekday, "cases": new_cases})
```

Report 2 visualisation:

In [13]:
```
# Create figure
fig3 = go.Figure()

# Add points

fig3.add_trace(go.Scatter(
    x=weekdays,
    y=avg_cases,
    mode='lines+markers+text',
    line=dict(color='lightgray', width=3),
    marker=dict(size=12, color='dodgerblue', line=dict(width=1.5, color='black')),
    text=[f"{val:.1f}" for val in avg_cases],                    # Add average values as text labels
    textposition="top center",
    textfont=dict(size=12),
    name="Average Cases"
))

# Custom layout for aesthetics

fig3.update_layout(
    title="Average Daily COVID-19 Cases by Weekday in Scotland",
    title_font_size=20,
    xaxis_title="Weekday",
    yaxis_title="Average Daily Cases",
    xaxis=dict(
        categoryorder='array',
        categoryarray=weekday_order,
        tickangle=0,
        tickfont=dict(size=12)
    ),
    yaxis=dict(
        gridcolor="lightgrey",
        zeroline=False,
        tickfont=dict(size=12)
    ),
    plot_bgcolor='white',
    paper_bgcolor='white',
    font=dict(family="Arial", size=13),
    height= 400,
    showlegend=False
)

fig3.show()
```



In [14]:
```
# Box plot with enhancements

fig4 = px.box(
    cases_by_day,
    x="weekday",
    y="cases",
    title="Distribution of Daily COVID-19 Cases by Weekday",
    labels={"cases": "Daily Cases", "weekday": "Weekday"},
    category_orders={"weekday": weekday_order},
    points="outliers"                    # Show outliers for extra insight
)

fig4.update_traces(
    marker=dict(size=4, color="black"),  # Outlier dot styling
    line=dict(width=1.5),                # Box edge thickness
    selector=dict(type='box')
)

fig4.update_layout(
    title_font_size=20,
    font=dict(family="Arial", size=13),
    xaxis=dict(
        title="Weekday",
        tickfont=dict(size=12),
        categoryorder='array',
        categoryarray=weekday_order
    ),
    yaxis=dict(
        title="Daily Cases",
        tickfont=dict(size=12),
        gridcolor="lightgrey"
    ),
    boxmode='group',
    plot_bgcolor='white',
    paper_bgcolor='white',
    showlegend=False,
    height=450,
    margin=dict(t=70, b=60)
)

fig4.show()
```



Report 2 mini-report:

Recognising fluctuation patterns in the number of infections based on the weekday can help healthcare institutions better manage resources for efficient healthcare service provision (4).

Based on the data provided from `nhs_covid_days.json` , Analysis of the fluctuation of infection rates depending on the day of the week was done and ratios of infections were displayed in a line graph and a box plot. The line graph illustrates a clear decrease of average infection rates moving towards weekends with it maximising midweek (4). To compensate for outliers that may affect averages, a box plot was created to illustrate the relationship between maximum infection rates for different weekdays. This revealed that while weekdays had greater variation and more extreme values (outliers), weekends showed tighter distributions with fewer large spikes. However, both the line graph and the box plot suggest that at the beginning of the week's days, infection rates spike the their maximum values again.

Results suggest that data taken during weekends might be under-representative due to operational factors like opening hours or patients'

References

1. Moolla I, Hiilamo H. *Health system characteristics and COVID-19 performance in high-income countries*. **BMC Health Services Research** [Internet]. 2023 Mar 13 [cited 2025 Mar 31];23(1):244. Available from: https://doi.org/10.1186/s12913-023-09206-z (https://doi.org/10.1186/s12913-023-09206-z)
2. Lupu D, Tiganasu R. *COVID-19 and the efficiency of health systems in Europe*. **Health Economics Review** [Internet]. 2022 Feb 12 [cited 2025 Mar 31];12(1):14. Available from: https://doi.org/10.1186/s13561-022-00358-y (https://doi.org/10.1186/s13561-022-00358-y)
3. Jacques O, Arpin E, Ammi M, Noël A. *The political and fiscal determinants of public health and curative care expenditures: Evidence from the Canadian provinces, 1980–2018*. **Canadian Journal of Public Health** [Internet]. 2023 Aug 1 [cited 2025 Mar 31];114(4):584–92. Available from: https://doi.org/10.17269/s41997-023-00751-y (https://doi.org/10.17269/s41997-023-00751-y)
4. Buckingham-Jeffery E, Morbey R, House T, Elliot AJ, Harcourt S, Smith GE. *Correcting for day of the week and public holiday effects: Improving a national daily syndromic surveillance service for detecting public health threats*. **BMC Public Health** [Internet]. 2017 May 19 [cited 2025 Mar 31];17(1):477. Available from: https://doi.org/10.1186/s12889-017-4372-y (https://doi.org/10.1186/s12889-017-4372-y)
5. Ellis DA, Jenkins R. *Weekday Affects Attendance Rate for Medical Appointments: Large-Scale Data Analysis and Implications*. **PLOS ONE** [Internet]. 2012 Dec 13 [cited 2025 Mar 31];7(12):e51365. Available from: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0051365 (https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0051365)

In [ ]: