Riyadh Restaurants Project
Data Engineering & Preprocessing Report

Data Engineer: Yazeed Almuhlaki

December 6, 2025

1 Project Context and Roles

This project analyzes a large point-of-interest dataset of restaurants in Riyadh. The team is divided into four roles:

- **Member 1 – Data Engineer (this report)**: Download, clean, and prepare the data; create a sample district subset; deliver clean files and documentation.
- **Member 2 – Spatial Analyst**: Use GeoPandas / GIS tools to create maps, spatial queries, and spatial statistics.
- **Member 3 – Business Analyst**: Perform non-spatial analysis (top cuisines, price vs. rating, etc.).
- **Member 4 – Dashboard Developer**: Integrate all outputs into a Streamlit / web dashboard and prepare the conclusion view.

This document describes only the work of the **Data Engineer** and provides the cleaned datasets and technical details needed by the other three roles.

2 Dataset Description

2.1 Source

- **Name:** Riyadh Restaurants (20K)
- **Origin:** Points of interest collected from Foursquare and shared as a public dataset on Kaggle.
- **Unit of observation:** One row per restaurant / cafe in Riyadh.

The raw dataset was downloaded once from Kaggle and stored as an untouched copy under `data/raw/` (file `riyadh_resturants_raw.csv`).

2.2 Main Columns

After inspection in the exploratory notebook (`01_explore_raw.ipynb`), the key columns used in the project are:

Column	Type	Description
name	String	Restaurant name (Arabic or English).
categories	String	Comma-separated list of categories (e.g., “Coffee Shop, Bakery”).
address	String	Informal address / location description.
lat	Numeric	Latitude (WGS 84, EPSG:4326).
lng	Numeric	Longitude (WGS 84, EPSG:4326).
price	Categorical (string)	Price level (e.g., cheap / moderate / expensive), as provided by the source.
likes	Numeric	Number of likes on Foursquare.
photos	Integer	Number of photos associated with the venue.
tips	Integer	Number of text tips / comments (count only).
rating	Numeric	Average user rating (0–10 scale).
ratingSignals	Numeric	Number of users who rated the place (often missing).

These columns are preserved in all cleaned outputs so that each team member can directly use them.

3 Project Folder Structure

All work is organized under a project root directory:

```
riyadh-restaurants-project/
  data/
    raw/
      riyadh_resturants_raw.csv
    clean/
      riyadh_restaurants_clean.csv
      riyadh_restaurants_clean.geojson
    boundaries/
      districts_sample_200.geojson
      districts_EPSG4326.geojson
    sample_district/
      restaurants_sample_district.geojson
      restaurants_sample_district.csv
  notebooks/
    01_explore_raw.ipynb        # initial exploration of raw CSV
    02_clean_data.ipynb         # full cleaning pipeline
    03_export_geojson.ipynb     # export cleaned CSV to GeoJSON (points)
  docs/
    (this LaTeX report and related documentation)
```

4 Data Engineering Workflow Overview

The main responsibilities of the Data Engineer are:

1. Keep an untouched raw copy of the Kaggle dataset.
2. Perform cleaning and basic validation:
   - Ensure numeric and valid coordinates (`lat`, `lng`),
   - Remove duplicates,
   - Handle missing values in `price`, `rating`, and `likes`,
   - Apply light text cleaning where needed.
3. Export the cleaned data in both CSV and GeoJSON formats.
4. Use GDAL to clip the cleaned restaurants to a study region and create a *Sample District* subset.
5. Document all steps so that Spatial, Business, and Dashboard members can trust and reuse the data.

Raw exploration is implemented in `01_explore_raw.ipynb`, the cleaning logic in `02_clean_data.ipynb`, and the GeoJSON export in `03_export_geojson.ipynb`. The GDAL commands for clipping are executed from the terminal and documented in this report.

5 Raw vs. Cleaned Data

5.1 Row Counts

- **Raw dataset:** 19 361 rows.
- **Cleaned dataset:** 19 355 rows.
- **Difference:** 6 rows removed in total as a result of the cleaning operations (duplicate removal and basic quality checks).
- **Sample district subset:** 1 804 restaurants inside the study region.

The cleaned dataset keeps almost all information while removing a very small number of problematic or duplicate records.

5.2 Missing Values Summary (After Cleaning)

After the cleaning pipeline was applied, the missing value counts per column in the cleaned dataset (`df_clean`) are:

Column	Missing values
name	0
categories	0
address	0
lat	0
lng	0
price	0
likes	0
photos	0
tips	0
rating	0
ratingSignals	11 409

- All columns required by the project description (`price`, `rating`, and coordinates) have **no missing values**.
- `ratingSignals` is missing for many restaurants; this reflects limitations of the original source and was intentionally *not* imputed.

6 Detailed Cleaning Steps

This section describes the exact decisions taken in the cleaning pipeline, matching the code in `02_clean_data.ipynb`. These details are important for reproducibility and for the other team members.

6.1 Coordinate Columns (`lat` and `lng`)

In the cleaning notebook, the latitude and longitude columns are explicitly converted to numeric:

```
df_clean["lat"] = pd.to_numeric(df_clean["lat"], errors="coerce")
df_clean["lng"] = pd.to_numeric(df_clean["lng"], errors="coerce")
```

- This ensures that any unexpected non-numeric values are turned into `NaN`.
- In this dataset, there are no missing values in `lat` or `lng` after conversion, so no additional rows needed to be dropped because of invalid coordinates.
- The coordinates are interpreted as WGS 84 (EPSG:4326), which is standard for Foursquare and web maps.

6.2 Duplicate Removal

To avoid counting the same restaurant multiple times, duplicates were addressed in two steps:

1. **Full-row duplicates:**

   ```
   df_clean = df_clean.drop_duplicates()
   ```

2. **Restaurant identity duplicates:**

   ```
   df_clean = df_clean.drop_duplicates(subset=["name", "lat", "lng"])
   ```

   This ensures that each physical restaurant (same name at the same coordinates) appears at most once.

Together with the other cleaning steps, these operations account for the 6 rows removed between the raw and cleaned datasets.

6.3 Handling Missing price

The price column represents a price level category. In the cleaning notebook, the following steps were applied:

1. **Normalization and case:**

   ```
   df_clean["price"] = df_clean["price"].str.strip().str.lower()
   ```

   This removes extra whitespace and stores price levels in lowercase for consistency (for example, "Cheap" and "cheap" become "cheap").

2. **Converting placeholder strings to NaN:**

   ```
   df_clean["price"] = df_clean["price"].replace("nan", np.nan)
   ```

   If the literal string "nan" appears, it is treated as a missing value.

3. **Missing value imputation:**

   ```
   price_mode = df["price"].mode()[0]
   df_clean["price"] = df_clean["price"].fillna(price_mode)
   ```

   Missing values in `price` are filled using the **global mode** computed from the original dataset (`df["price"]`), i.e. the most frequent price level overall.

After these steps, the `price` column contains no missing values.

6.4 Handling Missing rating

The `rating` column stores the average user rating on a 0–10 scale. In the exploratory notebook it was confirmed that `rating` already had the correct numeric type, so no explicit conversion (such as `to_numeric`) was required.

Missing values in `rating` were handled as follows:

1. **Compute the mean rating:**

```
rating_mean = df_clean["rating"].mean()
```

2. **Impute missing values:**

   ```
   df_clean["rating"] = df_clean["rating"].fillna(rating_mean)
   ```

   All missing ratings are replaced by the overall mean rating in the cleaned dataset.

As a result, the `rating` column has no missing values and is ready for statistical analysis by the Business Analyst.

6.5 Handling `likes`, `photos`, `tips`, and `ratingSignals`

- **`likes`:** Any missing values in `likes` were filled with `0`:

  ```
  df_clean["likes"] = df_clean["likes"].fillna(0)
  ```

  This treats missing likes as "no recorded likes" rather than removing the rows.

- **`photos` and `tips`:** These columns did not contain missing values in the cleaned dataset and were left as-is.

- **`ratingSignals`:** This column is missing for many restaurants (11 409 rows). Instead of guessing the number of raters, the missing values were **left as `NaN`**. The Business Analyst can choose how to handle this (for example, filter out missing values when using this column).

6.6 Text Normalization

Light text cleaning was applied to make string columns more consistent without changing their meaning. In the cleaning notebook the following code was used:

```
for col in ["name", "categories", "address"]:
    if col in df_clean.columns:
        df_clean[col] = df_clean[col].str.strip()
```

- Leading and trailing whitespace is removed from `name`, `categories`, and `address`.

- The original capitalization and internal punctuation (including apostrophes such as in "o'reilly" or "mcdonald's") are preserved.

No advanced text processing (tokenization, stemming, etc.) was done at the Data Engineering stage.

7 Exporting Clean Data for the Team

After the cleaning steps, the main dataset was saved in two formats.

7.1 Clean CSV

- **Path:** `data/clean/riyadh_restaurants_clean.csv`
- **Content:** 19 355 cleaned rows, with all columns described above.
- **Usage:** Recommended for the Business Analyst (non-spatial analysis) and Dashboard Developer (tables, charts).

7.2 Clean GeoJSON (Points)

To support spatial work, the cleaned CSV was converted to a GeoJSON file in `03_export_geojson.ipynb` by creating a point geometry from `lng` and `lat`. The relevant code is:

```
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point

df_clean = pd.read_csv("../data/clean/riyadh_restaurants_clean.csv")
geometry = [Point(xy) for xy in zip(df_clean["lng"], df_clean["lat"])]
gdf_rest = gpd.GeoDataFrame(
    df_clean,
    geometry=geometry,
    crs="EPSG:4326"
)
gdf_rest.to_file(
    "../data/clean/riyadh_restaurants_clean.geojson",
    driver="GeoJSON"
)
```

- **Path:** `data/clean/riyadh_restaurants_clean.geojson`
- **Content:** Same attributes as the CSV, plus a point geometry for each restaurant (CRS: EPSG:4326).
- **Usage:** Recommended for the Spatial Analyst (GeoPandas, QGIS, web maps) and for map-based visualizations in the dashboard.

8 Sample District Extraction Using GDAL

The project requirements include working on a focused “Sample District” or study region. A district boundary layer was provided as a GeoJSON file:

- `data/boundaries/districts_sample_200.geojson`

The clipping step is performed using GDAL command line tools (`ogr2ogr`) rather than a separate notebook; the commands are documented here so that any team member can reproduce the process.

8.1 District Boundaries Preparation

To match the restaurant coordinates, the districts were reprojected to WGS 84 (EPSG:4326) using `ogr2ogr`:

```
ogr2ogr -t_srs EPSG:4326 \
  data/boundaries/districts_EPSG4326.geojson \
  data/boundaries/districts_sample_200.geojson
```

This ensures that both the district polygons and restaurant points share the same coordinate reference system.

8.2 Clipping Restaurants to the Study Region

The cleaned restaurants GeoJSON was then clipped to the district polygons using `ogr2ogr` with `-clipsrc`. The command (run from the project root) is:

```
ogr2ogr -f "GeoJSON" \
  data/sample_district/restaurants_sample_district.geojson \
  data/clean/riyadh_restaurants_clean.geojson \
  -clipsrc data/boundaries/districts_EPSG4326.geojson
```

This operation keeps only those restaurant points that lie inside the polygon layer defined by `districts_EPSG4326.geojson`. The resulting layer has:

- **Geometry:** Point
- **CRS:** EPSG:4326 (WGS 84)
- **Feature count:** 1 804 restaurants

For convenience, the sample district dataset was also exported to CSV:

```
ogr2ogr -f "CSV" \
  data/sample_district/restaurants_sample_district.csv \
  data/sample_district/restaurants_sample_district.geojson
```

8.3 Interpretation

From the project perspective:

- The **full cleaned dataset** (19 355 rows) represents restaurants across the overall Riyadh area.
- The **sample district subset** (1 804 rows) represents restaurants inside the polygon region defined by the provided district boundaries. This subset serves as a focused study area for:
  - Spatial visualizations (Member 2),
  - Local non-spatial statistics (Member 3),
  - Example maps/charts in the dashboard (Member 4).

If needed, the Spatial Analyst can further refine the study region by selecting specific district names (using `-clipsrcwhere`) and regenerating more specialized subsets.

9 Guidance for Other Team Members

9.1 For the Spatial Analyst (Member 2)

- Use `data/clean/riyadh_restaurants_clean.geojson` for city-wide maps in GeoPandas or QGIS.
- Use `data/sample_district/restaurants_sample_district.geojson` for detailed maps of the study area.
- Both files are in EPSG:4326, which is directly compatible with most web map backgrounds (OpenStreetMap, etc.).

9.2 For the Business Analyst (Member 3)

- Use `data/clean/riyadh_restaurants_clean.csv` for global analyses (top categories, price vs. rating, distribution of ratings, etc.).
- Use `data/sample_district/restaurants_sample_district.csv` for focused local analyses in the study area.
- `price` and `rating` have no missing values, so they can be safely used in correlations and plots.
- `ratingSignals` is missing for many rows; consider filtering or handling `NaN` explicitly when using it.

9.3 For the Dashboard Developer (Member 4)

- Load the cleaned CSV into Streamlit (or another framework) for tables and non-spatial charts.
- Use the GeoJSON files for map-based visualizations (for example, with `pydeck` or `folium`).
- The separation between full and sample datasets allows:
  - One dashboard page for "All Riyadh",
  - Another page for the "Sample District" with more detailed views.

10 Conclusion

The Data Engineering phase has:

- Converted the raw Kaggle dataset (19 361 rows) into a clean, consistent dataset (19 355 rows).
- Ensured that price and rating information are complete and ready for analysis.
- Exported the data in both CSV and GeoJSON formats.
- Created a sample district subset (1 804 restaurants) by clipping to the provided district polygons using GDAL.

These outputs form the foundation for the Spatial Analyst, Business Analyst, and Dashboard Developer to complete the remaining project tasks.