```python
import dash
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import numpy as np
from dash.dependencies import Output, Input
import os
import plotly.express as px

from assets import whoop

access_token = whoop.get_access_token("trevor.liggett@gmail.com", os.getenv('WHOOP_PASSWORD'))


data = whoop.get_user_data_df(access_token,
                              start_date='2000-01-01T00:00:00.000Z',
                              end_date='2030-01-01T00:00:00.000Z',
                              url='https://api-7.whoop.com/users/{}/cycles')



data["Date"] = pd.to_datetime(data["date"], format="%Y-%m-%d")
data.sort_values("Date", inplace=True)

# Have to hard code stats at first
stats = ['sleep.score', 'sleep.qualityDuration']



external_stylesheets = [
    {
        "href": "https://fonts.googleapis.com/css2?"
        "family=Lato:wght@400;700&display=swap",
        "rel": "stylesheet",
    },
]
app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
server = app.server
app.title = "Sleep Analysis with WHOOP"

app.layout = html.Div(
    children=[
        html.Div(
            children=[
                html.Img(src="assets/sleep.png", className="header-emoji"),
                html.H1(
                    children="Sleep Analysis: Powered by WHOOP", className="header-title"
                ),
                html.P(
                    children="Monitoring and analyzing"
                    " sleep patterns between 2018 and 2021"
                    " using the WHOOP wearable.",
                    className="header-description",
                ),
            ],
            className="header",
        ),
        html.Div(
            children=[
                html.Div(
                    children=[
                        html.Div(
                            children="Date Range", className="menu-title"
                        ),
                        dcc.DatePickerRange(
                            id="date-range",
                            min_date_allowed=data.Date.min().date(),
                            max_date_allowed=data.Date.max().date(),
                            start_date=pd.to_datetime("2021-04-11",
format="%Y-%m-%d"),
                            end_date=data.Date.max().date(),
                        ),
                    ]
                ),
            ],
            className="menu",
        ),
        html.Div(
            children=[
                html.Div(
                    children=dcc.Graph(
                        id="need-chart",
                        config={"displayModeBar": False},
                    ),
                    className="card",
                ),
                html.Div(
                    children=dcc.Graph(
                        id="efficiency-chart",
                        config={"displayModeBar": False},
                    ),
                    className="card",
                ),
                html.Div(
                    children=dcc.Graph(
                        id="consistency-chart",
                        config={"displayModeBar": False},
                    ),
                    className="card",
                ),
                html.Div(
                    children=dcc.Graph(
                        id="pie-chart",
                        config={"displayModeBar": False},
                    ),
                    className="card",
                ),
            ],
            className="wrapper",
        ),
    ]
)


@app.callback(
    [Output("need-chart", "figure"), Output("efficiency-chart", "figure"),
Output("consistency-chart", "figure"), Output("pie-chart", "figure")],
    [
```

```python
        Input("date-range", "start_date"),
        Input("date-range", "end_date"),
    ],
)
def update_charts(start_date, end_date):
    mask = (data['Date'] > start_date) & (data['Date'] <= end_date)
    filtered_data = data.loc[mask]
    # create pie data
    names = ['sws', 'rem', 'light', 'wake']
    values = [filtered_data['sleep.sws.duration'].mean() / 3600000,
                            filtered_data['sleep.rem.duration'].mean() / 3600000,
                            filtered_data['sleep.light.duration'].mean() / 3600000,
                            filtered_data['sleep.wake.duration'].mean() / 3600000]

    pie_data = pd.DataFrame(list(zip(names, values)), columns =['names', 'values'])

    need_chart_figure = {
        "data": [
            {
                "x": filtered_data["Date"],
                "y": filtered_data["sleep.qualityDuration"] / 3600000,
                "type": "lines",
                "name": "Sleep Duration",
            },
            {
                "x": filtered_data["Date"],
                "y": filtered_data["sleep.needBreakdown.total"] / 3600000,
                "type": "lines",
                "name": "Sleep Need",
            },
        ],
        "layout": {
            "title": {"text": "Hours of Sleep vs Sleep Need", "x": 0.05, "xanchor": "left"},
            "xaxis": {"fixedrange": True},
            "yaxis": {"fixedrange": True},
            "colorway": ["#52B2BF", "#0A1172"],
        },
    }

    efficiency_chart_figure = {
        "data": [
            {
                "x": filtered_data["Date"],
                "y": filtered_data["sleep.efficiency"],
                "type": "lines",
                "hovertemplate": "%{y:.2f}<extra></extra> percent",
            },
        ],
        "layout": {
            "title": {
                "text": "Sleep Efficiency",
                "x": 0.05,
                "xanchor": "left",
            },
            "xaxis": {"fixedrange": True},
            "yaxis": {"tickprefix": "", "fixedrange": True},
            "colorway": ["#7A4988"],
        },
    }

    consistency_chart_figure = {
        "data": [
            {
                "x": filtered_data["Date"],
                "y": filtered_data["sleep.consistency"],
                "type": "lines",
                "hovertemplate": "%{y:.2f}<extra></extra> percent",
            },
        ],
        "layout": {
            "title": {
                "text": "Sleep Consistency",
                "x": 0.05,
                "xanchor": "left",
            },
            "xaxis": {"fixedrange": True},
            "yaxis": {"tickprefix": "", "fixedrange": True},
            "colorway": ["#7A4988"],
        },
    }

    pie_chart_figure = px.pie(pie_data, values=values, names=names,
 color_discrete_sequence=px.colors.sequential.Purp,
                                hole=.3,
                                title="Proportion of Sleep Spent in Major
Stages")

    return need_chart_figure, efficiency_chart_figure,
consistency_chart_figure, pie_chart_figure

if __name__ == "__main__":
    app.run_server(debug=True)

```