## Aim: To perform blending among multiple tables using tableau

**IDE: Tableau**

## Theory:

It is often necessary to combine data from multiple places—different tables or even data sources—to perform a desired analysis. Depending on the structure of the data and the needs of the analysis, there are several ways to combine the tables.

# Relationships vs Joins

The default method in Tableau Desktop is to use relationships. Relationships preserve the original tables’ level of detail when combining information. Relationships also allow for context-based joins to be performed on a sheet-by-sheet basis, making each data source more flexible. Relationships are the recommended method of combining data in most instances. For more information, see [How Relationships Differ from Joins](https://help.tableau.com/current/pro/desktop/en-us/datasource_relationships_learnmorepage.htm).

However, there may be times when you want to directly establish a join, either for control or for desired aspects of a join compared to a relationship, such as deliberate filtering or duplication.

# Common issues

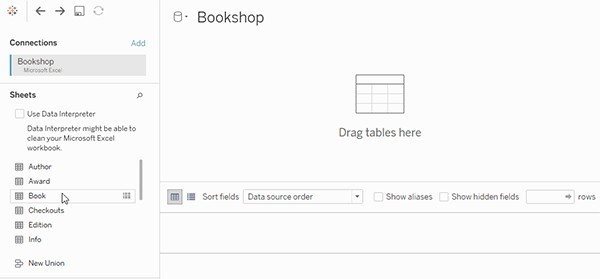
* To view, edit, or create joins, you must open a logical table in the relationship canvas—the area you see when you first open or create a data source—and access the join canvas.
* **Published Tableau data sources cannot be used in joins.** To combine published data sources, you must edit the original data sources to natively contain the join or use a data blend.
* When joining tables, the fields that you join on must be the same data type. If you change the data type after you join the tables, the join will break.
* Fields used in the join clause cannot be removed without breaking the join. To join data and be able to clean up duplicate fields, use Tableau Prep Builder instead of Desktop

# Create a join

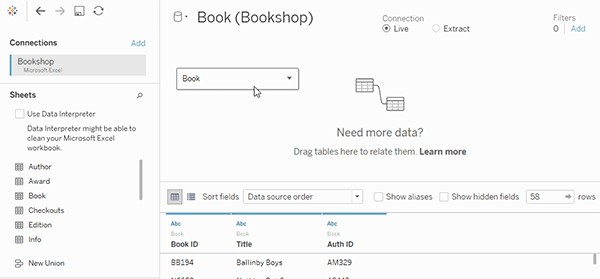
1. To create a join, connect to the relevant data source or sources. See [Connect to Your Data](https://help.tableau.com/current/pro/desktop/en-us/basicconnectoverview.htm).

These can be in the same data source (such as tables in a database or sheets in an Excel spreadsheet) or different data sources (this is known as a cross-database join). If you combined tables using a cross- database join, Tableau colors the tables in the canvas and the columns in the data grid to show you which connection the data comes from.

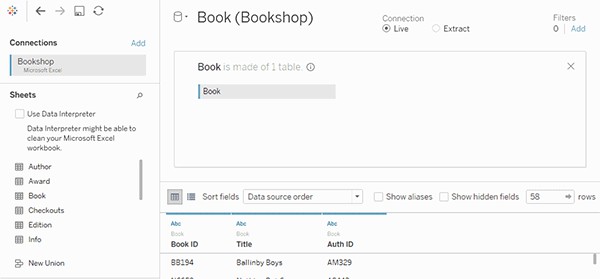
1. Drag the first table to the canvas.



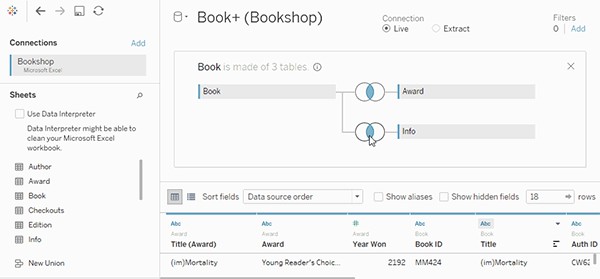
1. Select Open from the menu or double-click the first table to open the join canvas (physical layer).



1. Double-click or drag another table to the join canvas.



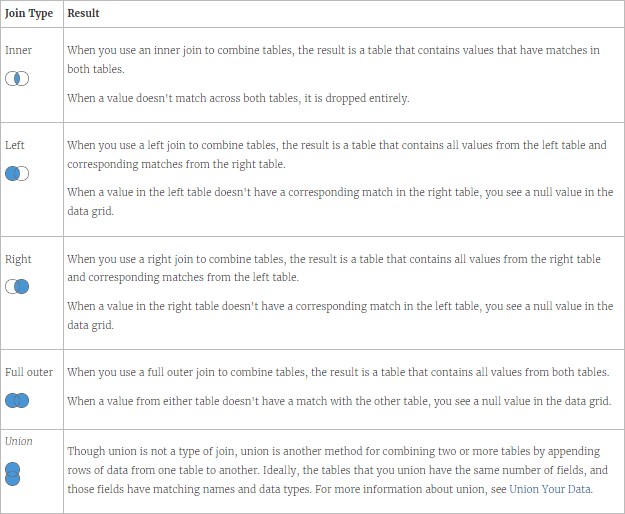
1. Click the join icon to configure the join. Add one or more join clauses by selecting a field from one of the available tables used in the data source, choosing a join operator, and a field from the added table.



1. When finished, close the join dialog and join canvas.

# Join types

In general, there are four types of joins that you can use in Tableau: inner, left, right, and full outer. If you aren't sure what join type you want to use to combine data from multiple tables, you should use relationships.



# Options to combine data

There are several ways to combine data, each with their own strengths and weaknesses.

Relationships are the default method and can be used in most instances, including across tables with different levels of detail. Relationships are flexible and adapt to the structure of the analysis on a sheet by sheet basis. However, you can't create relationships between tables from published data sources.

Joins combine tables by adding more columns of data across similar row structures. This can cause data loss or duplication if tables are at different levels of detail, and joins must be established before analysis can begin. You can't use a published data source in a join.

Blends, unlike relationships or joins, never combine the data directly. Instead, blends query each data source independently, aggregate the results to the appropriate level, then present the results together visually in the view. Because of this, blends can handle different levels of detail and also work with published data sources. Blends don't create a new, blended data source (and therefore can't be published as a "blended data source"). Instead, they are simply blended results visualized per sheet.

## Pre Lab Exercise:

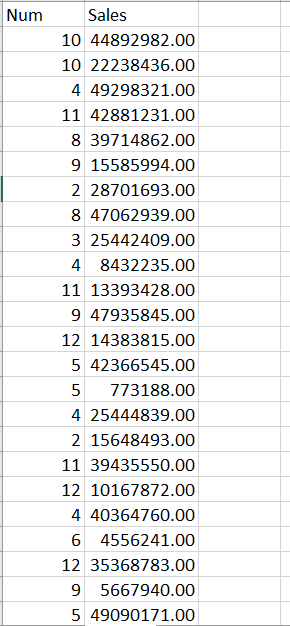
1. What is a join in Tableau?
2. How do you perform a join in Tableau?
3. What is the difference between data blending and traditional joins in Tableau?

## Tasks:

Perform the following tasks:

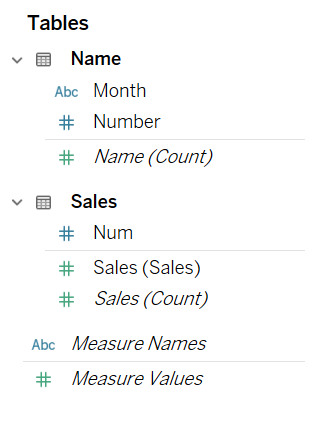
Task 1: Create two sheets with the same feature set

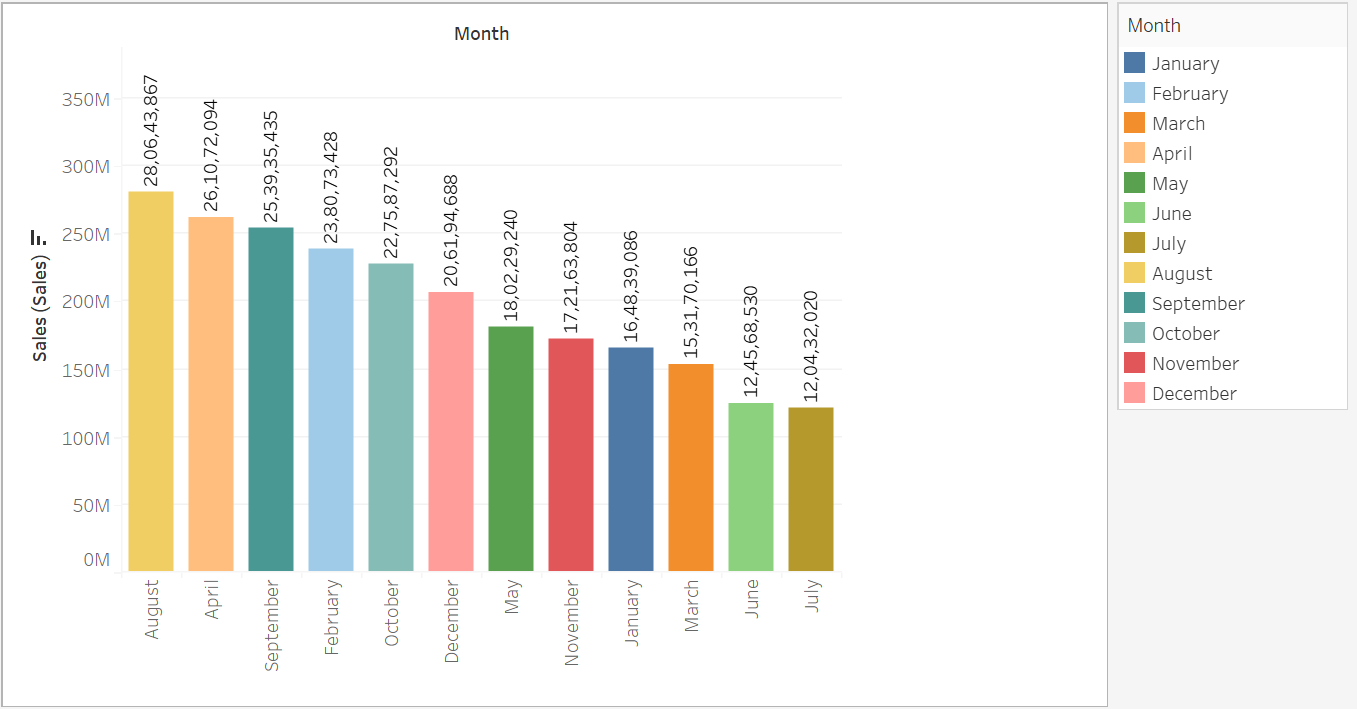
## Results:-



Task 2: Check in tableau to analyse the cross-items in-between the tables.

## Results:-

Task 3: Join the tables

Task 4: Perform the analysis in-between the two separate tables

## Observation and Result Analysis:

Write the final observation and process corresponding to each task

1. **Before joining two tables**
2. **After joining two tables**

## Post Lab Exercise:

**Exercise-1:**

Python Implementation of joining the two tables and performing the same analysis you did in task 4.

**Code:-**

import pandas as pd

import plotly.express as px

Dataset\_Sales\_and\_Numbers = pd.read\_excel("./Dataset.xlsx", "Sales")

Dataset\_Nubers\_and\_Months = pd.read\_excel("./Dataset.xlsx", "Name")

Merged\_Dataset = pd.merge(Dataset\_Sales\_and\_Numbers, Dataset\_Sales\_and\_Numbers, on="Number")

Final\_Dataset = Merged\_Dataset[["Month", "Sales"]]

Final\_Dataset = Final\_Dataset.groupby("Month").sum().reset\_index()

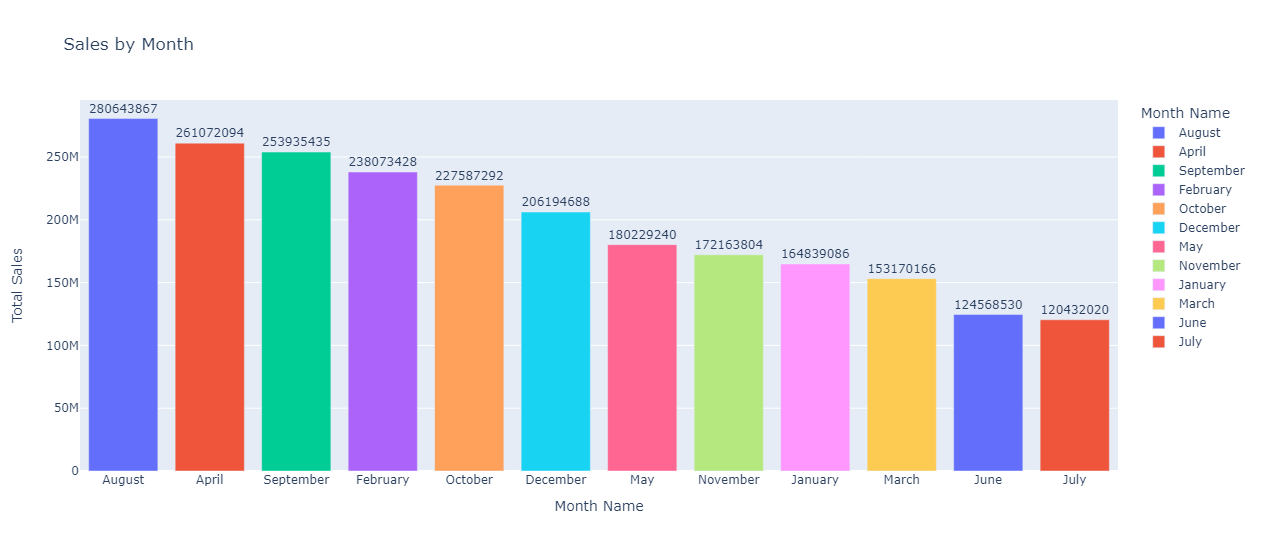
Final\_Dataset = Final\_Dataset.sort\_values(by="Sales", ascending=False)

Final\_Dataset["Sales\_Label"] = Final\_Dataset["Sales"].astype(str)

Figure = px.bar(Final\_Dataset,x="Month", y="Sales",color="Month", title="Sales by Month",labels={"Sales": "Total Sales", "Month": "Month Name"},text="Sales\_Label",)

Figure.update\_traces(textposition="outside")

Figure.write\_html("sales\_by\_month\_plot.html")

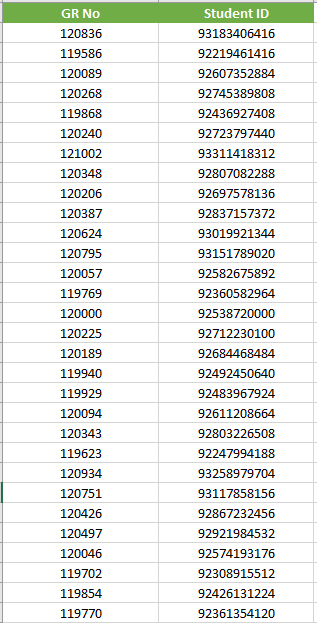
**Result :-**

## Exercise-2:

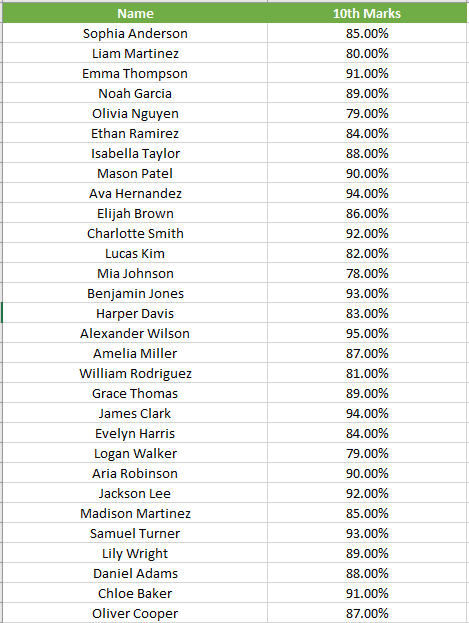
Make the 4 spreadsheets/tables with 30 dummy data Table 1: Gr. No | Student ID

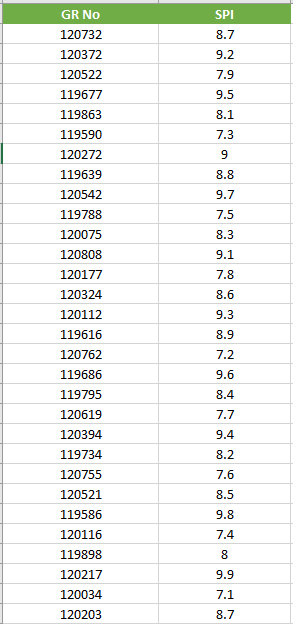
Table 2: Student ID | Name Table 3: Gr. No | SPI

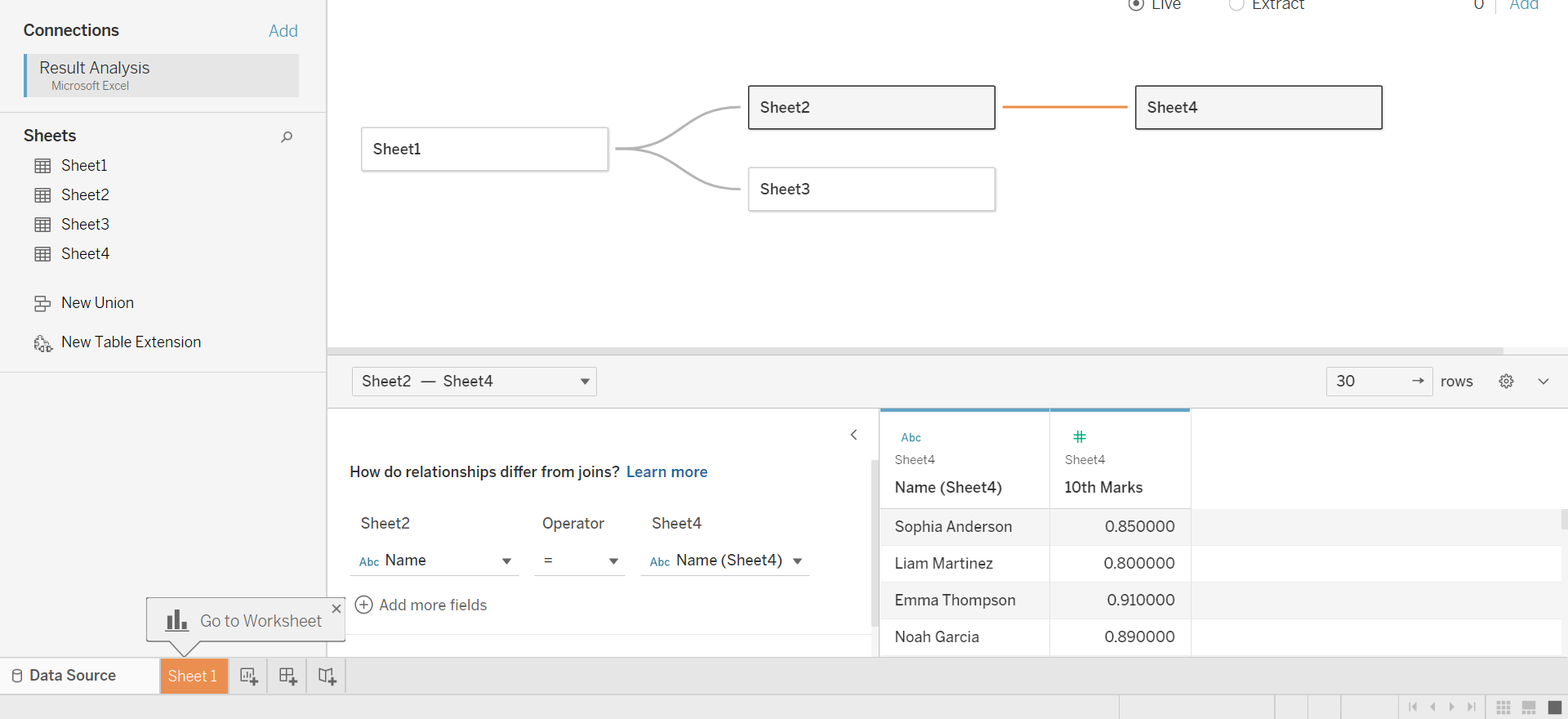
Table 4: Name | 10th marks

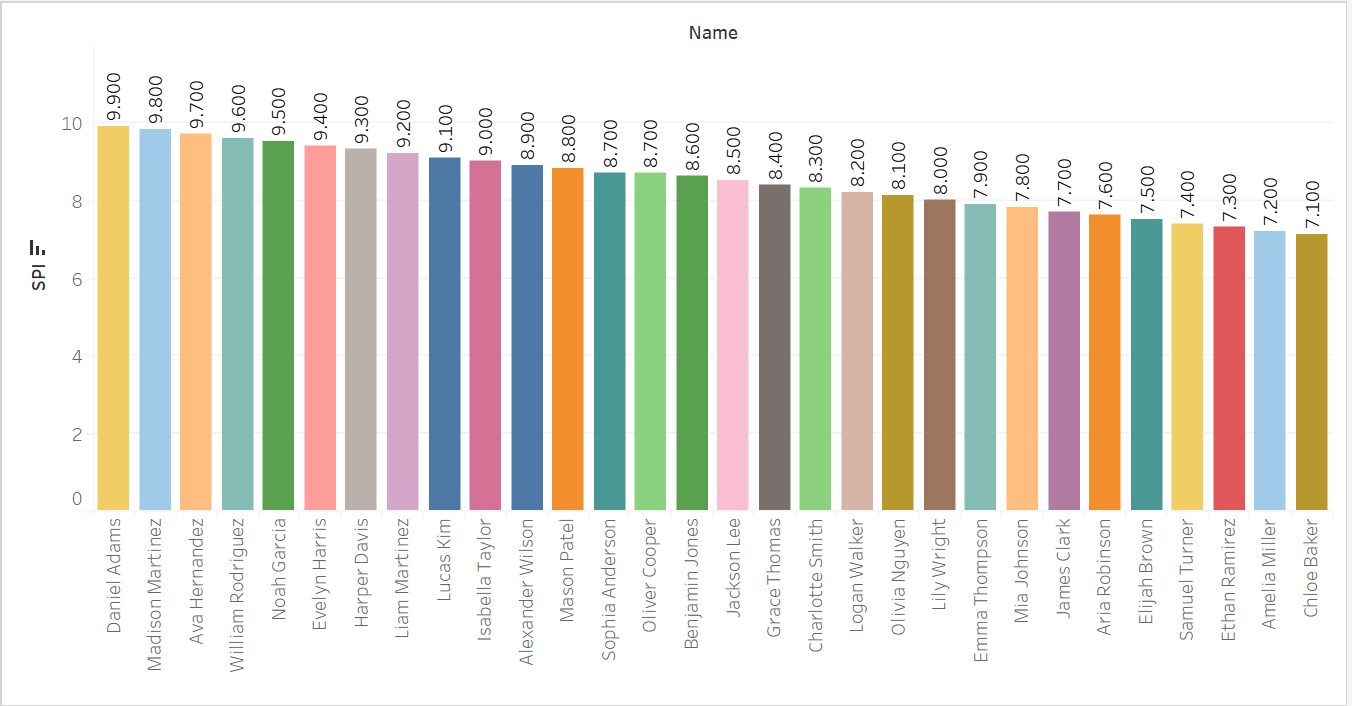


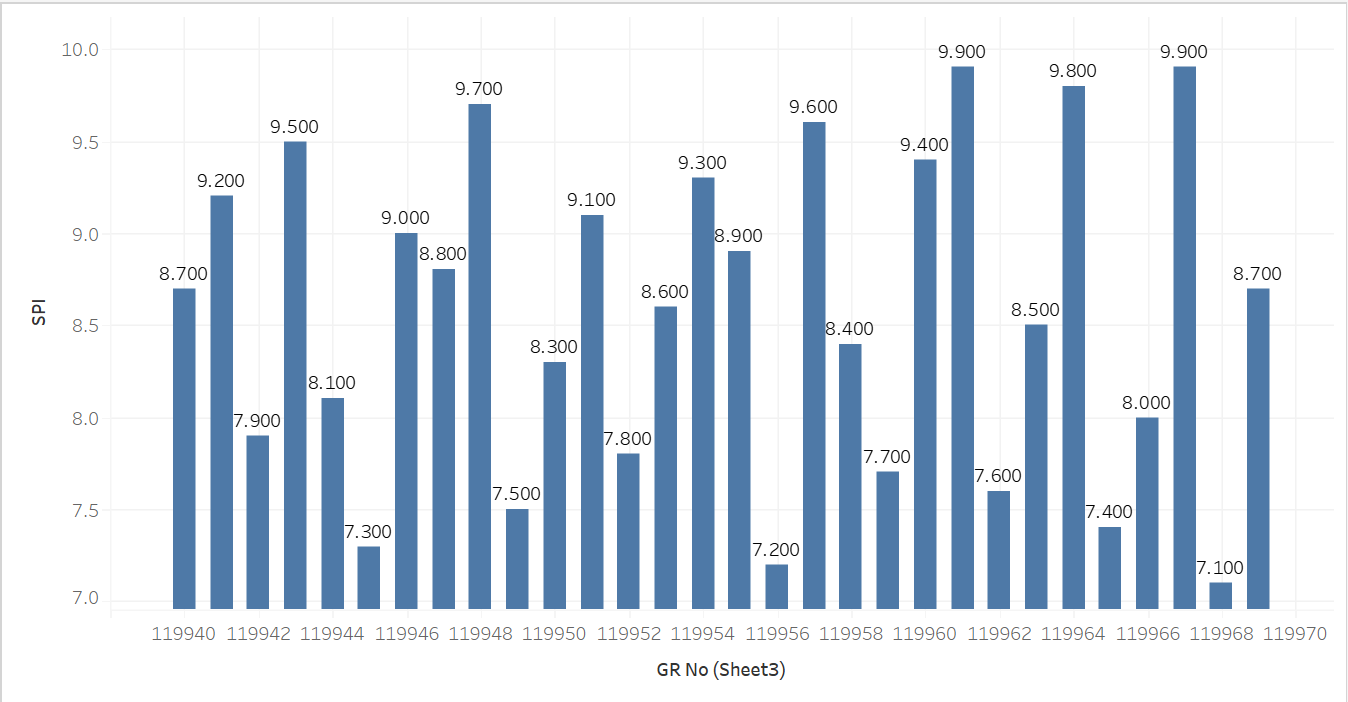






**Implementation Using Tableau :-**

Perform the analysis of (Name vs SPI) and arrange in descending order

Perform the analysis of (Gr. No. vs SPI) and arrange in descending order

**Implementation Using Python:-**

**Pre – Requisitesies :-**

import pandas as pd

import plotly.express as px

GRNo\_Student\_ID = pd.read\_excel('./Result Analysis.xlsx','Sheet1')

Student\_ID\_Name = pd.read\_excel('./Result Analysis.xlsx','Sheet2')

GRNo\_SPI = pd.read\_excel('./Result Analysis.xlsx','Sheet3')

Name\_10th\_Marks = pd.read\_excel('./Result Analysis.xlsx','Sheet4')

Task – 1 :-Perform the analysis of (Name vs SPI) and arrange it in descending order

**Code :-**

Joint\_Sheet2\_and\_Sheet2 = pd.merge(GRNo\_Student\_ID, Student\_ID\_Name,on='Student ID')

Joint\_GRNO\_Name\_SPI = pd.merge(Joint\_Sheet2\_and\_Sheet2,GRNo\_SPI,on='GR No')

Task\_1\_Datatset = Joint\_GRNO\_Name\_SPI[['Name', 'SPI']]

Task\_1\_Datatset = Task\_1\_Datatset.sort\_values(by='SPI' , ascending = False)

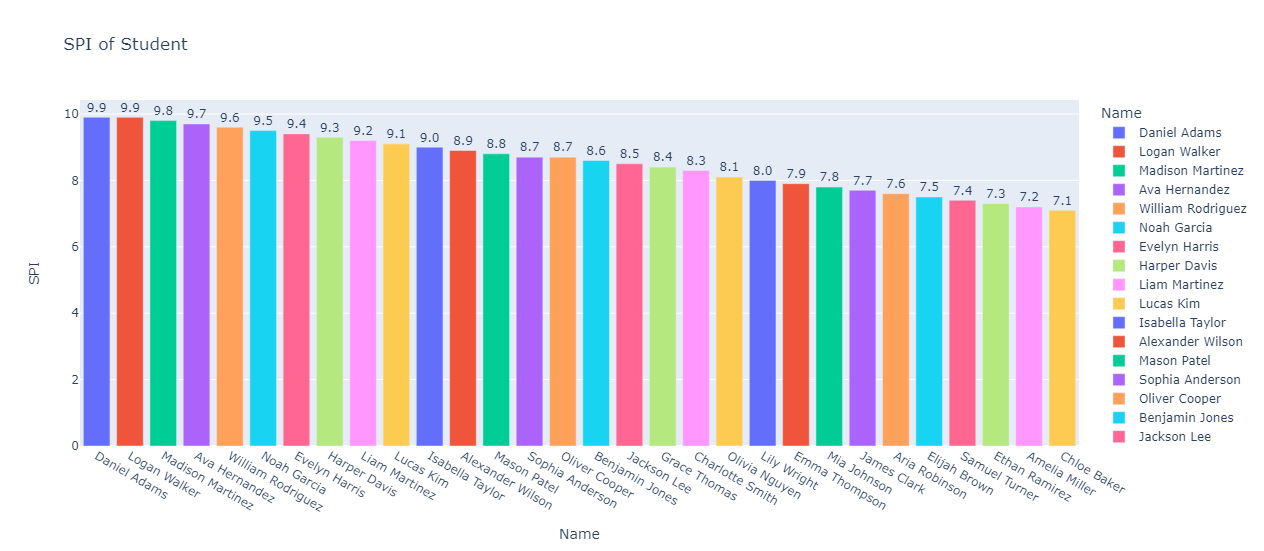
Task\_1\_Datatset['Text'] = Task\_1\_Datatset['SPI'].astype(str)

Task\_1\_Plot = px.bar(data\_frame=Task\_1\_Datatset,x='Name' , y='SPI' , color = 'Name' , title='SPI of Student' , text='Text' , labels={'SPI' : 'SPI' , 'Name' : 'Name'})

Task\_1\_Plot.update\_traces(textposition="outside")

Task\_1\_Plot.write\_html('Name vs SPI.html')

**Output :-**



Task – 2 :- Perform the analysis of (Gr. No. vs SPI) and arrange it in descending order

**Code :-**

Task\_2\_Dataset = GRNo\_SPI.sort\_values(by='SPI' , ascending = False)

Task\_2\_Dataset["Text"] = Task\_2\_Dataset['SPI'].astype(str)

Task\_2\_Plot = px.bar(data\_frame=Task\_2\_Dataset,x="GR No",y="SPI",color="GR No",title="SPI of Student",text="Text",labels={"SPI": "SPI", "GR No": "GR No"},)

Task\_2\_Plot.update\_traces(textposition="outside")

Task\_2\_Plot.write\_html("GR No vs SPI.html")

**Output :-**

