North Korea

Various maps of the North Korea

PROJECT WORK

Data Analysis course of the PTE TTK

Király Márk

The content of the presentation

PART 1: Brief description of the shapefiles	[QGIS]
PART 2: Sparse SQL queries	[PostGIS]
PART 3: Transferring the CSV dataset to PostgreSQL with Python and visualizing the results in the QGIS application	[Python, PostGIS, QGIS]
PART 4: Analyzing a CSV file with Pandas and creating various plots	[Python, Pandas, NumPy]

The datasets, files, and source codes related to the presentations are all available in my GitHub repository at the following link:
https://github.com/ProgrammerGnome/data-analysis-presentation

PART 1

Brief description of the shapefiles

- **Categories of Layers:**
  - Points (stations, settlements)
  - Lines (roads, train tracks)
  - Polygons (buildings, borders, cities)

- **Visual Representation Key:**
  - Red dot: Railway stations
  - Green dot: Settlements
  - Gray line: Roads
  - Red line: Train tracks
  - Gray filled shape: Buildings
  - Light green filled shape: Cities
  - Another colors filled shape: Borders (admin levels 2, 1, 0)

The sources of the dataset:

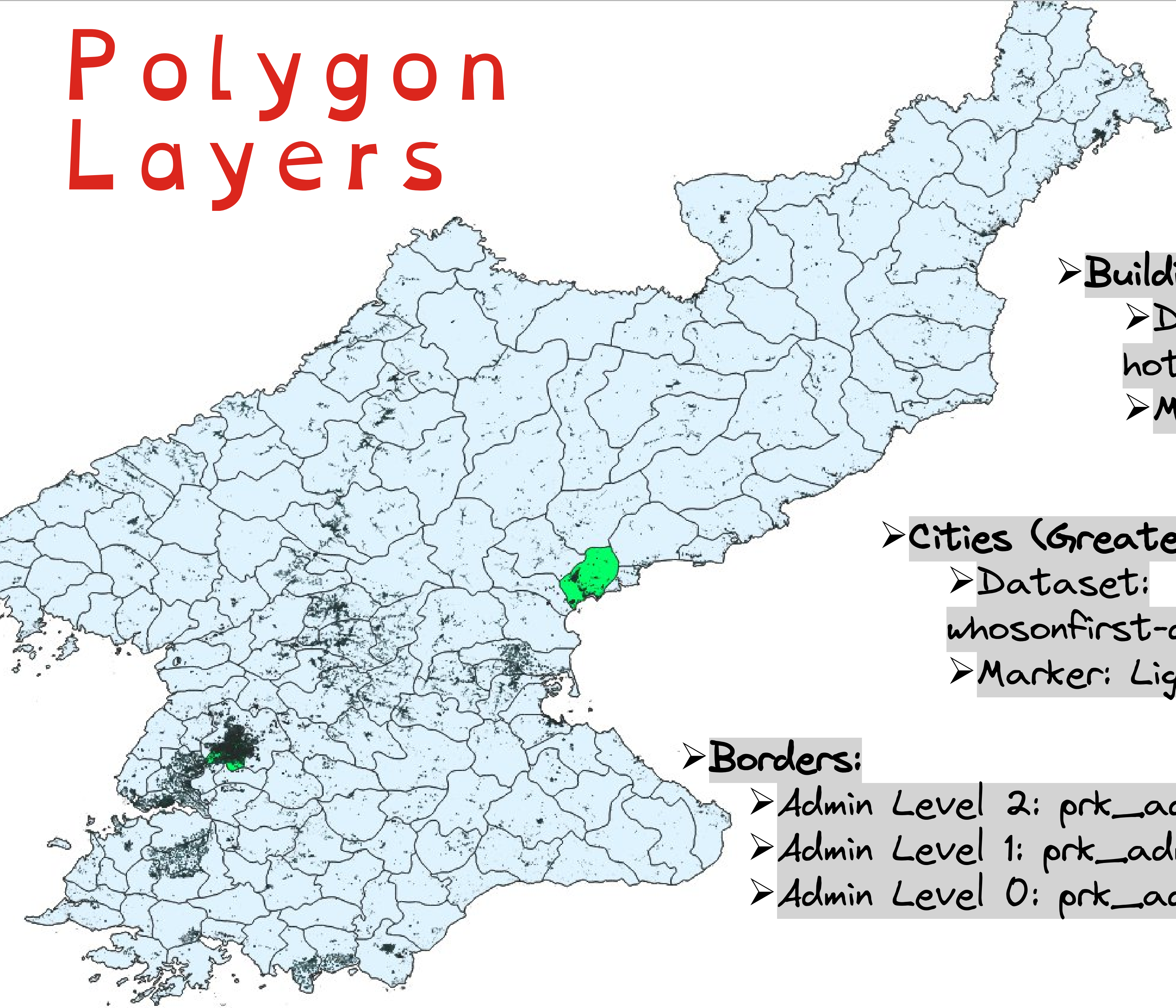
PostGIS layers:

https://data.humdata.org/group/prk?ext_geodata=1&q=&sort=last_modified%20desc&ext_page_size=25

CSV:

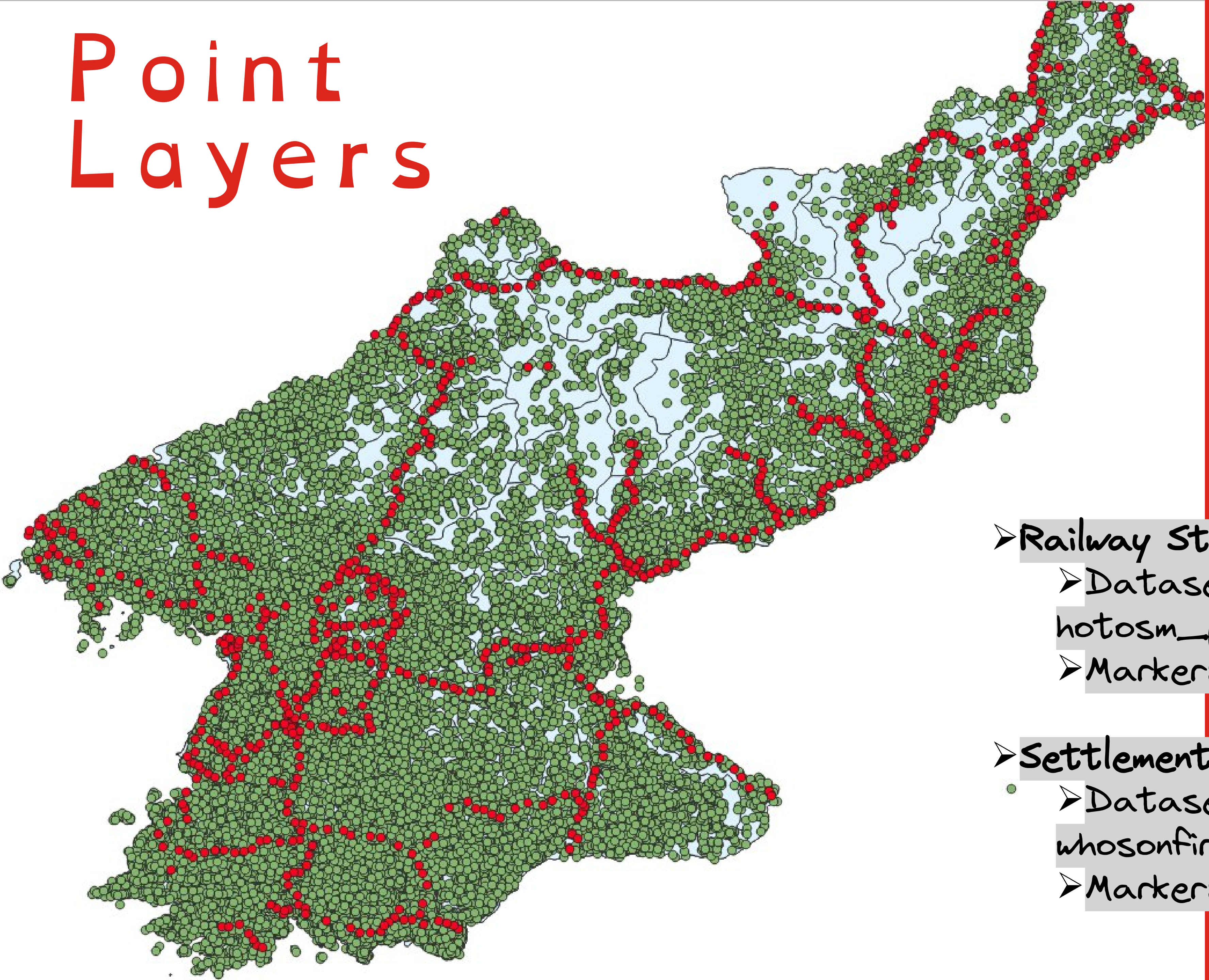
https://www.kaggle.com/datasets/fanbyprinciple/north-korea-missile-test-database

Polygon Layers



- **Buildings:**
  - Dataset: hotosm_prk_buildings_polygons_shp
  - Marker: Gray filled shape
- **Cities (Greatest):**
  - Dataset: whosonfirst-data-admin-kp-locality
  - Marker: Light green filled shape
- **Borders:**
  - Admin Level 2: prk_admbnda_adm2_wfp_20190624
  - Admin Level 1: prk_admbnda_adm1_wfp_20190624
  - Admin Level 0: prk_admbnda_adm0_wfp_20190624

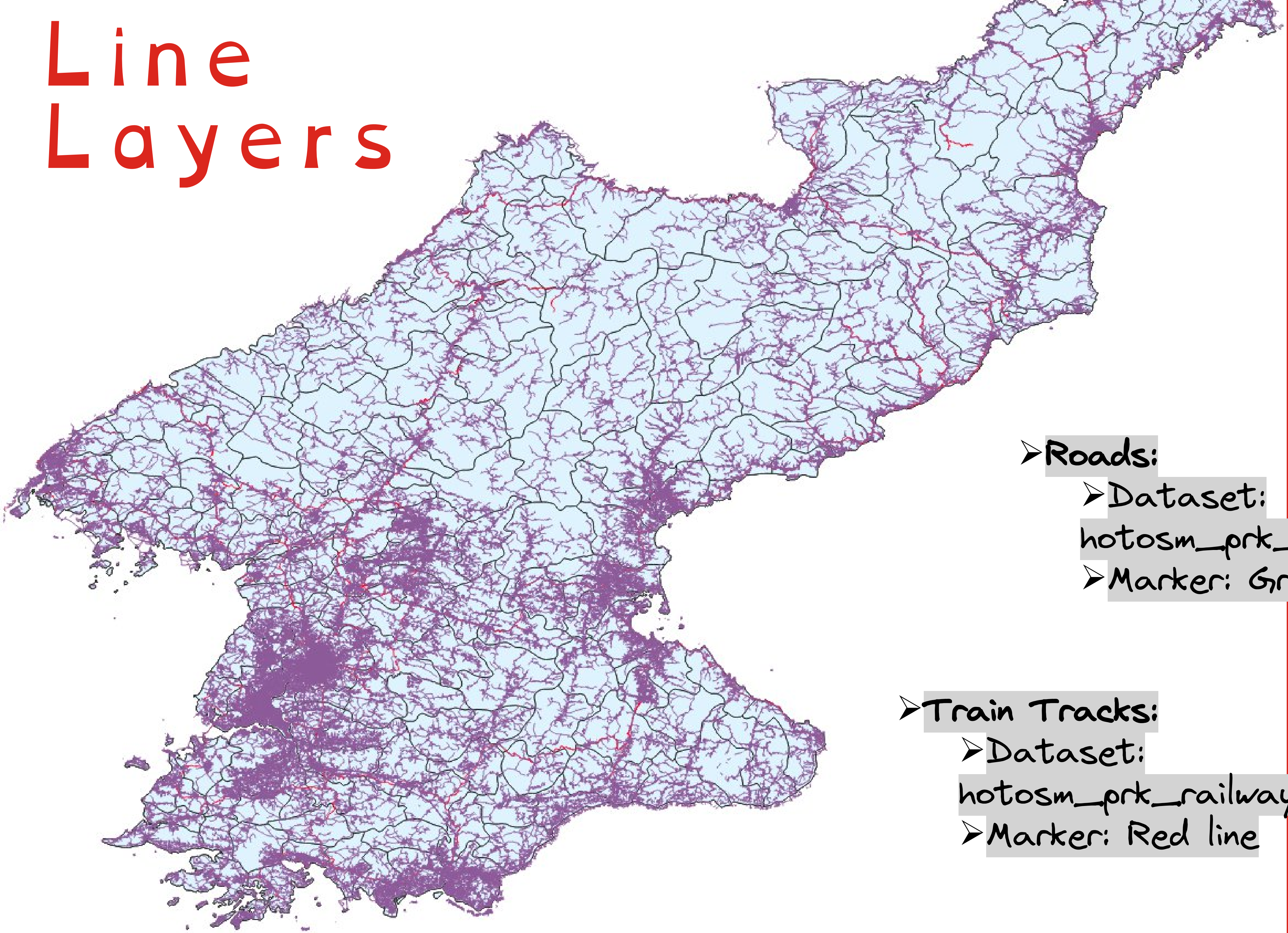
Point Layers



- **Railway Stations:**
  - Dataset: hotosm_prk_railways_points_shp
  - Marker: Red dot

- **Settlements:**
  - Dataset: whosonfirst-data-admin-kp-locality
  - Marker: Green dot

Line Layers



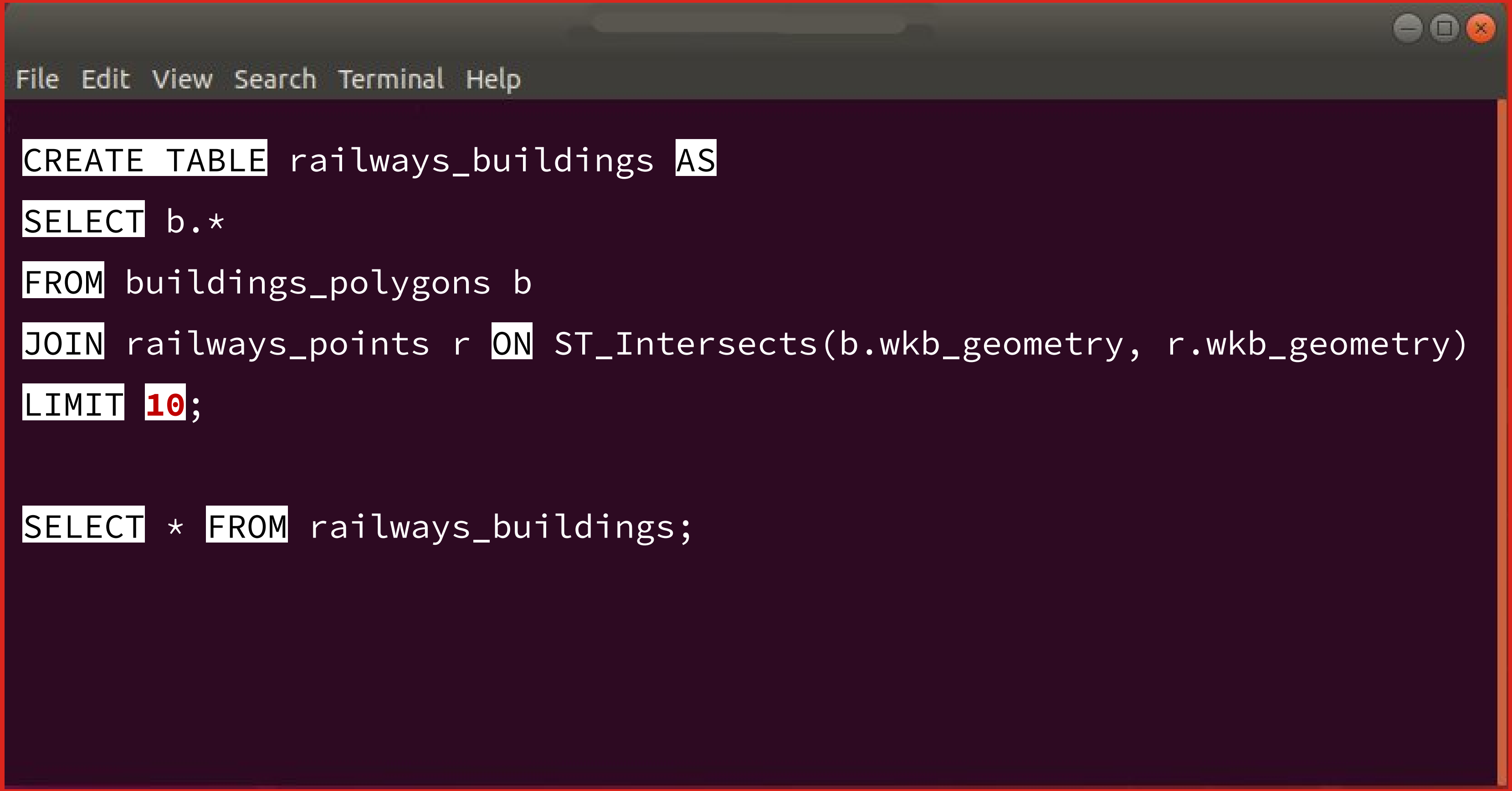
- **Roads:**
  - Dataset: hotosm_prk_roads_lines_shp
  - Marker: Gray line

- **Train Tracks:**
  - Dataset: hotosm_prk_railways_lines_shp
  - Marker: Red line

PART 2

Sparse SQL queries

-- Find 10 building polygons that intersect with railway points (railway stations)
-- and copy their data into a new table (railways_buildings)

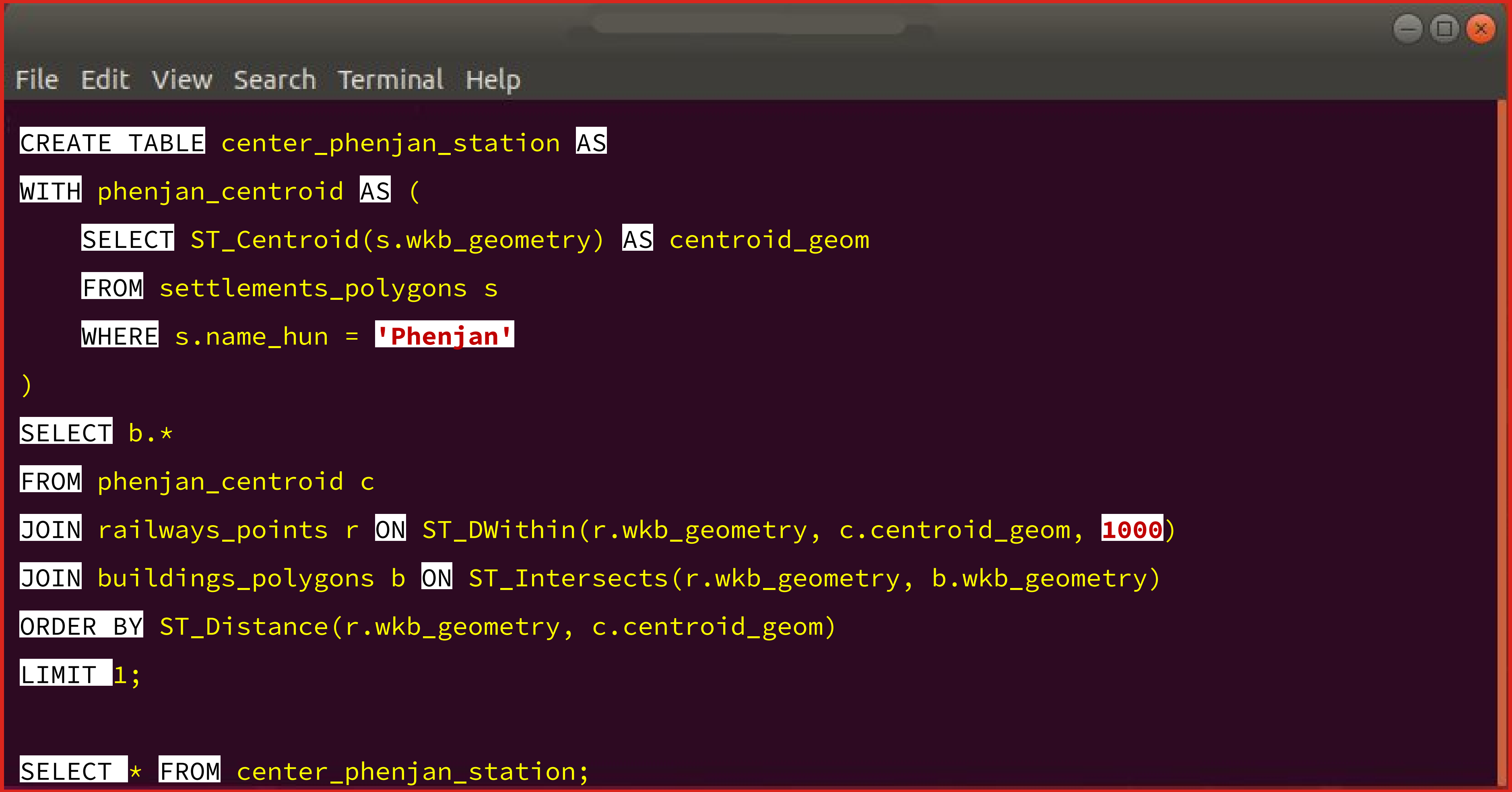


```
CREATE TABLE railways_buildings AS
SELECT b.*
FROM buildings_polygons b
JOIN railways_points r ON ST_Intersects(b.wkb_geometry, r.wkb_geometry)
LIMIT 10;

SELECT * FROM railways_buildings;
```

PART 2 - Sparse SQL queries

-- Find the railway station located near the centroid of the Pyongyang polygon (settlements_polygons.name_hun),
-- then retrieve the building_polygon related to that station, and copy the data into a new table (center_phenjan_station)
-- Search for the railway station and building near the centroid of the Pyongyang polygon

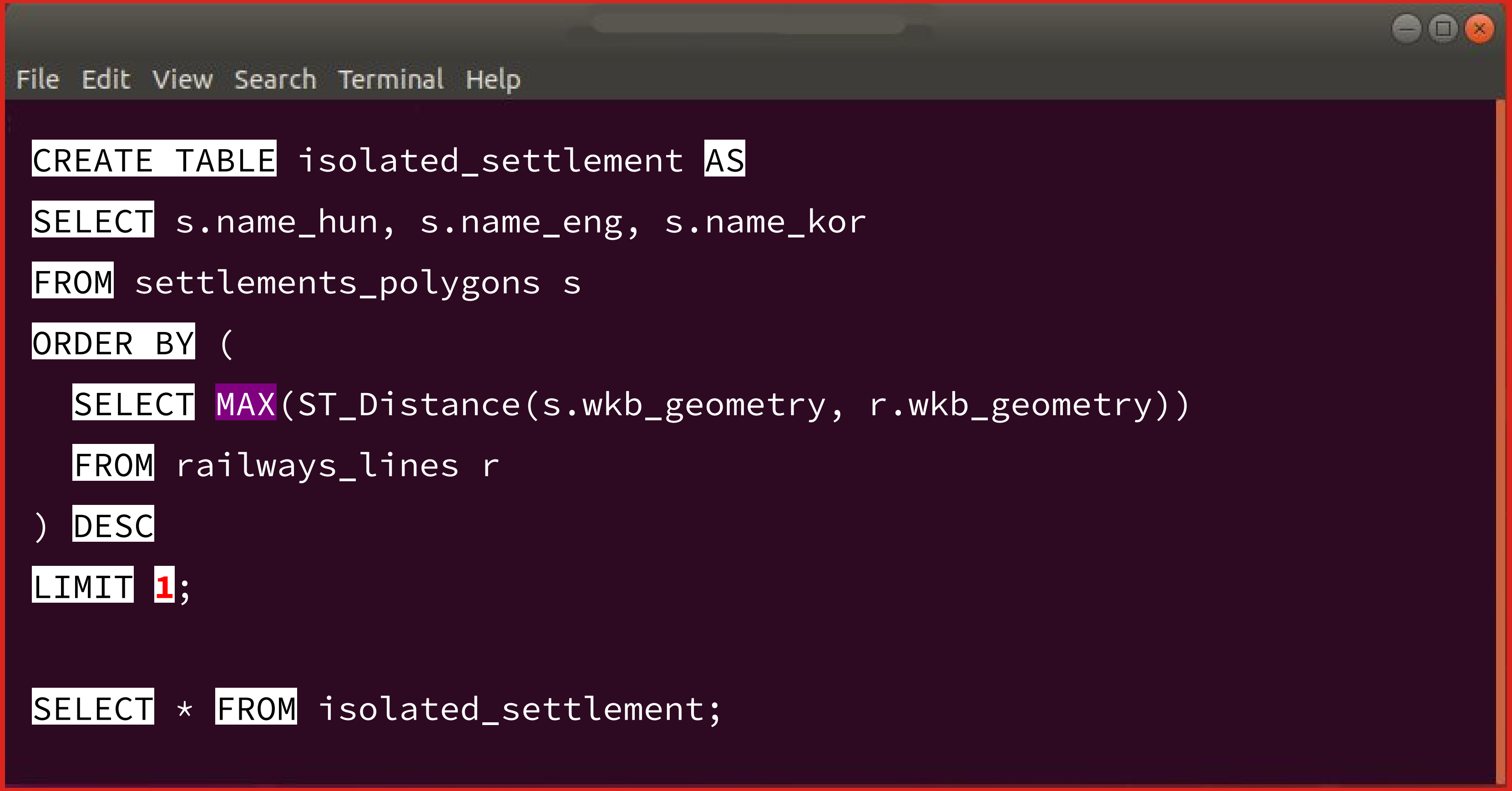


```
CREATE TABLE center_phenjan_station AS
WITH phenjan_centroid AS (
    SELECT ST_Centroid(s.wkb_geometry) AS centroid_geom
    FROM settlements_polygons s
    WHERE s.name_hun = 'Phenjan'
)
SELECT b.*
FROM phenjan_centroid c
JOIN railways_points r ON ST_DWithin(r.wkb_geometry, c.centroid_geom, 1000)
JOIN buildings_polygons b ON ST_Intersects(r.wkb_geometry, b.wkb_geometry)
ORDER BY ST_Distance(r.wkb_geometry, c.centroid_geom)
LIMIT 1;

SELECT * FROM center_phenjan_station;
```

PART 2 - Sparse SQL queries

-- Find the settlement (only its Hungarian, English, and Korean names)
that is the farthest from the railways_lines
-- and save the data into a new table (isolated_settlement)

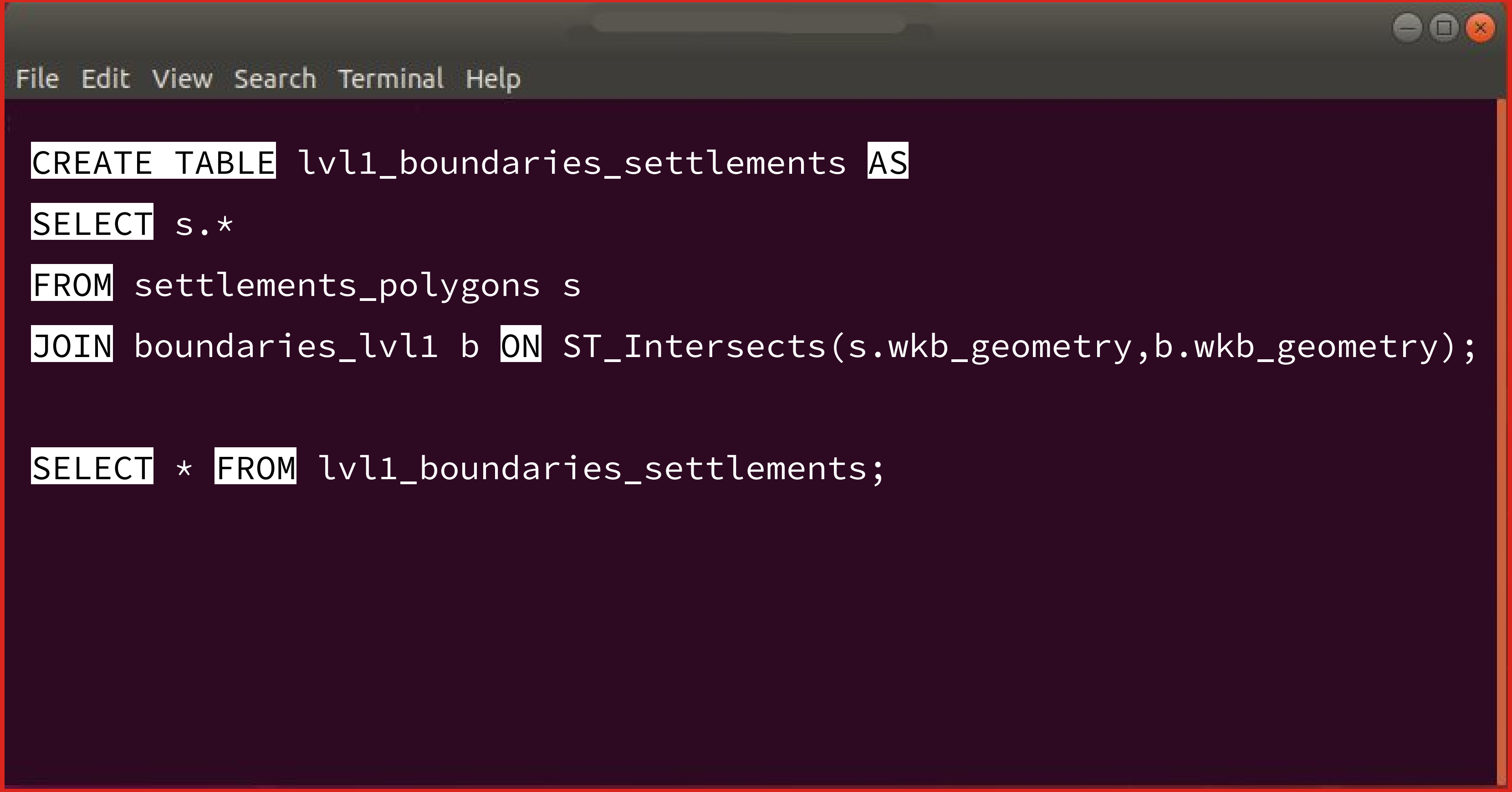


```
CREATE TABLE isolated_settlement AS
SELECT s.name_hun, s.name_eng, s.name_kor
FROM settlements_polygons s
ORDER BY (
  SELECT MAX(ST_Distance(s.wkb_geometry, r.wkb_geometry))
  FROM railways_lines r
) DESC
LIMIT 1;

SELECT * FROM isolated_settlement;
```

PART 2 - Sparse SQL queries

-- List all settlements that intersect with the boundaries_lvl1 line and save them into a new table (lvl1_boundaries_settlements)



```
CREATE TABLE lvl1_boundaries_settlements AS
SELECT s.*
FROM settlements_polygons s
JOIN boundaries_lvl1 b ON ST_Intersects(s.wkb_geometry,b.wkb_geometry);

SELECT * FROM lvl1_boundaries_settlements;
```

PART 3

Transferring the CSV dataset to PostgreSQL with Python and visualizing the results in the QGIS application

In Python, we extract the settlements names coordinates (and names) a CSV file and save them into a new table in a PostgreSQL database.

Finally, we display them using QGIS.

EXPERIMENT:

Hypothesis: These facilities must have a railway station or at least a railway line nearby.

Using Python, we extract data from the tables associated with the layers to determine:

Whether there is a station within a 10 km radius of the given settlement.

Whether there is a railway track within a 10 km radius.

We then combine and visualize these results in QGIS!

PART 3

Transferring the CSV dataset to PostgreSQL with Python and visualizing the results in the QGIS application

The Python code used to process the CSV file and load it into the database.

https://github.com/ProgrammerGnome/data-analysis-presentation/blob/main/DataAnalysis_project_Mark.py

Using the following SQL queries and table creation scripts can I implement the filtering mentioned on the previous slide.

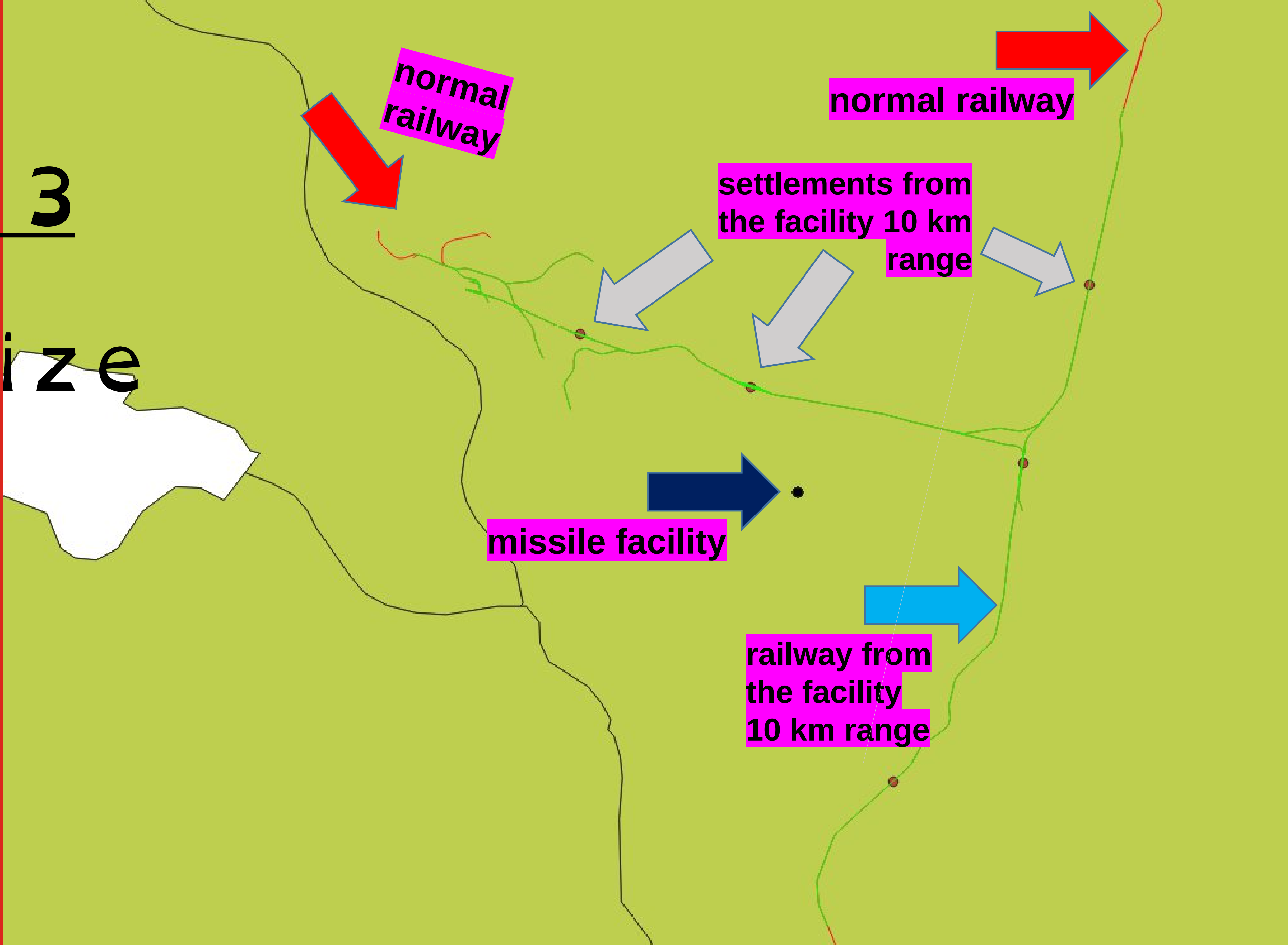
https://github.com/ProgrammerGnome/data-analysis-presentation/blob/main/SQL-scripts/facility_near_railway_buildings.sql (not working ?=> data issue?)

https://github.com/ProgrammerGnome/data-analysis presentation/blob/main/SQL-scripts/facility_near_railway_points.sql

https://github.com/ProgrammerGnome/data-analysis-presentation/blob/main/SQL-scripts/facility_near_railway_lines.sql

PART 3

Visualize the result in QGIS


normal railway
normal railway
settlements from the facility 10 km range
missile facility
railway from the facility 10 km range

PART 4

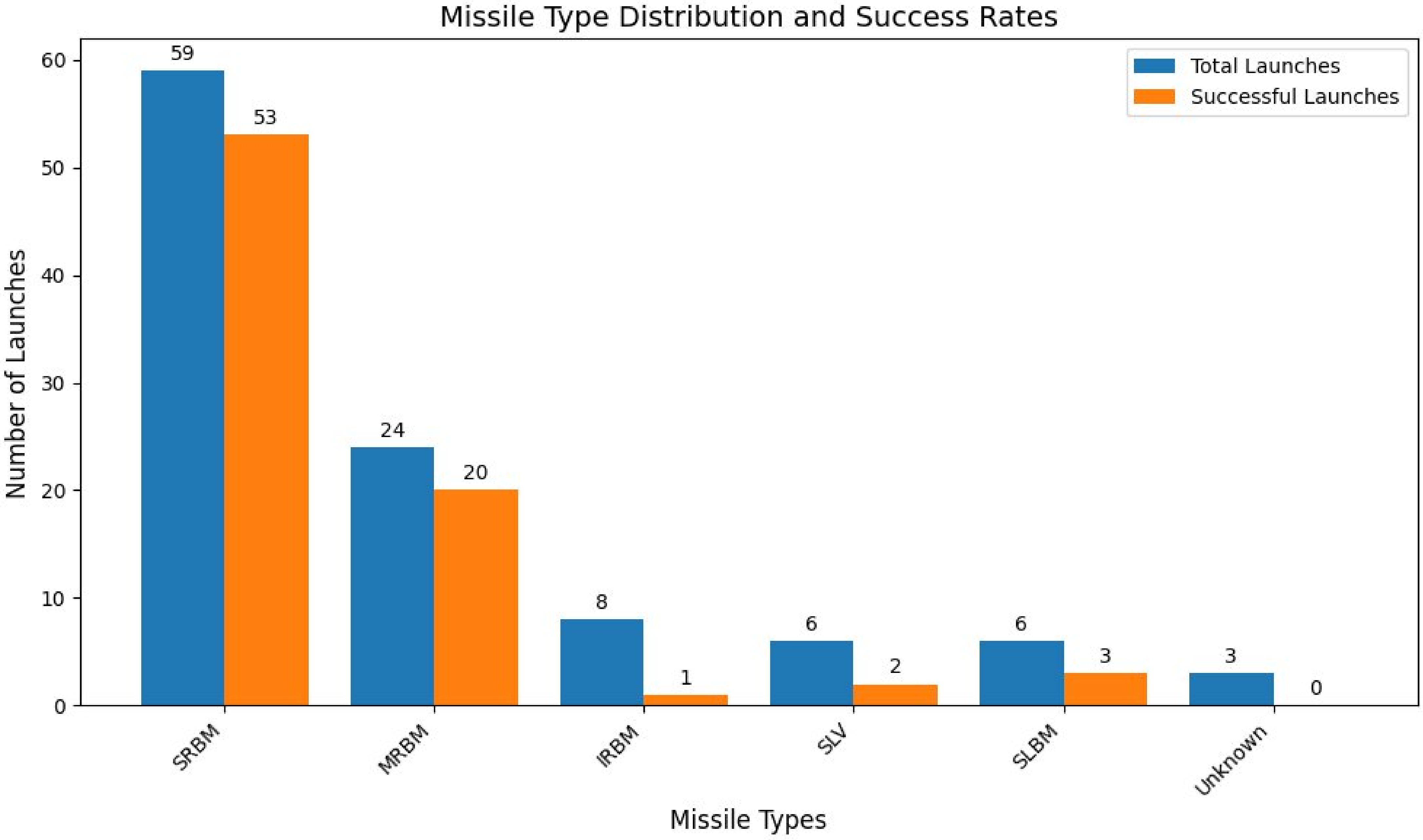
Analyze the CSV with Pandas

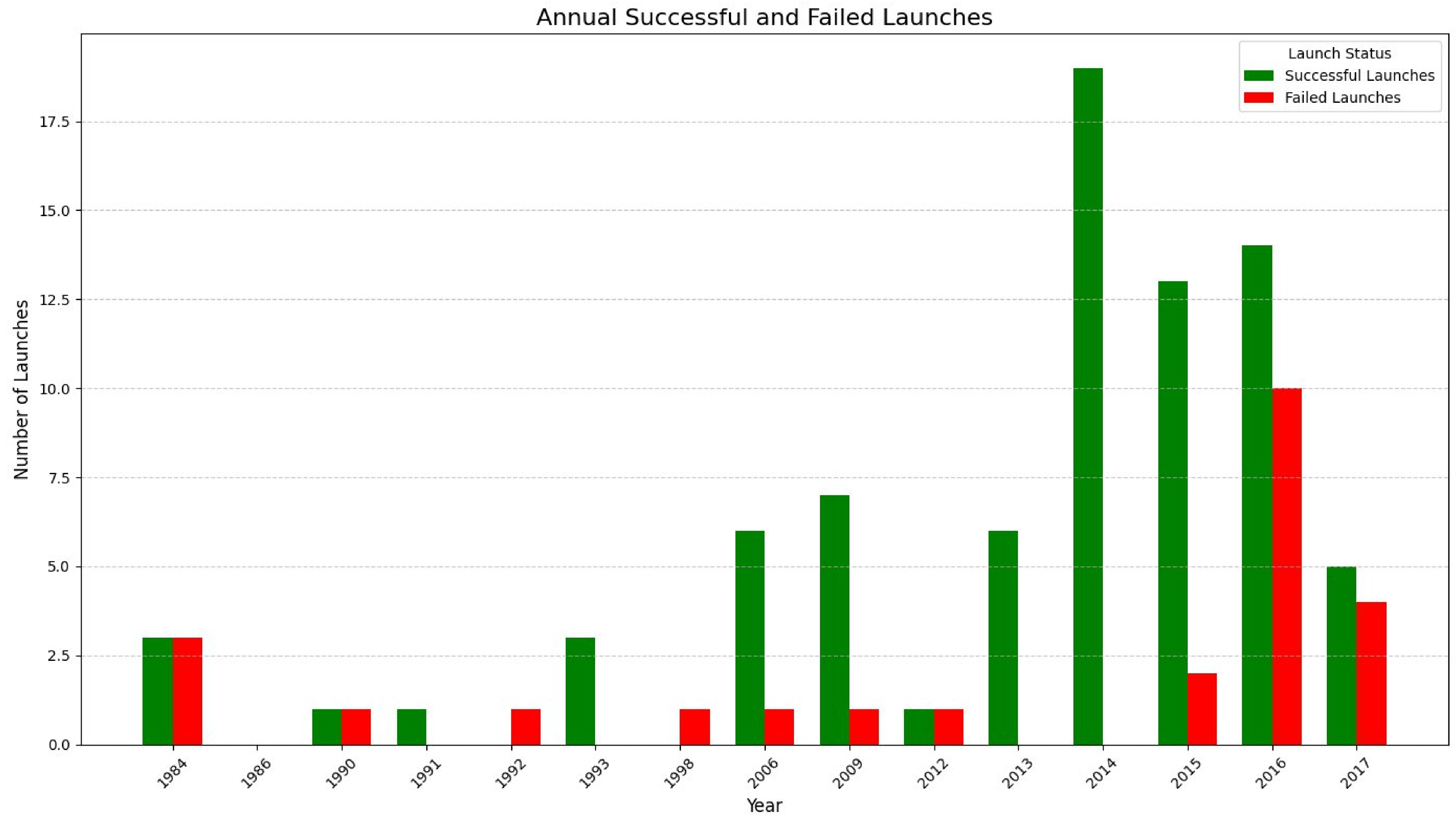
In the previous slides, we discussed missile facilities.
While we're at it, let's explore a few additional
interesting aspects related to them.

Using Pandas, we will now create a few charts based on
the CSV data inserted into PostgreSQL.

The source code:

https://github.com/ProgrammerGnome/data-analysis-presentation/blob/main/DataAnalysis_project_Mark.ipynb


Missile Type Distribution and Success Rates
Total Launches
Successful Launches
Number of Launches
0
10
20
30
40
50
60
59
53
24
20
8
1
6
2
6
3
3
0
SRBM
MRBM
IRBM
SLV
SLBM
Unknown
Missile Types


Annual Successful and Failed Launches
Launch Status
🟩 Successful Launches
🟥 Failed Launches
Number of Launches
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Year
1984
1986
1990
1991
1992
1993
1998
2006
2009
2012
2013
2014
2015
2016
2017

Thank you for your attention!