In [1]:
```
import numpy as np
import pandas as pd
import re

import plotly.express as px
import plotly.graph_objects as go

import dash
from dash import Dash, dcc, html, Input, Output, callback, State, dash_table
```

Data Wrangling

In [2]:
```
def data_wrangling():
    # create dataframe
    data = pd.read_csv('nigeria_agricultural_exports.csv')

    # remove (NEPC) from some values in Company column
    data['Company'] = [name.split('(')[0] for name in data['Company']]
    # convert company names to acronyms
    data['Company'] = [''.join(re.findall(r'[A-Z]', name)) for name in data['Company']]

    # convert date column to datetime
    data['Date'] = pd.to_datetime(data['Date'])

    # Create Total profit column
    data['Total Profit'] = data['Units Sold'] * data['Profit per unit']

    # Create month, quarter and year columns
    data['Month'] = [date.strftime("%B") for date in data['Date']]
    data['Year'] = data['Date'].dt.year
    data['Quarter'] = data['Date'].dt.quarter

    # Define the correct order of months
    month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October',
                   'November', 'December']

    # Convert Month column to a categorical type with the specified order
    data['Month'] = pd.Categorical(data['Month'], categories=month_order, ordered=True)

    # Create profit margin column
    data['Profit Margin'] = (data['Total Profit'] / data['Export Value']) * 100

    # rename export value column
    data.rename(columns = {'Export Value': 'Revenue'}, inplace=True)

    return data
```

Sales Performance

In [3]:
```
def sales_performance():
    data = data_wrangling().copy()

    top_selling_products = data[['Product Name', 'Units Sold']].groupby(
        'Product Name').sum('Units Sold').sort_values(by='Units Sold',ascending=False).reset_index()
    prod_fig = px.bar(top_selling_products, x= 'Product Name', y = 'Units Sold',
                      color='Units Sold',title='Top-Selling Products')
    prod_fig.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])

    # sales revenue by company
    company_revenue = data[['Company', 'Revenue']].groupby('Company').sum(
    'Revenue').sort_values(by = 'Revenue', ascending=False).reset_index()
    revenue_fig = px.bar(company_revenue, x = 'Company', y = 'Revenue',
                         color = 'Revenue', title = 'Sales Revenue by Company')
    revenue_fig.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],
                              font_color=colors['text'])

    # Product Sales variation by export country
    country_sales = data[['Export Country', 'Product Name', 'Units Sold']].groupby(
    ['Export Country', 'Product Name'])['Units Sold'].sum().unstack().reset_index()

    country_sales_fig = px.bar(country_sales,
             x='Export Country',
             y=country_sales.columns[1:],
             title='Product Sales by Export Country',
             labels={'value': 'Units Sold', 'Export Country': 'Country', 'variable':'Product'},
             barmode='group')
    country_sales_fig.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],
                                    font_color=colors['text'])

    # Average Revenue per Country
    country_revenue = data[['Export Country', 'Revenue']].groupby('Export Country').mean('Revenue').reset_index()
    country_revenue_fig = px.bar(country_revenue, x='Export Country', y='Revenue',
                                 title = 'Average Sales Revenue by Country')
    country_revenue_fig.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],
                                      font_color=colors['text'])
```

```
    # Total Units Sold
    total_sold_fig = px.histogram(data, x='Export Country', y='Units Sold',title = 'Total Units Sold by Country')
    total_sold_fig.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],
                                 font_color=colors['text'])

    # Is there any correlation between the units sold and the profit generated?
    corr_fig1 = px.scatter(data, x = 'Units Sold', y='Total Profit', title = 'Units Sold vs Total Profit')
    corr_fig1.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])



    return prod_fig, revenue_fig, country_sales_fig, country_revenue_fig, total_sold_fig, corr_fig1
```

Time Series Analysis

In [4]:
```
def time_series_analysis():
    data = data_wrangling().copy()

    # Sales variation over time (monthly, quarterly, annually)
    monthly_sales = data[['Month', 'Units Sold', 'Revenue']].groupby('Month').sum(['Units Sold','Revenue']).reset_index()
    # plot of monthly sales
    monthly_sales_fig = go.Figure()
    monthly_sales_fig.add_trace(go.Scatter(x = monthly_sales['Month'], y = monthly_sales['Units Sold'],line_color ="#0000ff"))
    # Customize the layout
    monthly_sales_fig.update_layout(
        title='Sales Variation by Month',
        xaxis_title='Month',
        yaxis_title='Units Sold',
        plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])

    # Quarter Sales
    quarter_data = data[['Quarter','Units Sold']].groupby('Quarter').sum('Units Sold').reset_index()
    quarter_sales_fig = go.Figure()
    quarter_sales_fig.add_trace(go.Scatter(x = quarter_data['Quarter'], y=quarter_data['Units Sold'], mode='lines+markers'))

    # Customize the layout
    quarter_sales_fig.update_layout(
        title='Sales Variation by Quarter',
        xaxis_title='Quarter',
        yaxis_title='Units Sold',
        plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])

    # seasonal change
    data_2020 = data[data['Year']==2020]
    data_2021 = data[data['Year']==2021]
    data_2022 = data[data['Year']==2022]
    data_2023 = data[data['Year']==2023]

    def get_month(i):
        month_data = i[['Month','Units Sold', 'Revenue']].groupby('Month').sum(['Units Sold', 'Revenue']).reset_index()
        return (month_data)

    month_fig1 = go.Figure()
    month_fig1.add_trace(go.Scatter(x=get_month(data_2020)['Month'],y=get_month(data_2020)['Revenue'],
                                    mode='lines+markers',name='Year 2020'))
    month_fig1.add_trace(go.Scatter(x=get_month(data_2021)['Month'],y=get_month(data_2021)['Revenue'],
                                    mode='lines+markers',name='Year 2021'))
    month_fig1.add_trace(go.Scatter(x=get_month(data_2022)['Month'],y=get_month(data_2022)['Revenue'],
                                    mode='lines+markers',name='Year 2022'))
    month_fig1.add_trace(go.Scatter(x=get_month(data_2023)['Month'],y=get_month(data_2023)['Revenue'],
                                    mode='lines+markers',name='Year 2023'))
    month_fig1.update_layout(title='Monthly Revenue Comparison',
                             xaxis_title='Month',
                             yaxis_title='Export Value (Revenue)',
                             plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])


    # Year Sales
    yearly_data = data[['Year', 'Units Sold', 'Revenue']].groupby('Year').sum(
        ['Units Sold','Export Revenue']).reset_index()
    # total units sold
    yearly_sales_fig = go.Figure()
    yearly_sales_fig.add_trace(go.Scatter(x=yearly_data['Year'], y=yearly_data['Units Sold'],mode='lines+markers'))
    yearly_sales_fig.update_layout(title='Total Units Sold per Year',
                                   xaxis_title ='Year',
                                   yaxis_title ='Units Sold',
                                   plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],
                                      font_color=colors['text'])
    # Revenue per year
    yearly_rev_fig = go.Figure()
    yearly_rev_fig.add_trace(go.Scatter(x=yearly_data['Year'], y=yearly_data['Revenue'],mode='lines+markers'))
    yearly_rev_fig.update_layout(title='Total Revenue per Year',
                             xaxis_title ='Year',
                             yaxis_title ='Export Value (Revenue)',
                             plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])

    # Is there any relationship between date of purchase and profit margin?
    corr_fig = px.scatter(data, x='Date', y='Profit Margin', color='Year',
                          size='Profit Margin', title='Date vs Profit Margin')
    corr_fig.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])

    return monthly_sales_fig, quarter_sales_fig, month_fig1, yearly_sales_fig, yearly_rev_fig, corr_fig
```

Cost Analysis

In [5]:
```
def cost_analysis():
    # What is the cost of goods sold (COGS) as a percentage of revenue?
    data = data_wrangling().copy()
    # Calculate COGS as a percentage of revenue
    new_data = data[['Product Name','Total Profit', 'Revenue']].groupby('Product Name').sum(
        ['Total Profit', 'Revenue']).reset_index()
    new_data['COGS'] = (((new_data['Revenue'] - new_data['Total Profit']) /new_data['Revenue']) * 100)
    new_data = new_data.sort_values(by='COGS', ascending=False)
    new_data = new_data[['Product Name', 'COGS']]

    # How does the COGS vary across different products?
    product_cogs_fig = px.bar(new_data, x = 'Product Name', y='COGS',color='COGS',
                              title = 'COGS per Product')
    product_cogs_fig.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],
                                   font_color=colors['text'])

    return new_data, product_cogs_fig
```

In [ ]:
```
def generate_table(dataframe, max_rows=10):
    return html.Table([
        html.Thead(
            html.Tr([html.Th(col) for col in dataframe.columns])
        ),
        html.Tbody([
            html.Tr([
                html.Td(dataframe.iloc[i][col]) for col in dataframe.columns
            ]) for i in range(min(len(dataframe), max_rows))
        ])
    ])
```

Geographic Data

In [6]:
```
def geographic_data():
    data = data_wrangling()

    # Number of Exports by Port
    port_export_total = data[['Destination Port', 'Units Sold']].groupby(
        'Destination Port').sum('Units Sold').sort_values(by='Units Sold', ascending=False).reset_index()
    port_export_total.rename(columns={'Units Sold': 'Export Volume'}, inplace=True)

    # rank destination port by export value (revenue)
    port_rev_fig = px.histogram(data, x='Destination Port', y='Revenue', title = 'Export Value (Revenue) of Destination Ports')
    port_rev_fig.update_layout(plot_bgcolor=colors['background'],
                               paper_bgcolor=colors['background'],font_color=colors['text'])

    # top export product per port
    port_data = data[['Destination Port', 'Product Name', 'Units Sold']].groupby(
    ['Destination Port', 'Product Name'])['Units Sold'].sum().unstack()
    export_mode = 'The most common means of export is by sea'
    # create a function to get max export value in each port and the name of the product
    def get_max_value_and_column(row):
        max_value = row.max()
        max_column = row.idxmax()
        return pd.Series([max_value, max_column], index=['Export Volume', 'Product Name'])

    # Apply the function to each row
    result = port_data.apply(get_max_value_and_column, axis=1).reset_index()

    return port_export_total,port_rev_fig, result
```

Performance Comparison

In [7]:
```
def performance_comparison():
    data = data_wrangling().copy()

    # How does each product perform in terms of profit margin?
    prod_profit_margin = data[['Product Name', 'Profit Margin']].groupby('Product Name').mean('Profit Margin').reset_index()
    fig = px.bar(prod_profit_margin.sort_values(by='Profit Margin'), x = 'Product Name', y='Profit Margin',
                title = 'Products\' Profit Margin Comparison')
    fig.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])

    # Can we compare the performance of different companies based on units sold and profit generated?
    company_data = data[['Company', 'Units Sold', 'Total Profit']].groupby('Company').sum(
        ['Units Sold', 'Total Profit']).reset_index()
    fig1 = px.bar(company_data.sort_values(by='Units Sold', ascending=False), x= 'Company', y = 'Units Sold',
                  title = 'Company Performance Based on Units Sold')

    fig1.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])

    fig2 =  px.bar(company_data.sort_values(by='Total Profit', ascending=False),x='Company', y = 'Total Profit',
                   title = 'Company Performance Based on Total Profit Generated')
    fig2.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])

    # Are there any outliers or underperforming products/companies that need attention?
    product_data = data[['Product Name', 'Units Sold', 'Total Profit']].groupby('Product Name').sum(
        ['Units Sold', 'Total Profit']).reset_index()
    fig3 = px.box(data, x='Product Name', y = 'Units Sold', title = 'Units Sold vs Product')
    fig3.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])

    fig4 =  px.box(data, x='Product Name', y = 'Total Profit', title = 'Total Profit vs Product')
    fig4.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])
```

```
    fig5 = px.box(data, x='Company',y='Revenue', title = 'Revenue vs Company')
    fig5.update_layout(plot_bgcolor=colors['background'],paper_bgcolor=colors['background'],font_color=colors['text'])
    return fig, fig1, fig2, fig3, fig4, fig5
```

In [ ]:
```
# adjust the function to change background color , header color and font color
def generate_table(dataframe, max_rows=10):
    return html.Table([
        html.Thead(
            html.Tr([html.Th(col,
                             style={'color': colors['text'], 'backgroundColor': colors['background'],'text-align':'',
                                    'padding':'10px','font-size':'20px'}) for col in dataframe.columns])
        ),
        html.Tbody([
            html.Tr([
                html.Td(dataframe.iloc[i][col], style={'color': colors['text'],
                                                       'backgroundColor': colors['background'],
                                                      'padding': '10px', 'fontSize': '18px'}) for col in dataframe.columns
            ]) for i in range(min(len(dataframe), max_rows))
        ], style={'backgroundColor': colors['background']})
    ],style={'width': '50%', 'border': '1px solid white'})
```

Dash App

In [8]:
```
def create_table(a):
    return dash_table.DataTable(a.to_dict('records'),[{"name": i, "id": i} for i in a.columns],
                                style_header={'backgroundColor': 'rgb(30, 30, 30)','color': colors['text'],
                                              'text-align':'left'},
                                 style_data={'backgroundColor': 'rgb(50, 50, 50)','color': colors['text']},
                                 style_cell={ 'border': '2px solid grey', 'padding':'10px','text-align':'left'},
                                 style_table = {'padding':'0px 40px 40px 40px', 'width':'500px'}
                                )
```

In [10]:
```
#initialize dash app
app = Dash(__name__)

# background colour
colors = {
    'background': '#111111',
    'text': '#7FDBFF'}

# app layout
app.layout = html.Div(style = {'backgroundColor': colors['background']},children = [html.H1('Nigeria Agro Export Report',
                                                                                             style={'text-align':'center',
                                                                                                    'padding-top':'20px',
                                                                                                    'color': colors['text']}),
                        # Dropdown Creation
                        # Add Outer Division
                       html.Div([html.Div([html.Div(
                           [html.H2('Report Type:', style={'margin-right':'1em','padding-left': '10px',
                                                           'color': colors['text']}),
                            # Add dropdown
                                   dcc.Dropdown(id = 'input-type',
                                                options = [{'label':'Sales Performance', 'value':'OPT1'},
                                                           {'label': 'Time Series Analysis', 'value': 'OPT2'},
                                                           {'label': 'Cost Analysis', 'value': 'OPT3'},
                                                           {'label': 'Geographic Data', 'value': 'OPT4'},
                                                           {'label': 'Performance Comparison', 'value': 'OPT5'}],
                                                placeholder = 'Select Report Type',
                                                style = {'Text-Align':'center','width': '80%',
                                                         'padding':'3px','font-size':'20px'})
                            # place them next to each other using division style
                           ], style = {'display':'flex'})])]),
                       html.Br(), html.Div(id = 'my-output' )
])

# add callback
@callback(Output(component_id='my-output', component_property='children'),
          Input(component_id='input-type', component_property='value')
)

def graph(chart):
    if chart == 'OPT1':
        a,b,c,d,e,f = sales_performance()
        return html.Div([html.Div([html.Div([dcc.Graph(figure = a)],style={'width': '50%',  'display': 'inline-block'}),
                        html.Div([dcc.Graph(figure = b)], style={'width': '50%',  'display': 'inline-block'})]),
                        dcc.Graph(figure = c),
                        html.Div([html.Div([dcc.Graph(figure = d)], style={'width': '50%',  'display': 'inline-block'}),
                        html.Div([dcc.Graph(figure = e)], style={'width': '50%',  'display': 'inline-block'})]),
                        dcc.Graph(figure = f)])

    if chart == 'OPT2':
        a,b,c,d,e,f = time_series_analysis()
        return html.Div([html.Div([html.Div([dcc.Graph(figure = a)], style={'width': '50%',  'display': 'inline-block'}),
                        html.Div([dcc.Graph(figure = b)],style={'width': '50%',  'display': 'inline-block'})]),
                        dcc.Graph(figure = c),
                        html.Div([html.Div([dcc.Graph(figure = d)], style={'width': '50%',  'display': 'inline-block'}),
                        html.Div([dcc.Graph(figure = e)], style={'width': '50%',  'display': 'inline-block'})]),
                        dcc.Graph(figure = f)])
    if chart =='OPT3':
        a,b = cost_analysis()
        return html.Div(
            [html.H3('Cost of Goods Sold',style={'color':colors['text'],'font-size':'20px', 'padding-left':'40px'}),
             create_table(a),
             dcc.Graph(figure = b)])

    if chart == 'OPT4':
```

```
        a,b,c = geographic_data()

        return html.Div([
            html.Div([html.H3('Volume of Exports per Port',
                              style={'color':colors['text'],'font-size':'20px', 'padding-left':'40px'}),
                      create_table(a)], style={'width': '50%',  'display': 'inline-block'}),
            html.Div([html.H3('Top Export Product per Port',
                              style={'color':colors['text'],'font-size':'20px', 'padding-left':'40px'}),
                      create_table(c)], style={'width': '50%',  'display': 'inline-block'}),
            dcc.Graph(figure=b)
        ])

    else:
        a,b,c,d,e,f = performance_comparison()
        return html.Div([dcc.Graph(figure = a ),
                         html.Div([html.Div([dcc.Graph(figure = b)],style={'width': '50%',  'display': 'inline-block'}),
                         html.Div([dcc.Graph(figure = c)],style={'width': '50%',  'display': 'inline-block'})]),
                         html.Div([html.Div([dcc.Graph(figure = d)],style={'width': '50%',  'display': 'inline-block'}),
                         html.Div([dcc.Graph(figure = e)], style={'width': '50%',  'display': 'inline-block'})]),
                         dcc.Graph(figure = f)])

# run app
if __name__ == '__main__':
    app.run_server(debug=True)
```

In [ ]: