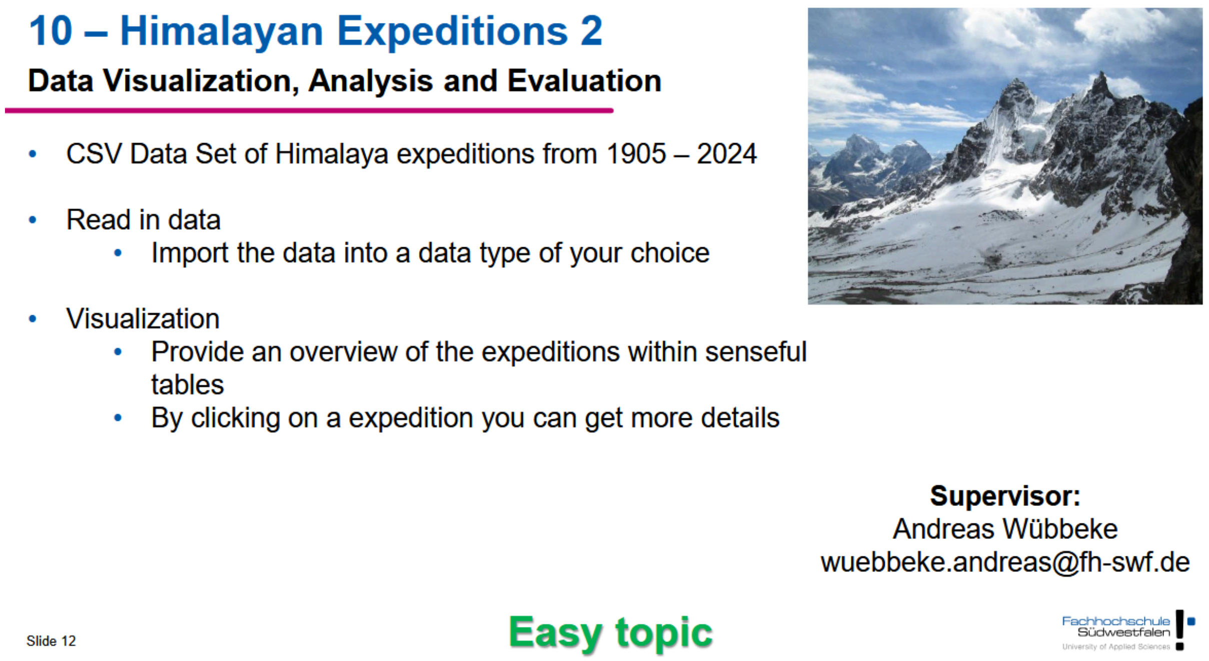
Prelude:

Group 10: Himalayan Expeditions 2



10 – Himalayan Expeditions 2

Data Visualization, Analysis and Evaluation

- CSV Data Set of Himalaya expeditions from 1905 – 2024
- Read in data
  - Import the data into a data type of your choice
- Visualization
  - Provide an overview of the expeditions within senseful tables
  - By clicking on a expedition you can get more details

Supervisor:
Andreas Wübbeke
wuebbeke.andreas@fh-swf.de

Slide 12

Easy topic

Fachhochschule Südwestfalen
University of Applied Sciences

Members:

- Kashira Titan Naratama
- Hansel Zorya

We were given 4 sets of Data:

- Exped.csv
- Members.csv
- Peaks.csv
- Refer.csv

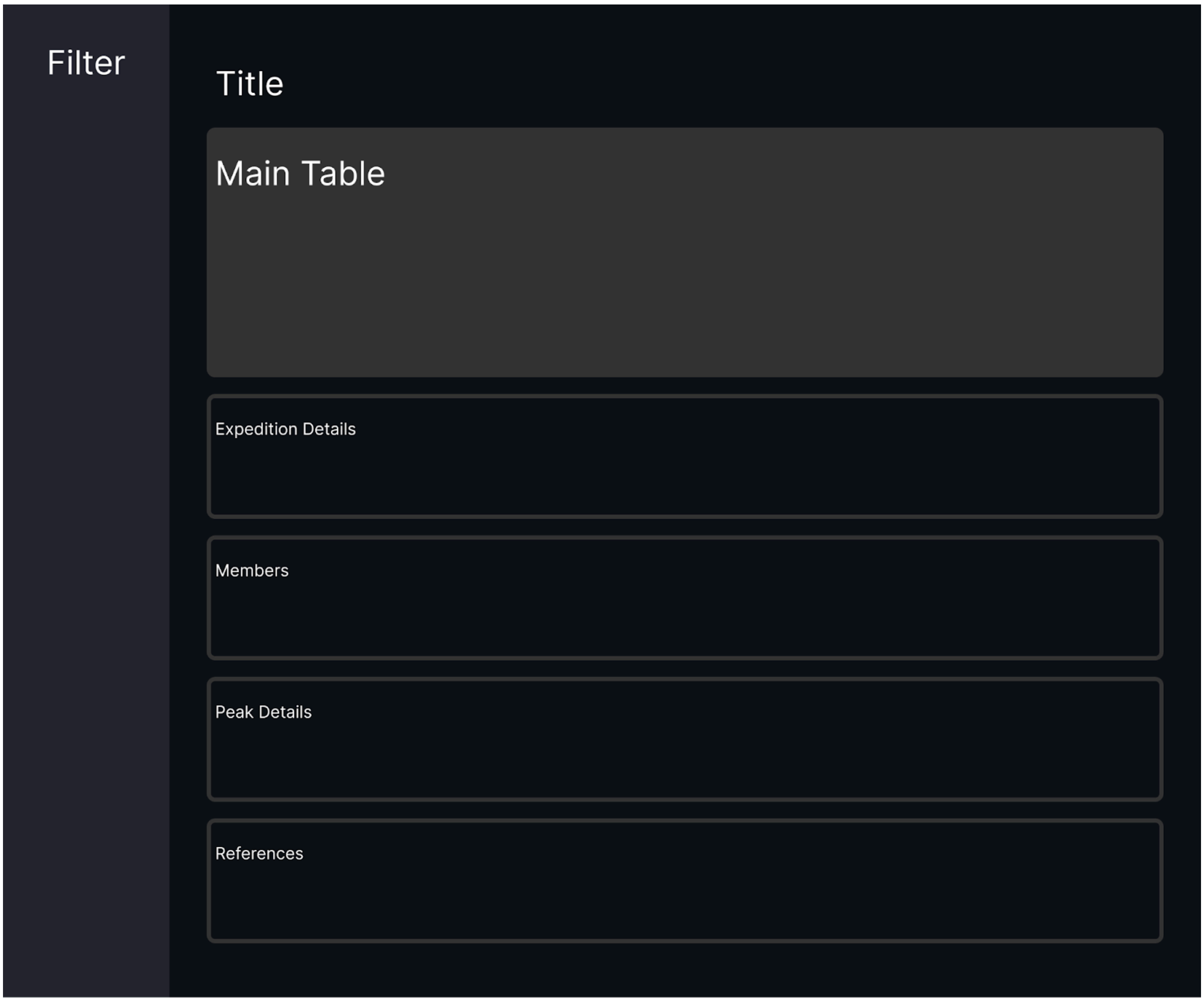
The objective of this assignment is to provide an overview of the data that was taken. The overview should be clickable and readable.

Challenge:
The 4 sets of data contain LOTS of data. Some of which are unnecessary and may need to be excluded. As such, we have decided to only use necessary data:

- Exped.csv:
  - Expid
  - Year
  - Leaders
  - Nation
  - Host
  - Sponsor
  - Highpoint
  - Hdeaths

- Members.csv
    - Fname
    - Lname
    - Status (role)
    - Death
    - Deathtype
- Peak.csv
    - Peakid
    - Pkname
    - Pkname2
    - Location
    - Heightm
- Refer
    - Refid
    - Ryear
    - Rauthor
    - Rtitle
    - Rpublisher

UI Concept



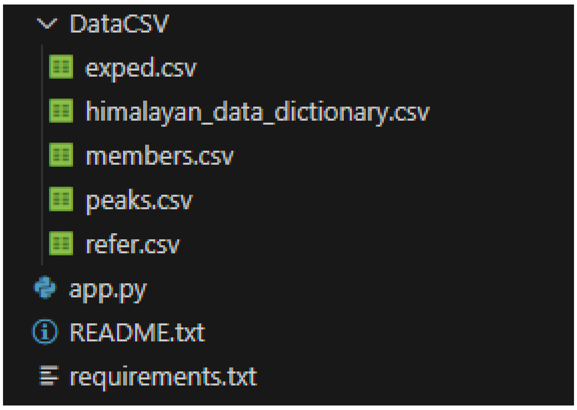
This is the concept of the UI of the visualization. It's a guideline following the capabilities that Streamlit can give.

The Filter is placed within a sidebar, so the user can easily filter out the data they want.
In the Main, there will be a Main Table. The rows inside should be able to be clicked. When the table is clicked, other tables should appear.

The Expedition Details, Members, Peak Details, and references should only pop up when a table is clicked within the Main Table.

Building the App

Files Layout



```
DataCSV
    exped.csv
    himalayan_data_dictionary.csv
    members.csv
    peaks.csv
    refer.csv
app.py
README.txt
requirements.txt
```

The DataCSVs will be placed in a folder for organized storing.

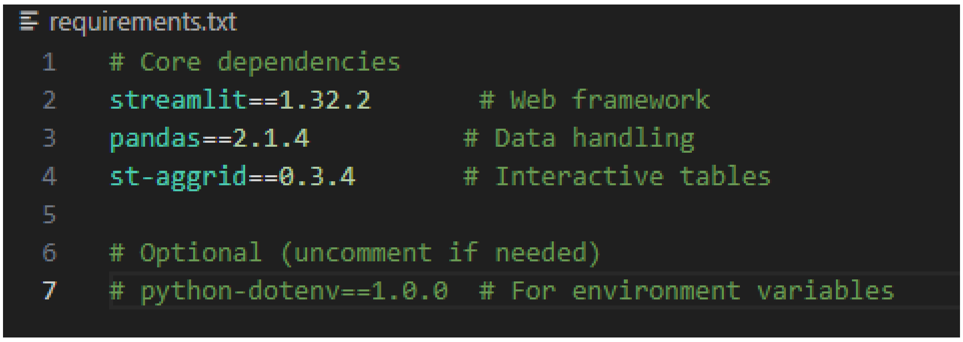
The app.py will be the main file to run the visualization.

A README.txt is placed for instructions on how to start the files.

Requirements.txt contains the required pip installs to use.

Adding The Imports

Pip Installs:

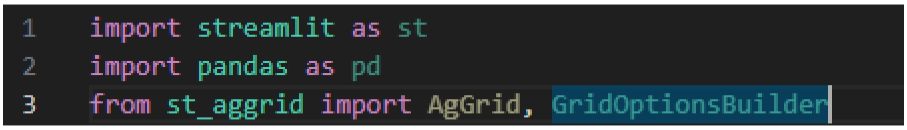


requirements.txt

```
# Core dependencies
streamlit==1.32.2       # Web framework
pandas==2.1.4          # Data handling
st-aggrid==0.3.4       # Interactive tables

# Optional (uncomment if needed)
# python-dotenv==1.0.0  # For environment variables
```

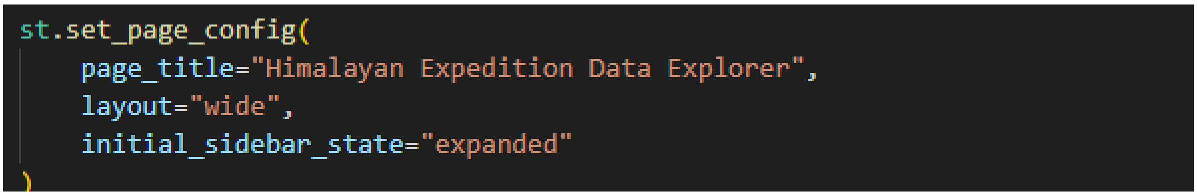
Imports:



```
import streamlit as st
import pandas as pd
from st_aggrid import AgGrid, GridOptionsBuilder
```

- Pandas handles the DataFrames
- Streamlit creates a webapp interface
- St_aggrid provides interactive tables and filtering

Setting the Page Configuration and Main App



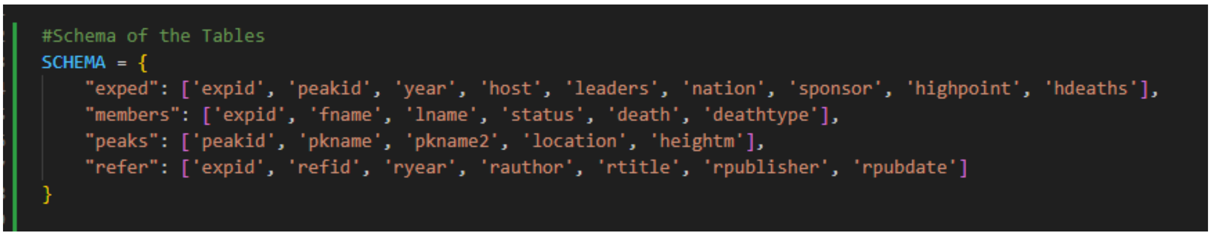
```
st.set_page_config(
    page_title="Himalayan Expedition Data Explorer",
    layout="wide",
    initial_sidebar_state="expanded"
)
```

The page will be named Himalayan Expedition Data Explorer.

The “wide” layout is a layout given by Streamlit. It expands the content to the full screen.

The sidebar, which will be used for the filter, will be “expanded” in its initial/default state.

Schema for The Tables



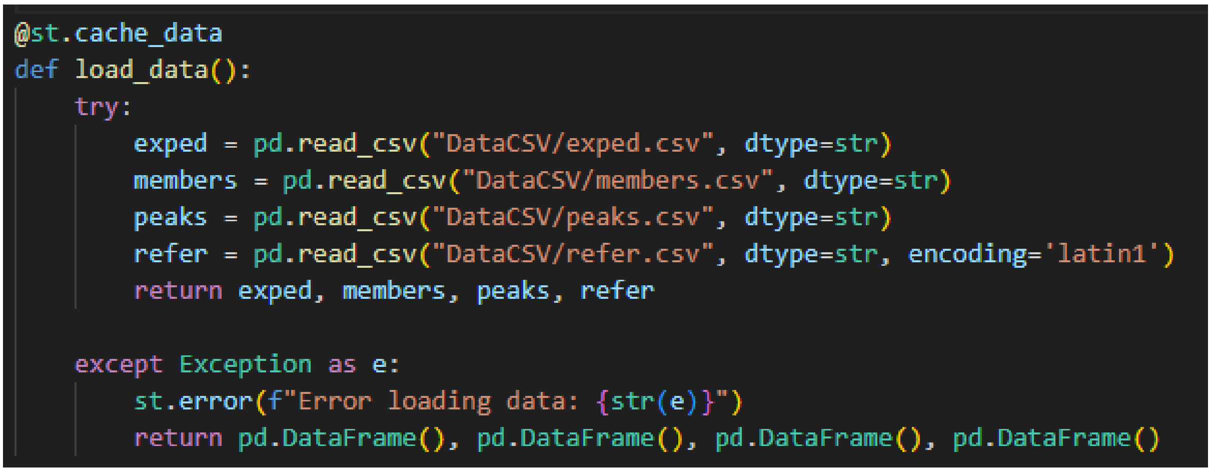
```
#Schema of the Tables
SCHEMA = {
    "exped": ['expid', 'peakid', 'year', 'host', 'leaders', 'nation', 'sponsor', 'highpoint', 'hdeaths'],
    "members": ['expid', 'fname', 'lname', 'status', 'death', 'deathtype'],
    "peaks": ['peakid', 'pkname', 'pkname2', 'location', 'heightm'],
    "refer": ['expid', 'refid', 'ryear', 'rauthor', 'rtitle', 'rpublisher', 'rpubdate']
}
```

The Schema defines the structure/data needed/given to the tables.

It ensures consistency within the data, and scalable in a way that it’s easy to add or remove fields.

Loading The Data

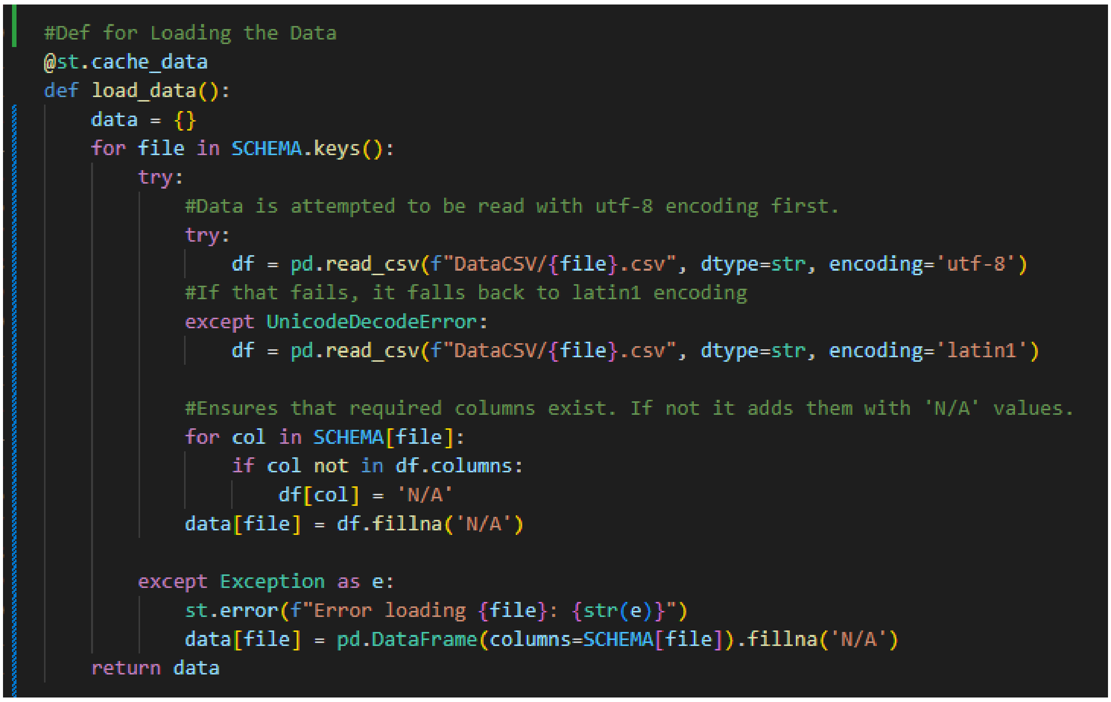
1st Attempt



```python
@st.cache_data
def load_data():
    try:
        exped = pd.read_csv("DataCSV/exped.csv", dtype=str)
        members = pd.read_csv("DataCSV/members.csv", dtype=str)
        peaks = pd.read_csv("DataCSV/peaks.csv", dtype=str)
        refer = pd.read_csv("DataCSV/refer.csv", dtype=str, encoding='latin1')
        return exped, members, peaks, refer

    except Exception as e:
        st.error(f"Error loading data: {str(e)}")
        return pd.DataFrame(), pd.DataFrame(), pd.DataFrame(), pd.DataFrame()
```

2nd Attempt



```python
#Def for Loading the Data
@st.cache_data
def load_data():
    data = {}
    for file in SCHEMA.keys():
        try:
            #Data is attempted to be read with utf-8 encoding first.
            try:
                df = pd.read_csv(f"DataCSV/{file}.csv", dtype=str, encoding='utf-8')
            #If that fails, it falls back to latin1 encoding
            except UnicodeDecodeError:
                df = pd.read_csv(f"DataCSV/{file}.csv", dtype=str, encoding='latin1')

            #Ensures that required columns exist. If not it adds them with 'N/A' values.
            for col in SCHEMA[file]:
                if col not in df.columns:
                    df[col] = 'N/A'
            data[file] = df.fillna('N/A')

        except Exception as e:
            st.error(f"Error loading {file}: {str(e)}")
            data[file] = pd.DataFrame(columns=SCHEMA[file]).fillna('N/A')
    return data
```

@st.cache_data caches the data to avoid reloading on every app interaction

In reading the file, the Data is attempted to be read with utf-8 encoding first. If it fails, it falls back to the latin1 encoding.

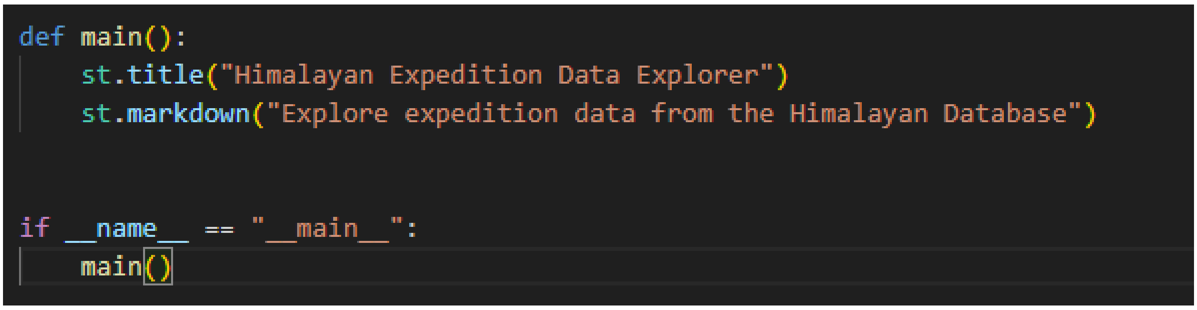
Each column in SCHEMA is checked to see if it exists. If not, it will be filled as ‘N/A’

At the end, each file that returns a NaN will be instead replaced with ‘N/A’.

Building The Main

Title and Data Loading

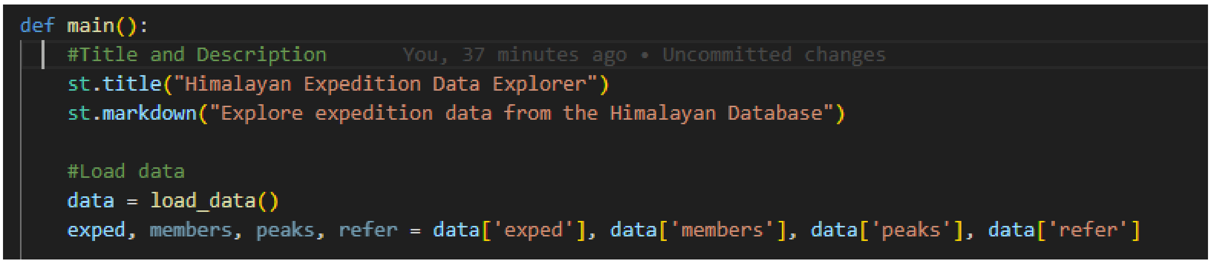
1st Attempt



```python
def main():
    st.title("Himalayan Expedition Data Explorer")
    st.markdown("Explore expedition data from the Himalayan Database")


if __name__ == "__main__":
    main()
```

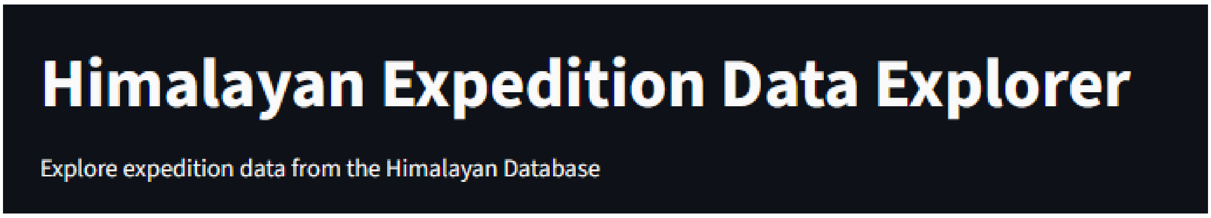
2nd Attempt



```python
def main():
    #Title and Description      You, 37 minutes ago • Uncommitted changes
    st.title("Himalayan Expedition Data Explorer")
    st.markdown("Explore expedition data from the Himalayan Database")

    #Load data
    data = load_data()
    exped, members, peaks, refer = data['exped'], data['members'], data['peaks'], data['refer']
```

Result:



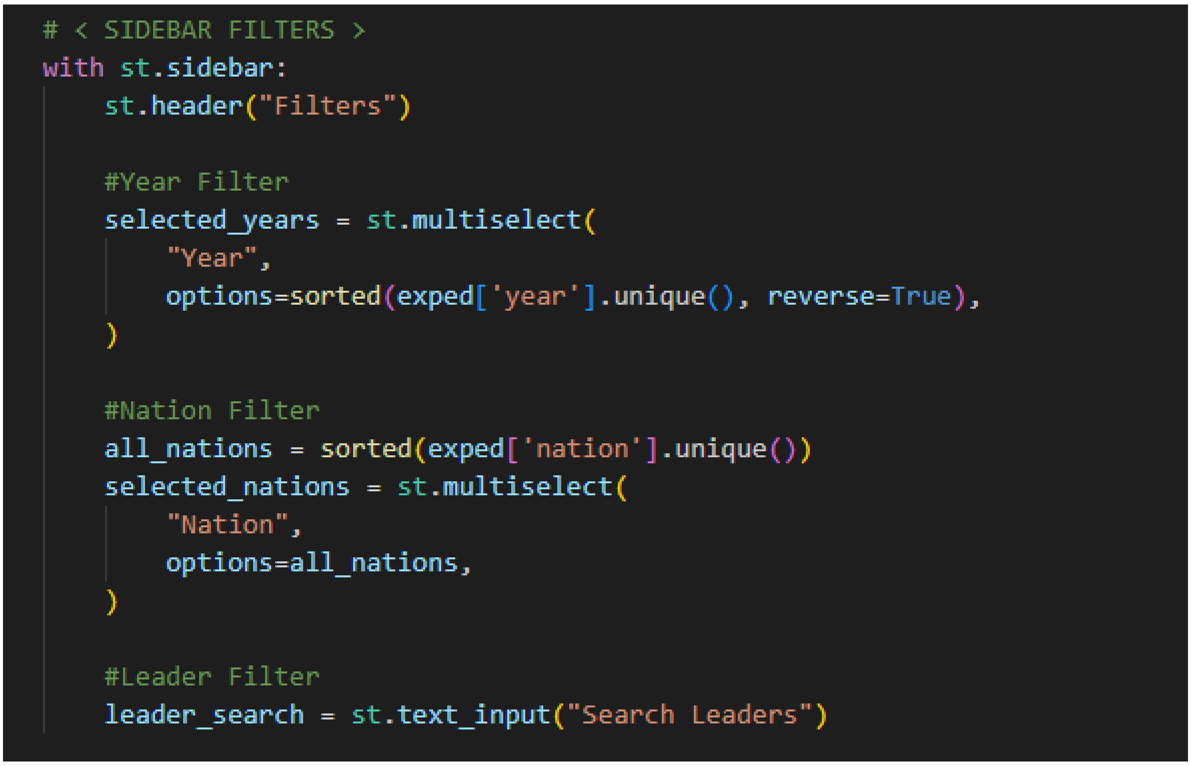
Himalayan Expedition Data Explorer

Explore expedition data from the Himalayan Database

A title is given to the page, named “Himalayan Expedition Data Explorer”, alongside the appropriate markdown right after.

Data is then loaded and unpacked into 4 variables.

Sidebar and Filter



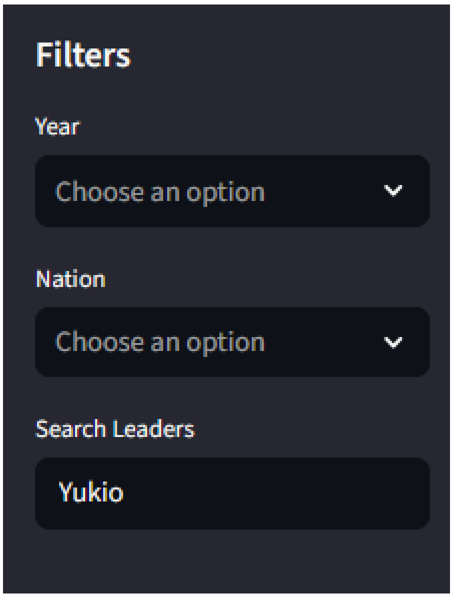
```
# < SIDEBAR FILTERS >
with st.sidebar:
    st.header("Filters")

    #Year Filter
    selected_years = st.multiselect(
        "Year",
        options=sorted(exped['year'].unique(), reverse=True),
    )

    #Nation Filter
    all_nations = sorted(exped['nation'].unique())
    selected_nations = st.multiselect(
        "Nation",
        options=all_nations,
    )

    #Leader Filter
    leader_search = st.text_input("Search Leaders")
```

Result:



Filters

Year

Choose an option

Nation

Choose an option

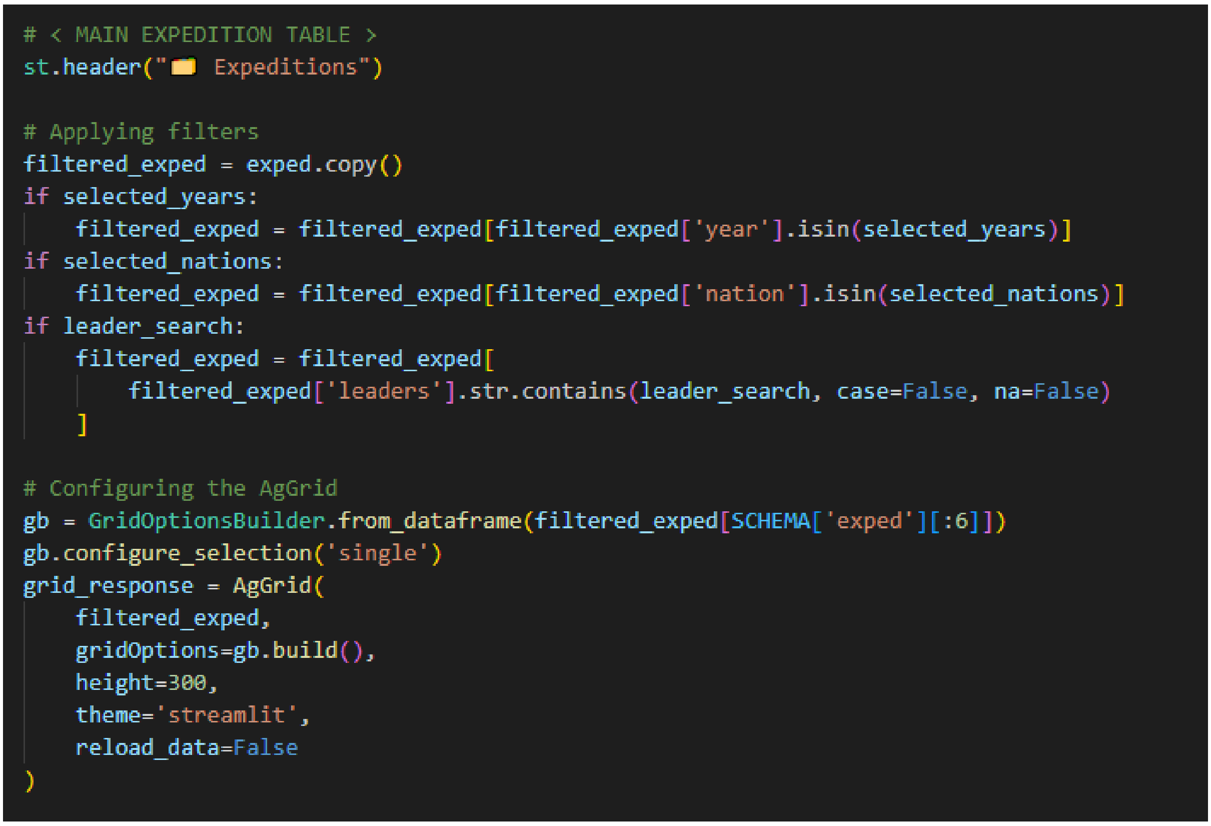
Search Leaders

Yukio

The filter bar allows the user to give filters to the main table by year, nation, and leader.

Main Expedition Table

Safe Selection Handling

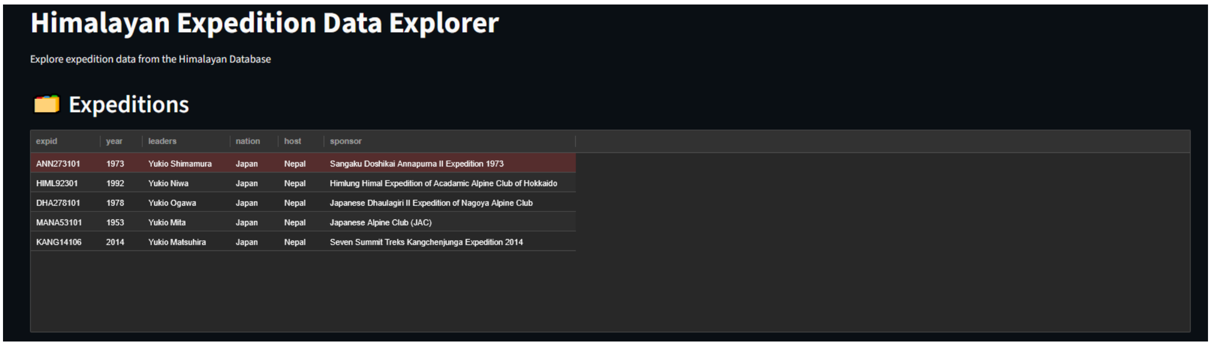


```
# < MAIN EXPEDITION TABLE >
st.header("📁 Expeditions")

# Applying filters
filtered_exped = exped.copy()
if selected_years:
    filtered_exped = filtered_exped[filtered_exped['year'].isin(selected_years)]
if selected_nations:
    filtered_exped = filtered_exped[filtered_exped['nation'].isin(selected_nations)]
if leader_search:
    filtered_exped = filtered_exped[
        filtered_exped['leaders'].str.contains(leader_search, case=False, na=False)
    ]

# Configuring the AgGrid
gb = GridOptionsBuilder.from_dataframe(filtered_exped[SCHEMA['exped'][:6]])
gb.configure_selection('single')
grid_response = AgGrid(
    filtered_exped,
    gridOptions=gb.build(),
    height=300,
    theme='streamlit',
    reload_data=False
)
```

Result:



Himalayan Expedition Data Explorer

Explore expedition data from the Himalayan Database

📁 Expeditions

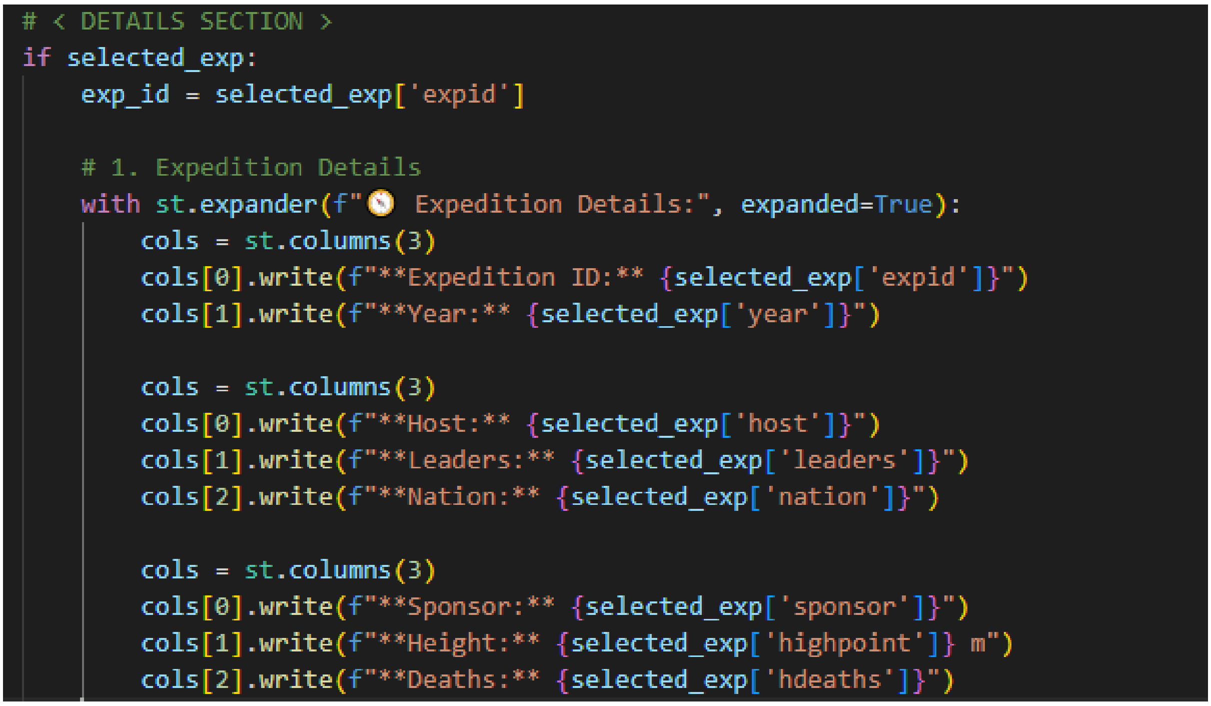
expid	year	leaders	nation	host	sponsor
ANN273101	1973	Yukio Shimamura	Japan	Nepal	Sangaku Doshikai Annapurna II Expedition 1973
HIML92301	1992	Yukio Niwa	Japan	Nepal	Himlung Himal Expedition of Acadamic Alpine Club of Hokkaido
DHA278101	1978	Yukio Ogawa	Japan	Nepal	Japanese Dhaulagiri II Expedition of Nagoya Alpine Club
MANA53101	1953	Yukio Mita	Japan	Nepal	Japanese Alpine Club (JAC)
KANG14106	2014	Yukio Matsuhira	Japan	Nepal	Seven Summit Treks Kangchenjunga Expedition 2014

First the filtering is applied before loading the table. This only happens if the user had made a selection in the filter sidebar.

Then the table is configured using AgGrid. With the first six columns of the SCHEMA used to create the main table.

Details Section

Expedition Details:

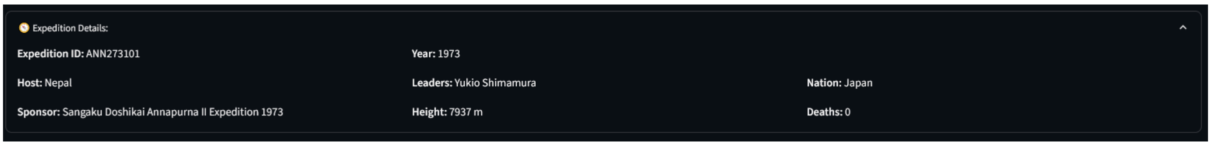


```python
# < DETAILS SECTION >
if selected_exp:
    exp_id = selected_exp['expid']

    # 1. Expedition Details
    with st.expander(f"🧭 Expedition Details:", expanded=True):
        cols = st.columns(3)
        cols[0].write(f"**Expedition ID:** {selected_exp['expid']}")
        cols[1].write(f"**Year:** {selected_exp['year']}")

        cols = st.columns(3)
        cols[0].write(f"**Host:** {selected_exp['host']}")
        cols[1].write(f"**Leaders:** {selected_exp['leaders']}")
        cols[2].write(f"**Nation:** {selected_exp['nation']}")

        cols = st.columns(3)
        cols[0].write(f"**Sponsor:** {selected_exp['sponsor']}")
        cols[1].write(f"**Height:** {selected_exp['highpoint']} m")
        cols[2].write(f"**Deaths:** {selected_exp['hdeaths']}")
```



🧭 Expedition Details:

Expedition ID: ANN273101 **Year:** 1973

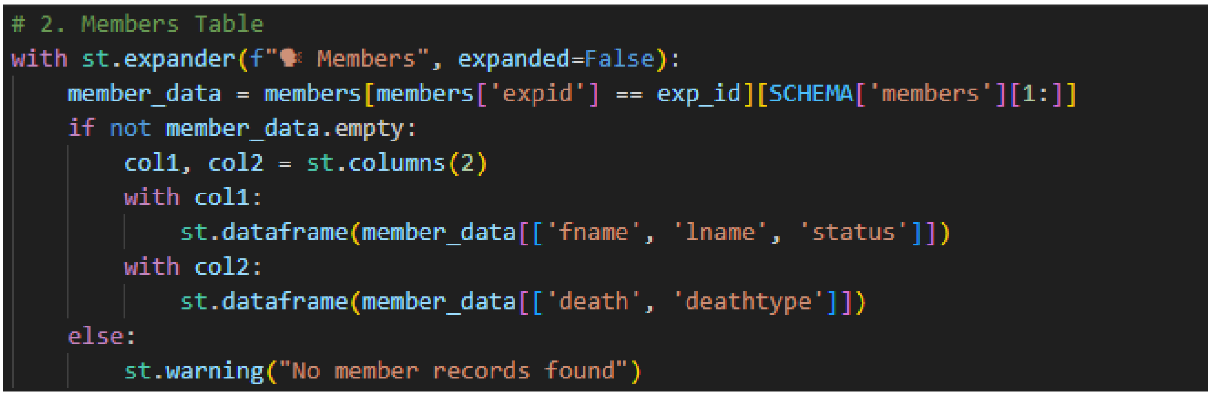
Host: Nepal **Leaders:** Yukio Shimamura **Nation:** Japan

Sponsor: Sangaku Doshikai Annapurna II Expedition 1973 **Height:** 7937 m **Deaths:** 0

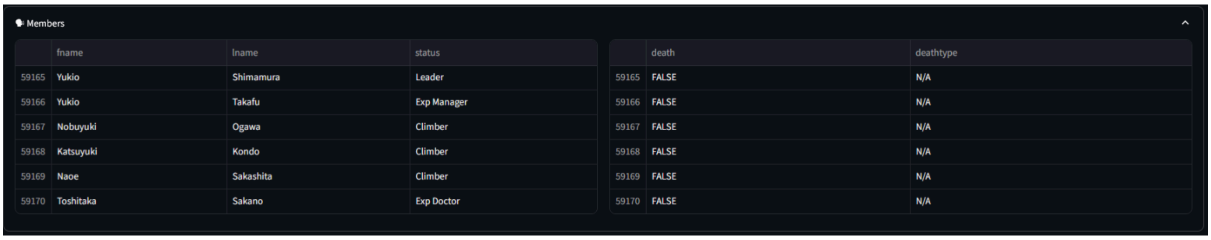
When a table is clicked within the main table, a collapsible section is created containing the details of the expedition.

The data is presented with columns.

Members Table:



```python
# 2. Members Table
with st.expander(f"👣 Members", expanded=False):
    member_data = members[members['expid'] == exp_id][SCHEMA['members'][1:]]
    if not member_data.empty:
        col1, col2 = st.columns(2)
        with col1:
            st.dataframe(member_data[['fname', 'lname', 'status']])
        with col2:
            st.dataframe(member_data[['death', 'deathtype']])
    else:
        st.warning("No member records found")
```



👣 Members

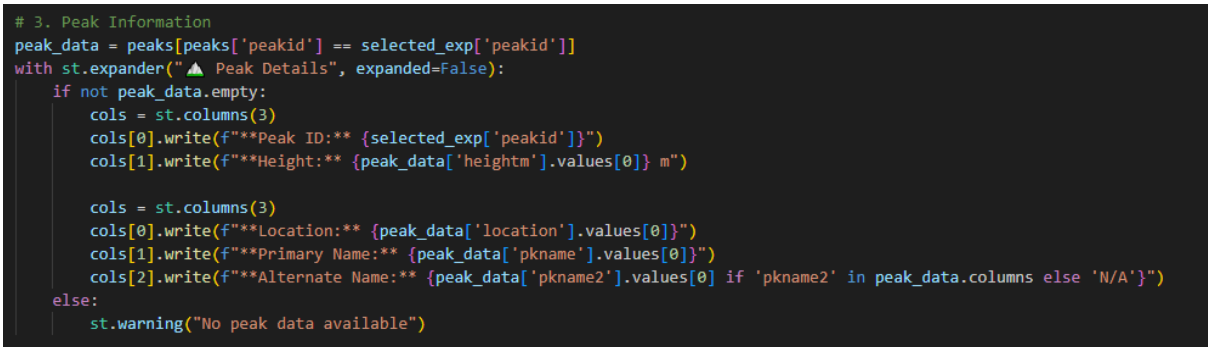
	fname	lname	status
59165	Yukio	Shimamura	Leader
59166	Yukio	Takafu	Exp Manager
59167	Nobuyuki	Ogawa	Climber
59168	Katsuyuki	Kondo	Climber
59169	Naoe	Sakashita	Climber
59170	Toshitaka	Sakano	Exp Doctor

	death	deathtype
59165	FALSE	N/A
59166	FALSE	N/A
59167	FALSE	N/A
59168	FALSE	N/A
59169	FALSE	N/A
59170	FALSE	N/A

When a table is clicked within the main table, another collapsible section is created containing the tables of the members in the expedition.

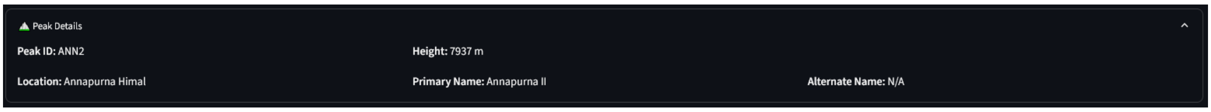
The data is presented with 2 tables due to its size. One containing the names and status(roles), the other pertaining their deaths (if it happened)

Peak Details:



```
# 3. Peak Information
peak_data = peaks[peaks['peakid'] == selected_exp['peakid']]
with st.expander("⛰ Peak Details", expanded=False):
    if not peak_data.empty:
        cols = st.columns(3)
        cols[0].write(f"**Peak ID:** {selected_exp['peakid']}")
        cols[1].write(f"**Height:** {peak_data['heightm'].values[0]} m")

        cols = st.columns(3)
        cols[0].write(f"**Location:** {peak_data['location'].values[0]}")
        cols[1].write(f"**Primary Name:** {peak_data['pkname'].values[0]}")
        cols[2].write(f"**Alternate Name:** {peak_data['pkname2'].values[0] if 'pkname2' in peak_data.columns else 'N/A'}")
    else:
        st.warning("No peak data available")
```



⛰ Peak Details

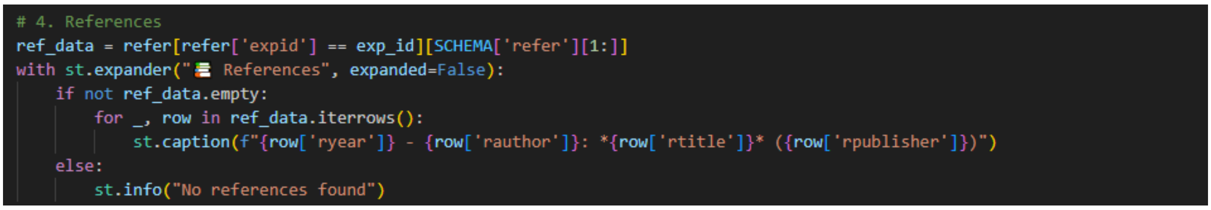
Peak ID: ANN2 **Height:** 7937 m

Location: Annapurna Himal **Primary Name:** Annapurna II **Alternate Name:** N/A

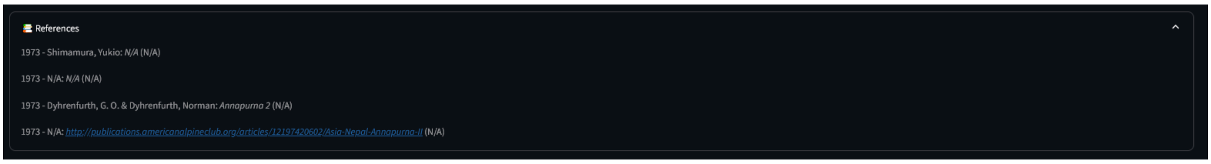
When a table is clicked within the main table, another collapsible section is created containing the detail of the expedition peak.

The data is presented with columns.

Reference:



```
# 4. References
ref_data = refer[refer['expid'] == exp_id][SCHEMA['refer'][1:]]
with st.expander("📚 References", expanded=False):
    if not ref_data.empty:
        for _, row in ref_data.iterrows():
            st.caption(f"{row['ryear']} - {row['rauthor']}: *{row['rtitle']}* ({row['rpublisher']})")
    else:
        st.info("No references found")
```



📚 References

1973 - Shimamura, Yukio: *N/A* (N/A)

1973 - N/A: *N/A* (N/A)

1973 - Dyhrenfurth, G. O. & Dyhrenfurth, Norman: *Annapurna 2* (N/A)

1973 - N/A: *http://publications.americanalpineclub.org/articles/12197420602/Asia-Nepal-Annapurna-II* (N/A)

When a table is clicked in the main table, the final collapsible section is created containing the references of the data of the expedition.

The data is presented in rows, using the APA Style of citation format.