AO19 Final: points

```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

```
# I read and prepare the points table.
point = pd.read_csv('/home/....../points.csv')
point = point.drop(columns=['Unnamed: 0', 'x', 'y'])
point
```

```
# I get the number of first serves.
point.groupby('server').count()[['rallyid']]
```

```
# I count the number of first serves in.
point[point['serve'] == 'second'].groupby('server').count()[['rallyid']]
```

```
# I check the number of points won by the server or the receiver after first or
second serve.
point[(point['server'] == 'Nadal') & (point['winner'] == 'Nadal') & (point['serve']
== 'first')].count()
```

```
# Number of aces.
point[(point['server'] == 'Nadal') & (point['reason'] == 'ace')].count()
```

```
# Number of double faults.
point[(point['server'] == 'Djokovic') & (point['serve'] == 'second') &
(point['reason'] == 'double_fault')].count()
```

```
# Number of points won.
point.groupby('winner').count()[['rallyid']]
```

```
# Reasons for points.
point.groupby('reason').count()[['rallyid']]
```

```
# I draw a bar chart of reasons for points.
reasons = ['ace', 'double_fault', 'net', 'out', 'winner']
counts = [11, 2, 37, 49, 43]
sorted_reasons, sorted_counts = zip(*sorted(zip(reasons, counts), key=lambda x:
x[1], reverse=True))
plt.bar(sorted_reasons, sorted_counts, color='skyblue')
plt.title('Reasons for points')
plt.xlabel('Reason')
plt.ylabel('Count')
plt.savefig('reasons_for_points.png')
plt.show()
```

```
# Points by the number of strikes.
point.groupby('strokes').count()[['rallyid']]
```

```
# I draw a bar chart of the frequency of number of strikes
strokes_counts = {
    1: 14,
    2: 28,
    3: 21,
    4: 19,
    5: 7,
    6: 6,
    7: 14,
    8: 5,
    9: 2,
    10: 2,
    11: 9,
    12: 4,
    13: 1,
    14: 3,
    15: 1,
    16: 3,
    17: 1,
    19: 1,
    22: 1
}

strokes = list(strokes_counts.keys())
counts = list(strokes_counts.values())
plt.figure(figsize=(10, 6))
plt.bar(strokes, counts, color='skyblue')
plt.xlabel('Number of Strikes')
plt.ylabel('Frequency')
plt.title('Frequency of Number of Strikes')
plt.xticks(strokes)  # Set x-axis ticks to be the unique values of 'strokes'
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.savefig('freq_num_strikes.png')
plt.show()
```

```
# Time of the points.
point.groupby('totaltime').max()[['rallyid']]
```

```
# I draw a bar chart of the frequencies of the time of points
frequencies = {
    '0-0.3': 55,
    '0.3-1': 9,
    '1-2': 14,
    '2-3': 12,
    '3-4': 8,
    '4-5': 17,
    '5-8': 12,
    '8-12': 9,
    '12-15': 4,
    'above 15': 1
}

bins = list(frequencies.keys())
counts = list(frequencies.values())
plt.figure(figsize=(10, 6))
plt.bar(bins, counts, color='skyblue')
plt.xlabel('Total Time Bins')
plt.ylabel('Frequency')
plt.title('Frequency of Total Time Bins')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.savefig('freq_time.png')
plt.show()
```

```
# Reason of points won by Djokovic.
djoko_point = point[point['winner'] == 'Djokovic']
djoko_win_reason = djoko_point.groupby('reason').size()
djoko_win_reason
```

```
# Reason of points won by Nadal.
nadal_point = point[point['winner'] == 'Nadal']
nadal_win_reason = nadal_point.groupby('reason').size()
nadal_win_reason
```

```
# I draw a bar chart about the reasons for points.
# Points for Djokovic
djokovic_data = {
    "Ace": 8,
    "Double Fault": 0,
    "Net": 14,
    "Out": 19,
    "Winner": 26
}
```

```
# Points for Nadal
nadal_data = {
    "Double Fault": 2,
    "Ace": 3,
    "Net": 23,
    "Out": 30,
    "Winner": 17
}

# Combine the datasets
combined_data = {
    "Djokovic": djokovic_data,
    "Nadal": nadal_data
}


reasons = list(djokovic_data.keys())  # Assuming both datasets have the same reasons
x = range(len(reasons))
bar_width = 0.35
fig, ax = plt.subplots()
for i, (player, data) in enumerate(combined_data.items()):
    ax.bar([pos + i * bar_width for pos in x], data.values(), bar_width,
label=player)
ax.set_xlabel('Reason')
ax.set_ylabel('Frequency')
ax.set_title('Reasons for rally endings')
ax.set_xticks([pos + bar_width / 2 for pos in x])
ax.set_xticklabels(reasons)
ax.legend()
plt.savefig('reason_for_points.png')
plt.show()
```

```
# Reasons for points for Djokovic as a server.
djokovic_serve_win = point[(point['server'] == 'Djokovic') & (point['winner'] ==
'Djokovic')]
djokovic_reasons = djokovic_serve_win.groupby('reason').size()
djokovic_reasons
```

```
# Reasons for points for Nadal as a receiver.
djoko_serve_lost = point[(point['server'] == 'Djokovic') & (point['winner'] ==
'Nadal')]
nadal_reasons = djoko_serve_lost.groupby('reason').size()
nadal_reasons
```

```
# Reasons for points for Nadal as a server.
nadal_serve_win = point[(point['server'] == 'Nadal') & (point['winner'] == 'Nadal')]
nadal_reasons = nadal_serve_win.groupby('reason').size()
nadal_reasons
```

```
# Reasons for points for Djokovic as a receiver.
nadal_serve_lost = point[(point['server'] == 'Nadal') & (point['winner'] ==
'Djokovic')]
djokovic_reasons = nadal_serve_lost.groupby('reason').size()
djokovic_reasons
```

```
# The avg of shots when Djokovic serves.
djokovic_serve = point[point['server'] == 'Djokovic']
avg_strikes = djokovic_serve['strokes'].mean()
avg_strikes
```

```
# The avg of shots when Nadal serves.
nadal_serve = point[point['server'] == 'Nadal']
avg_strikes = nadal_serve['strokes'].mean()
avg_strikes
```

```
# The avg of shots when Djokovic wins the point.
djokovic_wins = point[point['winner'] == 'Djokovic']
avg_strikes = djokovic_wins['strokes'].mean()
avg_strikes
```

```
# The avg of shots when Nadal wins the point.
nadal_wins = point[point['winner'] == 'Nadal']
avg_strikes = nadal_wins['strokes'].mean()
avg_strikes
```

```
# I draw a bar chart of time per rally.
plt.figure(figsize=(10, 6))
plt.bar(point['rallyid'], point['totaltime'], color='skyblue')
plt.title('Total Time per Rally')
plt.xlabel('Rally ID')
plt.ylabel('Total Time')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```