**Contents**

1. **Use case – Problem Statement**
2. **Project and Folder Structure**

**USECASE**

**📊 Use Case 1: "Identify Underperforming Stores"**

**🔍 Business Goal:**

We want to find out which stores are consistently generating lower sales than others and figure out if it's due to lack of promotions, holidays, or fewer customers.

**📋 Requirements:**

* Compare average daily sales of each store.
* Rank stores by average sales.
* Show:
  + Store
  + Average Sales
  + Average Customers
  + % of Days Open
* Flag stores in the **bottom 10 percentile** as underperformers.
* Optional: Compare with number of promotions (Promo) per store.

**📈 Use Case 2: "Promotion Effectiveness"**

**🔍 Business Goal:**

We need to justify the cost of our promos. I want to know: do promotions actually drive more sales?

**📋 Requirements:**

* Compare average sales on Promo = 1 vs Promo = 0, **for each store**.
* Bonus: Also compare average customers.
* Filter out closed days (Open = 0).
* Find top 5 stores where promotions had the **biggest positive impact**.
* Show:
  + Store
  + Avg Sales (Promo)
  + Avg Sales (No Promo)
  + % Increase

**🏖️ Use Case 3: "Holiday Impact Analysis"**

**🔍 Business Goal:**

Understand how holidays affect our business.

**📋 Requirements:**

* Group by StateHoliday and analyze average sales.
* Also compare Open = 1 vs Open = 0 on holidays.
* Show:
  + Holiday Type (0, a, b, c)
  + Avg Sales (Open)
  + Days Closed
  + Avg Customers
* Bonus: Trend sales over time for holidays across weeks/months.

**📅 Use Case 4: "Best Day to Launch Campaigns"**

**🔍 Business Goal:**

We want to launch new weekly offers. Which day of the week gives us the best footfall and sales?

**📋 Requirements:**

* For all stores, group by DayOfWeek (1 = Monday, 7 = Sunday)
* Show:
  + Average Sales
  + Average Customers
  + % of Days Open
* Identify **top 2 days** for launching new campaigns.
* Bonus: Visualize as a weekly sales profile.

**🕵️ Use Case 5: "Store Reliability Score"**

**🔍 Business Goal:**

We need a health score to measure how consistently a store operates.

**📋 Requirements:**

Create a reliability score based on:

* % of Days Open
* % of SchoolHoliday days closed
* Sales volatility (e.g., Standard Deviation)
* Show:
  + Store
  + Days Open %
  + Volatility
  + Reliability Score (you define the formula)

**🔁 Use Case 6: "Store Calendar Generator" *(Stretch goal)***

**🔍 Business Goal:**

I want a ready-made calendar that tells me for any store:

* What days it was open
* What kind of day it was (weekend, holiday, promo, etc.)

**📋 Requirements:**

* Create a derived table or view with:
  + Date
  + Store
  + Is\_Weekend
  + Is\_Holiday
  + Was\_Open
  + Was\_Promo
  + Sales

**FOLDER STRUCTURE**

**rossmann-dashboard/**

**│**

**├── app/ # Core Dash application**

**│ ├── \_\_init\_\_.py**

**│ ├── layout.py # Dash layout components**

**│ ├── callbacks.py # Dash interactivity**

**│ └── database.py # SQLite connection + queries**

**│**

**├── data/**

**│ └── rossmann.csv # Raw data**

**│**

**├── db/**

**│ └── rossmann.sqlite # Generated DB (not versioned)**

**│**

**├── scripts/**

**│ ├── create\_db.sql # SQL to create & populate tables**

**│ └── underperformers.sql # Query script**

**│**

**├── Dockerfile**

**├── requirements.txt**

**├── gunicorn.conf.py**

**├── app.py # Entry point for Dash**

**├── .gitignore**

**└── README.md**

**USECASE 1**

✅ Step 1: Business Use Case – Underperforming Stores

🎯 Business Goal:

“As the Sales Director, I want to identify consistently low-performing stores so I can allocate help or make closure decisions.”

🧑‍💻 Tech Lead Style Breakdown

🔹 Git Versioning

* Commit scripts and .py files
* DO NOT commit .sqlite unless it’s a milestone or frozen reference
* Add to .gitignore:

bash

CopyEdit

db/rossmann.sqlite

\_\_pycache\_\_/

\*.pyc

.env

📜 SQL Script (scripts/underperformers.sql)

sql

CopyEdit

-- Get stats per store

WITH store\_stats AS (

SELECT

Store,

COUNT(\*) AS total\_days,

SUM(Open) AS open\_days,

AVG(Sales) AS avg\_sales,

AVG(Customers) AS avg\_customers,

ROUND(100.0 \* SUM(Open) / COUNT(\*), 2) AS open\_rate

FROM rossmann

GROUP BY Store

)

SELECT

Store,

avg\_sales,

avg\_customers,

open\_rate,

RANK() OVER (ORDER BY avg\_sales ASC) AS sales\_rank

FROM store\_stats

WHERE open\_days > 50;

🚀 Step 2: Building the Dash App

🔹 app/database.py

python

CopyEdit

import sqlite3

import pandas as pd

def query\_db(query):

conn = sqlite3.connect("db/rossmann.sqlite")

df = pd.read\_sql\_query(query, conn)

conn.close()

return df

🔹 app/layout.py

python

CopyEdit

from dash import html, dcc

layout = html.Div([

html.H1("Underperforming Stores"),

dcc.Graph(id="store-graph")

])

🔹 app/callbacks.py

python

CopyEdit

from dash import Output, Input

import plotly.express as px

from .database import query\_db

def register\_callbacks(app):

@app.callback(

Output("store-graph", "figure"),

Input("store-graph", "id") # Dummy trigger

)

def update\_graph(\_):

query = open("scripts/underperformers.sql").read()

df = query\_db(query)

fig = px.bar(df, x="Store", y="avg\_sales", title="Underperforming Stores")

return fig

🔹 app.py

python

CopyEdit

from dash import Dash

from app.layout import layout

from app.callbacks import register\_callbacks

app = Dash(\_\_name\_\_)

app.layout = layout

register\_callbacks(app)

server = app.server # for gunicorn

if \_\_name\_\_ == "\_\_main\_\_":

app.run\_server(debug=True)

🐳 Step 3: Docker + Gunicorn

🔹 requirements.txt

nginx

CopyEdit

dash

pandas

plotly

gunicorn

🔹 Dockerfile

Dockerfile

CopyEdit

FROM python:3.11-slim

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

EXPOSE 8050

CMD ["gunicorn", "--config", "gunicorn.conf.py", "app:server"]

🔹 gunicorn.conf.py

python

CopyEdit

bind = "0.0.0.0:8050"

workers = 2

timeout = 120

🔹 Build & Run Docker

bash

CopyEdit

docker build -t rossmann-app .

docker run -p 8050:8050 rossmann-app

Then open <http://localhost:8050> in your browser 🎉

🧠 SQLite Scripting & Execution

🔹 Create DB from CSV

sql

CopyEdit

-- scripts/create\_db.sql

DROP TABLE IF EXISTS rossmann;

CREATE TABLE rossmann (

Store INT,

DayOfWeek INT,

Date TEXT,

Sales INT,

Customers INT,

Open INT,

Promo INT,

StateHoliday TEXT,

SchoolHoliday INT

);

.mode csv

.import data/rossmann.csv rossmann

🔹 Run This

bash

CopyEdit

sqlite3 db/rossmann.sqlite < scripts/create\_db.sql

✅ Next Steps

Let’s checkpoint this as Milestone 1:

* Folder structure created
* SQLite DB generated from CSV
* Dash app renders one graph using SQL
* Docker build works

Once you're done with this, tell me and I’ll guide you through:

* Use Case 2: Promo Effectiveness
* Adding filters/dropdowns to your Dash UI
* Deployment extensions like Docker Compose (with a volume mount, etc.)

Want a GitHub .gitignore template or prewritten Docker .env support too?

You said:

Add to `.gitignore`:

Where is the gitignore found? in which folder? Do I create it? I do not see it in my project folder

ChatGPT said:

Great question — yes, you create the .gitignore file yourself! It should be placed at the root level of your project folder — the same level as your Dockerfile, app.py, and README.md.

📁 Your Folder Should Look Like:

kotlin

CopyEdit

rossmann-dashboard/

│

├── .gitignore 👈 You create this here

├── Dockerfile

├── app.py

├── requirements.txt

├── README.md

├── data/

├── db/

├── app/

├── scripts/

└── outputs/

✍️ How to Create .gitignore

You can create it using:

🖥️ On Command Line

bash

CopyEdit

touch .gitignore

📄 Or, Manually (File Explorer)

1. Right-click → New → Text Document
2. Name it .gitignore exactly — no .txt at the end
   * On Windows: Enable file extensions to remove .txt
   * On macOS/Linux: It works directly

📝 What to Add in .gitignore

Here’s a good starter for your project:

gitignore

CopyEdit

# Ignore SQLite DBs

db/\*.sqlite

# Python cache

\_\_pycache\_\_/

\*.pyc

# Jupyter checkpoints

.ipynb\_checkpoints/

# Environment

.env

# OS-specific

.DS\_Store

Thumbs.db

Once done, commit it:

bash

CopyEdit

git add .gitignore

git commit -m "Add initial .gitignore"

Let me know if you want me to generate a custom .gitignore based on all your files too — I can automate that.