```
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output, State
import plotly.graph_objects as go
import plotly.express as px
from dash import no_update
# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()


# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
launch_sites = []
launch_sites.append({'label': 'All Sites', 'value': 'All Sites'})
for item in spacex_df["Launch Site"].value_counts().index:
launch_sites.append({'label': item, 'value': item})
app.layout = html.Div(children=[html.H1('SpaceX Launch Records
Dashboard',
style={'textAlign': 'center', 'color': '#503D36',
'font-size': 40}),
# TASK 1: Add a dropdown list to enable Launch Site selection
# The default select value is for ALL sites
dcc.Dropdown(id='site-dropdown', options = launch_sites, value = 'All
Sites', placeholder = "Select a Launch Site here", searchable = True),
html.Br(),
# TASK 2: Add a pie chart to show the total successful launches count
for all sites
# If a specific launch site was selected, show the Success vs. Failed
counts for the site
html.Div(dcc.Graph(id='success-pie-chart')),
html.Br(),

html.P("Payload range (Kg):"),
# TASK 3: Add a slider to select payload range
```

```
dcc.RangeSlider(id='payload-slider', min = 0, max = 10000, step = 1000,
value = [min_payload, max_payload], marks={ 2500: {'label': '2500
(Kg)'}, 5000: {'label': '5000 (Kg)'}, 7500: {'label': '7500 (Kg)'}}),

# TASK 4: Add a scatter chart to show the correlation between payload
and launch success
html.Div(dcc.Graph(id='success-payload-scatter-chart')),
])

# TASK 2:
# Add a callback function for `site-dropdown` as input,
`success-pie-chart` as output
@app.callback( Output(component_id='success-pie-chart',
component_property='figure'),
Input(component_id='site-dropdown', component_property='value')
)
# Add computation to callback function and return graph
def select(inputt):
if inputt == 'All Sites':
new_df = spacex_df.groupby(['Launch Site'])["class"].sum().to_frame()
new_df = new_df.reset_index()
fig = px.pie(new_df, values='class', names='Launch Site', title='Total
Success Launches by Site')
else:
new_df = spacex_df[spacex_df["Launch Site"] ==
inputt]["class"].value_counts().to_frame()
new_df["name"] = ["Failure", "Success"]
fig = px.pie(new_df, values='class', names='name', title='Total Success
Launches for ' + inputt)
return fig

# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as
inputs, `success-payload-scatter-chart` as output
@app.callback( Output(component_id='success-payload-scatter-chart',
component_property='figure'),
Input(component_id='site-dropdown', component_property='value'),
Input(component_id='payload-slider', component_property='value')
)
def scatter(input1, input2):
print(input1)
print(input2)
if input1 == 'All Sites':
```

```
new_df = spacex_df
new_df2 = new_df[new_df["Payload Mass (kg)"] >= input2[0]]
new_df3 = new_df2[new_df["Payload Mass (kg)"] <= input2[1]]
fig2 = px.scatter(new_df3, y="class", x="Payload Mass (kg)",
color="Booster Version Category")
else:
new_df = spacex_df[spacex_df["Launch Site"] == input1]
new_df2 = new_df[new_df["Payload Mass (kg)"] >= input2[0]]
new_df3 = new_df2[new_df["Payload Mass (kg)"] <= input2[1]]
#new_df2 = new_df[new_df["Payload Mass (kg)"] >= input2[0] &
new_df["Payload Mass (kg)"] <= input2[1]]
fig2 = px.scatter(new_df3, y="class", x="Payload Mass (kg)",
color="Booster Version Category")
return fig2

# Run the app
if __name__ == '__main__':
app.run_server()
```