Tanvi Sharma
Data Science
City of Austin

Democratizing Data

Python Powered Dashboards &
Open Data for Transparent Governance in Austin

Democratizing Data

Python Powered Dashboards &
Open Data for Transparent Governance in Austin

Tanvi Sharma
Data Science
City of Austin

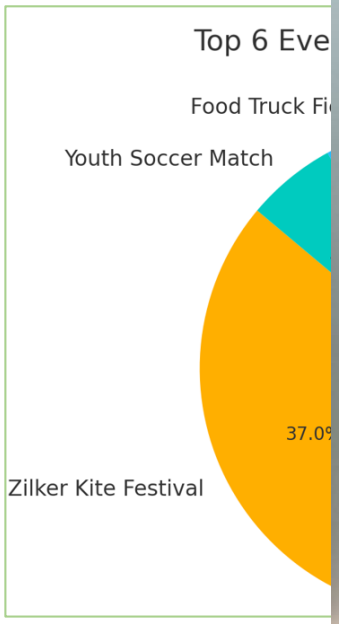
Event table

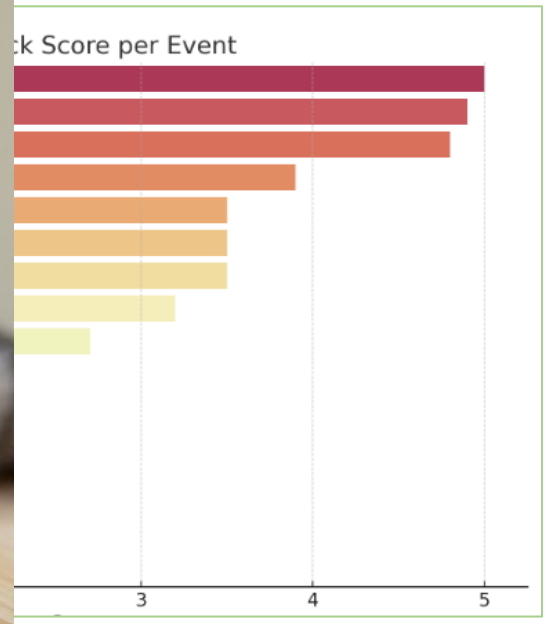
Event Name		Count	Program Budget
Zilker Kite Festival			3000
Outdoor Movie Night			4500
Youth Soccer Match			3000
Neighborhood BBQ Bash			1500
Live Jazz in the Park			2500
Senior Citizen Picnic			3500
Art in the Park			2500
Community Garden Fair			4500
Local Musicians Showcase			3500
Family Fun Gathering			1500
Food Truck Fiesta			4500
Weekend Wellness Fair			2000
DIY Craft Workshop			3000
Spring Picnic			1500
Fall Volunteer Drive			3000
Open Mic Evening			4000

Democratizing Data

Python Powered Dashboards &
Open Data for Transparent Governance in Austin

Tanvi Sharma
Data Science
City of Austin





Tanvi Sharma
Data Science
City of Austin

Power of Data Visualization

*"As humans, we process visuals **60,000 times** faster than text. Dashboards don't just show data, they tell a story."*

Tanvi Sharma
Data Science
City of Austin

About Me

- Tanvi Sharma – Gen Z Data Professional
- 5+ years in Data Science, ML, Visualization
- Data Enthusiast at City of Austin
- 14,000+ followers on LinkedIn sharing data insights
- Enjoy exploring U.S. national parks

Tanvi Sharma
Data Science
City of Austin

Why Data Transparency matters?

Trust

Builds public confidence in decisions

Accountability

Shows what's working (and what's not)

Equity

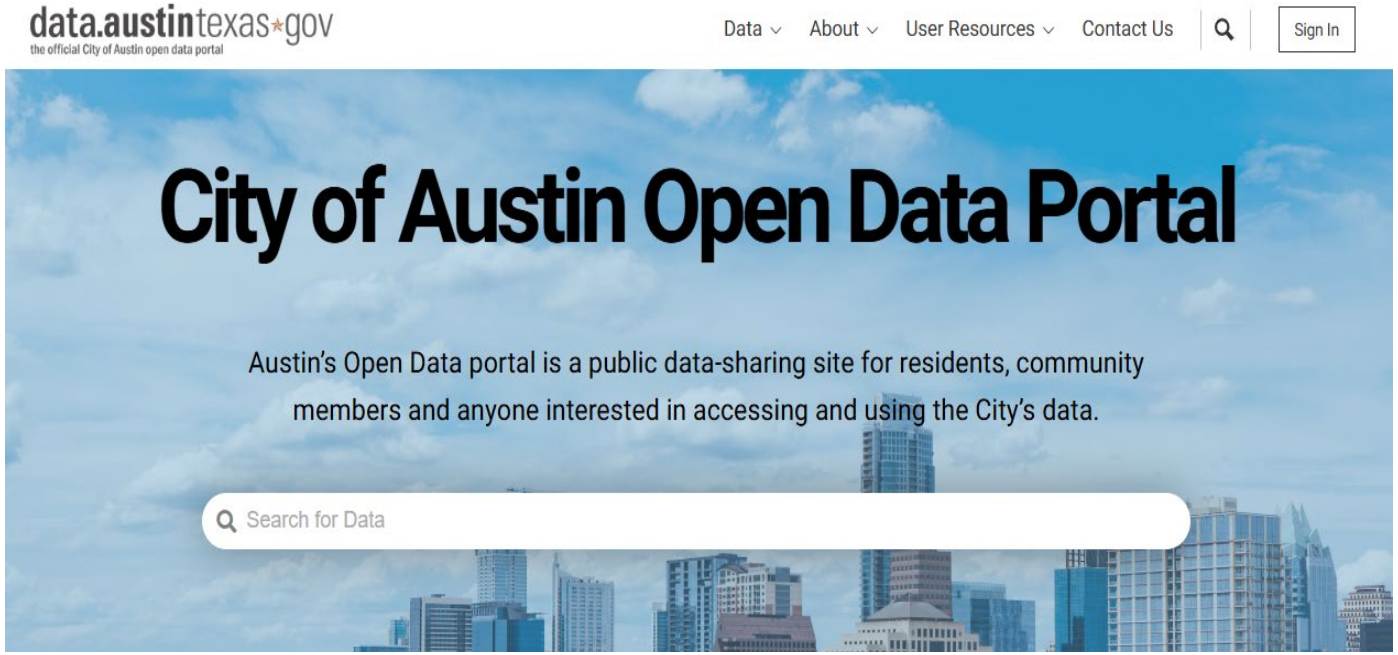
Highlights gaps and underserved areas

Collaboration

Encourages dialogue and Civic participation

Tanvi Sharma
Data Science
City of Austin

What is Open data?



- Freely available public data in readable formats
- No restrictions on use or sharing
- The engine powering transparency tools
- Used by cities like Austin to engage citizens
- Link: https://data.austintexas.gov/

Tanvi Sharma
Data Science
City of Austin

From Open portal to Clean Dataset

1. Real-Time Traffic Incident Reports

- Live traffic data updated every 5 minutes
- Fields: issue type, location, status, time
- Why it matters: Reflects real-time city movement
- **Goal:** Spot trends and types of incidents over time
- Link: Real-Time Traffic Incident Reports | Open Data | City of Austin Texas

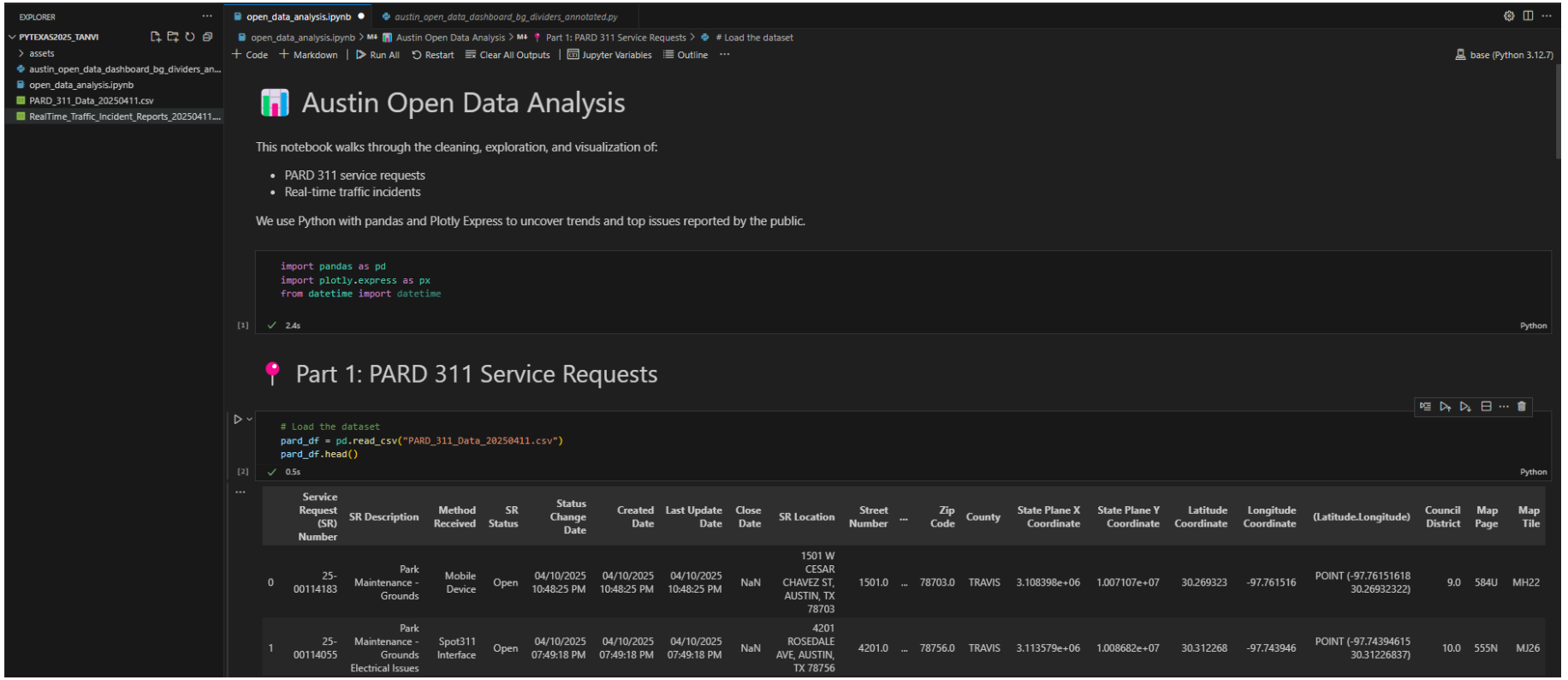
2. PARD 311 Service Requests

- Public service requests related to Austin parks
- Fields: service type, request date, status, location
- Why it matters: Shows real community needs
- **Goal:** Visualize request types, backlog, seasonality
- Link: PARD 311 Data | Open Data | City of Austin Texas

Democratizing Data

Python Powered Dashboards &
Open Data for Transparent Governance in Austin

Tanvi Sharma
Data Science
City of Austin



	Service Request (SR) Number	SR Description	Method Received	SR Status	Status Change Date	Created Date	Last Update Date	Close Date	SR Location	Street Number	...	Zip Code	County	State Plane X Coordinate	State Plane Y Coordinate	Latitude Coordinate	Longitude Coordinate	(Latitude,Longitude)	Council District	Map Page	Map Tile
0	25-00114183	Park Maintenance - Grounds	Mobile Device	Open	04/10/2025 10:48:25 PM	04/10/2025 10:48:25 PM	04/10/2025 10:48:25 PM	NaN	1501 W CESAR CHAVEZ ST, AUSTIN, TX 78703	1501.0	...	78703.0	TRAVIS	3.108398e+06	1.007107e+07	30.269323	-97.761516	POINT (-97.76151618 30.26932322)	9.0	584U	MH22
1	25-00114055	Park Maintenance - Grounds Electrical Issues	Spot311 Interface	Open	04/10/2025 07:49:18 PM	04/10/2025 07:49:18 PM	04/10/2025 07:49:18 PM	NaN	4201 ROSEDALE AVE, AUSTIN, TX 78756	4201.0	...	78756.0	TRAVIS	3.113579e+06	1.008682e+07	30.312268	-97.743946	POINT (-97.74394615 30.31226837)	10.0	555N	MJ26

- Use pandas to load CSVs from Austin's Open Data Portal
- Clean nulls, fix column names, filter by date/year
- Add computed fields (like "incident category" or year groups)

Democratizing Data

Python Powered Dashboards &
Open Data for Transparent Governance in Austin

Tanvi Sharma
Data Science
City of Austin

```
# Convert date columns to datetime
pard_df['Created Date'] = pd.to_datetime(pard_df['Created Date'], errors='coerce')
```

[3] ✓ 6.7s Python

```
C:\Users\SharmaT.COACD\AppData\Local\Temp\ipykernel_31180\1488975625.py:2: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected,
  pard_df['Created Date'] = pd.to_datetime(pard_df['Created Date'], errors='coerce')
```

```
# Top 10 request types
pard_type_summary = pard_df['SR Description'].value_counts().head(10).reset_index()
pard_type_summary.columns = ['SR Description', 'Count']
pard_type_summary
```

[4] ✓ 0.0s Python

	SR Description	Count
0	Park Maintenance - Grounds	45008
1	Park Maintenance - Grounds Plumbing Issues	5196
2	Park Maintenance - Pool Issues	3342
3	Park Maintenance - Grounds Electrical Issues	3127
4	Park Maintenance - Building Plumbing Issues	2116
5	Park Maintenance - Building Issues	1153
6	Park Maintenance - Building A/C & Heating Issues	697
7	Park Maintenance - Building Electrical Issues	390
8	Park Maintenance - Cemeteries	301
9	ZZ Park Maintenance - Pool Issues	95

```
# Requests over time
pard_timeline = pard_df.groupby(pard_df['Created Date'].dt.date).size().reset_index(name='Request Count')
pard_timeline.head()
```

[5] ✓ 0.0s Python

	Created Date	Request Count
0	2014-01-01	1
1	2014-01-02	3
2	2014-01-03	14
3	2014-01-04	3

- df['Date'] = pd.to_datetime(...) → Parsing dates
- df = df[df['Year'] >= 2021] → Filtering recent years
- df['Category'] = df['Desc'].str.extract(...) → Creating new fields
- .groupby() and .agg() → Summary stats for visualization

Tanvi Sharma
Data Science
City of Austin

Once we had our dataset clean, it is time to build something interactive.

Tanvi Sharma
Data Science
City of Austin

Building Interactive Dashboards with Dash

Dash-ing Toward Visualization

austin_open_data_dashboard_bg_dividers_annotated.py > ...

```python
import pandas as pd
from dash import Dash, dcc, html, Input, Output
import plotly.express as px

# Initialize the Dash app
app = Dash(__name__)

# Load and process PARD 311 data
df_pard = pd.read_csv("PARD_311_Data_20250411.csv")
df_pard['Created Date'] = pd.to_datetime(df_pard['Created Date'], errors='coerce')
df_pard['Year'] = df_pard['Created Date'].dt.year

# Load and process Traffic data
df_traffic = pd.read_csv("RealTime_Traffic_Incident_Reports_20250411.csv")
df_traffic['Published Date'] = pd.to_datetime(df_traffic['Published Date'], errors='coerce')
df_traffic['Year'] = df_traffic['Published Date'].dt.year
df_traffic['Published Date Only'] = df_traffic['Published Date'].dt.date

# Define layout and structure of the app
app.layout = html.Div([
    html.Link(rel='stylesheet', href='/assets/styles.css'),
    html.H1("City of Austin Open Data Dashboard", style={'textAlign': 'center'}),
    dcc.Tabs(id="tabs", value='tab-pard', children=[
        dcc.Tab(label='PARD 311', value='tab-pard'),
        dcc.Tab(label='Traffic Incidents', value='tab-traffic'),
    ]),
    html.Div(id='tabs-content')
])
```

- **.py File (Core Dash App Logic)** Define the dashboard layout using Dash components (like html.Div, dcc.Dropdown, and dcc.Graph)

Democratizing Data

Python Powered Dashboards &
Open Data for Transparent Governance in Austin

Tanvi Sharma
Data Science
City of Austin

```python
# Update tab content dynamically based on active tab selection
@app.callback(Output('tabs-content', 'children'), Input('tabs', 'value'))
def render_tab(tab):
    if tab == 'tab-pard':
        years = sorted(df_pard['Year'].dropna().unique())
        options = [{'label': 'All Years', 'value': 'all'}] + [{'label': str(y), 'value': y} for y in years]
        return html.Div([
            html.Label("Filter by Year:", style={'color': 'white'}),
            dcc.Dropdown(id='pard-year-dropdown', options=options, value='all'),
            html.Div(id='pard-charts', className='tab-section pard-bg')
        ])
    elif tab == 'tab-traffic':
        years = sorted(df_traffic['Year'].dropna().unique())
        options = [{'label': 'All Years', 'value': 'all'}] + [{'label': str(y), 'value': y} for y in years]
        return html.Div([
            html.Label("Filter by Year:", style={'color': 'white'}),
            dcc.Dropdown(id='traffic-year-dropdown', options=options, value='all'),
            html.Div(id='traffic-charts', className='tab-section traffic-bg')
        ])
```

- Dropdown filter lets users choose year (or 'All Years') to explore trends
- Graphs update automatically using Dash callbacks

Tanvi Sharma
Data Science
City of Austin

For PARD 311 data

```python
# Update PARD 311 plots based on selected year
@app.callback(Output('pard-charts', 'children'), Input('pard-year-dropdown', 'value'))
def update_pard_charts(selected_year):
    filtered_df = df_pard.copy() if selected_year == 'all' else df_pard[df_pard['Year'] == selected_year]
    year_label = "All Years" if selected_year == 'all' else str(selected_year)

    top_types = filtered_df['SR Description'].value_counts().head(10).reset_index()
    top_types.columns = ['SR Description', 'Count']
    timeline = filtered_df.groupby(filtered_df['Created Date'].dt.date).size().reset_index(name='Request Count')

    return [
        html.Div(dcc.Graph(figure=px.bar(top_types, x='SR Description', y='Count',
                                         title=f'Top PARD 311 Service Requests - {year_label}')), className='graph-block'),
        html.Hr(),
        html.Div(dcc.Graph(figure=px.line(timeline, x='Created Date', y='Request Count',
                                          title=f'PARD 311 Requests Over Time - {year_label}')), className='graph-block')
    ]
```

Tanvi Sharma
Data Science
City of Austin

For Real-Time Traffic Incident data

```python
# Update Traffic Incident plots based on selected year
@app.callback(Output('traffic-charts', 'children'), Input('traffic-year-dropdown', 'value'))
def update_traffic_charts(selected_year):
    filtered_df = df_traffic.copy() if selected_year == 'all' else df_traffic[df_traffic['Year'] == selected_year]
    year_label = "All Years" if selected_year == 'all' else str(selected_year)

    top_types = filtered_df['Issue Reported'].value_counts().head(10).reset_index()
    top_types.columns = ['Issue Reported', 'Count']
    timeline = filtered_df.groupby('Published Date Only').size().reset_index(name='Incident Count')

    return [
        html.Div(dcc.Graph(figure=px.bar(top_types, x='Issue Reported', y='Count',
                 title=f'Top Traffic Incident Types - {year_label}')), className='graph-block'),
        html.Hr(),
        html.Div(dcc.Graph(figure=px.line(timeline, x='Published Date Only', y='Incident Count',
                 title=f'Traffic Incidents Over Time - {year_label}')), className='graph-block')
    ]
```

Tanvi Sharma
Data Science
City of Austin

How to run this locally

```
# Run the Dash server locally
if __name__ == '__main__':
    app.run(debug=True)
```

- if __name__ == '__main__' block is what lets us launch the dashboard locally on our machine.
- By default output URL will be: http://127.0.0.1:8050/
- Tip: Replace 127.0.0.1 with your machine's IP if you want others on your Wi-Fi to access the dashboard—like from a tablet or second laptop.

Tanvi Sharma
Data Science
City of Austin

Styling with Custom CSS

open_data_analysis.ipynb | # styles_annotated.css

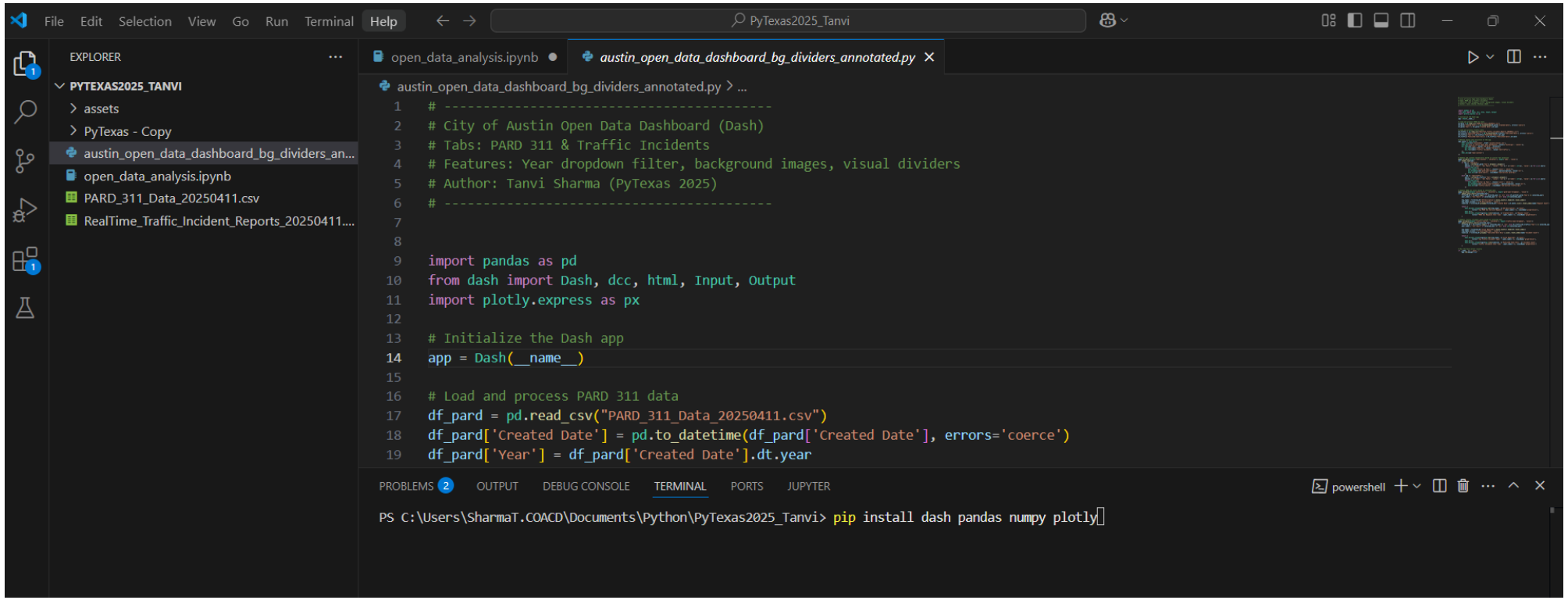
assets > # styles_annotated.css > .graph-block

```css
/* ----------------------------------------
   Custom Styling for City of Austin Dashboard
   Applies background images and styles per tab
   Author: Tanvi Sharma (PyTexas 2025)
---------------------------------------- */


.tab-section.pard-bg {
    background-image: url("/assets/311PARD.jpg");
    background-size: cover;
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: center;
    padding: 20px;
    background-color: rgba(255, 255, 255, 0.2);
}
.tab-section.traffic-bg {
    background-image: url("/assets/Traffic_light.jpg");
    background-size: cover;
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: center;
    padding: 20px;
    background-color: rgba(255, 255, 255, 0.2);
}
.graph-block {
    background-color: rgba(255,255,255,0.85);
    padding: 15px;
    border-radius: 8px;
    margin-bottom: 20px;
}

```

- Adjusted padding, image size, and chart spacing
- Added borders/dividers between dashboard sections for readability

Tanvi Sharma
Data Science
City of Austin

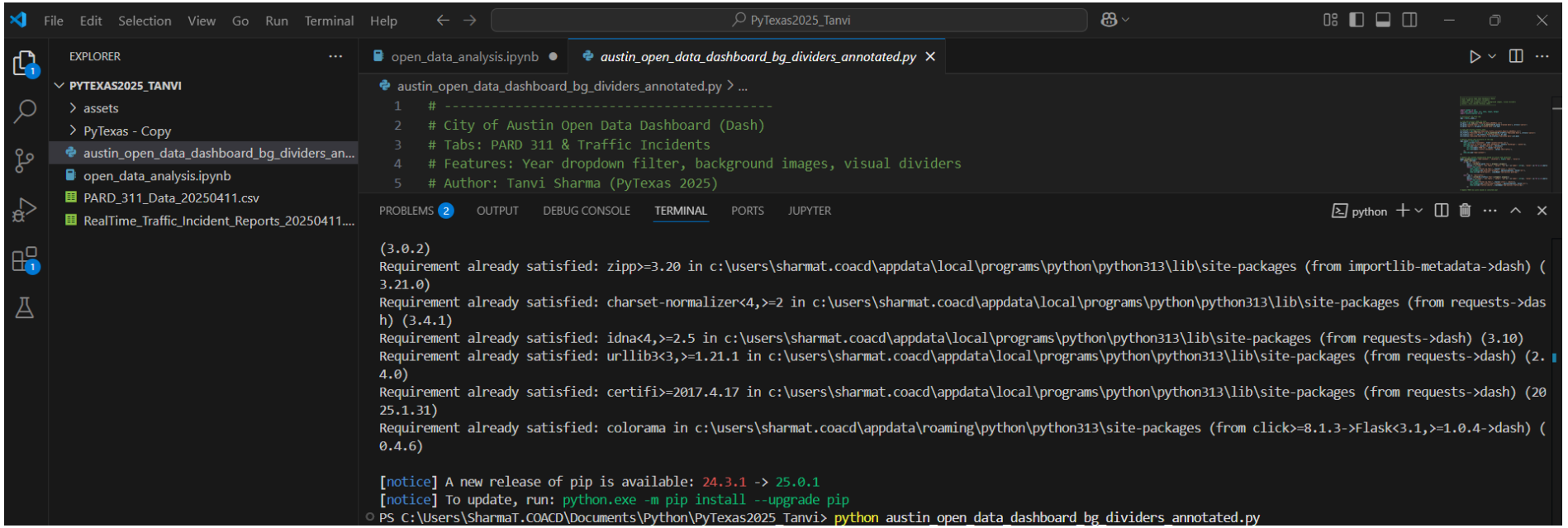
How to Run it from Terminal



Democratizing Data

Python Powered Dashboards & Open Data for Transparent Governance in Austin

Tanvi Sharma
Data Science
City of Austin



Code: python austin_open_data_dashboard_bg_dividers_annotated.py

```
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app 'austin_open_data_dashboard_bg_dividers_annotated'
 * Debug mode: on
```

Democratizing Data

Python Powered Dashboards &
Open Data for Transparent Governance in Austin

Tanvi Sharma
Data Science
City of Austin

City of Austin Open Data Dashboard

PARD 311 | Traffic Incidents

All Years

Democratizing Data

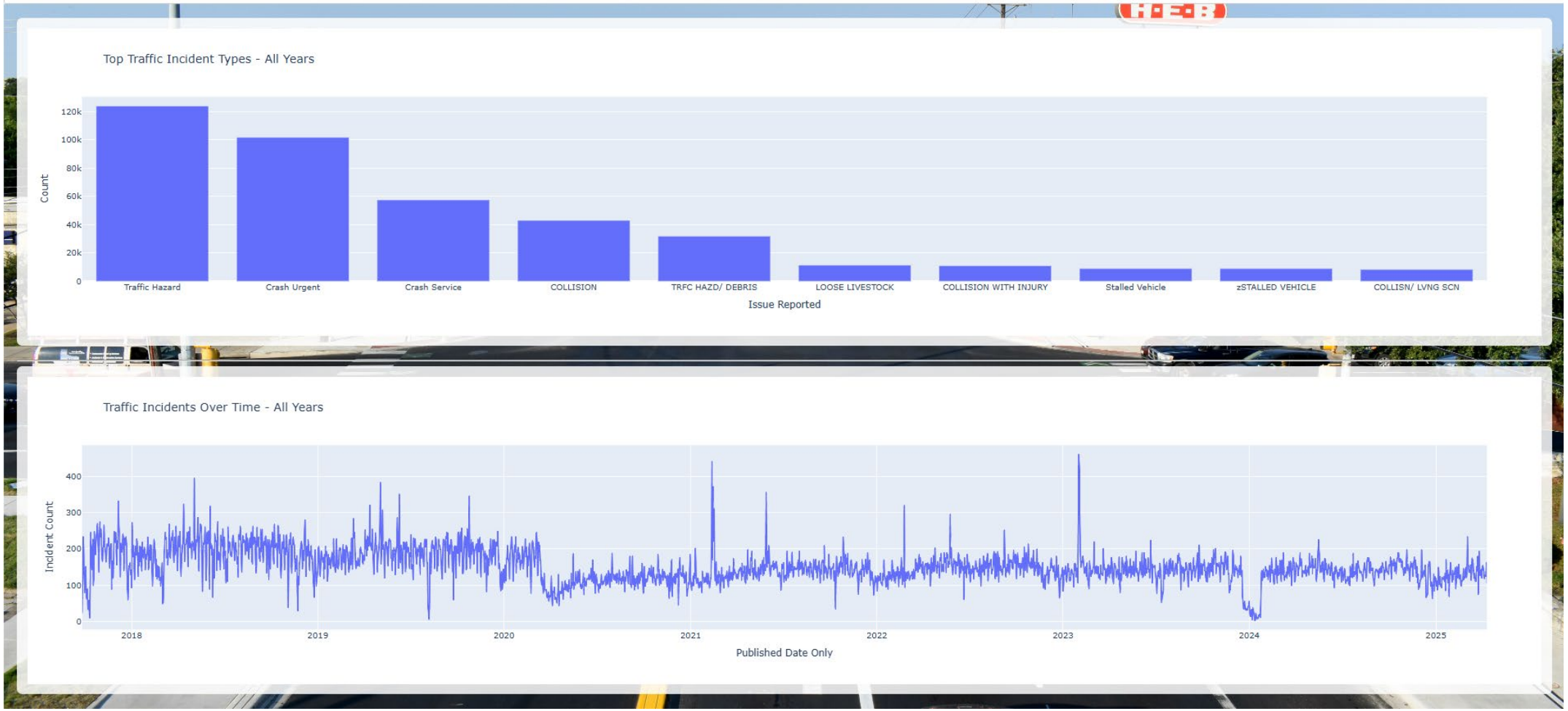
Python Powered Dashboards &
Open Data for Transparent Governance in Austin

Tanvi Sharma
Data Science
City of Austin

City of Austin Open Data Dashboard

PARD 311 | Traffic Incidents

All Years



Tanvi Sharma
Data Science
City of Austin

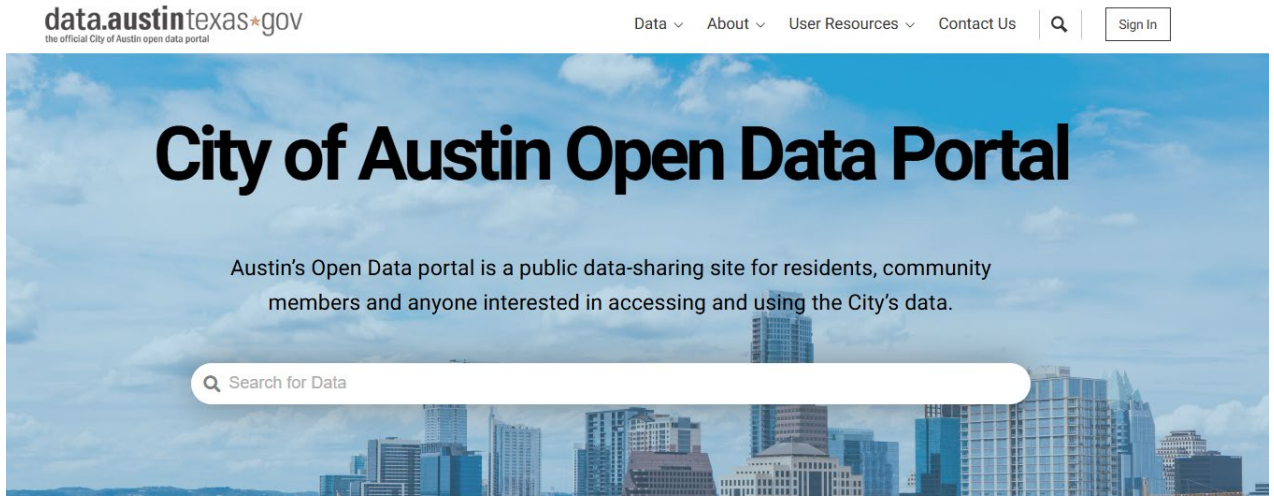
Why Choose Dash Over others?

Tool Comparison: Dash Vs Others

	Tool	Best For	Pros	Cons	
1	Jupyter Notebook	Exploratory data analysis, prototyping, documentation	Free, great for prototyping, rich ecosystem (pandas, matplotlib)	Not for production dashboards, limited interactivity	
2	Power BI	Quick drag-and-drop dashboards, integrates well with Microsoft stack	Easy to use, good sharing in MS ecosystem, real-time dashboarding	License cost, less customizable, Microsoft-centric	
3	Tableau	Beautiful dashboards, storytelling with data, enterprise BI	Visually stunning, enterprise-level governance & sharing	Expensive, less control over custom logic	
4	Dash	Fully customizable interactive dashboards in Python, shareable as web apps	Open-source, fully in Python, highly interactive, customizable UI	Requires Python knowledge, more setup needed	

Tanvi Sharma
Data Science
City of Austin

Conclusion & Call to Action



- Explore the power of open data in your community.
- Leverage Dash to create interactive, customized dashboards.
- Share insights and drive informed decision-making.

Democratizing Data

Python Powered Dashboards &
Open Data for Transparent Governance in Austin

Tanvi Sharma
Data Science
City of Austin