Assignment 5: Interactive Data Visualization with Plotly and Dash

Muhammad Raees (mr2714), Ali Khalid (ak5013), Kaleem Nawaz Khan (kk5271), ISTE-782, Spring 2023

Summary

In this dashboard, we explored and visualized a dataset (http://roycekimmons.com/tools/generated_data/exams) that contains information about the scores of students in math, reading, and writing. Together with the exam results, it also lists the student's ethnicity or race, gender, the level of education of their parents, and if they have access to regular meals and test preparation classes. We examined the data in detail in the last assignment, and in this work, we would like to provide interactivity features to the users through a Plotly Dashboard to examine, evaluate, and interact with the dataset. The interactive features will provide users to dynamically select the data, apply filters, analyze, and visually interpret the results. The main graph of this interactive dashboard is a scatter plot, which allows users to visualize and evaluate the test scores of students and identify patterns. To make the graph more interactive, we provided control to the users to dynamically select the scores to compare. For instance, a user can compare any type of test score with another type of test score (including the overall score which we calculate as an average). To add more interactivity, we allow users to compare the scores across various differentiating factors like gender, ethnicity, whether they get lunches, practice, etc. We provide these options to the users through a set of drop-drop options at the top. Additionally, the user can filter out data based on the education level of parents with multi-selection in a drop-down. Our data does not inherently contain a range of numeric data which might be useful for a slider. However, we use the ethnic background of the user on a categorical slider to choose from a set of ethnicities present in the dataset. Evidently, we can visualize and interact with data more vividly through the Plotly Dash application. We designed the application into multiple pages so that the visualization is separated effectively. The second graph on the homepage shows the correlation between each type of score. All the graphs are interactive with all the controls provided in the application. We also provide the distribution of the data similar to the last assignment. The user can navigate to the "Distribution" page from the top of the page to access different distributions.

Table of Contents

Introduction

We deployed the application to a remote server. The live application can be accessed here here (https://students-mv7p.onrender.com/). The application can also be run on the local server by running the last cell independently and by accessing the link http://127.0.0.1:8051/ (http://127.0.0.1:8051/). The last line of code shows the port number. If the port is not available, try changing the port and running again.

The application may take some time (30 secs) to load on the remote server. The server hibernates the application when not accessed by the users becuase of free version. Please try refreshing the page after 30 second or 1 minute interval if the application fails to load on the first try.

The following code imports the needed libraries; most libraries will come along with Anaconda installation. Following components may be installed if not present on the device using pip.

- pip install dash
- pip install dash-bootstrap-components
- pip install plotly-express

In [1]:
```
import dash
import dash_bootstrap_components as dbc
from jupyter_dash import JupyterDash

from dash.dependencies import Output, Input
from dash import html, dcc

import plotly.graph_objects as go
from urllib.parse import unquote

import numpy as np
import pandas as pd

import plotly.express as px
```

Data Set

We are going to perform exploratory data analysis of a dataset (http://roycekimmons.com/tools/generated_data/exams) that contains the information about the score of students in different subjects like math, reading, and writing. This dataset also contains the information about the parents's education, ethnicity/race, gender, lunch, and test preparation of student.

In [2]:
```
exams = pd.read_csv('https://raw.githubusercontent.com/raeeschaudhary/coursera_test/main/exams.csv')
overall = (exams.math_score + exams.reading_score + exams.writing_score)/3
exams["overall"] = overall
```

Data Visualizations

We will try to find the varriables that effect the performance of student overall and also in each individual subject. For this purpoe we will use different Plotly Dash (https://plotly.com/dash/) components to build interactive graphs and dashboards. According to Tufte (2001) the plots and analysis based on these plots are only as good as the data itself. Therefore in the next two sections we will explore the distribution of categorical and numerical features so that our conclusions are well informed and incorporate and unbalance in the dataset.

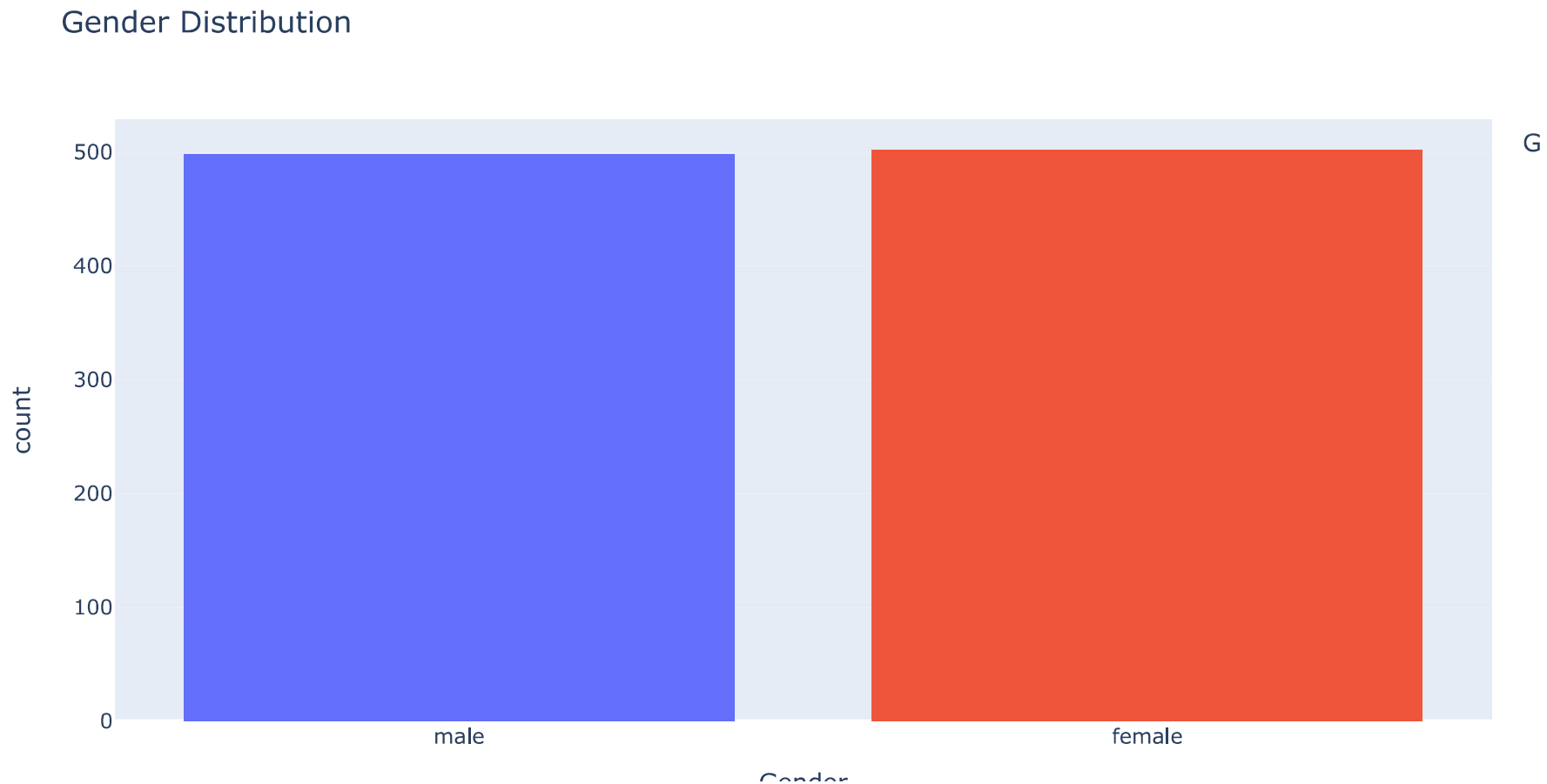
Distribution of Features

In this section we can see the distribution of features in the data through interactive Plotyly graphs.

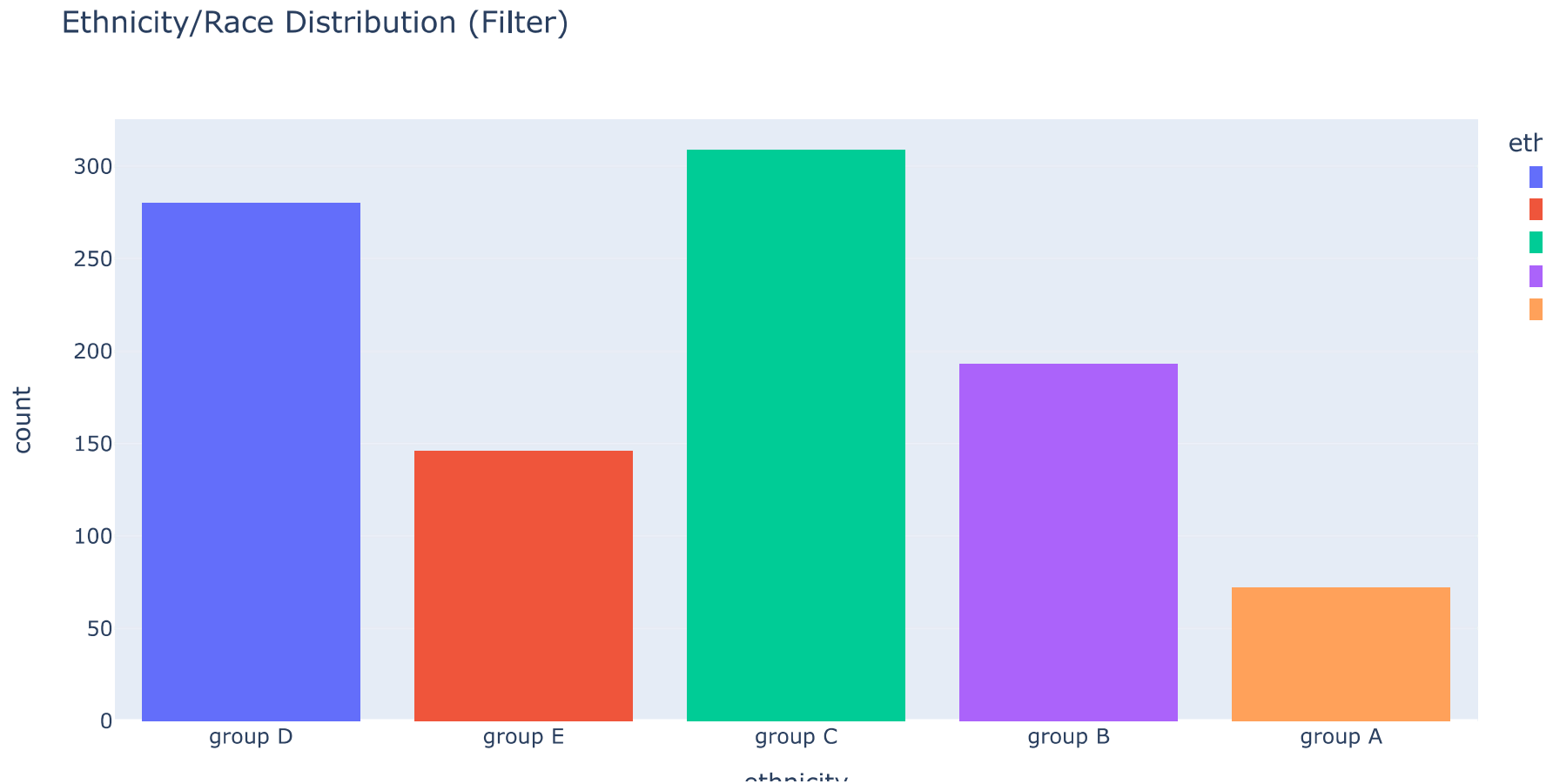
In [3]:
```
def display_dist(exams):
    filtered = exams.copy()
    fig1 = px.histogram(filtered, x="gender", color="gender", labels={'gender':'Gender'},
                        title='Gender Distribution')
    fig2 = px.histogram(filtered, x="ethnicity", color="ethnicity", labels={'ethniciy':'Ethniciy/Race'},
                        title='Ethnicity/Race Distribution (Filter)')
    fig3 = px.histogram(filtered, x="parental_level_of_education", color="parental_level_of_education",
                        labels={'parental_level_of_education':'Parental Level of Education'},
                        title='Parental Level of Education Distribution (Filter)')
    fig4 = px.histogram(filtered, color="lunch", x="lunch", labels={'lunch':'Lunch Program'},
                        title='Lunch Program Distribution')
    fig5 = px.histogram(filtered, x="test_preparation_course", color="test_preparation_course",
                        labels={'test_preparation_course':'Test Preparation Course'},
                        title='Test Preparation Distribution')
    fig6 = px.histogram(filtered, x="math_score", labels={'math_score':'Math Score'},
                        title='Math Score Distribution')
    fig7 = px.histogram(filtered, x="writing_score", labels={'writing_score':'Writing Score'},
                        title='Writing Score Distribution')
    fig8 = px.histogram(filtered, x="reading_score", labels={'reading_score':'Reading Score'},
                        title='Reading Score Distribution')



    return fig1, fig2, fig3, fig4, fig5, fig6, fig7, fig8
```

In [4]:
```
fig1, fig2, fig3, fig4, fig5, fig6, fig7, fig8 = display_dist(exams)
```

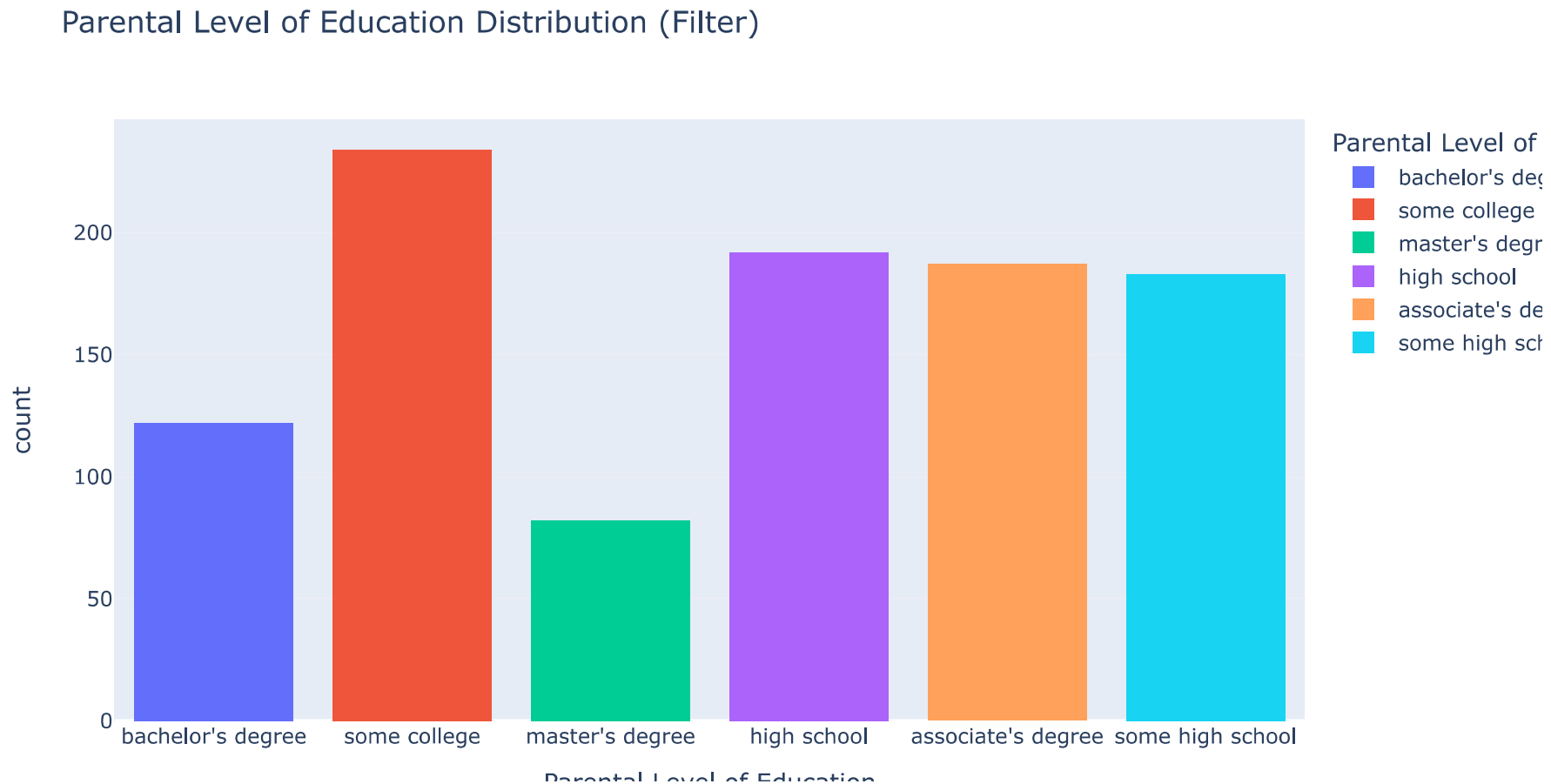
In [5]:
```
fig1.show()
```



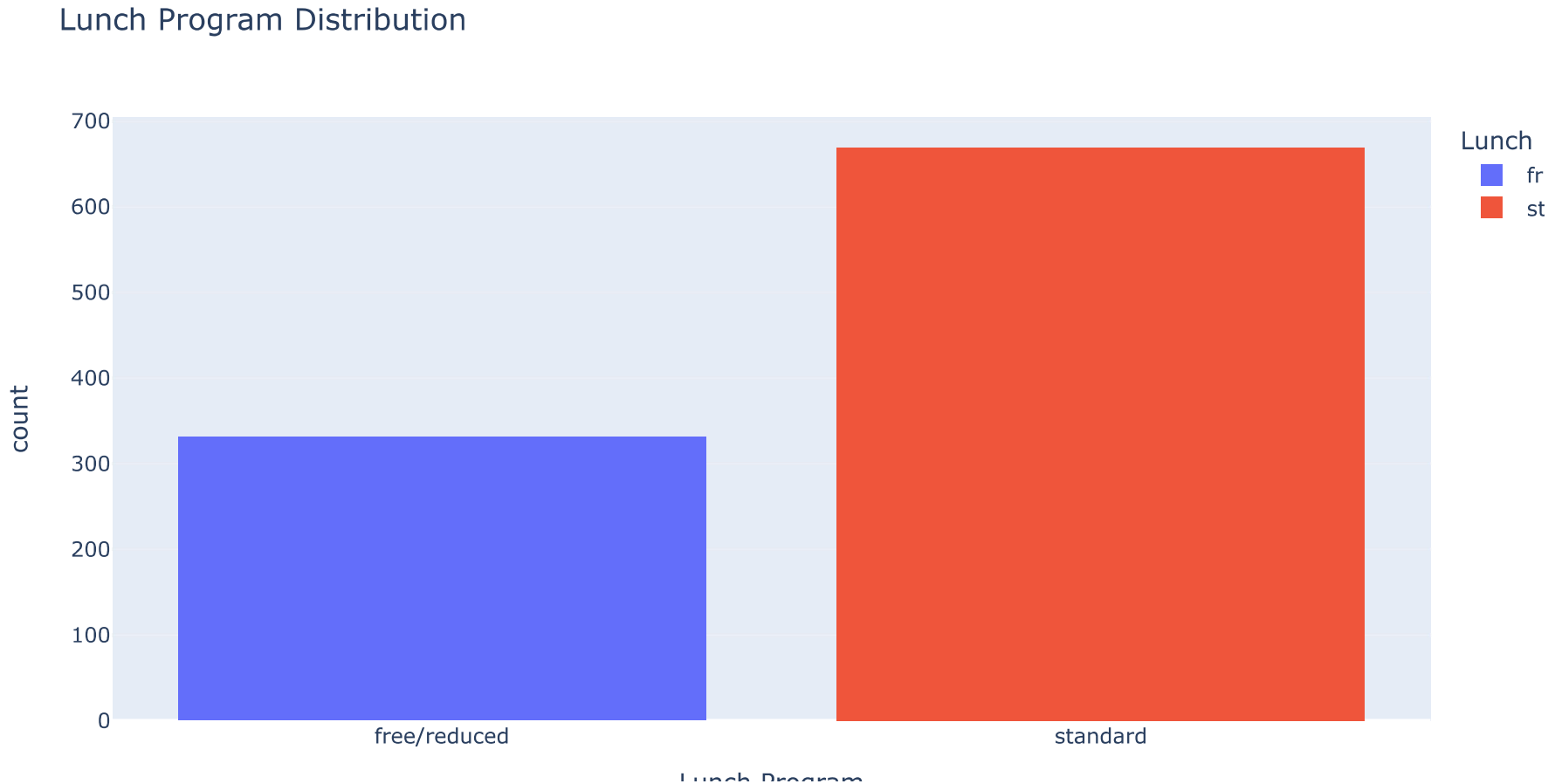
In [6]:
```
fig2.show()
```



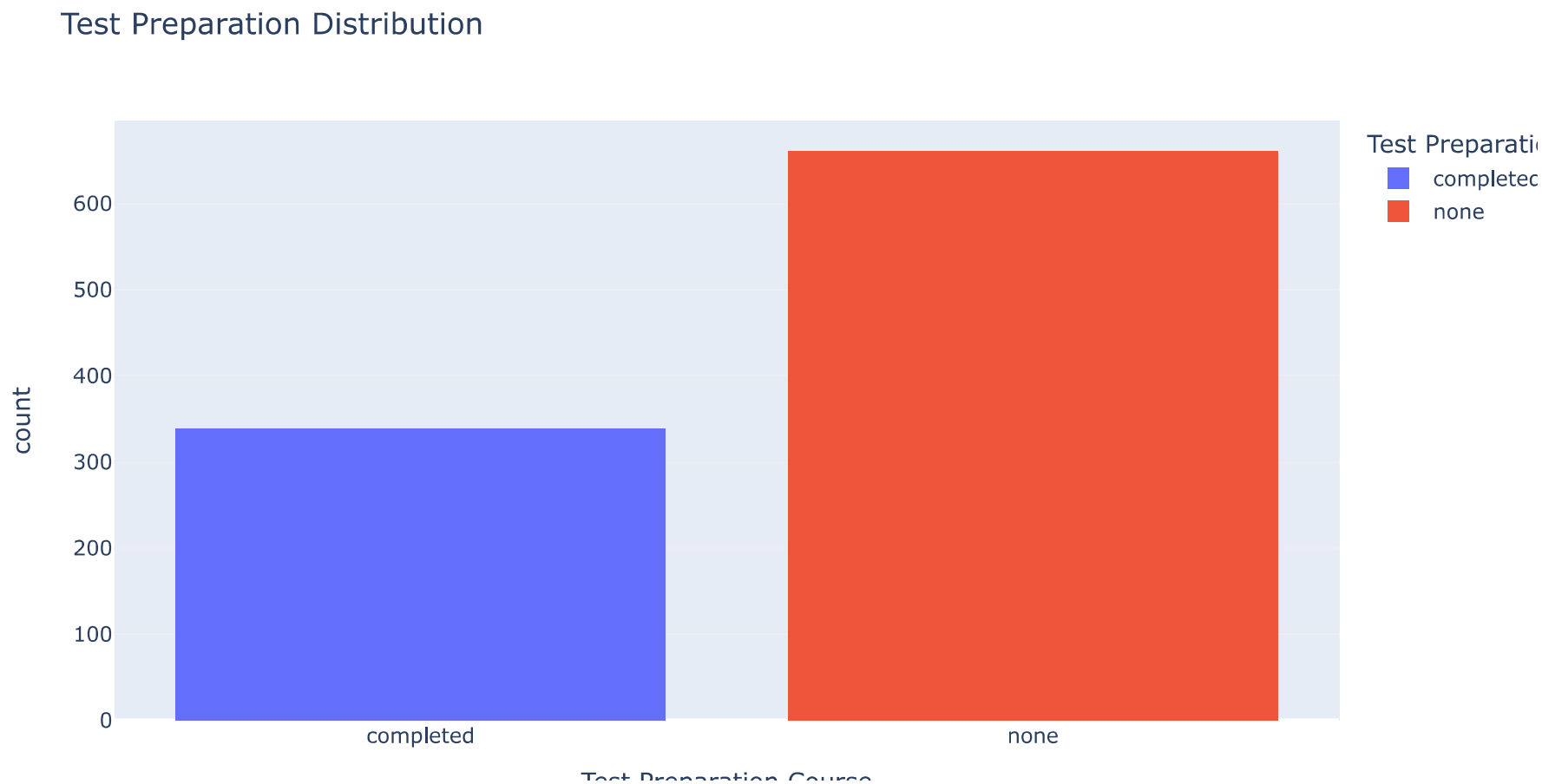
In [7]:
```
fig3.show()
```



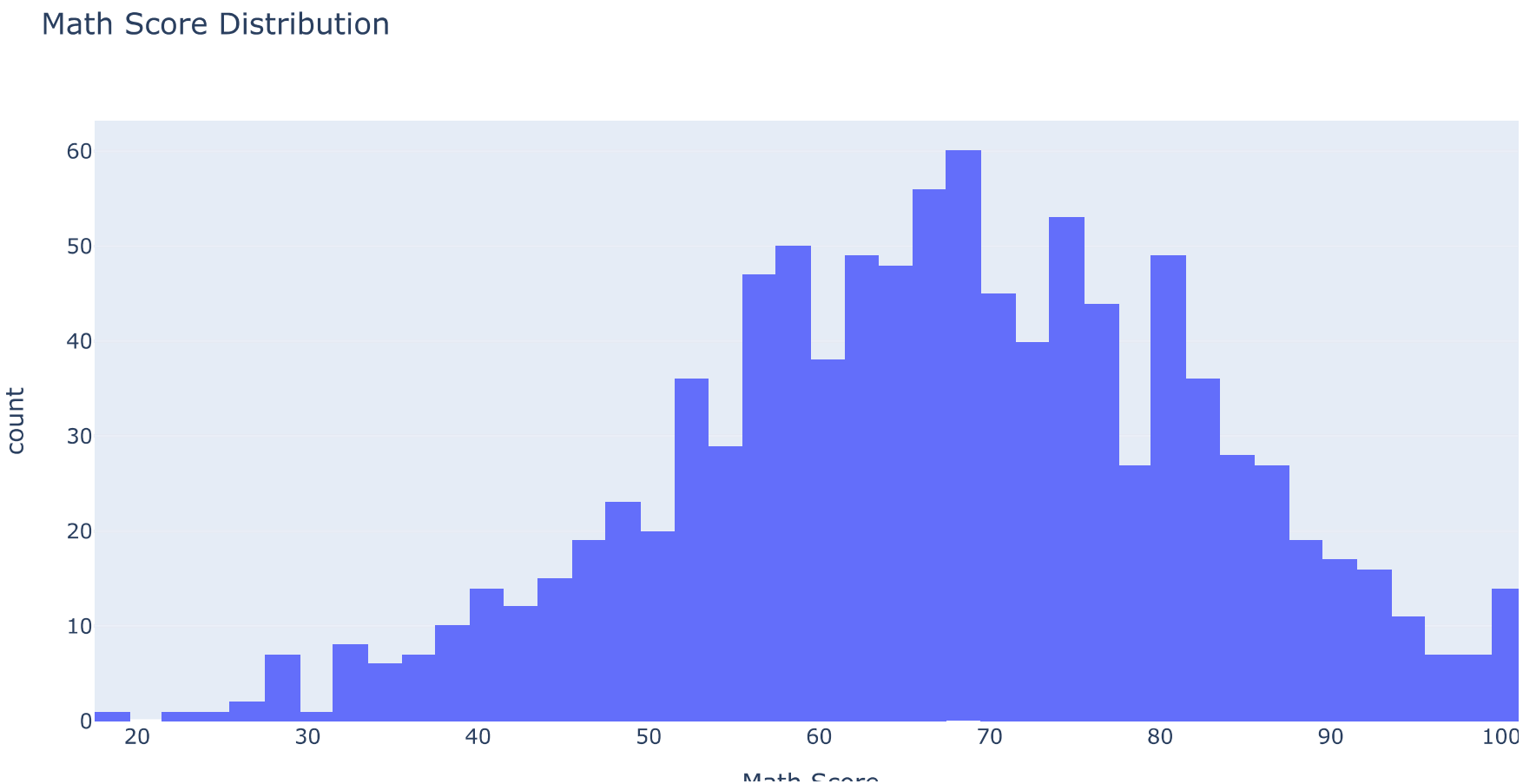
In [8]:
```
fig4.show()
```



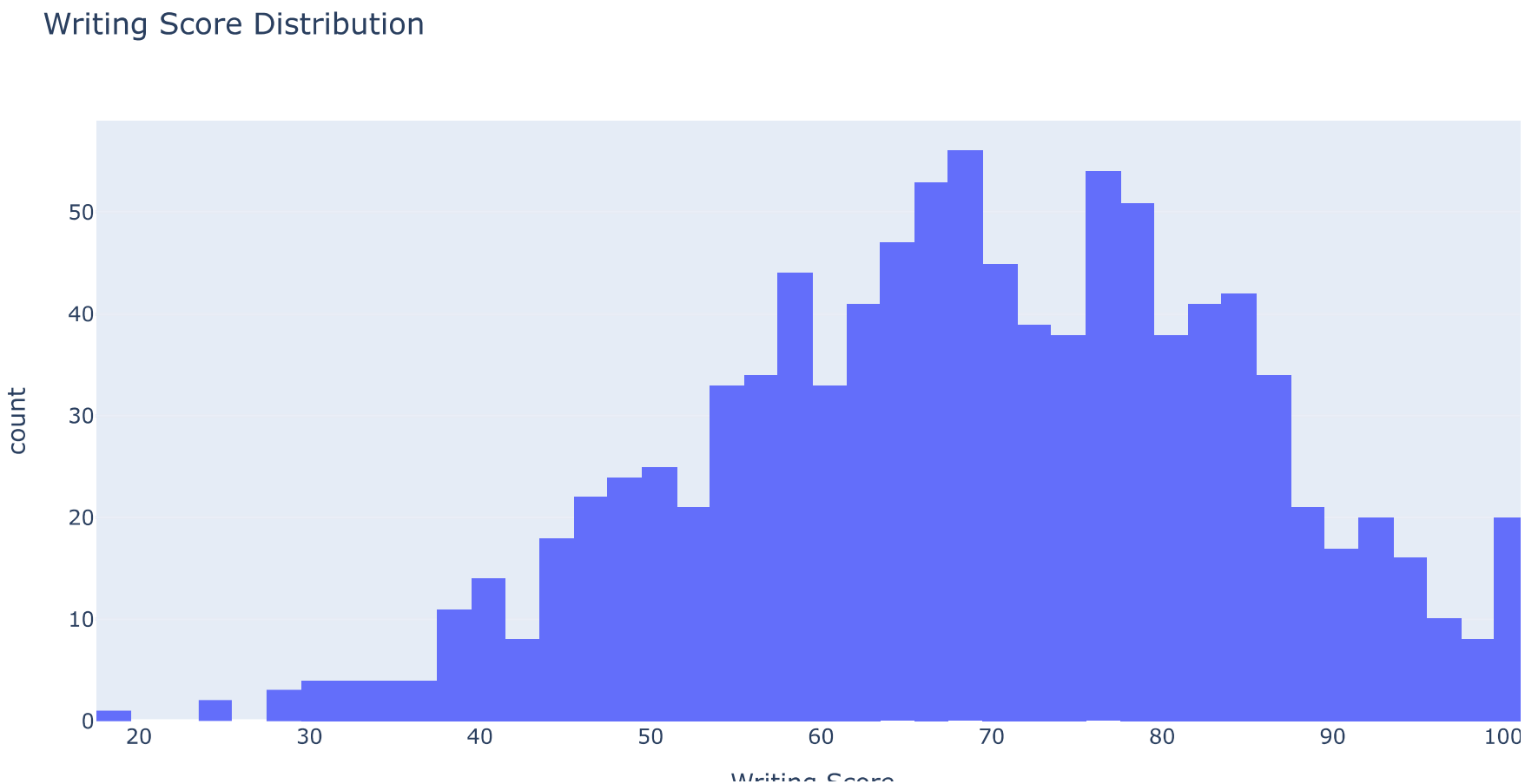
In [9]:
```
fig5.show()
```



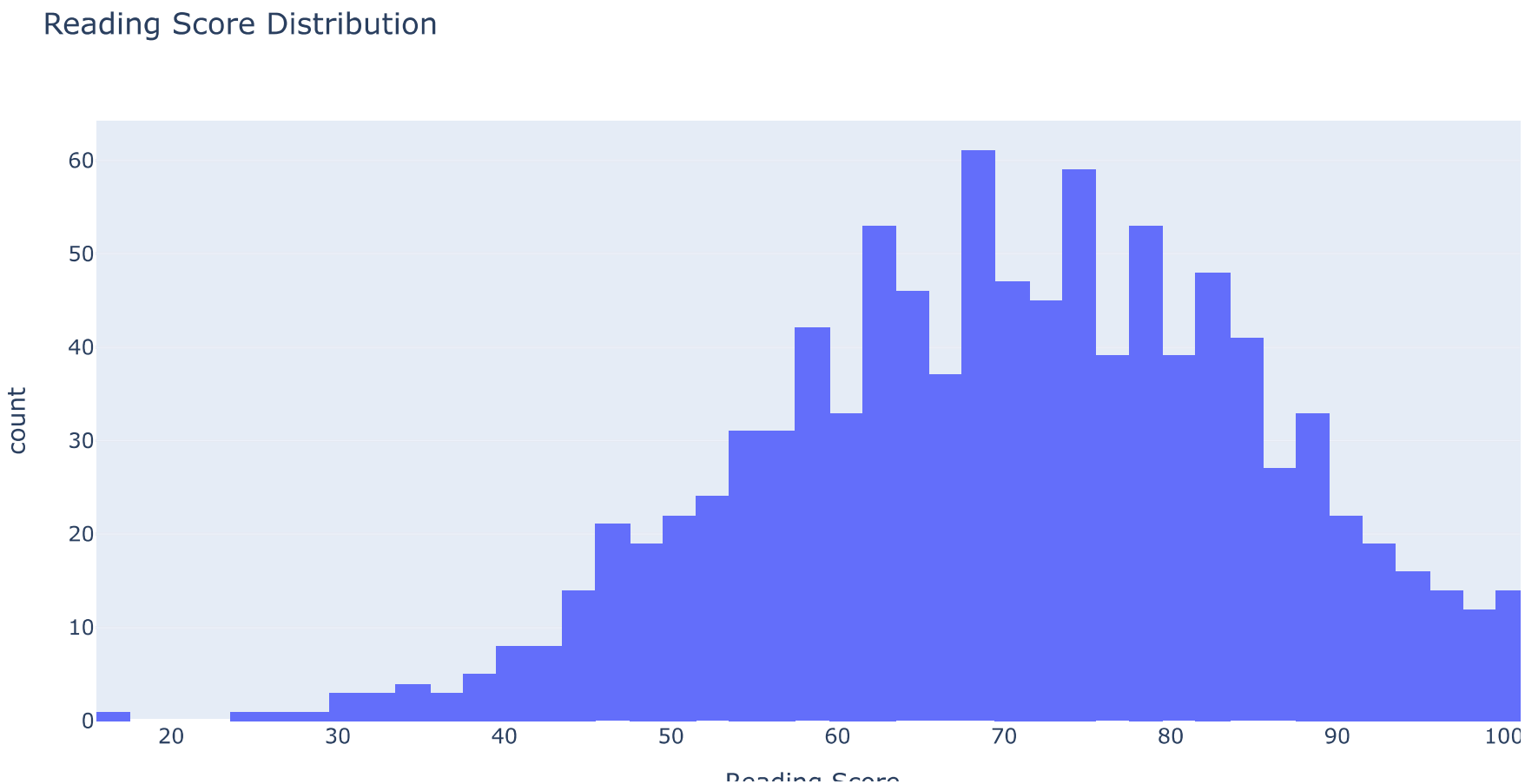
In [10]:
```
fig6.show()
```



In [11]:
```
fig7.show()
```



In [12]:
```
fig8.show()
```

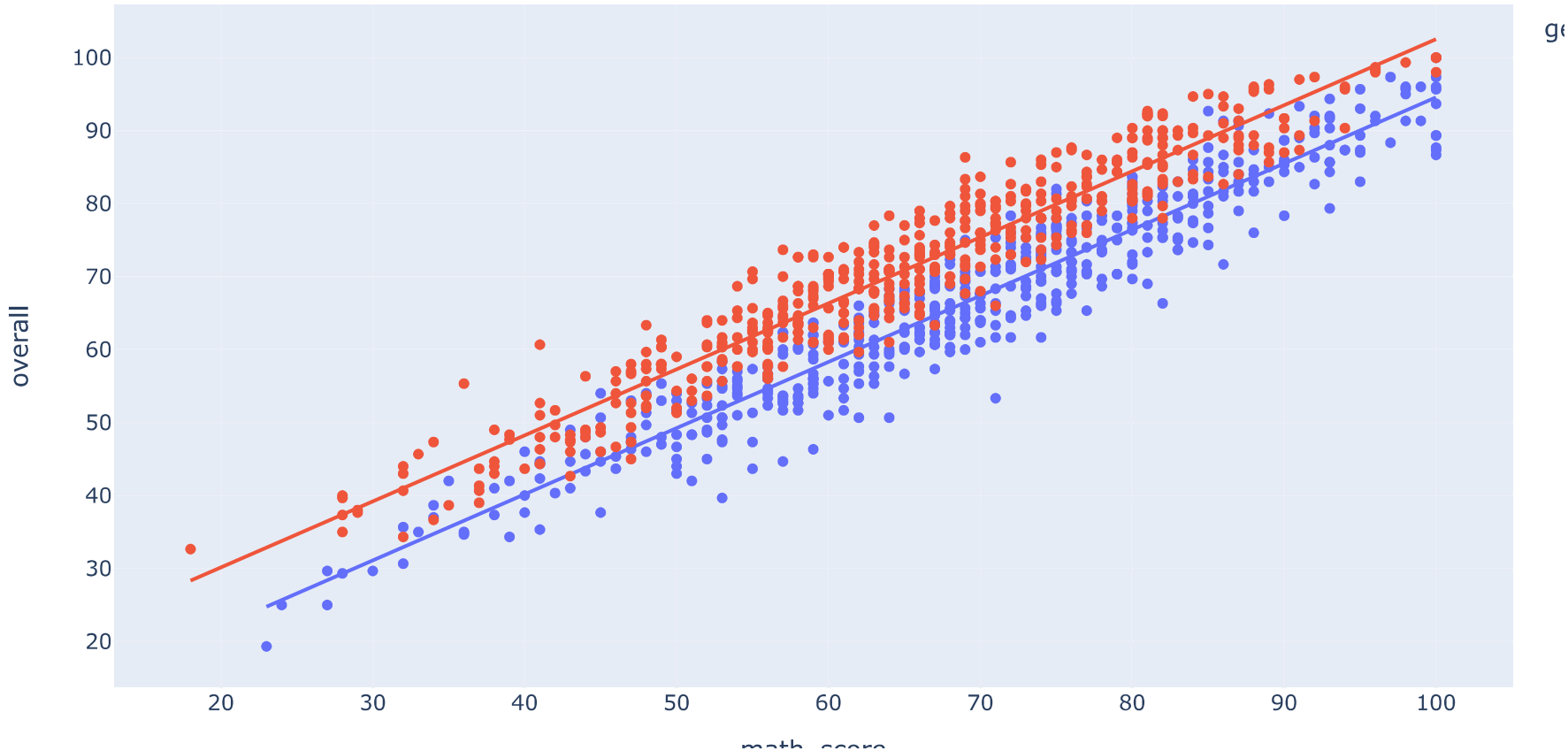


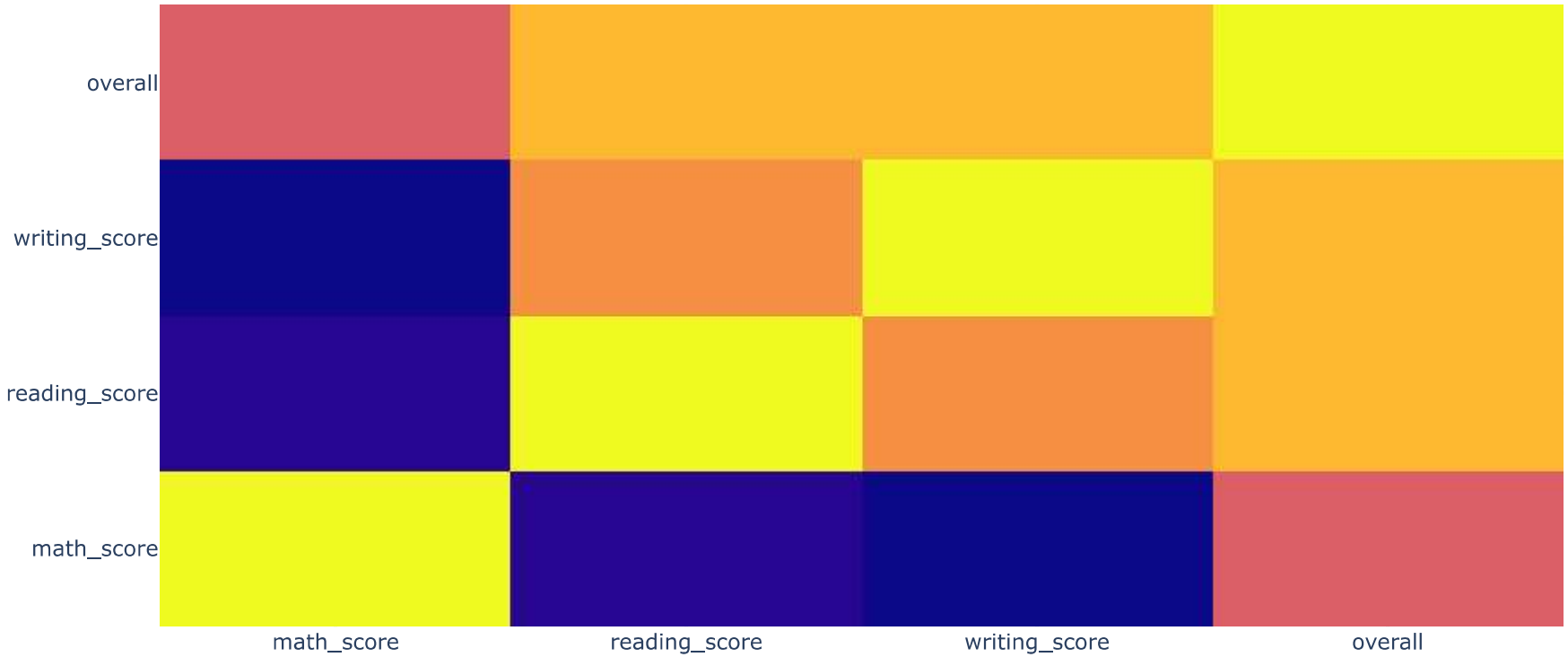
Trends in Scores

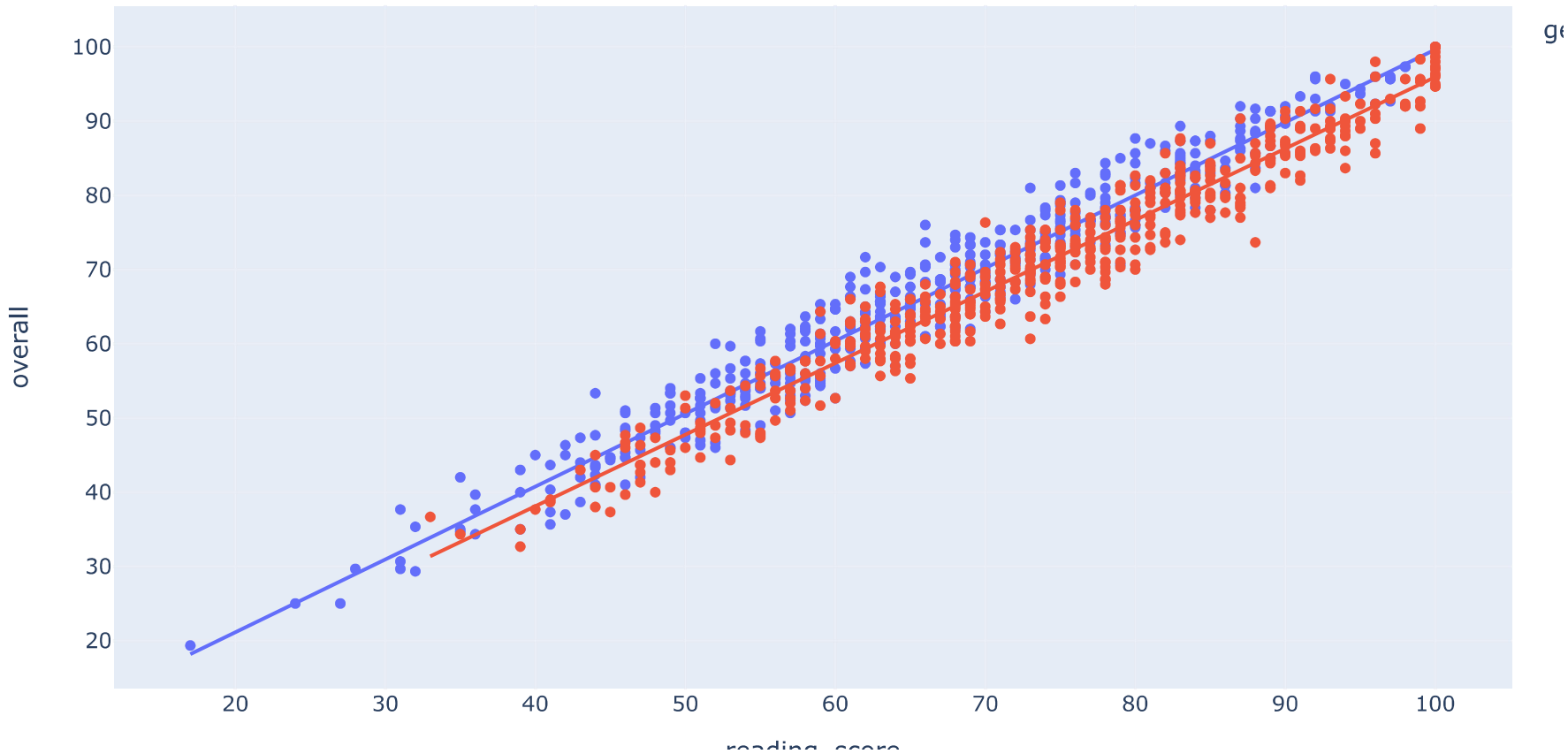
Tufte (2001) suggests that the information content of the graphic should be high quality. A set of scatterplots and heatmaps shows the effect of information with respect to observed patterns and correlations.

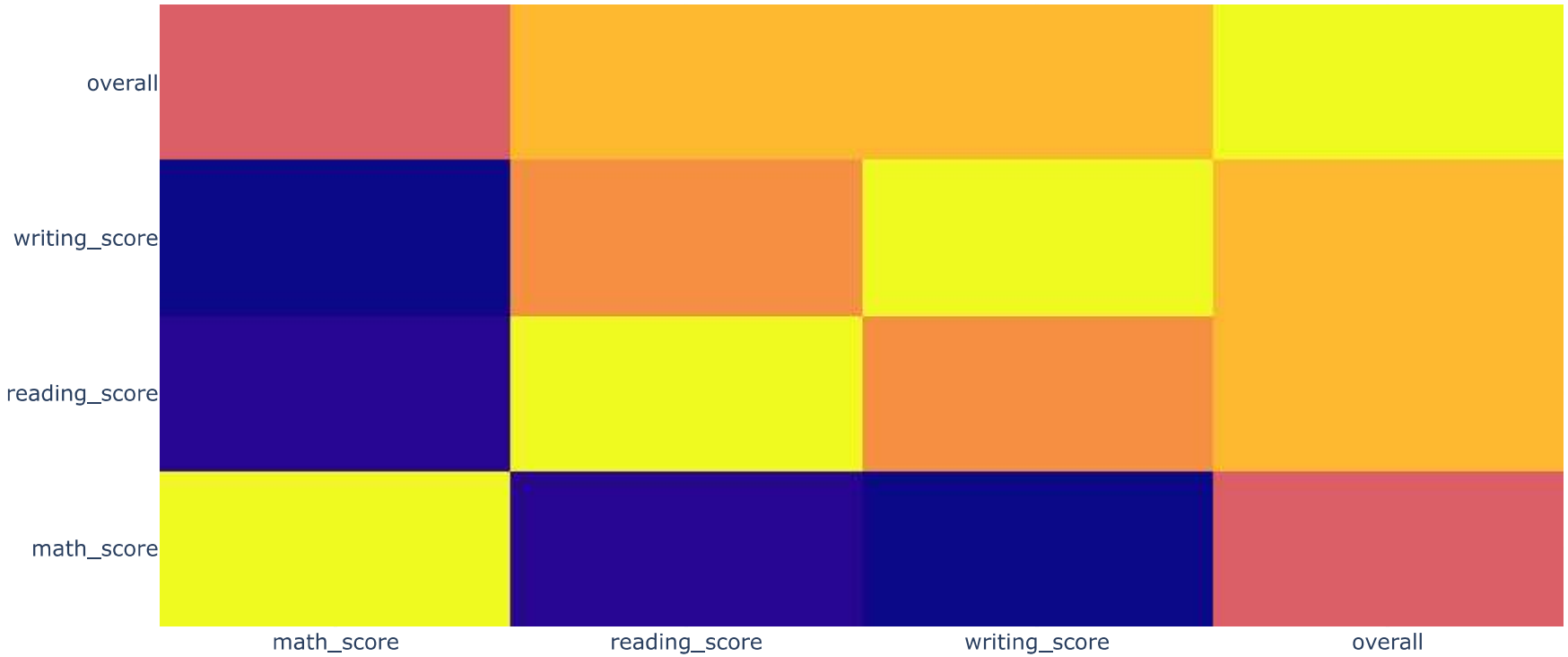
In [13]:
```
def display_main(exams, diff, ques, comp):
    filtered = exams.copy()
    fig1 = px.scatter(filtered, x=ques, y=comp, color=diff, hover_data=[diff], trendline="ols")
    cols = ['math_score', 'reading_score', 'writing_score', 'overall']
    df_corr = filtered[cols].corr().round(2)
    fig2 = go.Figure()
    fig2.add_trace(go.Heatmap(x = df_corr.columns, y = df_corr.index, z = np.array(df_corr)))
    return fig1, fig2
```

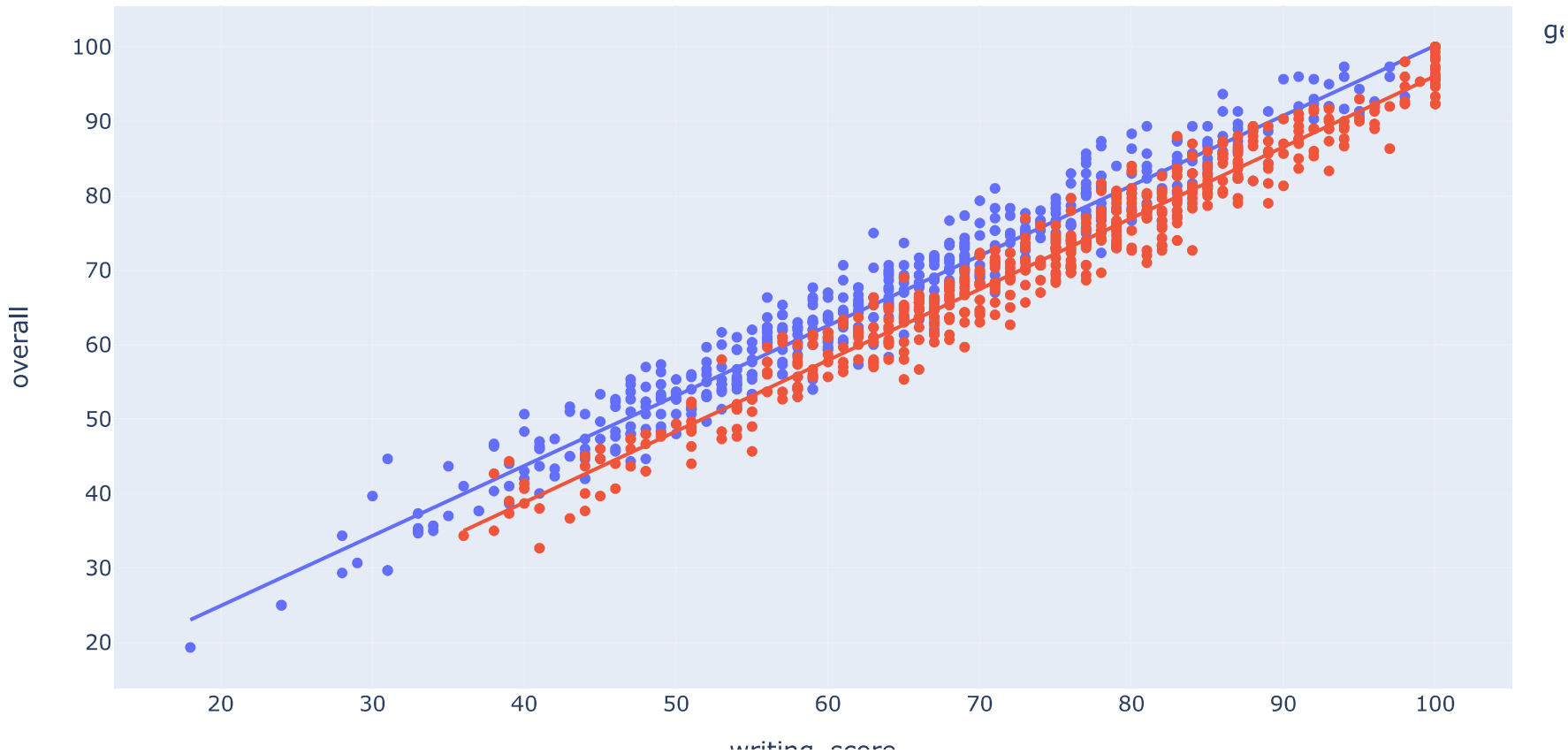
In [14]:
```
for ques in ['math_score', 'reading_score', 'writing_score', 'overall']:
    diff = 'gender'
    comp = 'overall'
    fig1, fig2 = display_main(exams, diff, ques, comp)
    fig1.show()
    fig2.show()
```

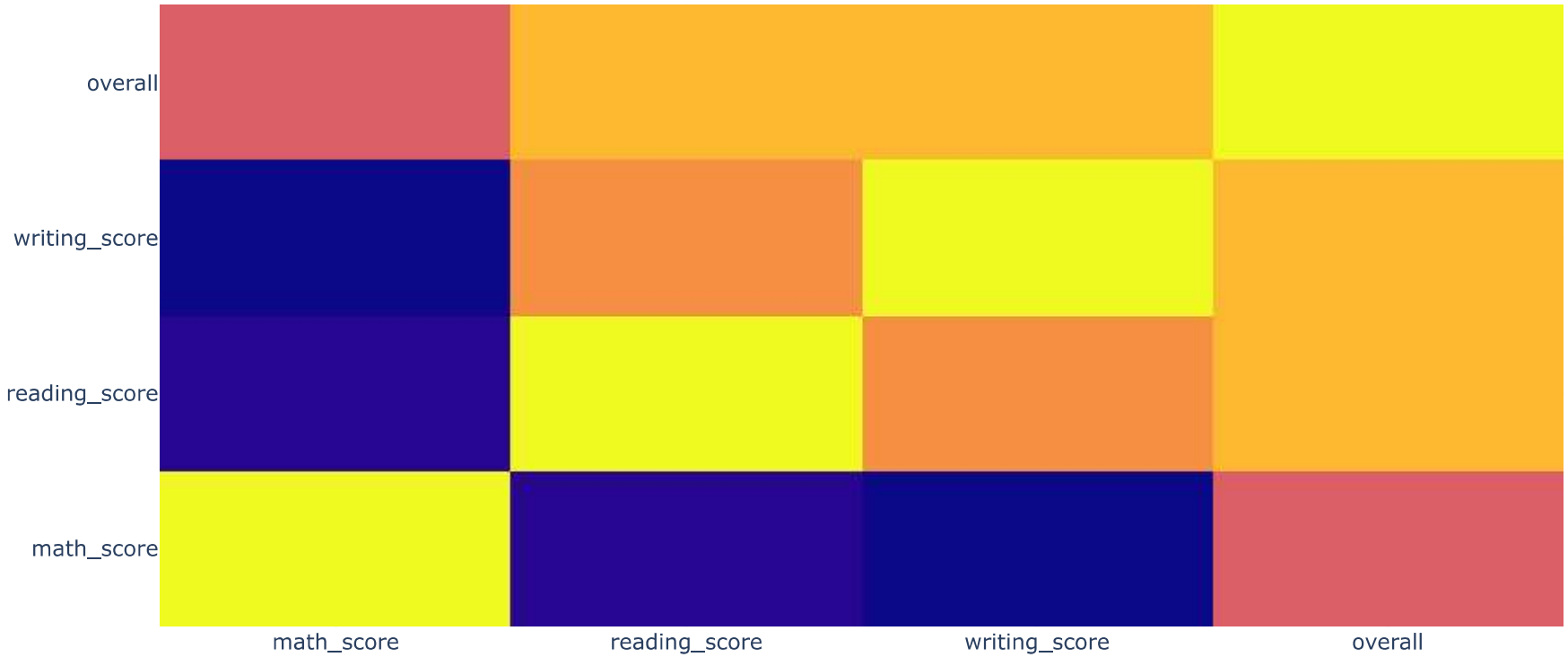


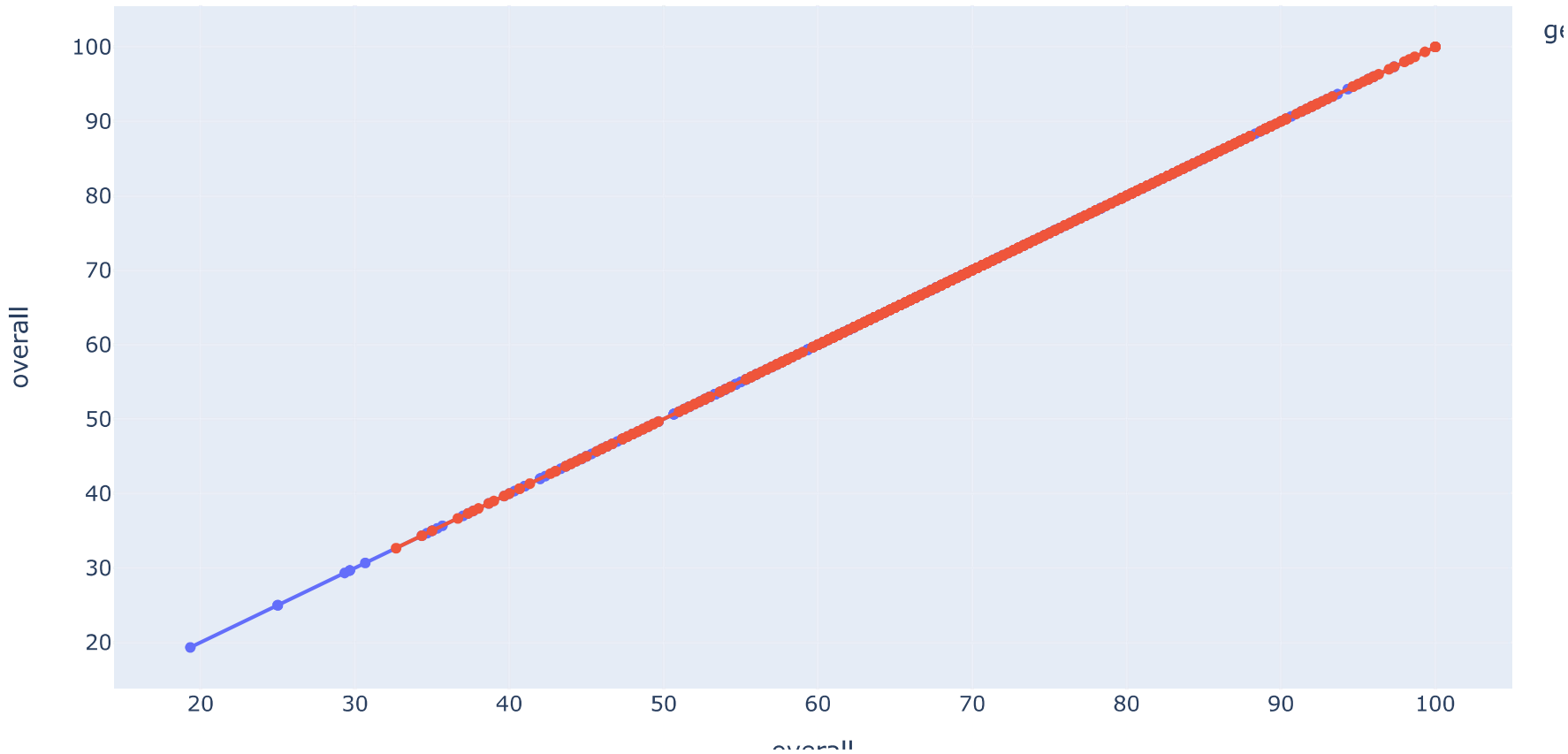

overall
writing_score
reading_score
math_score
math_score
reading_score
writing_score
overall

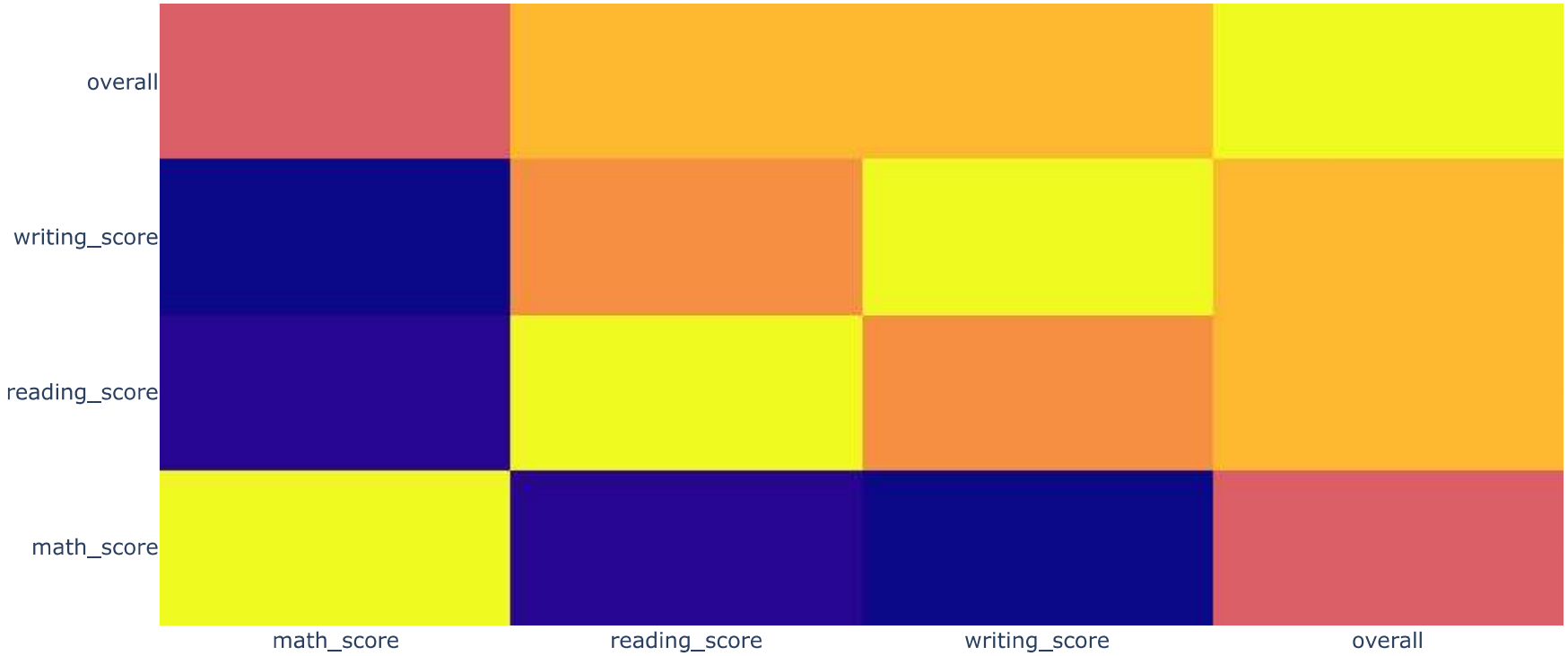

overall
100
90
80
70
60
50
40
30
20
20
30
40
50
60
70
80
90
100


overall
writing_score
reading_score
math_score
math_score
reading_score
writing_score
overall


overall
100
90
80
70
60
50
40
30
20
20
30
40
50
60
70
80
90
100


overall
writing_score
reading_score
math_score
math_score
reading_score
writing_score
overall

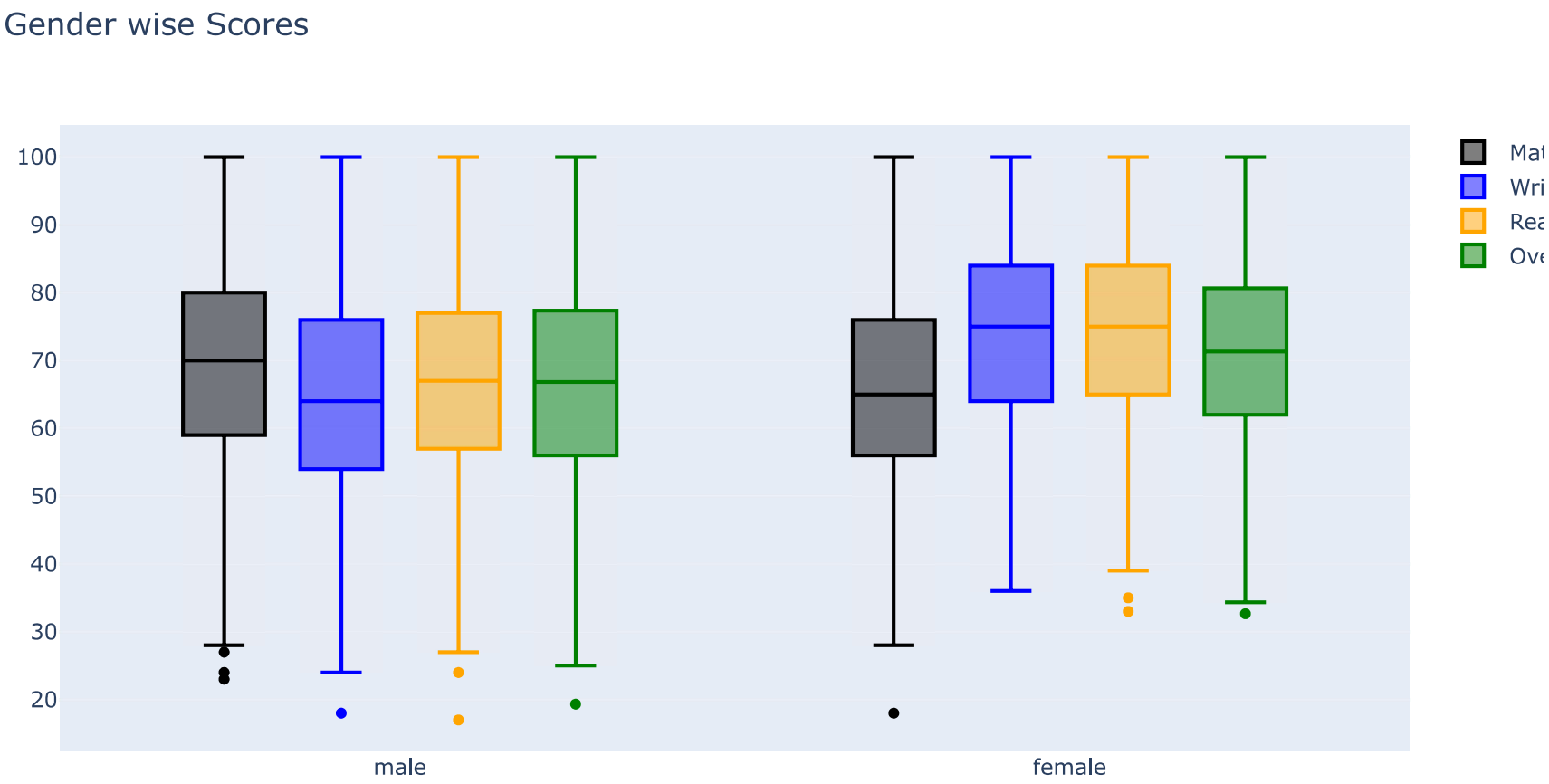

overall
100
90
80
70
60
50
40
30
20
20
30
40
50
60
70
80
90
100
overall



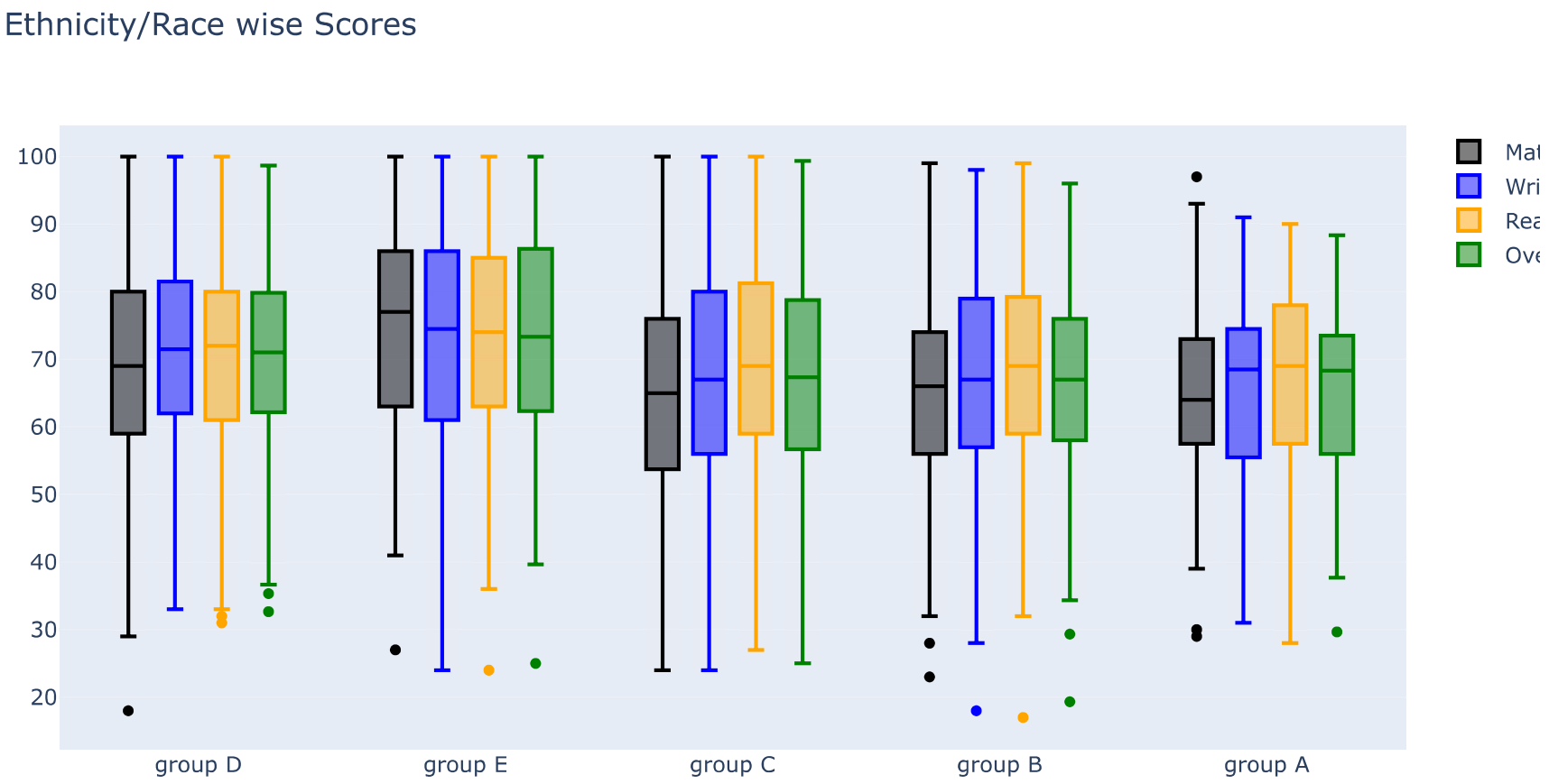
Distribution of Scores

The following plots show the spread of scores across various variables through interactive boxplots. Plots also give information about the quartile ranges.

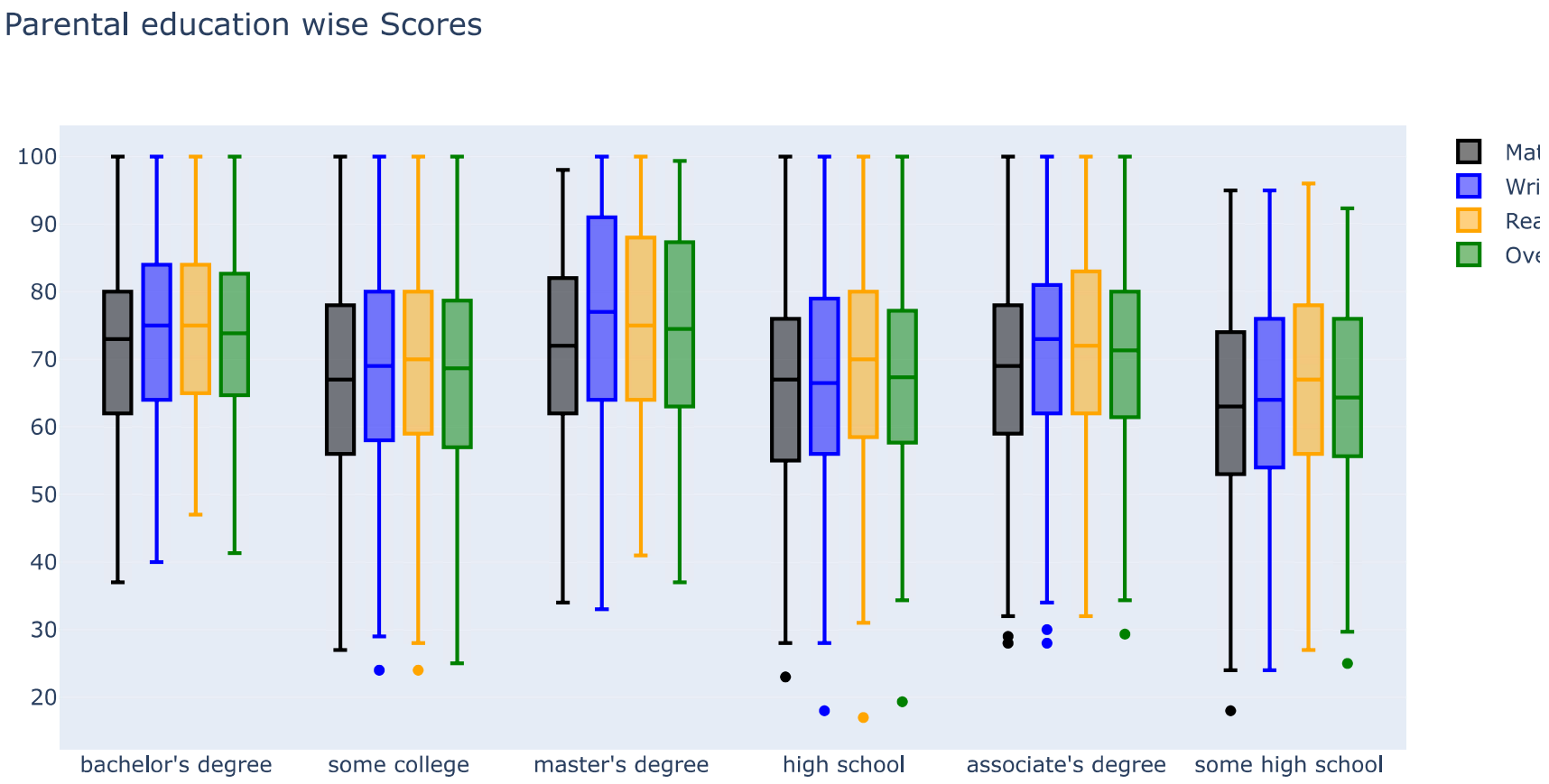
In [15]:
```
fig = go.Figure()
x=exams['gender']
fig.add_trace(go.Box(y=exams['math_score'], x=x, name='Math Score', marker_color='black'))
fig.add_trace(go.Box(y=exams['writing_score'], x=x, name='Writing Score', marker_color='blue'))
fig.add_trace(go.Box(y=exams['reading_score'], x=x, name='Reading Score', marker_color='orange'))
fig.add_trace(go.Box(y=exams['overall'], x=x, name='Overall Score', marker_color='green'))
fig.update_layout(title="Gender wise Scores", boxmode='group')
fig.show()
```



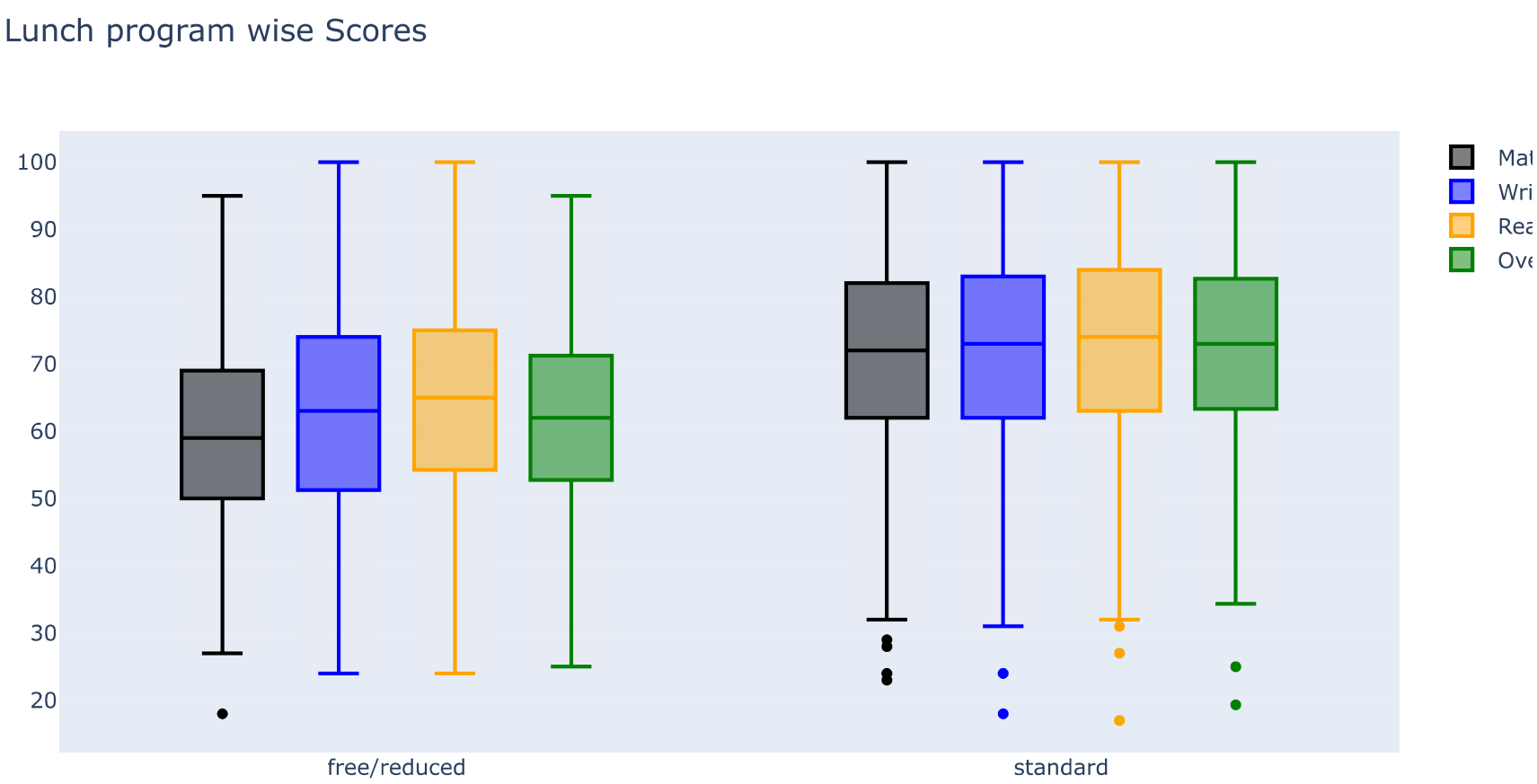
In [16]:
```
fig = go.Figure()
x=exams['ethnicity']
fig.add_trace(go.Box(y=exams['math_score'], x=x, name='Math Score', marker_color='black'))
fig.add_trace(go.Box(y=exams['writing_score'], x=x, name='Writing Score', marker_color='blue'))
fig.add_trace(go.Box(y=exams['reading_score'], x=x, name='Reading Score', marker_color='orange'))
fig.add_trace(go.Box(y=exams['overall'], x=x, name='Overall Score', marker_color='green'))
fig.update_layout(title="Ethnicity/Race wise Scores", boxmode='group')
fig.show()
```



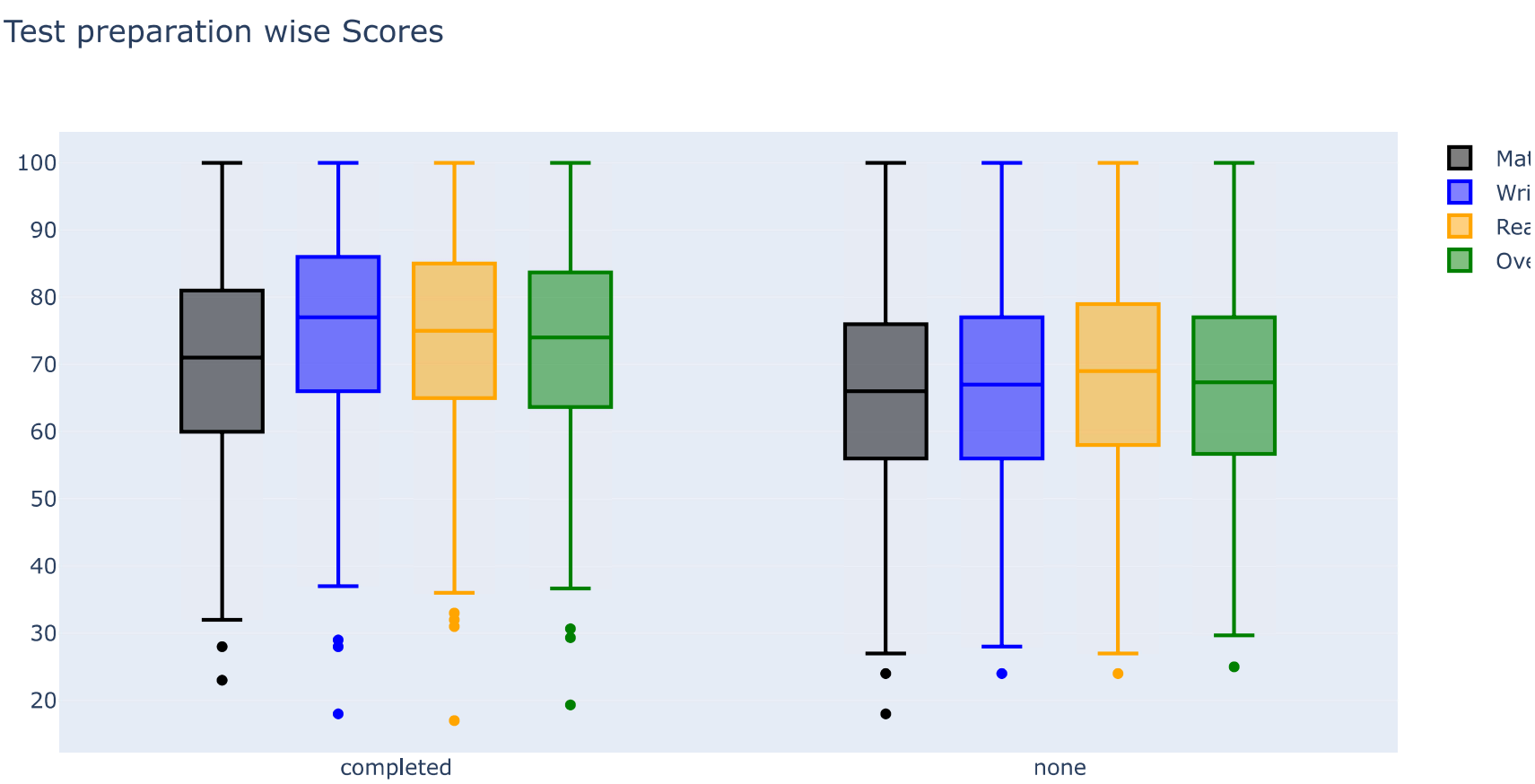
In [17]:
```
fig = go.Figure()
x=exams['parental_level_of_education']
fig.add_trace(go.Box(y=exams['math_score'], x=x, name='Math Score', marker_color='black'))
fig.add_trace(go.Box(y=exams['writing_score'], x=x, name='Writing Score', marker_color='blue'))
fig.add_trace(go.Box(y=exams['reading_score'], x=x, name='Reading Score', marker_color='orange'))
fig.add_trace(go.Box(y=exams['overall'], x=x, name='Overall Score', marker_color='green'))
fig.update_layout(title="Parental education wise Scores", boxmode='group')
fig.show()
```



In [18]:
```
fig = go.Figure()
x=exams['lunch']
fig.add_trace(go.Box(y=exams['math_score'], x=x, name='Math Score', marker_color='black'))
fig.add_trace(go.Box(y=exams['writing_score'], x=x, name='Writing Score', marker_color='blue'))
fig.add_trace(go.Box(y=exams['reading_score'], x=x, name='Reading Score', marker_color='orange'))
fig.add_trace(go.Box(y=exams['overall'], x=x, name='Overall Score', marker_color='green'))
fig.update_layout(title="Lunch program wise Scores", boxmode='group')
fig.show()
```



In [19]:
```
fig = go.Figure()
x=exams['test_preparation_course']
fig.add_trace(go.Box(y=exams['math_score'], x=x, name='Math Score', marker_color='black'))
fig.add_trace(go.Box(y=exams['writing_score'], x=x, name='Writing Score', marker_color='blue'))
fig.add_trace(go.Box(y=exams['reading_score'], x=x, name='Reading Score', marker_color='orange'))
fig.add_trace(go.Box(y=exams['overall'], x=x, name='Overall Score', marker_color='green'))
fig.update_layout(title="Test preparation wise Scores", boxmode='group')
fig.show()
```



Dash Application

We deployed the application to a remote server. The live application can be accessed here here (https://students-mv7p.onrender.com/). The application can also be run on the local server by running the last cell independently and by accessing the link http://127.0.0.1:8051/ (http://127.0.0.1:8051/). The last line of code shows the port number. If the port is not available, try changing the port and running again.

The application may take some time (30 secs) to load on the remote server. The server hibernates the application when not accessed by the users becuase of free version. Please try refreshing the page after 30 second or 1 minute interval if the application fails to load on the first try.

In [20]:

```
import dash
import dash_bootstrap_components as dbc
from jupyter_dash import JupyterDash

from dash.dependencies import Output, Input
from dash import html, dcc

import plotly.graph_objects as go
from urllib.parse import unquote

import numpy as np
import pandas as pd

import plotly.express as px

app = JupyterDash(__name__, external_stylesheets=[dbc.themes.COSMO])
#server=app.server
exams = pd.read_csv('https://raw.githubusercontent.com/raeeschaudhary/coursera_test/main/exams.csv')
overall = (exams.math_score + exams.reading_score + exams.writing_score)/3
exams["overall"] = overall


def make_empty_fig():
    fig = go.Figure()
    fig.layout.paper_bgcolor = '#E5ECF6'
    fig.layout.plot_bgcolor = '#E5ECF6'
    return fig

main_layout = html.Div([
    html.Div([
    dbc.NavbarSimple(
    children=[
        dbc.NavItem(dbc.NavLink("Distribution", href="dist")),
        dbc.NavItem(dbc.NavLink("Score Ranges", href="score")),
    ],
    brand="Student Performance - Data Visualization",
    brand_href="/",
    color="primary",
    dark=True,
    ),
    dbc.NavbarSimple([
```

```
        ]),
    dcc.Location(id='location'),
    html.Div(id='main_content'),
    html.Br(),
]),
    html.Br(),
],  style={'backgroundColor': '#E5ECF6'})

main_dashboard = html.Div([
        dbc.Row([
            dbc.Col(lg=1),
            dbc.Col([
                    dbc.Label('Differentiating Variable'),
                    html.Br(),
                    dcc.Dropdown(id='diff_dropdown',
                         value='gender', options=[{'label': v, 'value': v}
                                                  for v in ['gender', 'ethnicity', 'parental_level_of_education', 'lun
                ], md=12, lg=4),
                dbc.Col([
                    dbc.Label('Score to Compare'),
                    dcc.Dropdown(id='ques_dropdown',
                         value='math_score',
                         options=[{'label': v, 'value': v}
                                  for v in ['math_score', 'reading_score', 'writing_score', 'overall']]),
                ], md=12, lg=3),
                dbc.Col([
                    dbc.Label('Compare Score Against'),
                    dcc.Dropdown(id='comp_dropdown',
                         value='overall',
                         options=[{'label': v, 'value': v}
                                  for v in ['math_score', 'reading_score', 'writing_score', 'overall']]),
                ], md=12, lg=3),
        ], style={'backgroundColor': '#E5ECF6'}),
        dbc.Row([
        dbc.Col(lg=1),
        dbc.Col([
            dbc.Label('Filter Education of Parents'),
            dcc.Dropdown(id='edu_selector',
                         multi=True,
                         placeholder='Select one or more',
                         options=[{'label': edu, 'value': edu}
                                  for edu in exams['parental_level_of_education'].drop_duplicates().sort_values()]),
```

```
        dcc.Graph(id='comparison_graph',
                  figure=make_empty_fig()),
        ], md=12, lg=5),
    dbc.Col([
        dbc.Label('Filter Ethnicity'),
        dcc.Slider(1, 6, step=None, id='ethnicity_slider',
                   marks={
                       1: 'Group A',
                       2: 'Group B',
                       3: 'Group C',
                       4: 'Group D',
                       5: 'Group E',
                       6: "All"
                   },
                   value=6
                  ),
        dcc.Graph(id='heatmap_graph',
                  figure=make_empty_fig()),
        html.Br(),
        ], md=12, lg=5),
    ]),
dbc.Row([
    dbc.Col(lg=1),
    dbc.Col([
        html.H1('Assignment 5: Interactive Data Visualization with Plotly and Dash'),
        html.Hr(),
        html.H3('Muhammad Raees (mr2714), Ali Khalid (ak5013), Kaleem Nawaz Khan (kk5271)'),
        html.H3('ISTE-782, Spring 2023'),
        html.Hr(),
        html.Div([
            html.P('In this dashboard, we explored and visualized a dataset'),
            html.A(dbc.Button('View Dataset', id='record-info-btn',
                  className='btn btn-orange align-middle btn btn-secondary'),
                   href='http://roycekimmons.com/tools/generated_data/exams'),
            html.P('Dataset contains information about the scores of students in math, reading, ' +
                   'and writing. Together with the exam results, it also lists the students ' +
                   'ethnicity or race, gender, the level of education of their parents, and if ' +
                   'they have access to regular meals and test preparation classes. We examined ' +
                   'the data in detail in the last assignment, and in this work, we would like to ' +
                   'provide interactivity features to the users through a Plotly Dashboard to examine, ' +
                   'evaluate, and interact with the dataset. The interactive features will provide users to ' +
```

```
                    'dynamically select the data, apply filters, analyze, and visually interpret the results. ' +
                    'The main graph of this interactive dashboard is a scatter plot, which allows users to ' +
                    'visualize and evaluate the test scores of students and identify patterns. To make the ' +
                    'graph more interactive, we provided control to the users to dynamically select the scores ' +
                    'to compare. For instance, a user can compare any type of test score with another type of ' +
                    'test score (including the overall score which we calculate as an average). ' +
                    'To add more interactivity, we allow users to compare the scores across various ' +
                    'differentiating factors like gender, ethnicity, whether they get lunches, practice, etc. ' +
                    'We provide these options to the users through a set of drop-drop options at the top. ' +
                    'Additionally, the user can filter out data based on the education level of parents ' +
                    'with multi-selection in a drop-down. Our data does not inherently contain a range of ' +
                    'numeric data which might be useful for a slider. However, we use the ethnic background of the
                    'user on a categorical slider to choose from a set of ethnicities present in the dataset. ' +
                    'Evidently, we can visualize and interact with data more vividly through the Plotly Dash ' +
                    'application. We designed the application into multiple pages so that the visualization ' +
                    'is separated effectively. The second graph on the homepage shows the correlation between each
                    'type of score. All the graphs are interactive with all the controls provided in the applicatic
                    'We also provide the distribution of the data similar to the last assignment. ' +
                    'The user can navigate to the "Distribution" page from the top of the page to access ' +
                    'different distributions.'),
                html.H3('Explore the other tabs from the top navigation to learn more about the data through interacti
            ] ,style={'text-align': 'justify'}),

            ], md=12, lg=10),
        dbc.Col([

        ], md=12, lg=5),
    ]),
], style={'backgroundColor': '#E5ECF6'})

dist_dashboard = html.Div([
    dbc.Row([
        dbc.Col(lg=1),
        dbc.Col([
            dbc.Label('Filter Education of Parents'),
            dcc.Dropdown(id='edu_selector1',
                         multi=True,
                         placeholder='Select one or more',
                         options=[{'label': edu, 'value': edu}
                                  for edu in exams['parental_level_of_education'].drop_duplicates().sort_values()]),
            html.Br(),
            dcc.Graph(id='gender_dist_graph',
```

```
                    figure=make_empty_fig()),
            html.Br(),
            dcc.Graph(id='parental_dist_graph',
                    figure=make_empty_fig()),
            html.Br(),
            dcc.Graph(id='test_dist_graph',
                    figure=make_empty_fig()),
            html.Br(),
            dcc.Graph(id='read_dist_graph',
                    figure=make_empty_fig()),
            ], md=12, lg=5),
        dbc.Col([
            dbc.Label('Filter Ethnicity'),
            dcc.Slider(1, 6, step=None, id='ethnicity_slider1',
                       marks={
                           1: 'Group A',
                           2: 'Group B',
                           3: 'Group C',
                           4: 'Group D',
                           5: 'Group E',
                           6: "All"
                       },
                       value=6
                    ),
            html.Br(),
            dcc.Graph(id='ethnicity_dist_graph',
                    figure=make_empty_fig()),
            html.Br(),
            dcc.Graph(id='lunch_dist_graph',
                    figure=make_empty_fig()),
            html.Br(),
            dcc.Graph(id='math_dist_graph',
                    figure=make_empty_fig()),
            html.Br(),
            dcc.Graph(id='write_dist_graph',
                    figure=make_empty_fig()),
            ], md=12, lg=5),

    ]),

], style={'backgroundColor': '#E5ECF6'})
```

```
score_dashboard = html.Div([
    dbc.Row([
        dbc.Col(lg=1),
        dbc.Col([
            dbc.Label('Filter Education of Parents'),
            dcc.Dropdown(id='edu_selector2',
                         multi=True,
                         placeholder='Select one or more',
                         options=[{'label': edu, 'value': edu}
                                  for edu in exams['parental_level_of_education'].drop_duplicates().sort_values()]),
            html.Br(),
            ], md=12, lg=5),
        dbc.Col([
            dbc.Label('Filter Ethnicity'),
            dcc.Slider(1, 6, step=None, id='ethnicity_slider2',
                       marks={
                           1: 'Group A',
                           2: 'Group B',
                           3: 'Group C',
                           4: 'Group D',
                           5: 'Group E',
                           6: "All"
                       },
                       value=6
                      ),
            ], md=12, lg=5),
        ]),
    dbc.Row([
        dbc.Col(lg=1),
        dbc.Col([
            dcc.Graph(id='gender_score_graph',
                      figure=make_empty_fig()),
            html.Br(),
            dcc.Graph(id='ethnicity_score_graph',
                      figure=make_empty_fig()),
            html.Br(),
            dcc.Graph(id='parental_score_graph',
                      figure=make_empty_fig()),
            html.Br(),
            dcc.Graph(id='lunch_score_graph',
                      figure=make_empty_fig()),
            dcc.Graph(id='test_score_graph',
```

```
                        figure=make_empty_fig()),
                ], md=12, lg=10),
        ]),

], style={'backgroundColor': '#E5ECF6'})

app.validation_layout = html.Div([
    main_layout,
    main_dashboard,
    dist_dashboard,
    score_dashboard,
])


app.layout = main_layout

def filter_data(edu_levels, ethnicity, filtered):
    group = ""
    if edu_levels:
        filtered = filtered[filtered['parental_level_of_education'].isin(edu_levels)]
    if ethnicity == 1:
        group = 'group A'
        filtered = filtered[filtered['ethnicity'] == group]
    elif ethnicity == 2:
        group = "group B"
        filtered = filtered[filtered['ethnicity'] == group]
    elif ethnicity == 3:
        group = "group C"
        filtered = filtered[filtered['ethnicity'] == group]
    elif ethnicity == 4:
        group = "group D"
        filtered = filtered[filtered['ethnicity'] == group]
    elif ethnicity == 5:
        group = "group E"
        filtered = filtered[filtered['ethnicity'] == group]
    else:
        group = ""
    return filtered

#this method updates the layout to order and main
@app.callback(Output('main_content', 'children'),
              Input('location', 'pathname'))
```

```
def display_content(pathname):
    if unquote(pathname[1:]) in ['dist']:
        return dist_dashboard
    elif unquote(pathname[1:]) in ['score']:
        return score_dashboard
    else:
        return main_dashboard

#This method plots the main figures with input from user
@app.callback(Output('comparison_graph', 'figure'),
              Output('heatmap_graph', 'figure'),
              Input('edu_selector', 'value'),
              Input('ethnicity_slider', 'value'),
              Input('diff_dropdown', 'value'),
              Input('ques_dropdown', 'value'),
             Input('comp_dropdown', 'value'))
def display_main(edu_levels, ethnicity, diff, ques, comp):
    filtered = exams.copy()
    filtered = filter_data(edu_levels, ethnicity, filtered)
    fig1 = px.scatter(filtered, x=ques, y=comp, color=diff, hover_data=[diff], trendline="ols")
    cols = ['math_score', 'reading_score', 'writing_score', 'overall']
    df_corr = filtered[cols].corr().round(2)
    fig2 = go.Figure()
    fig2.add_trace(go.Heatmap(x = df_corr.columns, y = df_corr.index, z = np.array(df_corr)))
    return fig1, fig2

#This method plots the main figures with input from user
@app.callback(Output('gender_dist_graph', 'figure'),
              Output('ethnicity_dist_graph', 'figure'),
              Output('parental_dist_graph', 'figure'),
              Output('lunch_dist_graph', 'figure'),
              Output('test_dist_graph', 'figure'),
              Output('math_dist_graph', 'figure'),
              Output('read_dist_graph', 'figure'),
              Output('write_dist_graph', 'figure'),
              Input('edu_selector1', 'value'),
              Input('ethnicity_slider1', 'value'),
             )
def display_dist(edu_levels, ethnicity):
    filtered = exams.copy()
    filtered = filter_data(edu_levels, ethnicity, filtered)
    fig1 = px.histogram(filtered, x="gender", color="gender", labels={'gender':'Gender'}, title='Gender Distribution')
```

```
    fig2 = px.histogram(filtered, x="ethnicity", color="ethnicity", labels={'ethniciy':'Ethniciy/Race'}, title='Ethnic
    fig3 = px.histogram(filtered, x="parental_level_of_education", color="parental_level_of_education",
                        labels={'parental_level_of_education':'Parental Level of Education'},
                        title='Parental Level of Education Distribution (Filter)')
    fig4 = px.histogram(filtered, color="lunch", x="lunch", labels={'lunch':'Lunch Program'}, title='Lunch Program Dis
    fig5 = px.histogram(filtered, x="test_preparation_course", color="test_preparation_course",
                        labels={'test_preparation_course':'Test Preparation Course'},
                        title='Test Preparation Distribution')
    fig6 = px.histogram(filtered, x="math_score", labels={'math_score':'Math Score'}, title='Math Score Distribution')
    fig7 = px.histogram(filtered, x="writing_score", labels={'writing_score':'Writing Score'}, title='Writing Score Di
    fig8 = px.histogram(filtered, x="reading_score", labels={'reading_score':'Reading Score'}, title='Reading Score Di

    return fig1, fig2, fig3, fig4, fig5, fig6, fig7, fig8


#This method plots the main figures with input from user
@app.callback(Output('gender_score_graph', 'figure'),
              Output('ethnicity_score_graph', 'figure'),
              Output('parental_score_graph', 'figure'),
              Output('lunch_score_graph', 'figure'),
              Output('test_score_graph', 'figure'),
              Input('edu_selector2', 'value'),
              Input('ethnicity_slider2', 'value'),
             )
def display_scores_box(edu_levels, ethnicity):
    filtered = exams.copy()
    filtered = filter_data(edu_levels, ethnicity, filtered)
    fig1 = go.Figure()
    x =exams['gender']
    fig1.add_trace(go.Box(y=filtered['math_score'], x=x, name='Math Score', marker_color='black'))
    fig1.add_trace(go.Box(y=filtered['writing_score'], x=x, name='Writing Score', marker_color='blue'))
    fig1.add_trace(go.Box(y=filtered['reading_score'], x=x, name='Reading Score', marker_color='orange'))
    fig1.add_trace(go.Box(y=filtered['overall'], x=x, name='Overall Score', marker_color='green'))
    fig1.update_layout(title="Gender wise Scores", boxmode='group')

    fig2 = go.Figure()
    x=exams['ethnicity']
    fig2.add_trace(go.Box(y=filtered['math_score'], x=x, name='Math Score', marker_color='black'))
    fig2.add_trace(go.Box(y=filtered['writing_score'], x=x, name='Writing Score', marker_color='blue'))
    fig2.add_trace(go.Box(y=filtered['reading_score'], x=x, name='Reading Score', marker_color='orange'))
    fig2.add_trace(go.Box(y=filtered['overall'], x=x, name='Overall Score', marker_color='green'))
    fig2.update_layout(title="Ethnicity/Race wise Scores", boxmode='group')
```

```
    fig3 = go.Figure()
    x=exams['parental_level_of_education']
    fig3.add_trace(go.Box(y=filtered['math_score'], x=x, name='Math Score', marker_color='black'))
    fig3.add_trace(go.Box(y=filtered['writing_score'], x=x, name='Writing Score', marker_color='blue'))
    fig3.add_trace(go.Box(y=filtered['reading_score'], x=x, name='Reading Score', marker_color='orange'))
    fig3.add_trace(go.Box(y=filtered['overall'], x=x, name='Overall Score', marker_color='green'))
    fig3.update_layout(title="Parental education wise Scores", boxmode='group')

    fig4 = go.Figure()
    x=exams['lunch']
    fig4.add_trace(go.Box(y=filtered['math_score'], x=x, name='Math Score', marker_color='black'))
    fig4.add_trace(go.Box(y=filtered['writing_score'], x=x, name='Writing Score', marker_color='blue'))
    fig4.add_trace(go.Box(y=filtered['reading_score'], x=x, name='Reading Score', marker_color='orange'))
    fig4.add_trace(go.Box(y=filtered['overall'], x=x, name='Overall Score', marker_color='green'))
    fig4.update_layout(title="Lunch program wise Scores", boxmode='group')

    fig5 = go.Figure()
    x=exams['test_preparation_course']
    fig5.add_trace(go.Box(y=filtered['math_score'], x=x, name='Math Score', marker_color='black'))
    fig5.add_trace(go.Box(y=filtered['writing_score'], x=x, name='Writing Score', marker_color='blue'))
    fig5.add_trace(go.Box(y=filtered['reading_score'], x=x, name='Reading Score', marker_color='orange'))
    fig5.add_trace(go.Box(y=filtered['overall'], x=x, name='Overall Score', marker_color='green'))
    fig5.update_layout(title="Test preparation wise Scores", boxmode='group')

    return fig1, fig2, fig3, fig4, fig5

#Here you can give the port number to run the app on the desired port
app.run_server(port=8051)
```

```
Dash app running on http://127.0.0.1:8051/ (http://127.0.0.1:8051/)
```

In [ ]:

In [ ]: