CPSC 471
T01-G05 Final Demo

December 2, 2025

Sam Safe, Aniket Bulusu, Sultan Alzoghaibi

Summary of Project

- Primary question: Trends in voting turnout and preferences based on ward-level characteristics in the city of Calgary.
- Data Driven Application
- Displaying a GeoJSON map with all wards & associated characteristics
- Users can explore how different ward characteristics influence either voter turnout or winners by selecting the specific factors they want to analyze.
- A meaningful data visualization will pop up and display the user's inquiries
  - Does an increase in crime rate have higher voter turnout?
  - Do wards with more recreational communities have a higher voter turnout?

Prominent Changes

- Convert relational schema to database schema .sql file
- Focus on voter turnout
- Create a loading script in Python that populates schema database in PostgreSQL
  - Use pgAdmin to do this through GUI as well
  - Data cleaned for easier access
- Implemented UI using the Dash framework
- Add visualizations via Plotly, Pandas, statsmodels
- Implement geospatial data visualization using Folium, GeoPandas, Shapely
- Use SQLAlchemy ORM to bake in functions for SQL queries
- Dockerized the entire project to simplify launch across platforms

Opportunities for Growth | Major Entities & Datasets

OPPORTUNITIES FOR GROWTH:

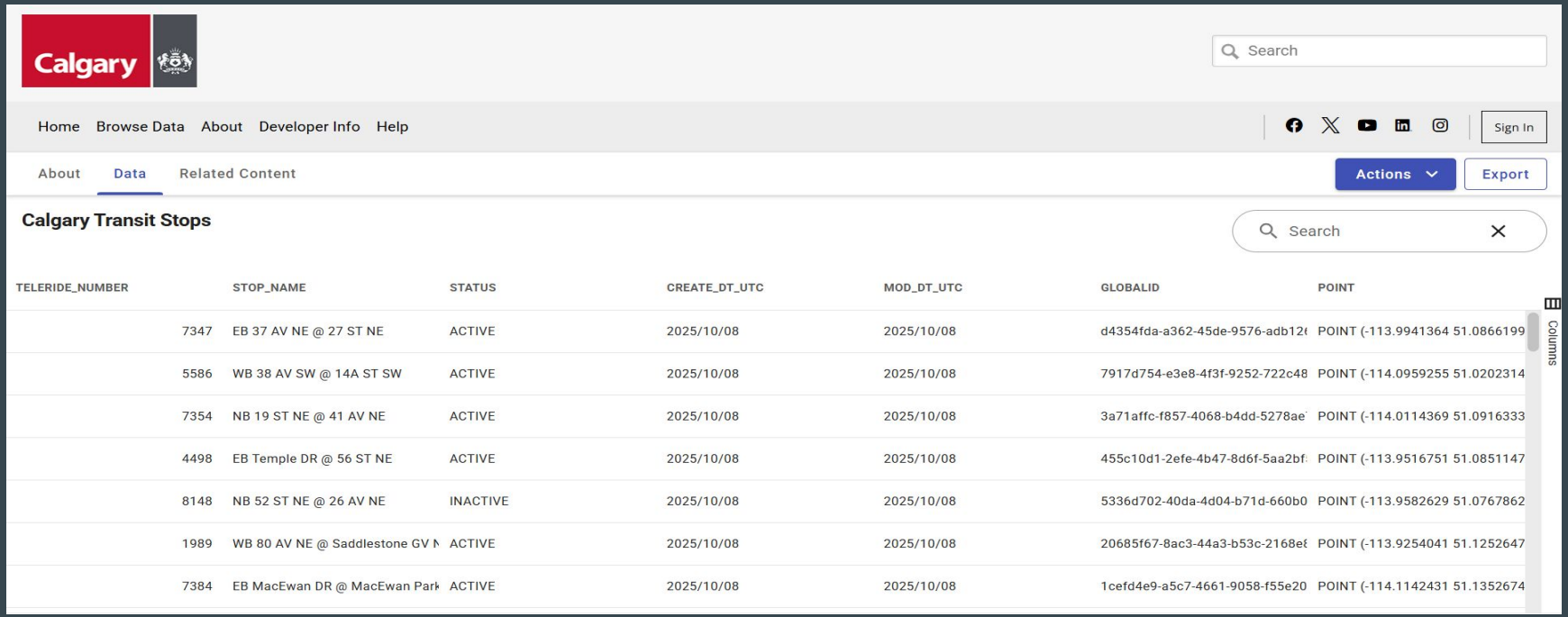
- Finding a meaningful way of presenting the data instead of just having several visualizations for the user to look at. We took the opportunity to regroup, re-discuss our thoughts of what the user would experience when looking through our project.
- Implemented a GeoJSON map to give users a pleasing visual to have as reference when filtering amongst wards
- Include a dedicated reference section with legible citations giving structure and format towards our final submission

MAJOR ENTITIES & DATASETS

- _Ward_Election_Results was our most important dataset as that contained all data regarding voting by ward and different types of candidates
- All datasets must contain Ward as that was the foreign key to join our tables and have the appropriate data for our visualizations
- The datasets from the milestone demo have not changed at all

Project Progress | Database

- Take Open Calgary Data Portal Data → Filter dataset for relevant data



TELERIDE_NUMBER	STOP_NAME	STATUS	CREATE_DT_UTC	MOD_DT_UTC	GLOBALID	POINT
7347	EB 37 AV NE @ 27 ST NE	ACTIVE	2025/10/08	2025/10/08	d4354fda-a362-45de-9576-adb12(	POINT (-113.9941364 51.0866199
5586	WB 38 AV SW @ 14A ST SW	ACTIVE	2025/10/08	2025/10/08	7917d754-e3e8-4f3f-9252-722c48	POINT (-114.0959255 51.0202314
7354	NB 19 ST NE @ 41 AV NE	ACTIVE	2025/10/08	2025/10/08	3a71affc-f857-4068-b4dd-5278ae	POINT (-114.0114369 51.0916333
4498	EB Temple DR @ 56 ST NE	ACTIVE	2025/10/08	2025/10/08	455c10d1-2efe-4b47-8d6f-5aa2bf	POINT (-113.9516751 51.0851147
8148	NB 52 ST NE @ 26 AV NE	INACTIVE	2025/10/08	2025/10/08	5336d702-40da-4d04-b71d-660b0	POINT (-113.9582629 51.0767862
1989	WB 80 AV NE @ Saddlestone GV N	ACTIVE	2025/10/08	2025/10/08	20685f67-8ac3-44a3-b53c-2168e(	POINT (-113.9254041 51.1252647
7384	EB MacEwan DR @ MacEwan Park	ACTIVE	2025/10/08	2025/10/08	1cefd4e9-a5c7-4661-9058-f55e20	POINT (-114.1142431 51.1352674

Project Progress | Database Cont.

- Do various data mining techniques → Filter through addresses of stops → Categorize by Ward, Active_Stops, Inactive_Stops, Total_Stops

ward_num	active_sto	inactive_st	total_stops
1	452	81	533
2	405	28	433
3	272	18	290
4	527	66	593
5	465	139	604
6	389	72	461
7	456	222	678
8	392	166	558
9	679	212	891
10	495	62	557
11	522	113	635
12	458	112	570
13	292	42	334
14	425	55	480

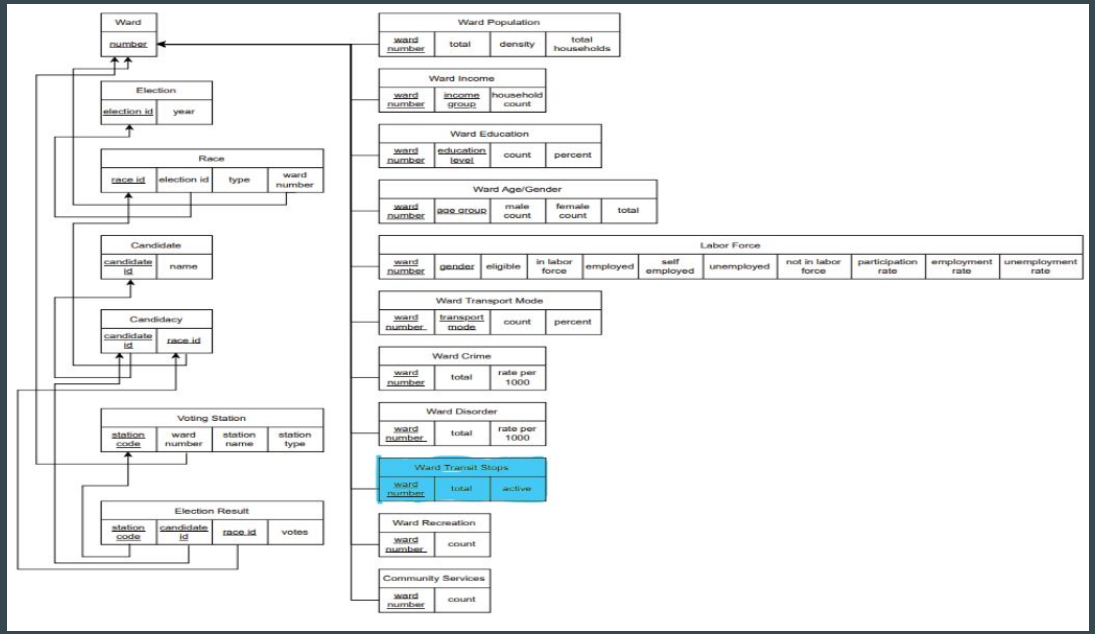
```sql
CREATE TABLE ward_transit_stops (
    ward_number INTEGER PRIMARY KEY REFERENCES ward(ward_number),
    total INTEGER NOT NULL,
    active INTEGER NOT NULL,
    inactive INTEGER
);
```

```python
# data standardiser
def load_csv(filename):
    df = pd.read_csv(DATA_DIR / filename)
    df.columns = (
        df.columns.str.strip()
        .str.lower()
        .str.replace(" ", "_")
        .str.replace("/", "_")
    )
    return df
```

```python
def load_ward_transit_stops(engine):
    print("Loading transport stops...")
    df = load_csv("_Ward_Transit_Stops.csv")
    transit_df = pd.DataFrame ({
        'ward_number': df['ward_num'],
        'total': df['total_stops'],
        'active': df['active_stops'],
        'inactive': df['inactive_stops']
    })
    transit_df.to_sql('ward_transit_stops', engine
```

Project Progress | Database Cont.

- Just like the other datasets, everything had gone through a similar process like this and translated to the database schema with Ward Number being a foreign key



Project Progress | Other Technical Aspects

- Use references for Dash and Plotly to build an aesthetic application frontend
- Dockerized containers for the project with automated entry point script following tutorials and online guides
- GeoJSON and data processing pipeline through pandas and WKT
- Interactive dashboard with Dash uses Pythonic callbacks to dynamically update visualizations

Supplemental Learning

- Connecting a Frontend (Dashboard) to our Backend (Data Visualizations & Queries) since when users are looking through our project, if they want to specifically filter data and look a particular ward characteristic with relation to voter turnout, the specific visualization and associated query must be called so the user can see the correct display of what we want to showcase. The progress made on this was definitely demanding as the data pipeline process had to go through various steps (CSV -> cleaned by pandas -> Postgres -> Dash queries using SQL). Furthermore, it was an invaluable learning experience on how data can be connected to a dashboard through different types of methods.
- Implementing a GeoJSON map for our frontend to give a user a reference of something to look at while trying to think of what type of data they want to select to visualize. The progress made on this was not too bad as it was a matter of having some sort of display as our main point of contention which was the Map of Calgary.

Team Dynamics

- We are communicating well making sure we kept tentative group deadlines so we have that much needed time to tweek any aspects of our project
- Regarding team meetings, sometimes we get in a little debuggle of how we want to move forward, but we always hear everyone's thoughts and opinions and find a general middle ground of how we want progress in our project
- We have learned that through peer evaluations, in-person meetups, or team meetings that we all have a voice and that if something is not quite how we envision, we make it vocal and heard so that we can correct those mistakes and have an even more fully fledged and successful project. This was always a very good aspect to have in our team as it always showed that we had an attention to detail in how our workflow was structured and ultimately produce a clean, structured, and successful project.

References

- NeuralNine. (2024, September 13). Build Data Apps in Python with Plotly Dash. YouTube. https://www.youtube.com/watch?v=pLU7ZLPhyX8
- Colgan, M. (2020, July 7). Developing and Deploying Data-Driven Apps. YouTube. https://www.youtube.com/watch?v=KreNYemH198
- Hello dash. Plotly. (n.d.). https://dash.plotly.com/layout
- Guides. Docker Documentation. (2024, December 20). https://docs.docker.com/guides/
- PostgreSQL 18.1 documentation. PostgreSQL Documentation. (2025, November 13). https://www.postgresql.org/docs/current/index.html
- SQLALCHEMY 2.0 documentation. SQLAlchemy Unified Tutorial - SQLAlchemy 2.0 Documentation. (n.d.). https://docs.sqlalchemy.org/en/20/tutorial/index.html
- mCoding. (2024, August 2). Docker Tutorial for Beginners. YouTube. https://www.youtube.com/watch?v=b0HMimUb4f0
- A practical introduction to databases — a practical introduction to databases. (n.d.). https://runestone.academy/ns/books/published/practical_db/index.html
- W3schools.com. W3Schools Online Web Tutorials. (n.d.-a). https://www.w3schools.com/postgresql/index.php
- Ark Coding. (n.d.). Streamlit vs Dash - Which one is better? Interactive Dashboard with Python. YouTube. https://www.youtube.com/watch?v=tXHXDRog37
- W3Schools. (n.d.). *Matplotlib Pyplot*. Www.w3schools.com. https://www.w3schools.com/python/matplotlib_pyplot.asp
- *NumPy Tutorial*. (n.d.). Www.w3schools.com. https://www.w3schools.com/python/numpy/default.asp