Project Report: Analysis of Weather and Sales Correlation

Abdurrahman Rafif

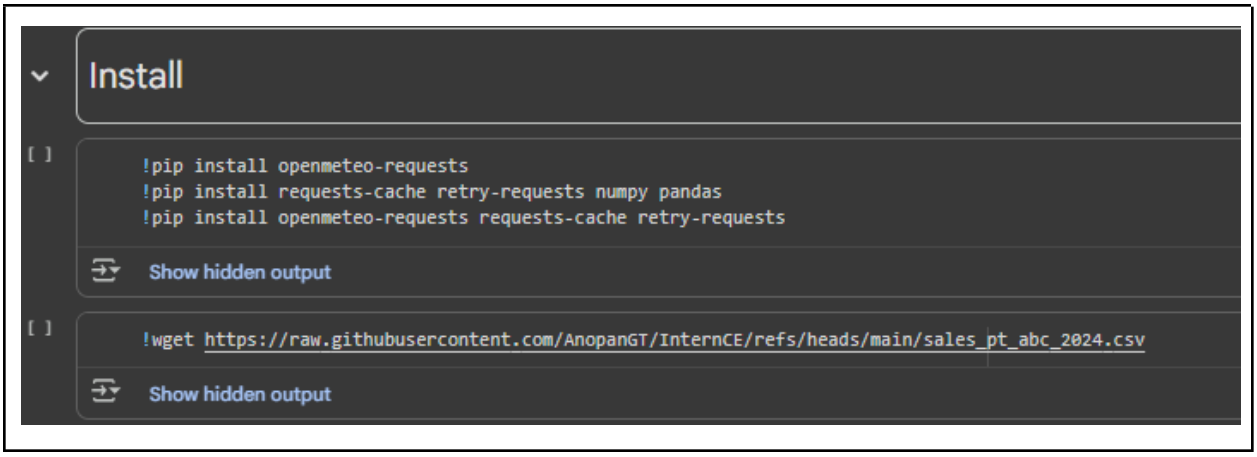
Final Result:
https://lookerstudio.google.com/reporting/53e3179c-6042-4b26-8861-30a16ad60972

Colab Link:
https://colab.research.google.com/drive/1TJK0ilUyQM4vKakqWqoXGgP_9yPFAbo0?usp=sharing

PPT Link:
https://www.canva.com/design/DAGy9dZd1XI/IC6CuLFTJuwMBjorAZqQlg/edit?utm_content=DAGy9dZd1XI&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Data Retrieval using API (I use open-meteo because it's free)

Install



Usage

"I use Google Collab"

Using Jakarta Pusat location, start-end 2024, and using variables that are deemed necessary for the future, then save to cuaca_jakarta_2024.csv

```
import openmeteo_requests
```

```
import pandas as pd
import requests_cache
from retry_requests import retry

# Setup the Open-Meteo API client with cache and retry on error
cache_session = requests_cache.CachedSession('.cache', expire_after =
-1)
retry_session = retry(cache_session, retries = 5, backoff_factor = 0.2)
openmeteo = openmeteo_requests.Client(session = retry_session)

# Make sure all required weather variables are listed here
# The order of variables in hourly or daily is important to assign them
correctly below
url = "https://archive-api.open-meteo.com/v1/archive"
params = {
  "latitude": -6.1818,
  "longitude": 106.8223,
  "start_date": "2024-01-01",
  "end_date": "2024-12-31",
   "daily": ["weather_code", "temperature_2m_mean", "temperature_2m_max",
"apparent_temperature_mean",   "apparent_temperature_max",   "rain_sum",
"precipitation_sum",     "precipitation_hours",     "sunshine_duration",
"relative_humidity_2m_mean", "cloud_cover_mean"],
  "timezone": "Asia/Bangkok",
}
responses = openmeteo.weather_api(url, params=params)

# Process first location. Add a for-loop for multiple locations or
weather models
response = responses[0]
print(f"Coordinates: {response.Latitude()}°N {response.Longitude()}°E")
print(f"Elevation: {response.Elevation()} m asl")
print(f"Timezone:
{response.Timezone()}{response.TimezoneAbbreviation()}")
print(f"Timezone difference to GMT+0: {response.UtcOffsetSeconds()}s")

# Process daily data. The order of variables needs to be the same as
requested.
daily = response.Daily()
```

```
daily_weather_code = daily.Variables(0).ValuesAsNumpy()
daily_temperature_2m_mean = daily.Variables(1).ValuesAsNumpy()
daily_temperature_2m_max = daily.Variables(2).ValuesAsNumpy()
daily_apparent_temperature_mean = daily.Variables(3).ValuesAsNumpy()
daily_apparent_temperature_max = daily.Variables(4).ValuesAsNumpy()
daily_rain_sum = daily.Variables(5).ValuesAsNumpy()
daily_precipitation_sum = daily.Variables(6).ValuesAsNumpy()
daily_precipitation_hours = daily.Variables(7).ValuesAsNumpy()
daily_sunshine_duration = daily.Variables(8).ValuesAsNumpy()
daily_relative_humidity_2m_mean = daily.Variables(9).ValuesAsNumpy()
daily_cloud_cover_mean = daily.Variables(10).ValuesAsNumpy()

daily_data = {"date": pd.date_range(
  start = pd.to_datetime(daily.Time(), unit = "s", utc = True),
  end = pd.to_datetime(daily.TimeEnd(), unit = "s", utc = True),
  freq = pd.Timedelta(seconds = daily.Interval()),
  inclusive = "left"
)}

daily_data["weather_code"] = daily_weather_code
daily_data["temperature_2m_mean"] = daily_temperature_2m_mean
daily_data["temperature_2m_max"] = daily_temperature_2m_max
daily_data["apparent_temperature_mean"]                                    =
daily_apparent_temperature_mean
daily_data["apparent_temperature_max"] = daily_apparent_temperature_max
daily_data["rain_sum"] = daily_rain_sum
daily_data["precipitation_sum"] = daily_precipitation_sum
daily_data["precipitation_hours"] = daily_precipitation_hours
daily_data["sunshine_duration"] = daily_sunshine_duration
daily_data["relative_humidity_2m_mean"]                                    =
daily_relative_humidity_2m_mean
daily_data["cloud_cover_mean"] = daily_cloud_cover_mean

daily_dataframe = pd.DataFrame(data = daily_data)

nama_file_cuaca = 'cuaca_jakarta_2024.csv'
daily_dataframe.to_csv(nama_file_cuaca, index=False)
print(f"\n✅ Data cuaca berhasil disimpan ke file: {nama_file_cuaca}")
```

Output:

```
Coordinates: -6.151142120361328°N 106.84210205078125°E
Elevation: 15.0 m asl
Timezone: b'Asia/Bangkok'b'GMT+7'
Timezone difference to GMT+0: 25200s

✅ Data cuaca berhasil disimpan ke file: cuaca_jakarta_2024.csv
```

Merging 'cuaca_jakarta_2024.csv' and 'sales_pt_abc_2024.csv'

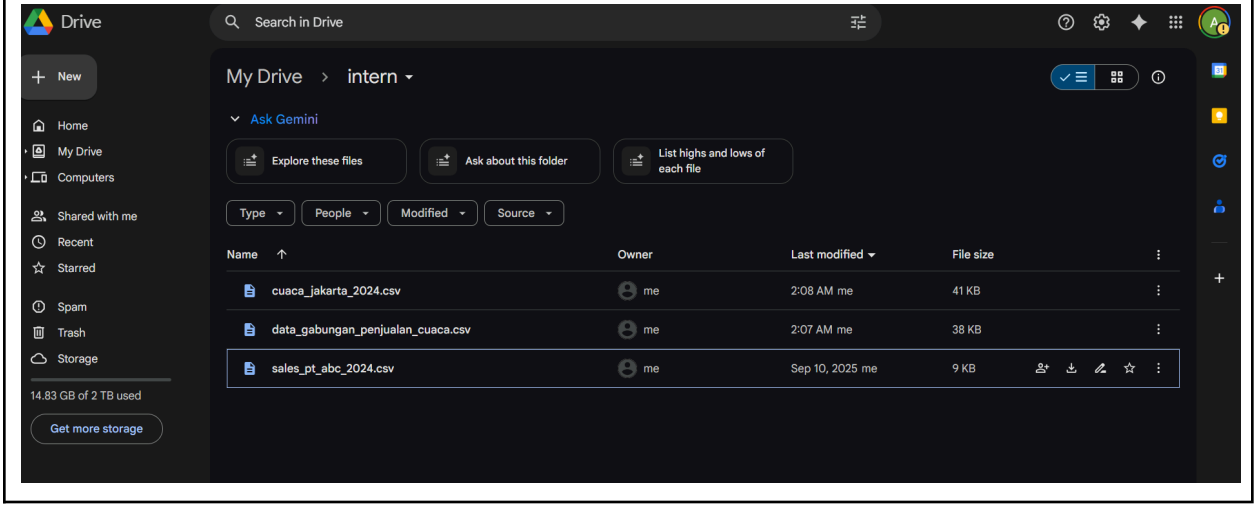
By reading both files, then cleaning and formatting the data, then saving it to 'data_gabungan_penjualan_cuaca.csv'

```
import pandas as pd

# Nama file input dan output
file_cuaca = 'cuaca_jakarta_2024.csv'
file_penjualan = 'sales_pt_abc_2024.csv'
file_output = 'data_gabungan_penjualan_cuaca.csv'

try:
    # 1. Baca kedua file CSV
    df_cuaca = pd.read_csv(file_cuaca)
    df_penjualan = pd.read_csv(file_penjualan)
    print("✅ Berhasil membaca kedua file.")

    # 2. Membersihkan Data dan Menyamakan Tipe Data Tanggal
    print("\nMembersihkan dan memformat data...")

    # --- Proses Data Penjualan ---
    df_penjualan.columns = df_penjualan.columns.str.strip()
                                        df_penjualan['Sales']          =
df_penjualan['Sales'].astype(str).str.replace(r'[       \"$,]',       '',
regex=True)
         df_penjualan['Sales']   =   pd.to_numeric(df_penjualan['Sales'],
errors='coerce')
     # Mengubah ke datetime dan HANYA MENGAMBIL TANGGALNYA (menghapus
jam)
```

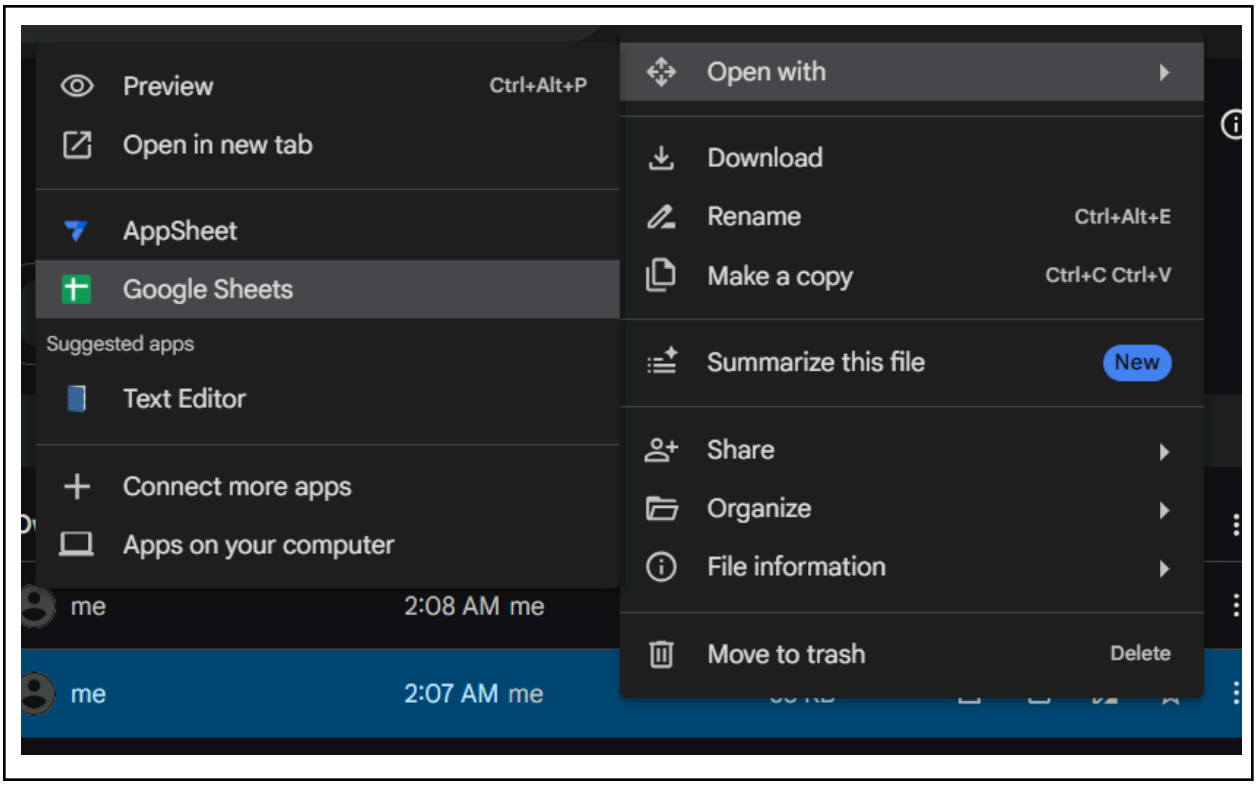
```
          df_penjualan['Date']   =   pd.to_datetime(df_penjualan['Date'],
format='%d/%m/%Y').dt.date

    # --- Proses Data Cuaca ---
      # Mengubah ke datetime, menghapus timezone, dan HANYA MENGAMBIL
TANGGALNYA
                                              df_cuaca['date']           =
pd.to_datetime(df_cuaca['date']).dt.tz_localize(None).dt.date

    print("Format selesai disamakan.")

    # 3. Investigasi Rentang Tanggal
    print("\n--- Hasil Investigasi Tanggal ---")
                print(f"File    Penjualan    ->    Tanggal    Pertama:
{df_penjualan['Date'].min()},              Tanggal              Terakhir:
{df_penjualan['Date'].max()}")
           print(f"File   Cuaca                  ->   Tanggal   Pertama:
{df_cuaca['date'].min()}, Tanggal Terakhir: {df_cuaca['date'].max()}")
    print("---------------------------------\n")

    # 4. Menggabungkan Kembali Data
    print("Mencoba menggabungkan data kembali...")
    df_penjualan.rename(columns={'Date': 'date'}, inplace=True)

     # Mengubah kembali kolom 'date' menjadi tipe data yang sama sebelum
merge
    df_penjualan['date'] = pd.to_datetime(df_penjualan['date'])
    df_cuaca['date'] = pd.to_datetime(df_cuaca['date'])

          df_gabungan   =   pd.merge(df_penjualan,   df_cuaca,   on='date',
how='inner')

    # 5. Simpan dan Tampilkan Hasil
    if len(df_gabungan) > 0:
      df_gabungan.to_csv(file_output, index=False)
         print(f"🎉 SELAMAT! Data gabungan berhasil disimpan ke file:
'{file_output}'")
      print("\nBerikut adalah 5 baris pertama dari data gabungan Anda:")
      display(df_gabungan.head())
```

```
            print(f"\nTotal baris data yang berhasil digabung: {len(df_gabungan)}")
    else:
        print("❌ GAGAL: Masih tidak ada tanggal yang cocok setelah perbaikan. Cek hasil investigasi di atas.")

except FileNotFoundError as e:
        print(f"\n❌ ERROR: File tidak ditemukan! Pastikan file '{e.filename}' sudah diunggah ke Colab.")
except Exception as e:
    print(f"\n❌ Terjadi error: {e}")
```

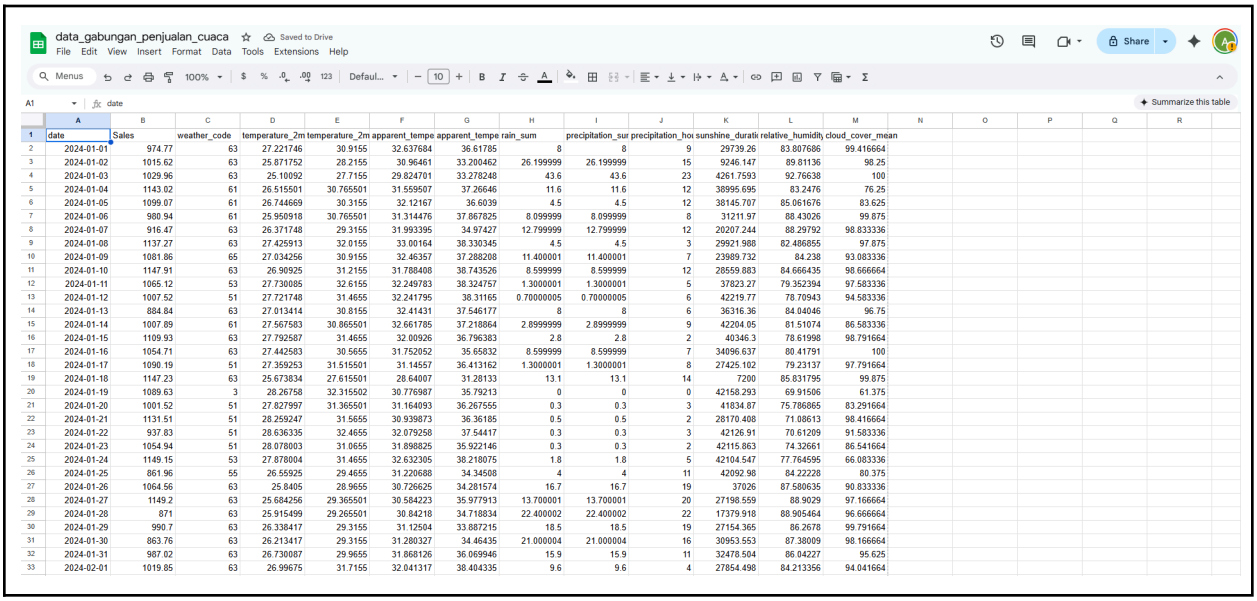
Upload File to Drive to simplify Looker Studio creation:



Open with Google Sheets:

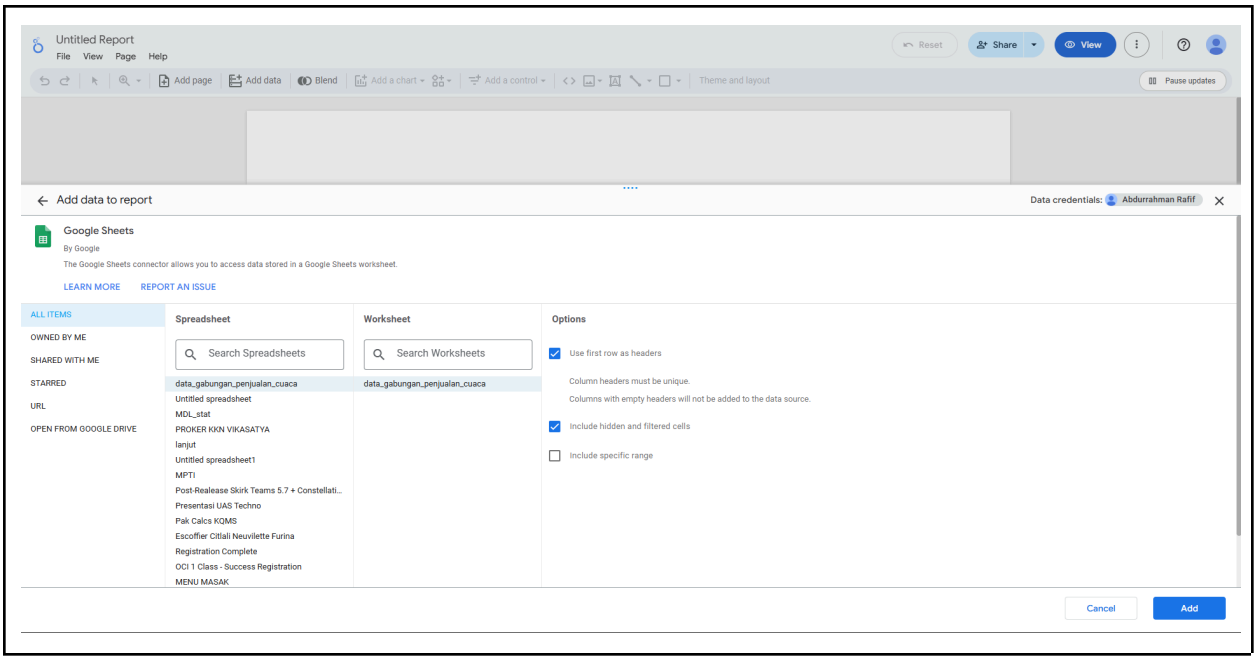


Google Sheets View:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	date	Sales	weather_code	temperature_2m	temperature_2m	apparent_tempe	apparent_tempe	rain_sum	precipitation_sur	precipitation_hou	sunshine_durati	relative_humidity	cloud_cover_mean
2	2024-01-01	974.77	63	27.221746	30.9155	32.637684	36.61785	8	8	9	29739.26	83.807686	99.416664
3	2024-01-02	1015.62	63	25.871752	28.2155	30.96461	33.200462	26.199999	26.199999	15	9246.147	89.81136	98.25
4	2024-01-03	1029.96	63	25.10092	27.7155	29.824701	33.278248	43.6	43.6	23	4261.7593	92.76638	100
5	2024-01-04	1143.02	61	26.515501	30.765501	31.559507	37.26646	11.6	11.6	12	38995.695	83.2476	76.25
6	2024-01-05	1099.07	61	26.744669	30.3155	32.12167	36.6039	4.5	4.5	12	38145.707	85.061676	83.625
7	2024-01-06	980.94	61	25.950918	30.765501	31.314476	37.867825	8.099999	8.099999	8	31211.97	88.43026	99.875
8	2024-01-07	916.47	63	26.371748	29.3155	31.993395	34.97427	12.799999	12.799999	12	20207.244	88.29792	98.833336
9	2024-01-08	1137.27	63	27.425913	32.0155	33.00164	38.330345	4.5	4.5	3	29921.988	82.486855	97.875
10	2024-01-09	1081.86	65	27.034256	30.9155	32.46357	37.288208	11.400001	11.400001	7	23989.732	84.238	93.083336
11	2024-01-10	1147.91	63	26.90925	31.2155	31.788408	38.743526	8.599999	8.599999	12	28559.883	84.666435	98.666664
12	2024-01-11	1065.12	53	27.730085	32.6155	32.249783	38.324757	1.3000001	1.3000001	5	37823.27	79.352394	97.583336
13	2024-01-12	1007.52	51	27.721748	31.4655	32.241795	38.31165	0.70000005	0.70000005	6	42219.77	78.70943	94.583336
14	2024-01-13	884.84	63	27.013414	30.8155	32.41431	37.546177	8	8	6	36316.36	84.04046	96.75
15	2024-01-14	1007.89	61	27.567583	30.865501	32.661785	37.218864	2.8999999	2.8999999	9	42204.05	81.51074	86.583336
16	2024-01-15	1109.93	63	27.792587	31.4655	32.00926	36.796383	2.8	2.8	2	40346.3	78.61998	98.791664
17	2024-01-16	1054.71	63	27.442583	30.5655	31.752052	35.65832	8.599999	8.599999	7	34096.637	80.41791	100
18	2024-01-17	1090.19	51	27.359253	31.515501	31.14557	36.413162	1.3000001	1.3000001	8	27425.102	79.23137	97.791664
19	2024-01-18	1147.23	63	25.673834	27.615501	28.64007	31.28133	13.1	13.1	14	7200	85.831795	99.875
20	2024-01-19	1089.63	3	28.26758	32.315502	30.776987	35.79213	0	0	0	42158.293	69.91506	61.375
21	2024-01-20	1001.52	51	27.827997	31.365501	31.164093	36.267555	0.3	0.3	3	41834.87	75.786865	83.291664
22	2024-01-21	1131.51	51	28.259247	31.5655	30.939873	36.36185	0.5	0.5	2	28170.408	71.08613	98.416664
23	2024-01-22	937.83	51	28.636335	32.4655	32.079258	37.54417	0.3	0.3	3	42126.91	70.61209	91.583336
24	2024-01-23	1054.94	51	28.078003	31.0655	31.898825	35.922146	0.3	0.3	2	42115.863	74.32661	86.541664
25	2024-01-24	1149.15	53	27.878004	31.4655	32.632305	38.218075	1.8	1.8	5	42104.547	77.764595	66.083336
26	2024-01-25	861.96	55	26.55925	29.4655	31.220688	34.34508	4	4	11	42092.98	84.22228	80.375
27	2024-01-26	1064.56	63	25.8405	28.9655	30.726625	34.281574	16.7	16.7	19	37026	87.580635	90.833336
28	2024-01-27	1149.2	63	25.684256	29.365501	30.584223	35.977913	13.700001	13.700001	20	27198.559	88.9029	97.166664
29	2024-01-28	871	63	25.915499	29.265501	30.84218	34.718834	22.400002	22.400002	22	17379.918	88.905464	96.666664
30	2024-01-29	990.7	63	26.338417	29.3155	31.12504	33.887215	18.5	18.5	19	27154.365	86.2678	99.791664
31	2024-01-30	863.76	63	26.213417	29.3155	31.280327	34.46435	21.000004	21.000004	16	30953.553	87.38009	98.166664
32	2024-01-31	987.02	63	26.730087	29.9655	31.868126	36.069946	15.9	15.9	11	32478.504	86.04227	95.625
33	2024-02-01	1019.85	63	26.99675	31.7155	32.041317	38.404335	9.6	9.6	4	27854.498	84.213356	94.041664

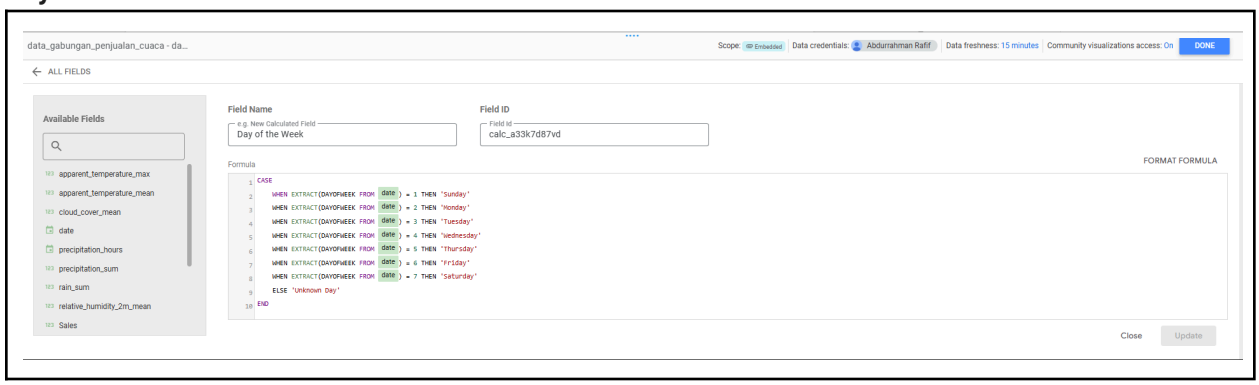
On the Looker page after creating a blank report, Add data to report (the Google Sheets that has been added)



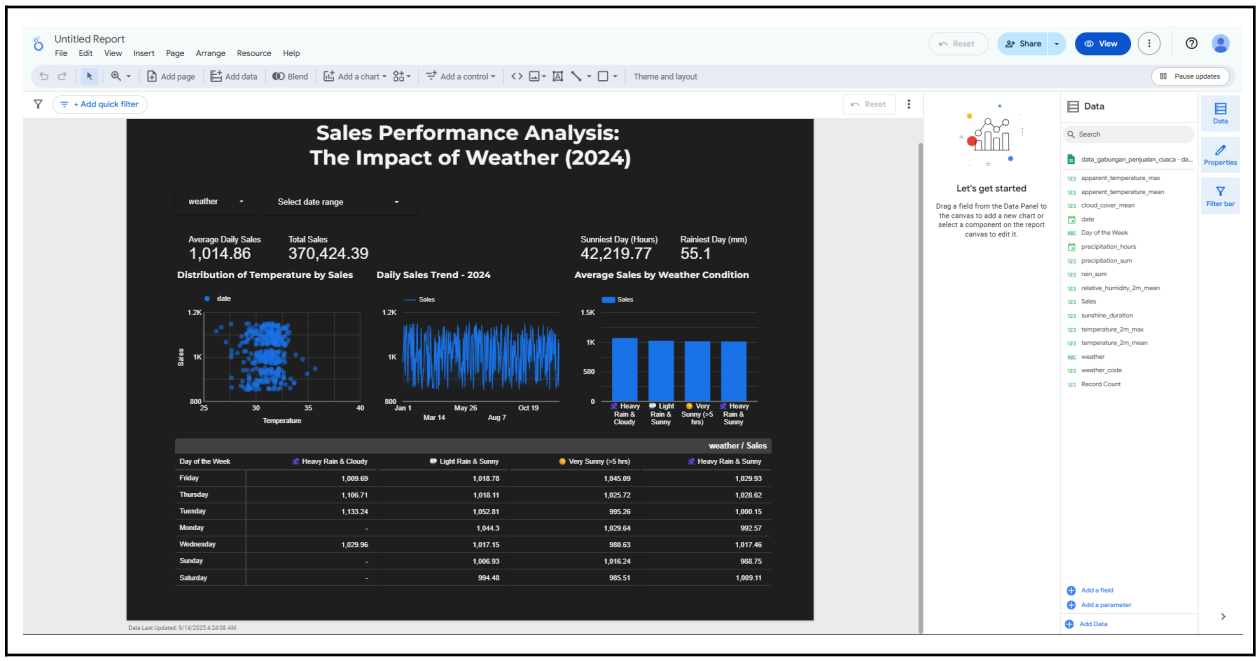
Creation Process by adding Charts, Controls, Fields, etc.

Weather Field for Weather Classification:

Day of the Week Field:



Final Looker Dashboard Result:



1. **Bonus Task: AI Modeling Using Python:**

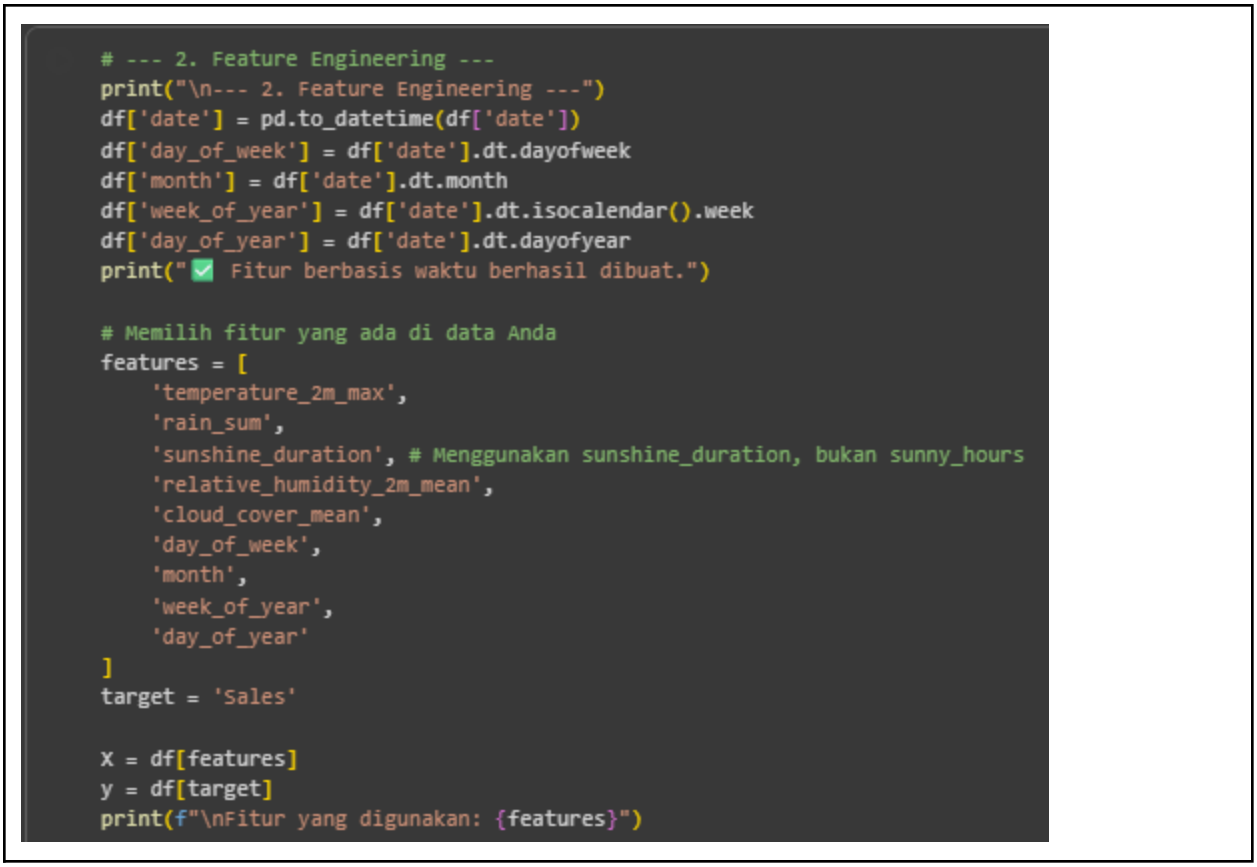
I'm using a linear regression and random forest model to predict future sales based on weather conditions.

RANDOM FOREST

Loading combined data:

```
[ ]
print("--- 1. Memuat Data Gabungan ---")
file_path = 'data_gabungan_penjualan_cuaca.csv'
df = pd.read_csv(file_path)

# Mengisi nilai kosong (jika ada)
df.fillna(df.mean(numeric_only=True), inplace=True)
print("✅ Data berhasil dimuat dan dibersihkan.")
# --- 2. Feature Engineering ---
print("\n--- 2. Feature Engineering ---")
df['date'] = pd.to_datetime(df['date'])
df['day_of_week'] = df['date'].dt.dayofweek
df['month'] = df['date'].dt.month
df['week_of_year'] = df['date'].dt.isocalendar().week
df['day_of_year'] = df['date'].dt.dayofyear
print("✅ Fitur berbasis waktu berhasil dibuat.")

# Memilih fitur yang ada di data Anda
features = [
    'temperature_2m_max',
    'rain_sum',
    'sunshine_duration', # Menggunakan sunshine_duration, bukan sunny_hours
    'relative_humidity_2m_mean',
    'cloud_cover_mean',
    'day_of_week',
    'month',
    'week_of_year',
    'day_of_year'
]
target = 'Sales'

X = df[features]
y = df[target]
print(f"\nFitur yang digunakan: {features}")
```

Feature Engineering:



```python
# --- 2. Feature Engineering ---
print("\n--- 2. Feature Engineering ---")
df['date'] = pd.to_datetime(df['date'])
df['day_of_week'] = df['date'].dt.dayofweek
df['month'] = df['date'].dt.month
df['week_of_year'] = df['date'].dt.isocalendar().week
df['day_of_year'] = df['date'].dt.dayofyear
print("✅ Fitur berbasis waktu berhasil dibuat.")

# Memilih fitur yang ada di data Anda
features = [
    'temperature_2m_max',
    'rain_sum',
    'sunshine_duration', # Menggunakan sunshine_duration, bukan sunny_hours
    'relative_humidity_2m_mean',
    'cloud_cover_mean',
    'day_of_week',
    'month',
    'week_of_year',
    'day_of_year'
]
target = 'Sales'

X = df[features]
y = df[target]
print(f"\nFitur yang digunakan: {features}")
```

Model Training and Model Evaluation:

```python
# --- 3. Model Training ---
print("\n--- 3. Model Training ---")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_model = RandomForestRegressor(n_estimators=100, random_state=42, oob_score=True)
rf_model.fit(X_train, y_train)
print("✅ Model Random Forest berhasil dilatih.")
```

```
--- 3. Model Training ---
✅ Model Random Forest berhasil dilatih.
```

```python
# --- 4. Model Evaluation ---
print("\n--- 4. Model Evaluation ---")
predictions = rf_model.predict(X_test)
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
oob = rf_model.oob_score_

print(f"R-squared (R²): {r2:.4f}")
print(f"Out-of-Bag (OOB) Score: {oob:.4f}")
print(f"Mean Absolute Error (MAE): ${mae:.2f}")
```

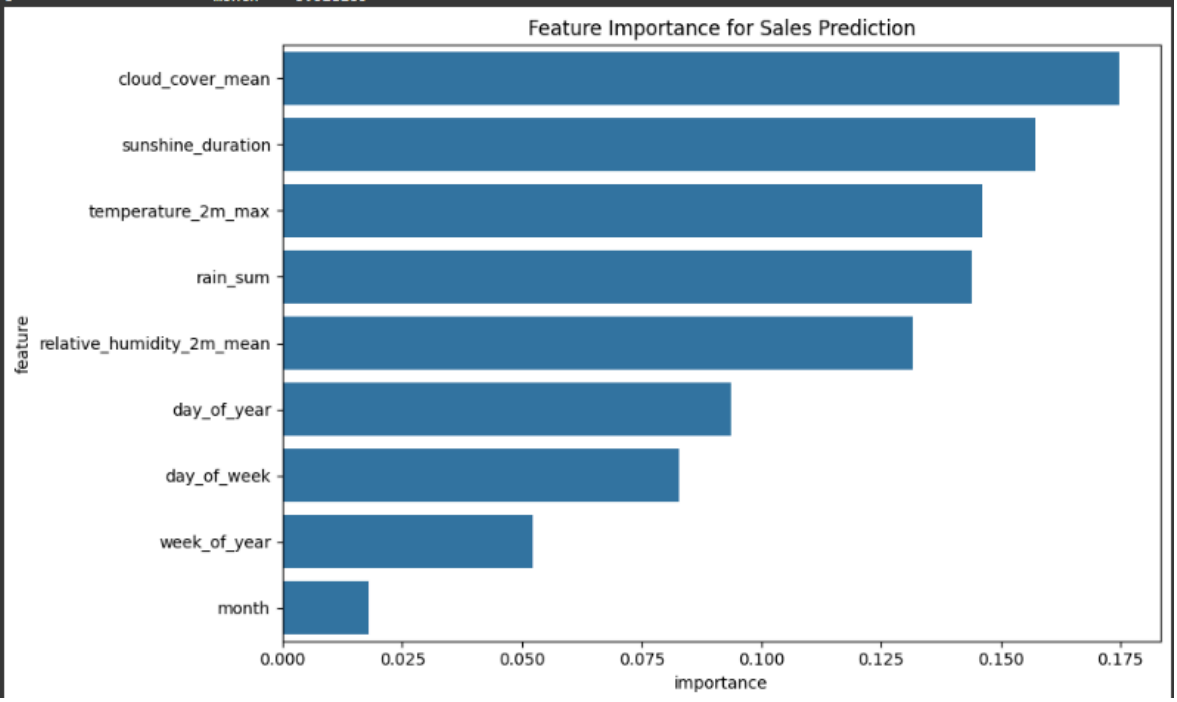
```
--- 4. Model Evaluation ---
R-squared (R²): -0.0430
Out-of-Bag (OOB) Score: -0.1108
Mean Absolute Error (MAE): $75.05
```

Feature Improtance:

```python
# --- 5. Feature Importance ---
print("\n--- 5. Feature Importance ---")
feature_importances = pd.DataFrame({'feature': features, 'importance': rf_model.feature_importances_})
feature_importances = feature_importances.sort_values('importance', ascending=False)

print(feature_importances)

plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=feature_importances)
plt.title('Feature Importance for Sales Prediction')
plt.tight_layout()
plt.savefig('feature_importance.png')
plt.show()
```

Output:

```
--- 5. Feature Importance ---
                      feature  importance
4          cloud_cover_mean    0.174584
2         sunshine_duration    0.157108
0        temperature_2m_max    0.146035
1                  rain_sum    0.143825
3  relative_humidity_2m_mean    0.131553
8               day_of_year    0.093575
5               day_of_week    0.082885
7              week_of_year    0.052326
6                     month    0.018109
```

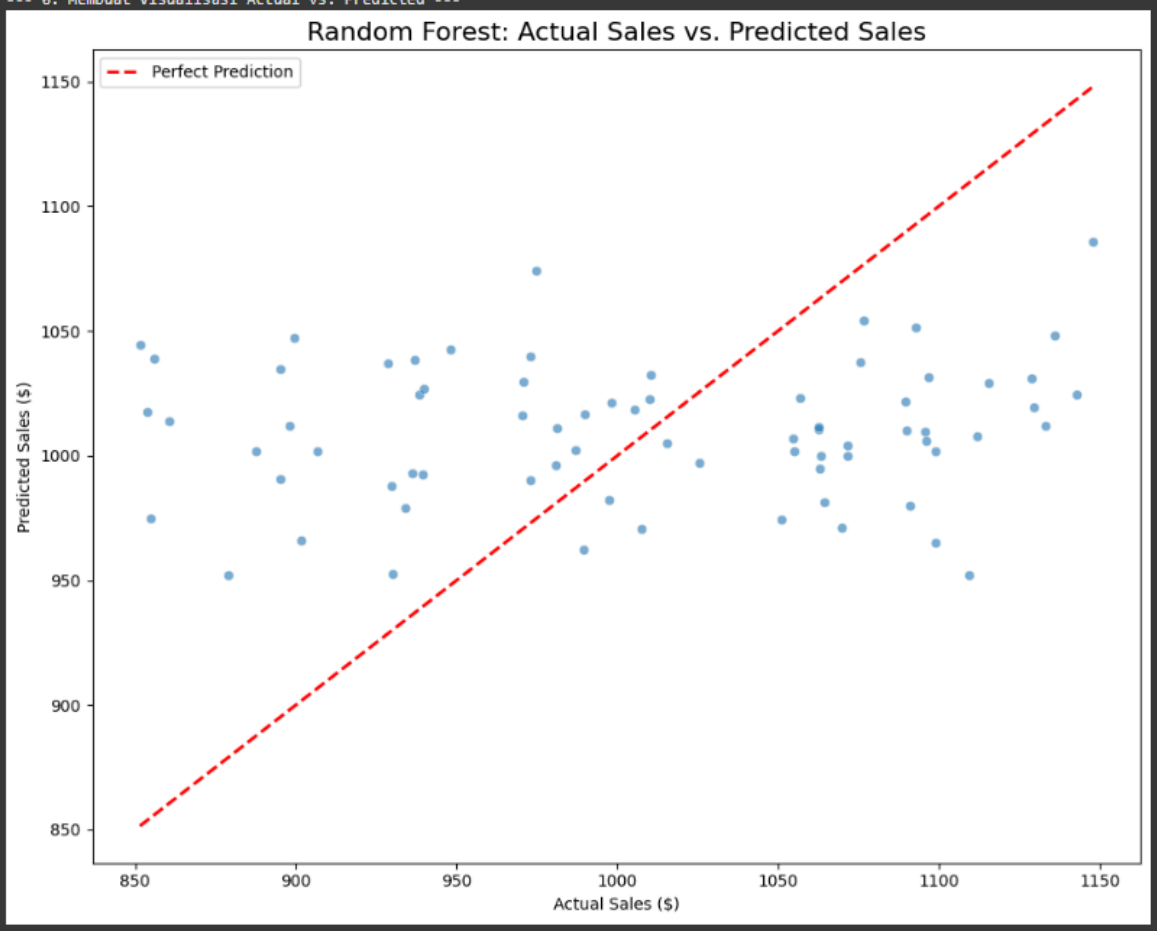


Visualisasai Actual vs Predicted Sales:

```python
# --- 6. Visualisasi Actual vs. Predicted Sales ---
print("\n--- 6. Membuat Visualisasi Actual vs. Predicted ---")

plt.figure(figsize=(10, 8))
sns.scatterplot(x=y_test, y=predictions, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r', linewidth=2, label='Perfect Prediction')
plt.title('Random Forest: Actual Sales vs. Predicted Sales', fontsize=16)
plt.xlabel('Actual Sales ($)')
plt.ylabel('Predicted Sales ($)')
plt.legend()
plt.tight_layout()
plt.savefig('actual_vs_predicted.png')
plt.show()

print("\n✅ Visualisasi Actual vs. Predicted berhasil dibuat dan disimpan sebagai 'actual_vs_predicted.png'")
```

Output:

```
--- 6. Membuat Visualisasi Actual vs. Predicted ---
```




```
✅ Visualisasi Actual vs. Predicted berhasil dibuat dan disimpan sebagai 'actual_vs_predicted.png'
```

LINEAR REGRESSION

Loading Data::

Linear Regression

```
[ ]
print("--- 1. Memuat Data ---")
file_path = 'data_gabungan_penjualan_cuaca.csv'
df = pd.read_csv(file_path)

# Mengisi nilai kosong (jika ada)
df.fillna(df.mean(numeric_only=True), inplace=True)
print("✅ Data berhasil dimuat.")
```

```
--- 1. Memuat Data ---
✅ Data berhasil dimuat.
```

Feature Engineering:

```
[ ]
# --- 2. Feature Engineering ---
print("\n--- 2. Feature Engineering ---")
df['date'] = pd.to_datetime(df['date'])
df['day_of_week'] = df['date'].dt.dayofweek
df['month'] = df['date'].dt.month
df['week_of_year'] = df['date'].dt.isocalendar().week
df['day_of_year'] = df['date'].dt.dayofyear
print("✅ Fitur berbasis waktu berhasil dibuat.")

# Memilih fitur yang sama seperti sebelumnya
features = [
    'temperature_2m_max', 'rain_sum', 'sunshine_duration',
    'relative_humidity_2m_mean', 'cloud_cover_mean',
    'day_of_week', 'month', 'week_of_year', 'day_of_year'
]
target = 'Sales'

X = df[features]
y = df[target]
```

```
--- 2. Feature Engineering ---
✅ Fitur berbasis waktu berhasil dibuat.
```

Model Training and Model Evaluation:

```
[ ]
# --- 3. Model Training ---
print("\n--- 3. Model Training (Linear Regression) ---")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
print("✅ Model Linear Regression berhasil dilatih.")
```

```
--- 3. Model Training (Linear Regression) ---
✅ Model Linear Regression berhasil dilatih.
```

```
[ ]
# --- 4. Model Evaluation ---
print("\n--- 4. Hasil Evaluasi Linear Regression ---")
predictions = lr_model.predict(X_test)
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

print(f"R-squared (R²): {r2:.4f}")
print(f"Mean Absolute Error (MAE): ${mae:.2f}")
```

```
--- 4. Hasil Evaluasi Linear Regression ---
R-squared (R²): -0.0405
Mean Absolute Error (MAE): $75.50
```

Visualisasi Hasil Prediksi:

```python
# --- 5. Visualisasi Hasil Prediksi ---
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=predictions, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r', linewidth=2) # Garis ideal
plt.title('Linear Regression: Actual Sales vs. Predicted Sales')
plt.xlabel('Actual Sales ($)')
plt.ylabel('Predicted Sales ($)')
plt.tight_layout()
plt.savefig('linear_regression_results.png')
plt.show()
```