Excel→PPT Agent (LangGraph + Gemini): Overview & Runbook

Give it an Excel file and a PowerPoint template; it plans/creates charts (optionally following your exact chart requirements), drops editable/native charts into named shapes in the PPT, and adds a short Insights section powered by Gemini.

Graph structure (flow of steps)

```
Validate

  ↓

Load Mapping (optional YAML/JSON)

  ↓

Load Excel

  ↓

Plan Charts (Gemini, schema-only)

  ↓

Enforce Requirements (overlay your required chart specs)

  ↓

Render/Prepare Chart Payloads (data series + PNG fallback)

  ↓

Generate Insights (Gemini, aggregates-only)

  ↓

Bind to PPT (native editable charts + mapping to named shapes)

  ↓

End
```

Each node — purpose & fallbacks

1) Validate

- Checks file paths exist (Excel, PPTX).

- Creates a temp folder for images (used as fallback).
- If a path is wrong → stops with a helpful error.

2) Load Mapping (optional)

- Reads --map YAML/JSON if provided.
- Expected keys (all optional):
- • charts: { <chart_id>: <shape_name> }
- • insights: <shape_name>
- • requirements: either
-    – simple: { <chart_id>: 'line' | 'bar' }
-    – explicit: { type, x, y, agg, top_k }
- No mapping file? The agent still works.

3) Load Excel

- Reads the first sheet; best-effort parses columns containing "date" as datetimes.
- Keeps data in memory (no uploads).

4) Plan Charts (Gemini)

- Sends only a schema/profile of the data (column names, basic stats, a few top values); no raw rows.
- Asks Gemini to suggest 1–2 charts as strict JSON (ChartPlan): id, type (line|bar), x, y, agg, top_k.
- If the call fails, falls back to a sensible default (time-series line or top-K bar).

5) Enforce Requirements (your overlay)

- Applies optional requirements from mapping:
- • If you provide type only (e.g., ts: line), it picks suitable columns for you.
- • If you provide explicit columns (x/y/agg/top_k), it validates and fills gaps with heuristics.
- Ensures required charts exist even if Gemini didn't propose them.

6) Render/Prepare Chart Payloads

- Builds native chart payloads (categories, values, series_name, type, title).
- Also saves a small PNG per chart as a last-resort fallback.

7) Generate Insights (Gemini)

- Computes tiny aggregates locally (sum/mean for a few numeric columns) and sends those plus chart titles.
- Returns structured output: a 1–2 sentence summary + 3–5 bullets.
- If the model call fails, writes a friendly placeholder.

8) Bind to PPT (native, editable charts)

- For each mapped chart ID → look up the named shape anywhere in the template:

- 1) If the shape already contains a chart, replace its data (editable).
- 2) If it's a placeholder, insert a native chart into it.
- 3) Otherwise, add a native chart at the shape's position/size.
- 4) If native fails, drop the PNG fallback.
- Unmapped charts get new slides with native charts.
- Insights write into the mapped text box; otherwise create a 'Key Insights' slide.
- Saves as <template_basename>_filled.pptx.

Shared state (carried between steps)

- excel_path, ppt_template_path — input paths
- mapping_path — optional YAML/JSON file
- mapping — parsed charts↔shapes, insights target, and requirements
- df — DataFrame loaded from the Excel (first sheet)
- chart_plan — final plan after Gemini + requirement overlay
- charts — prepared payloads for native PPT (and a PNG fallback path)
- insights — summary + bullets for the insights slide
- ppt_out_path — saved output file path
- log — human-readable steps/fallbacks

Inputs

Required

- --excel path/to/data.xlsx
- --template path/to/template.pptx
- .env with GOOGLE_API_KEY=... (or GEMINI_API_KEY)

Optional

- --map path/to/mapping.yaml (or .json)
- charts: { chart_id: shape_name }
- insights: TEXT_insights
- requirements:
- • simple: { chart_id: 'line' | 'bar' }
- • explicit: { type, x, y, agg, top_k }

Output

- A filled deck saved next to your template, e.g., template_filled.pptx.
- Charts are editable/native unless the PNG fallback was needed.
- Console log shows requirement enforcement, missing shapes, and any fallbacks.

Requirements (packages & openness)

- langchain-google-genai — Open-source wrapper for Gemini in LangChain (service requires API key).
- google-genai — Open-source SDK client; connects to proprietary Gemini API.
- langgraph — Open-source orchestration for agent graphs.
- pandas — Open-source data frames.
- numpy — Open-source numerics.
- matplotlib — Open-source plotting (used only to create PNG fallback images).
- python-pptx — Open-source PowerPoint writer/editor (native charts).
- pydantic — Open-source data validation / structured schemas.
- python-dotenv — Open-source .env loader.
- pyyaml — Open-source YAML parser (only if you use YAML mapping files).

Install (one time)

```
pip install -U langchain-google-genai google-genai langgraph pandas numpy
matplotlib python-pptx pydantic python-dotenv pyyaml
```

How to run

Minimal (no mapping)

```
python
excel_to_ppt_agent_langchain_env_mapping_requirements_columns_insights.py --
excel ./data.xlsx --template ./template.pptx
```

With mapping + requirements

```
python
excel_to_ppt_agent_langchain_env_mapping_requirements_columns_insights.py --
excel ./data.xlsx --template ./template.pptx --map ./mapping.yaml
```

Example mapping.yaml

```
charts:

  revenue_trend: CHART_revenue_trend

  top_customers: CHART_top_customers

insights: TEXT_insights

requirements:

  revenue_trend:

    type: line

    x: date
    y: revenue
    agg: sum
  top_customers:
    type: bar
    x: customer
    y: revenue
    top_k: 10
```

Edge cases and fallbacks

- Required chart not supported by data (e.g., no datetime for a line): we synthesize a best-effort version (line vs row index or bar) and log it.
- Mapping points to a shape that doesn't exist: logged; we create a new slide and place a native chart.
- Mapped chart ID wasn't produced: requirement overlay tries to produce it; if still unavailable, remaining charts are placed and the miss is logged.
- Native chart insertion fails: PNG fallback is used, so the deck still renders.

Template tips

- Use the Selection Pane in PowerPoint to rename shapes where you want content.
- Name chart targets like CHART_<id> (e.g., CHART_revenue_trend) and the insights box TEXT_insights.
- Avoid grouped targets for placement; plain placeholders or shapes work best.