Project 3 Write-up: Starbucks Locations App

Group 10: Kim Khue Nguyen, Noah Stevens, Marcanthony Solorzano, Gina Butler

Purpose:

The objective of this project is to apply the skills acquired throughout weeks one through fourteen of the Data Analytics and Visualization Bootcamp to develop a data application with interactive visualizations for data analysis. This project involves analyzing datasets and creating an application that provides insights through interactive elements.

Inspiration:

Our team selected the topic of Worldwide Starbucks Locations based on our personal experiences with Starbucks in different parts of the world. We were inspired by the desire to understand the distribution of Starbucks stores globally and to uncover any patterns or trends in store locations.

Our inspiration came from the following datasets:

- **Starbucks Locations Worldwide 2021:** https://www.kaggle.com/datasets/kukuroo3/starbucks-locations-worldwide-2021-version
- **Starbucks Store Location Analysis:** https://nycdatascience.com/blog/r/starbucks-store-location-analysis/?utm_source=chatgpt.com

Both datasets sparked our curiosity about the number of Starbucks stores worldwide and whether there were any noticeable trends in their distribution.

Datasets Used:

To ensure we worked with the most recent and comprehensive data, we used the following datasets:

- **Starbucks Locations Worldwide 2023:** https://www.kaggle.com/datasets/omarsobhy14/starbucks-store-location-2023
- **Country Mapping - ISO, Continent, Region: https://www.kaggle.com/datasets/andradaolteanu/country-mapping-iso-continent-region**

The Starbucks Locations Worldwide 2023 dataset provided a more updated list of Starbucks store locations, while the Country Mapping dataset allowed us to enhance our analysis by incorporating detailed geographical data, including continents and regions.

I. Data Cleaning Process

As a team, we conducted data cleaning and preprocessing using Jupyter Notebook. Our main steps included:

```
df_Starbucks.head()
```

	Brand	Store Number	Store Name	Ownership Type	Street Address	City	State/Province	Country	Postcode	Phone Number	Timezone	Longitude	Latitude
0	Starbucks	47370-257954	Meritxell, 96	Licensed	Av. Meritxell, 96	Andorra la Vella	7	AD	AD500	376818720	GMT+1:00 Europe/Andorra	1.53	42.51
1	Starbucks	22331-212325	Ajman Drive Thru	Licensed	1 Street 69, Al Jarf	Ajman	AJ	AE	NaN	NaN	GMT+04:00 Asia/Dubai	55.47	25.42
2	Starbucks	47089-256771	Dana Mall	Licensed	Sheikh Khalifa Bin Zayed St.	Ajman	AJ	AE	NaN	NaN	GMT+04:00 Asia/Dubai	55.47	25.39
3	Starbucks	22126-218024	Twofour 54	Licensed	Al Salam Street	Abu Dhabi	AZ	AE	NaN	NaN	GMT+04:00 Asia/Dubai	54.38	24.48
4	Starbucks	17127-178586	Al Ain Tower	Licensed	Khaldiya Area, Abu Dhabi Island	Abu Dhabi	AZ	AE	NaN	NaN	GMT+04:00 Asia/Dubai	54.54	24.51

1. **Merging Data: We performed a LEFT JOIN to merge country and region data between the `continent.csv` and `Starbucks.csv` while keeping only relevant Starbucks location data.**

```
# Re-merging with the correct key (using alpha-2 from Starbucks dataset before replacement)
df_merged = df_Starbucks.merge(df_Countries[['alpha-2', 'name', 'region']],
                               left_on='Country', right_on='alpha-2', how='left')

# Replace the alpha-2 country code with the full country name
df_merged['Country'] = df_merged['name']

# Drop unnecessary columns from the merge
df_merged.drop(columns=['alpha-2', 'name'], inplace=True)
```

```
df_merged.head()
```

	Brand	Store Number	Store Name	Ownership Type	Street Address	City	State/Province	Country	Postcode	Phone Number	Timezone	Longitude	Latitude	region
0	Starbucks	47370-257954	Meritxell, 96	Licensed	Av. Meritxell, 96	Andorra la Vella	7	Andorra	AD500	376818720	GMT+1:00 Europe/Andorra	1.53	42.51	Europe
1	Starbucks	22331-212325	Ajman Drive Thru	Licensed	1 Street 69, Al Jarf	Ajman	AJ	United Arab Emirates	NaN	NaN	GMT+04:00 Asia/Dubai	55.47	25.42	Asia
2	Starbucks	47089-256771	Dana Mall	Licensed	Sheikh Khalifa Bin Zayed St.	Ajman	AJ	United Arab Emirates	NaN	NaN	GMT+04:00 Asia/Dubai	55.47	25.39	Asia
3	Starbucks	22126-218024	Twofour 54	Licensed	Al Salam Street	Abu Dhabi	AZ	United Arab Emirates	NaN	NaN	GMT+04:00 Asia/Dubai	54.38	24.48	Asia
4	Starbucks	17127-178586	Al Ain Tower	Licensed	Khaldiya Area, Abu Dhabi Island	Abu Dhabi	AZ	United Arab Emirates	NaN	NaN	GMT+04:00 Asia/Dubai	54.54	24.51	Asia

2. **Dropping Unnecessary Columns: The following columns were removed to streamline our dataset:**
   - **"Store Name"**
   - **"Brand"**

- "Ownership Type"
- "Postcode"
- "alpha-2"
- "Phone Number"
- "Timezone"
- "Street Address"

```
UNWANTED_COLUMNS = ["Store Name", "Brand", "Ownership Type", "Postcode", "Phone Number", "Timezone", "Street Address"]
df_merged = df_merged.drop(columns=UNWANTED_COLUMNS).dropna()
df_merged.head()
```

	Store Number	City	State/Province	Country	Longitude	Latitude	region
0	47370-257954	Andorra la Vella	7	Andorra	1.53	42.51	Europe
1	22331-212325	Ajman	AJ	United Arab Emirates	55.47	25.42	Asia
2	47089-256771	Ajman	AJ	United Arab Emirates	55.47	25.39	Asia
3	22126-218024	Abu Dhabi	AZ	United Arab Emirates	54.38	24.48	Asia
4	17127-178586	Abu Dhabi	AZ	United Arab Emirates	54.54	24.51	Asia

Rename the columns:

```
df_merged.rename(columns = {'Store Number':'store_number', 'City':'city', 'State/Province':'state_province', 'Country':'country', 'Longitude': 'longitude
df_merged
```

	store_number	city	state_province	country	longitude	latitude	region
0	47370-257954	Andorra la Vella	7	Andorra	1.53	42.51	Europe
1	22331-212325	Ajman	AJ	United Arab Emirates	55.47	25.42	Asia
2	47089-256771	Ajman	AJ	United Arab Emirates	55.47	25.39	Asia
3	22126-218024	Abu Dhabi	AZ	United Arab Emirates	54.38	24.48	Asia
4	17127-178586	Abu Dhabi	AZ	United Arab Emirates	54.54	24.51	Asia
...	...	...	...	...	...	...	...
25596	21401-212072	Thành Ph? H? Chí Minh	SG	Vietnam	106.70	10.78	Asia
25597	24010-226985	Thành Ph? H? Chí Minh	SG	Vietnam	106.71	10.72	Asia
25598	47608-253804	Johannesburg	GT	South Africa	28.04	-26.15	Africa
25599	47640-253809	Menlyn	GT	South Africa	28.28	-25.79	Africa
25600	47609-253286	Midrand	GT	South Africa	28.11	-26.02	Africa

25583 rows × 7 columns

3. **Filtering for English-language Countries: We reduced the dataset to include only countries represented in English to maintain consistency.**
4. **Creating a SQLite Database: After cleaning, we stored the processed data in an SQLite database, enabling efficient queries for further analysis.**

II. Application Backend

`app.py` and `sqlHelper.py`

Our Starbucks Locations App is powered by a Flask backend that manages API requests, database queries, and serves interactive visualizations. Below is an in-depth explanation of the two key components:

---

1. Flask Application (`app.py`)

The `app.py` file is the core of our application, handling routing, API endpoints, and rendering HTML templates. It uses Flask, a lightweight Python web framework, to serve content dynamically.

Key Features:

a. Flask Setup

```
app = Flask(__name__)
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0  # Remove caching
```

- **Initializes the Flask application.**
- **Disables caching to ensure updated content is served.**

b. HTML Page Routes

```
@app.route("/")
def welcome():
    return render_template("index.html")
```

- **Renders different pages (`index.html`, `dashboard.html`, `map.html`, etc.).**
- **Uses Flask's `render_template()` function to serve HTML templates.**

c. API Endpoints

These endpoints fetch Starbucks store data dynamically from a database and return JSON responses for frontend use.

```
@app.route("/api/v1.0/bar_data/<country_input>")
def bar_data(country_input):
    df = sqlHelper.queryBarData(country_input)
    return jsonify(df.to_dict(orient="records"))
```

- **The API retrieves bar chart data for a specific country using SQL queries.**
- **Converts the query result from a Pandas DataFrame to JSON format.**

Other similar API endpoints include:

- `/api/v1.0/table_data` → **Fetches structured tabular data.**
- `/api/v1.0/burst_data` → **Retrieves burst analysis data.**
- `/api/v1.0/map_data` → **Fetches store location data for the interactive map.**

d. Caching Control

To prevent users from seeing outdated data, Flask adds HTTP headers to disable browser caching.

```
@app.after_request
def add_header(r):
    r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate,
public, max-age=0"
    r.headers["Pragma"] = "no-cache"
    r.headers["Expires"] = "0"
    return r
```

- **Ensures users always see the latest version of the app's data and visualizations.**

e. Running the Application

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

- **Starts the Flask web server in debug mode for development.**
- **Debug mode should be turned off for production.**

---

2. Database Helper (`sqlHelper.py`)

The `sqlHelper.py` file acts as a bridge between the Flask app and the SQLite database. It contains SQL queries to retrieve Starbucks location data efficiently.

Key Features:

a. SQL Query Execution

Each function in `sqlHelper.py` executes SQL queries and returns Pandas DataFrames for further processing.

Example: Querying Starbucks store data by country

```
def queryBarData(self, country_input):
    query = f"SELECT country, COUNT(*) as store_count FROM starbucks
WHERE country = '{country_input}' GROUP BY country;"
    return pd.read_sql(query, self.connection)
```

- **Retrieves the number of Starbucks stores per country.**
- **Uses SQL aggregation (`COUNT(*)`) to count store locations.**
- **Returns a Pandas DataFrame for further transformation.**

b. Queries for Different API Routes

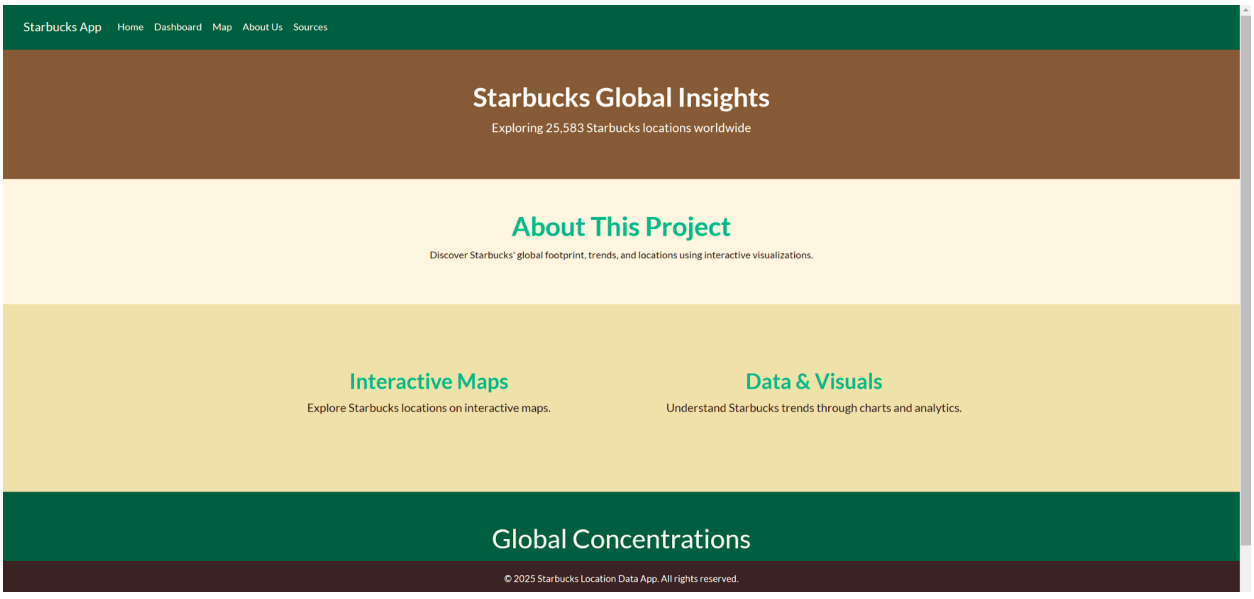
Other query functions include:

- **`queryTableData()` → Fetches structured tabular data for the dashboard.**
- **`queryBurstData()` → Retrieves burst analysis data for advanced insights.**
- **`queryMapData()` → Returns latitude/longitude for Starbucks store locations.**

III. Application Frontend

Data Visualization and Analysis Techniques

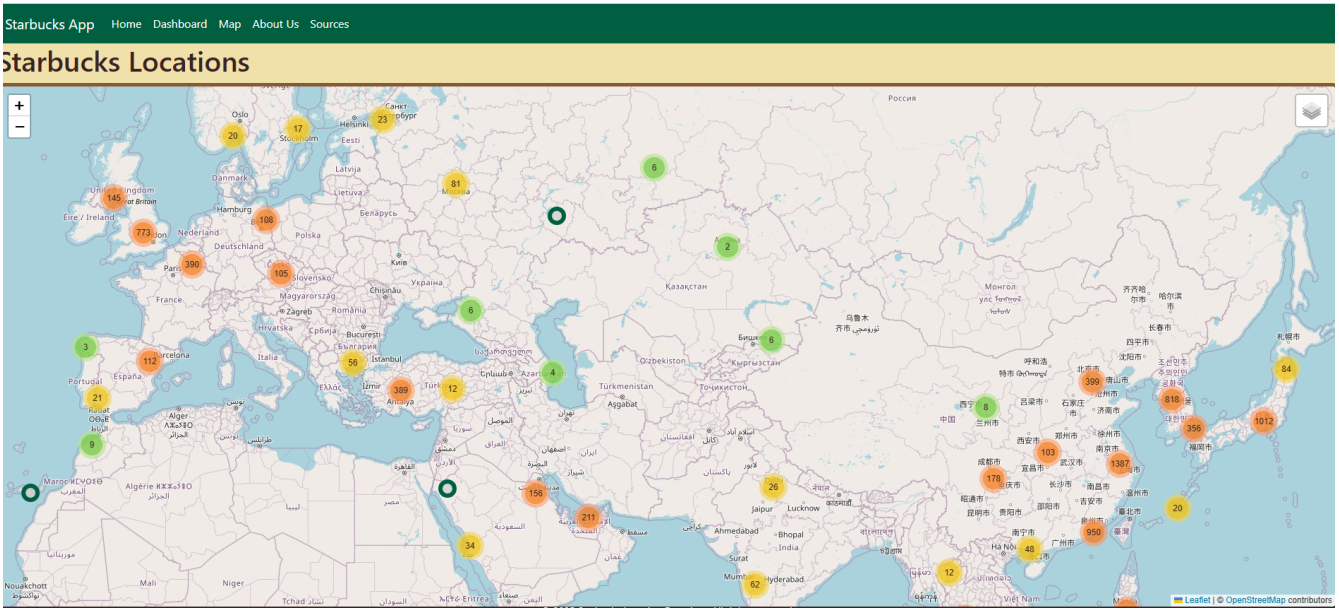
This structured data cleaning and integration enabled us to develop a robust data application with interactive visualizations to explore Starbucks' global footprint effectively. Below are the details of the visualization techniques and tools used in our application.



By integrating these visualization and analysis techniques, our Starbucks Locations App offers a comprehensive platform for exploring and understanding the global footprint of Starbucks stores.

1. Interactive Maps

We utilized geospatial visualization tools to create dynamic maps that allow users to explore Starbucks locations worldwide. These maps enable users to zoom in on specific regions, filter locations by country or city, and visualize the density of stores in various areas. The interactive nature of the maps enhances user engagement and provides a comprehensive understanding of Starbucks' geographical distribution.



a. Creating a unique marker

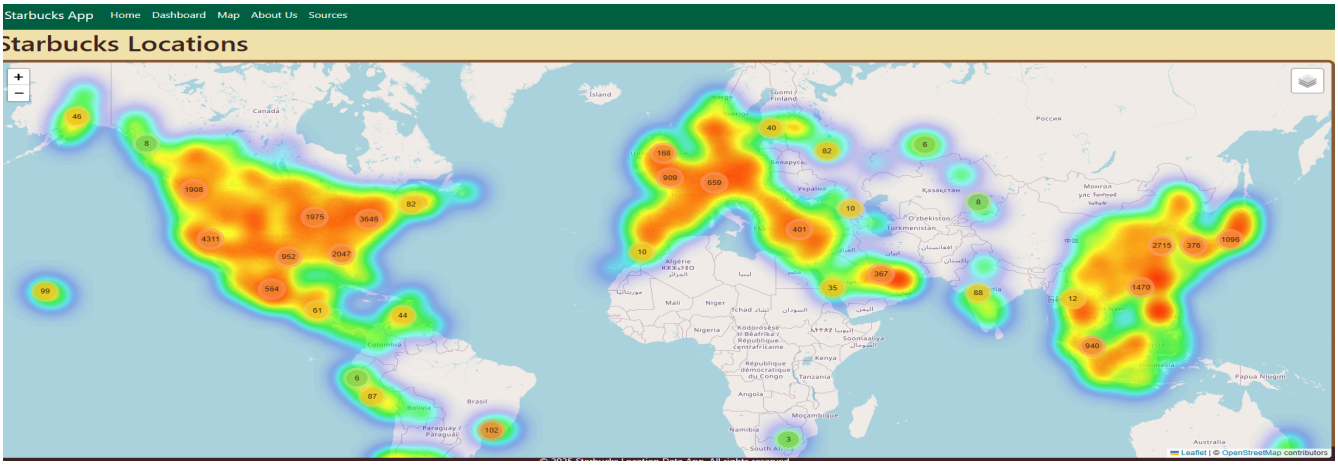
Created a circular marker that shows the store number, city, country and region.

Example: Code for marker.

```
let marker = L.circleMarker([row.latitude, row.longitude], {
        radius: 10, // Size of the circle
        color: "#006241", // Outline color
        fillColor: "#D3D3D3", // Fill color
        fillOpacity: 0.8, // Transparency
        weight: 6 // Border thickness
      }).bindPopup(`<h1>${row.store_number}<h1><h2>${row.city},
${row.country}</h2><h3>${row.region}</h3>`);
      markers.addLayer(marker);
```

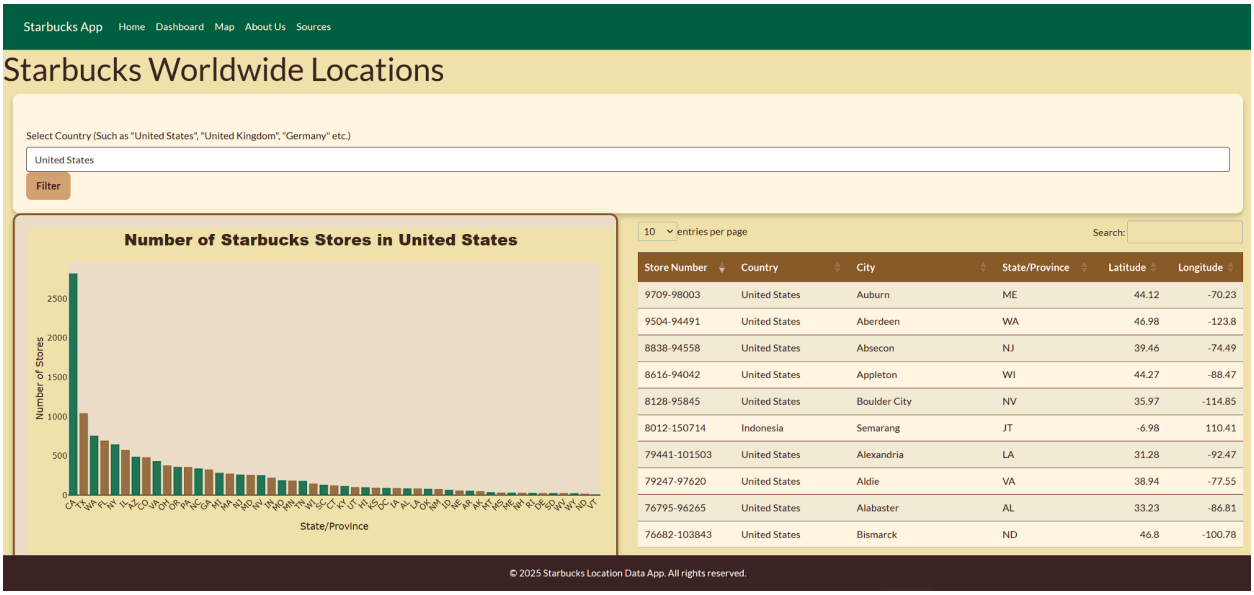
2. Global Concentration Analysis

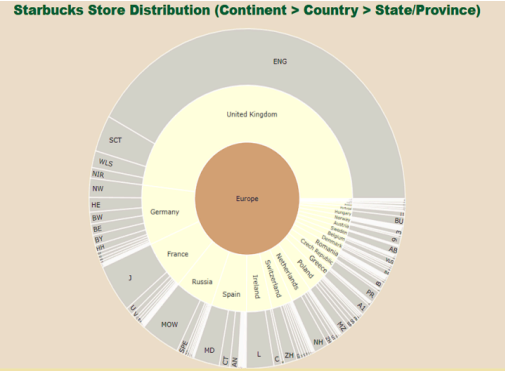
We conducted an analysis of Starbucks' global concentrations by examining the density of stores in different regions. This involved calculating the number of stores relative to population size and geographic area, providing insights into market saturation and potential areas for expansion. The results are presented through heat maps and density plots, offering a visual representation of Starbucks' market presence.

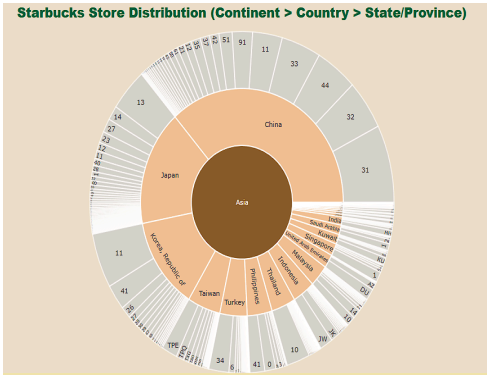


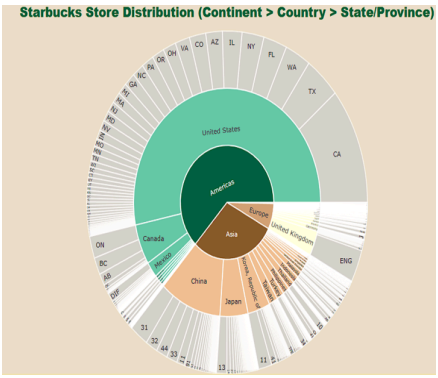
3. Data Dashboards

Our application features dashboards that present data through various plots and graphs. These visualizations include bar charts, Sunburst, interactive maps, and tables that depict trends such as the number of stores per country, growth over time, and regional comparisons. The dashboards are designed to be user-friendly, allowing for easy interpretation of complex data sets.









4. Data Filtering and Search Functionality

To enhance user experience, we incorporated data filtering and search capabilities. Users can search for specific locations, filter data by various criteria such as country, city, or store type, and view corresponding visualizations. This functionality allows for tailored data exploration, enabling users to focus on areas of interest. These are separated so that the table can be used to find specific locations, as the bar chart is made specifically for countries.

5. CSS Styling

The `style.css` has 238 lines of code and styles every portion of the webpage, and helps with such functionalities as the Navbar collapsing when the window is small, ensuring the footer sticks to the bottom of the page, and creating boxes and styling for the table, plots and map. The following is what we used for the footer, specifically where the sticky position allows the footer to always be at the bottom of the page, whether at the top, middle or bottom of the page.

```
Example:
/* Footer Styling */
footer {
background-color: #3e2723; /* Dark Coffee */
…… More Code in between ……
width: 100%;
position: sticky;
}
```

===

In our Starbucks Locations App, we employed various visualization and analysis techniques to provide users with interactive insights into Starbucks' global presence.