In [1]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.renderers.default = "iframe"

import warnings
warnings.filterwarnings("ignore")

floor = pd.read_csv("floor.csv")
floor
```

```
### adding calendar day to floor data
from datetime import datetime
floor['Weekday'] = (datetime.strptime(d, '%Y-%m-%d %H:%M:%S').strftime('%a') for d in floor['Timestamp

### converting hours
floor['Hour Copy'] = [
    '8:00 a.m.', '9:00 a.m.', '2:00 p.m.', '3:00 p.m.', '4:00 p.m.', '5:00 p.m.',
    '6:30 a.m.', '7:30 a.m.', '8:30 a.m.', '9:30 a.m.', '10:30 a.m.', '11:30 a.m.',
    '12:30 p.m.', '1:30 p.m.', '2:30 p.m.', '3:30 p.m.', '4:30 p.m.', '5:30 p.m.', '6:30 p.m.',
    '10:30 a.m.', '11:30 a.m.', '12:30 p.m.', '1:30 p.m.', '2:30 p.m.', '3:30 p.m.', '4:30 p.m.',
    '5:30 p.m.']

floor['Hour Copy'] = floor['Hour'].map(hour_convert)
floor.head()
```

Out[1]:

	Location	Timestamp	Average Occupancy	Peak Occupancy	Time Interval	Location Path	Hour	Quarter	Weekday	Hour Copy
0	UCSB Library	2021-09-22 16:00:00	12	2	1hour	UCSB > UCSB Library	16	3rd Quarter	Wednesday	4 p.m.
1	2nd Floor Ocean	2021-09-22 16:00:00	12	2	1hour	UCSB > UCSB Library > 2nd Floor > 2nd Floor Ocean	16	3rd Quarter	Wednesday	4 p.m.
2	2nd Floor	2021-09-22 16:00:00	12	2	1hour	UCSB > UCSB Library > 2nd Floor	16	3rd Quarter	Wednesday	4 p.m.
3	UCSB Library	2021-09-26 10:00:00	30	39	1hour	UCSB > UCSB Library	10	4th Quarter	Saturday	12 p.m.
4	2nd Floor Ocean	2021-09-26 12:00:00	3	4	1hour	UCSB > UCSB Library > 2nd Floor > 2nd Floor Ocean	12	4th Quarter	Saturday	12 p.m.

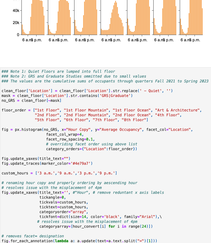
In [2]:
```
condition = floor['Location'] == 'UCSB Library'
### without UCSB Overall totals
class_floor = floor[~condition]
class_floor['Hour'].unique()
class_floor['Location']
class_floor.shape
floor_copy = floor.copy(floor_copy['Quarter'].notna())

quarters = floor_copy['Quarter'].unique().tolist()
# quarters.remove('')
quarters.sort()
['Fall 2021', 'Winter 2022', 'Spring 2022', 'Summer 2022', 'Fall 2022', 'Winter 2023', 'Spring 2023']

class_floor.head()
```

Out[2]:

	Location	Timestamp	Average Occupancy	Peak Occupancy	Time Interval	Location Path	Hour	Quarter	Weekday	Hour Copy
1	2nd Floor Ocean	2021-09-22 16:00:00	12	2	1hour	UCSB > UCSB Library > 2nd Floor > 2nd Floor Ocean	16	3rd Quarter	Wednesday	4 p.m.
2	2nd Floor	2021-09-22 16:00:00	12	2	1hour	UCSB > UCSB Library > 2nd Floor	16	3rd Quarter	Wednesday	4 p.m.
4	2nd Floor Ocean	2021-09-26 12:00:00	3	4	1hour	UCSB > UCSB Library > 2nd Floor > 2nd Floor Ocean	12	4th Quarter	Saturday	12 p.m.
5	2nd Floor Mountain	2021-09-26 12:00:00	2	0	1hour	UCSB > UCSB Library > 2nd Floor > 2nd Floor Mountain	12	4th Quarter	Saturday	12 p.m.
6	2nd Floor	2021-09-26 12:00:00	3	4	1hour	UCSB > UCSB Library > 2nd Floor	12	4th Quarter	Saturday	12 p.m.

Histograms

In [3]:
```
floor_total = floor.groupby(floor_total['Quarter'].notna())
floor_total['Hour Copy'] = floor_total['Hour'].map(hour_convert)
floor_total.sort_values(inplace=True)

fig = px.histogram(floor_total, x='Hour Copy', y='Average Occupancy', facet_col='Quarter')

fig.update_traces(marker_color='#f19ec1')

custom_hours = ['9 a.m.','9 p.m.']

fig.update_axes(title_text='', y='Hour', # remove redundant x axis labels
                tickangle=0,
                ticklevelsize=hours,
                tickformat=hours,
                tickfont=dict(size=24, color='black', family='Arial'),
                categoryorder='array',
                categoryarray=hour_convert[12 for i in range(24)])

fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")[1]))

# removes facetn designation
fig.for_each_axis(lambda a:
    if type(fig.layout[axis]) == go.layout.XAxis:
        fig.layout[axis].title.text = ''
    # removes y axis labels for all subplots
    if type(fig.layout[axis]) == go.layout.YAxis:
        fig.layout[axis].title.text = '')

# equal sizing
# annotations used to remove 'Wednesday' created by stacked y axis labels
annotations = list(fig.layout.annotations)
    [ga.layout.annotation(
        x=x,
        y=0.5,
        text=x[:3],
        font=dict(size=14, color = 'black'
        textangle=0,
        showarrow=False,
        xref='paper',
        yref='paper'
        )
    for x in
    ],
    )
template="simple_white",
title_text='Popular times of the week to visit the library by quarter and hour',
font=dict(family='Arial', size=14, color='black'))

fig.show()
```

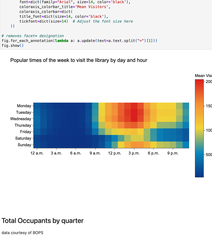


Note 1: Quiet Floors are lumped into full floor
Note 2: Ocean and Mountain are lumped together due to small values
The values are the cumulative sum of occupancy through quarters Fall 2021 to Spring 2023

```
class_floor['location'] = class_floor['location'].str.replace(' - Quiet', '')
mask = class_floor['location'].isin(['UCSB Overall'])
no_OOS = class_floor[~mask]

floor_order = ['1st Floor', '1st Floor Mountain', '1st Floor Ocean', 'Art & Architecture',
               '2nd Floor', '2nd Floor Mountain', '2nd Floor Ocean', '4th Floor',
               '5th Floor', '6th Floor', '7th Floor', '8th Floor']

fig = px.histogram(no_OOS, x='Hour Copy', y='Average Occupancy', facet_col='location',
                   category_orders={'location':floor_order},
                   facet_row_spacing=0.1,
                   # overriding facet order using above list
                   category_orders={"location":floor_order})

fig.update_traces(title_text="")
fig.update_traces(marker_color='#ea7b47')

custom_hours = ['9 a.m.','9 a.m.','9 p.m.','9 p.m.']

# removing hour copy and property ordering by descending hour
fig.update_traces(title_text="", y=no_OOS['Hour'],
fig.update_axes(title_text="", y='Hour', # remove redundant x axis labels
                tickangle0,
                ticklevelsize=hours,
                tickformat=hours,
                categoryorder='array',
                tickfont=dict(size=24, color='black', family='Arial'),),
                # resolves issue with the displacement of dpm
                categoryarray=hour_convert[12 for i in range(24)])

# removes facetn designation
fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")[1]))

for axis in fig.layout:
    if type(fig.layout[axis]) == go.layout.YAxis:
        fig.layout[axis].title.text = ''

# equal sizing
# annotations used to remove 'Wednesday' created by stacked y axis labels
annotations = list(fig.layout.annotations)
    [ga.layout.annotation(
        x=x,
        y=0.5,
        text=x[:3],
        font=dict(size=14, color = 'black'
        textangle=0,
        showarrow=False,
        xref='paper',
        yref='paper'
        )
    for x in
    ],
    )
template="simple_white",
title_text=f'Popular times of the week to visit the library by day and hour',
font=dict(family='Arial', size=14, color='black'))

fig.show()
```

Heat Map

In [4]:
```
### Heat Map using mean visitors relative to weekn and hour instead of total

week_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

# calculate the mean of 'Average Occupancy' based on 'Weekday' and 'Hour Copy'
mean = pd.pivot_table(floor_total, values='Average Occupancy', index='Weekday', columns='Hour Copy', agg
mean = mean.reindex(week_order)

# Create heatmap using pandas pivot table from above
fig = px.imshow(mean, text_auto=True, aspect="auto",
                color_continuous_scale='portland')

custom_hours = ['12 a.m.','3 a.m.','6 a.m.','9 a.m.','12 p.m.','3 p.m.','6 p.m.','9 p.m.']

fig.update_axes(title_text="",

                title_text="",
                ticklevelsize=6,
                tickformat=hours,
                ticklevelsize=hours,
                tickfont=dict(size=24, color='black', family='Arial'),
                # used to reorder weeks
                categoryorder='array',
                categoryarray=hour_convert[12 for i in range(24)])

fig.update_layout(
    template="simple_white",
    title_text=f'Heat Map',
    title_text='Popular times of the week to visit the library by day and hour',
    font=dict(family='Arial', size=24, color='black'),
    colorscale_colorbar_title= 'Mean Visitors',
    colorscale_colorbar=dict(
        title_text='Mean Visitors', color='black'),
        tickfont=dict(size=24 # adjust the font size here
    )))

# removes facetn designation
fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")[1]))
fig.show()
```



Total Occupants by quarter

data courtesy of BDPS

In [5]:
```
df = {'Fall':{2015SB,81393,8613987,85929,27,81933,6660741,
    'Winter':{7592008,8873458,87499,8802175,241,5147974,7969802,
    'Spring':{7592008,8873458,87499,8802175,241,5147974,7969802,
    'Summer':{26421,2170,1600,0,0,0
```