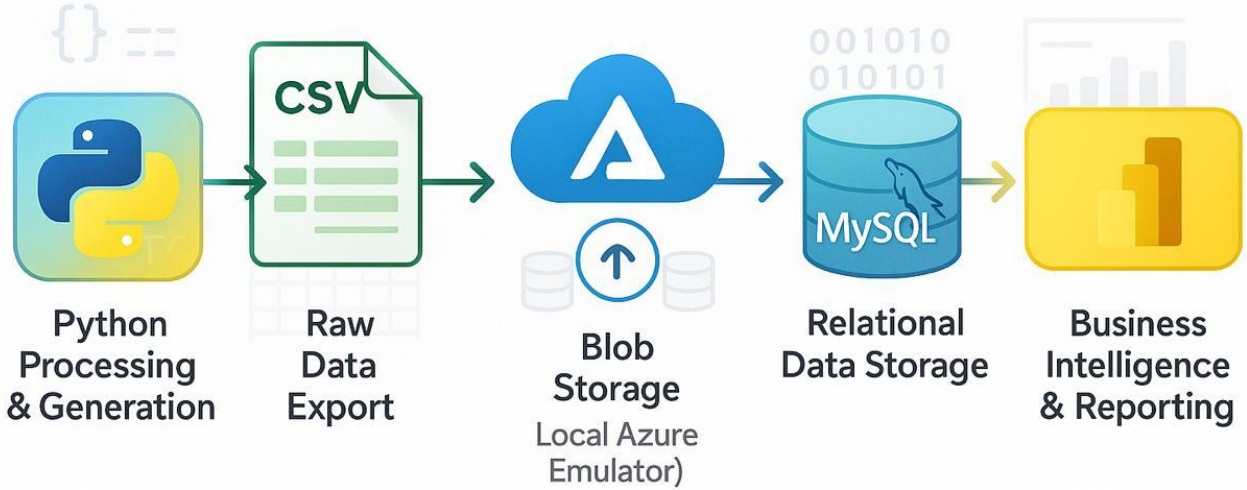
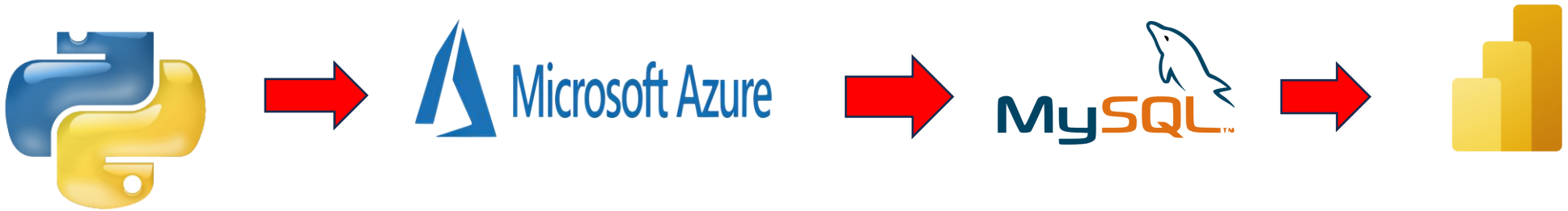
- **Title:** Tesla Financial Performance Dashboard (2015–2024)
- **Subtitle:** A Data-Driven Approach Using Python, SQL, and Power BI
- **Your Name: Tanmay Sharma**
- **Role: Data/Business-Analyst**



Project Objective

- **Goal:** Analyse Tesla's financial growth using simulated data and visualize key KPIs.
- **Tools Used:** Python (Data Simulation), MySQL (Data Storage & Querying), Power BI (Dashboarding)
- **Cloud Insight:** Used Azurite to simulate Azure Blob Storage for cloud understanding.



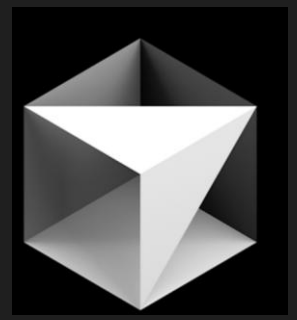
Dataset Creation (Python)

•**Explain:**

- •generated synthetic data using NumPy and Pandas.
- •Data included: Year, Revenue_Billion, Profit_Billion, Stock Prices (Open, Close, High, Low)

•**Highlight:** Why I used random data – to simulate real-world trends.

- Since real Tesla financial and stock data may not always be accessible, I used Python to generate **realistically structured random data** (via NumPy) to **simulate real-world financial trends** such as revenue growth, profit fluctuations, and market volatility.
- This approach allowed me to create a **credible mock dataset** for building a complete end-to-end analytics project — including SQL integration and Power BI dashboarding — without depending on external APIs or scraping.



```python
import numpy as np
import pandas as pd
from datetime import datetime, timedelta

# ---------- CONFIG ----------
RNG_SEED = 42
np.random.seed(RNG_SEED)
Ctrl+L to chat, Ctrl+K to generate
START_DATE = pd.Timestamp("2018-01-01")
END_DATE = pd.Timestamp("2025-08-08")  # current project date
TICKER = "TSLA"

# Output filenames (these match the MySQL table names)
STOCK_CSV = "tesla_stock_prices.csv"
QUARTERLY_CSV = "tesla_quarterly_financials.csv"
PROD_CSV = "tesla_production_sales.csv"

# ---------- 1) STOCK PRICES (daily) ----------
def generate_stock_prices(start, end, init_price=20.0, drift=0.0007, vol=0.03):
    """
    Geometric Brownian Motion with daily seasonality & occasional jumps to mimic news.
    start, end: pandas.Timestamp
    init_price: starting close price (USD)
    drift: average daily drift
    vol: daily volatility
    """
    dates = pd.bdate_range(start=start, end=end)  # business days
    n = len(dates)
    # daily returns ~ N(drift, vol)
    rand = np.random.normal(loc=drift, scale=vol, size=n)
    # occasional jumps
    jumps = np.random.choice([0, 1, -1], size=n, p=[0.98, 0.01, 0.01]) * np.random.uniform(0.03, 0.25, size=n)
    returns = rand + jumps
    price = np.zeros(n)
    price[0] = init_price
    for i in range(1, n):
        price[i] = price[i-1] * np.exp(returns[i])
    # Build OHLCV with realistic intraday ranges
    open_p = price * (1 + np.random.normal(0, 0.002, n))
    close_p = price
    high_p = np.maximum(open_p, close_p) * (1 + np.abs(np.random.normal(0.002, 0.01, n)))
    low_p = np.minimum(open_p, close_p) * (1 - np.abs(np.random.normal(0.002, 0.01, n)))
```

```python
    # Build OHLCV with realistic intraday ranges
    open_p = price * (1 + np.random.normal(0, 0.002, n))
    close_p = price
    high_p = np.maximum(open_p, close_p) * (1 + np.abs(np.random.normal(0.002, 0.01, n)))
    low_p = np.minimum(open_p, close_p) * (1 - np.abs(np.random.normal(0.002, 0.01, n)))
    # Volume simulated with trend and noise
    base_vol = 30_000_00  # base in hundreds (adjusts for scale)
    vol_trend = np.linspace(0.8, 1.8, n)  # rising interest over years
    volume = (base_vol * vol_trend * (1 + np.random.normal(0, 0.3, n))).astype(int)
    df = pd.DataFrame({
        "trade_date": dates,
        "open_price": np.round(open_p, 2),
        "high_price": np.round(high_p, 2),
        "low_price": np.round(low_p, 2),
        "close_price": np.round(close_p, 2),
        "volume": volume
    })
    # Ensure no negative or zero prices
    for col in ["open_price", "high_price", "low_price", "close_price"]:
        df[col] = df[col].clip(lower=0.01)
    return df

# ---------- 2) QUARTERLY FINANCIALS ----------
def generate_quarterly_financials(start_q="2018Q1", end_date=END_DATE):
    # build quarters from 2018-Q1 to the quarter covering end_date
    quarters = pd.period_range(start=start_q, end=end_date.to_period("Q"), freq="Q")
    rows = []
    revenue_base = 2e9  # start ~$2B per quarter in 2018 (example)
    revenue_growth_annual = 0.45  # aggressive growth per year early; we model tapering
    for q in quarters:
        years_since_2018 = q.year - 2018 + (q.quarter - 1)/4
        # taper growth over time -> logistic-like slowdown
        growth_factor = (1 + revenue_growth_annual) ** (years_since_2018) / (1 + 0.02*years_since_201
        noise = np.random.normal(0, 0.08)
        revenue = revenue_base * growth_factor * (1 + noise)
        # net income margin gradual improvement with noise
        margin_base = 0.02 + 0.005 * (q.year - 2018)  # increases over years
        margin = np.clip(margin_base + np.random.normal(0, 0.03), -0.2, 0.4)
        net_income = revenue * margin
        # EPS: scale net income / outstanding shares proxy (use a pseudo number)
        shares = 1_000_000_000  # used only to derive EPS magnitude
        eps = (net_income / shares) * 1.0  # keep EPS in dollars
```

```python
# ---------- 3) PRODUCTION & DELIVERIES (quarterly) ----------
def generate_production_sales(q_fin_df):
    rows = []
    # baseline values (2018)
    model_sx_prod_base = 20000   # low numbers early
    model_sx_del_base = 19000
    model_3y_prod_base = 40000
    model_3y_del_base = 38000
    for idx, r in q_fin_df.iterrows():
        quarter = r["quarter"]
        # growth tied loosely to revenue growth
        revenue = r["revenue"]
        # scale factors to convert revenue to production units (toy model)
        total_scale = revenue / 1e9  # billions
        # production roughly proportional to scale with noise
        model_sx_prod = int(max(0, model_sx_prod_base * (1 + 0.12 * (total_scale - 2)) * (1 + np.random.normal(0, 0.12))))
        model_sx_del = int(max(0, model_sx_del_base * (1 + 0.12 * (total_scale - 2)) * (1 + np.random.normal(0, 0.12))))
        model_3y_prod = int(max(0, model_3y_prod_base * (1 + 0.35 * (total_scale - 2)) * (1 + np.random.normal(0, 0.15))))
        model_3y_del = int(max(0, model_3y_del_base * (1 + 0.35 * (total_scale - 2)) * (1 + np.random.normal(0, 0.15))))
        rows.append({
            "quarter": quarter,
            "model_s_x_production": model_sx_prod,
            "model_s_x_deliveries": model_sx_del,
            "model_3_y_production": model_3y_prod,
            "model_3_y_deliveries": model_3y_del
        })
    df = pd.DataFrame(rows)
    return df

# ---------- RUN GENERATION ----------
if __name__ == "__main__":
    # 1. Stock prices
    print("Generating stock prices...")
    stock_df = generate_stock_prices(START_DATE, END_DATE, init_price=22.0, drift=0.0009, vol=0.035)
    stock_df.to_csv(STOCK_CSV, index=False, date_format="%Y-%m-%d")
    print(f"Saved {STOCK_CSV} ({len(stock_df)} rows)")
```

TESLA

Name	Date modified	Type	Size
Script.py	08-08-2025 13:48	PY File	7 KB
sql_script.py	08-08-2025 13:18	PY File	2 KB
tesla_financial_metrics.csv	08-08-2025 12:46	CSV File	8 KB
tesla_production_sales.csv	08-08-2025 13:48	CSV File	2 KB
tesla_quarterly_financials.csv	08-08-2025 13:48	CSV File	2 KB
tesla_stock_prices.csv	08-08-2025 13:48	CSV File	90 KB

Why I Used Azurite in This Project:

- 🌐 **Cloud Simulation Without Internet Cost**: As a fresher,
- I wanted to simulate real Azure Blob Storage without requiring a paid subscription.
- 💾 **Local CSV Upload & Access**: I used Azurite to upload Tesla's financial and stock data in .
- csv format for further integration.
- ⚙️ **Practical Cloud Integration**: Demonstrated how a cloud pipeline would look in a **real-world cloud analytics environment**, useful for large-scale companies like MAANG.
- 🛠️ **Integrated With Python**: Used Python's azure-storage-blob SDK to connect and read data from the Azurite emulator.

Key Benefits:

- ✅ No cloud charges
- ✅ Works offline
- ✅ Prepares you for real Azure Blob Storage
- ✅ Valuable for Data Engineering + Data Analyst practice


```
Azurite Blob service is starting at http://127.0.0.1:10000
Azurite Blob service is successfully listening at http://127.0.0.1:10000
Azurite Queue service is starting at http://127.0.0.1:10001
Azurite Queue service is successfully listening at http://127.0.0.1:10001
Azurite Table service is starting at http://127.0.0.1:10002
```

Microsoft Azure Storage Explorer

File Edit View Help

EXPLORER

Search for resources

Collapse all Refresh all

- Quick Access
- Emulator & Attached
  - Storage Accounts
    - (Attached Containers)
    - (Emulator - Default Ports) (Key)
    - Amazon24 (Key)
    - Amazon_Sentiments (Key)
    - AmazonCLV (Key)
    - Azurite24 (Key)
    - Netflix_Content_Recommendation (Key)
    - StockAnalysis (Key)
      - Blob Containers
        - amazon
        - amazonclv
        - amazonsentiments
        - netflix
        - netflixcontentrecommendation
        - teslastockmarketanalysis
        - View all
      - Queues
      - Tables

teslastockmarketanalysis | Get Started

Upload | Download | Open | Preview | New Folder | Select All | Properties | Delete | Undelete | Manage History | Folder Statistics | Refresh

Active blobs (default) | teslastockmarketanalysis

Name	Access Tier	Access Tier Last Modified	Last Modified	Blob Type	Content Ty
Tesla					Folder
tesla_financial_metrics.csv	Hot (inferred)	08-08-2025 19:05	08-08-2025 19:05	Block Blob	application/
tesla_production_sales.csv	Hot (inferred)	08-08-2025 19:05	08-08-2025 19:05	Block Blob	application/
tesla_quarterly_financials.csv	Hot (inferred)	08-08-2025 19:05	08-08-2025 19:05	Block Blob	application/
tesla_stock_prices.csv	Hot (inferred)	08-08-2025 19:05	08-08-2025 19:05	Block Blob	application/

TESLA

Showing 1 to 5 of 5 cached items

1 | Load more

Actions | Properties

Node Display Name	teslastockmarketanalysis
URL	http://127.0.0.1:10000/devstor
Custom Domain	
Type	Blob Container
HNS Enabled	false
Lease State	available
Lease Status	unlocked
Public Read Access	off

Activities

Clear completed | Clear successful

Transfer from 'T:\GitHub\Financial report\Tesla\' to 'devstoreaccount1/teslastockmarketanalysis/' complete: 4 items transferred (used SAS, discovery completed)
Started at: 08-08-2025 19:05, Duration: 5 seconds

Copy AzCopy Command to Clipboard

Successfully added new connection.

Database Setup (MySQL)

- **Mention:**
  - Tables Created: tesla_financials, tesla_stock_prices
  - SQL used to import data and perform analysis.
- **Sample Query:** "Find Most Profitable Year", with result.

```sql
CREATE TABLE tesla_stock_prices (
    id INT AUTO_INCREMENT PRIMARY KEY,
    date DATE NOT NULL,
    open_price DECIMAL(10,2),
    high_price DECIMAL(10,2),
    low_price DECIMAL(10,2),
    close_price DECIMAL(10,2),
    volume BIGINT
);
```

```sql
CREATE TABLE tesla_financials (
    Year INT PRIMARY KEY,
    Revenue_Billion DECIMAL(10, 2),
    Profit_Billion DECIMAL(10, 2)
);
```

Queries Covered:

- Total Revenue and Profit
- Most Profitable Year
- YoY Revenue & Profit Growth
- Revenue vs Profit

```sql
-- Find the Most Profitable Year
Select Year as Profitable_Year,
sum(Profit_Billion) as Total_Profit
from tesla_financials
Group by Year
Order by Profitable_Year DESC;
```

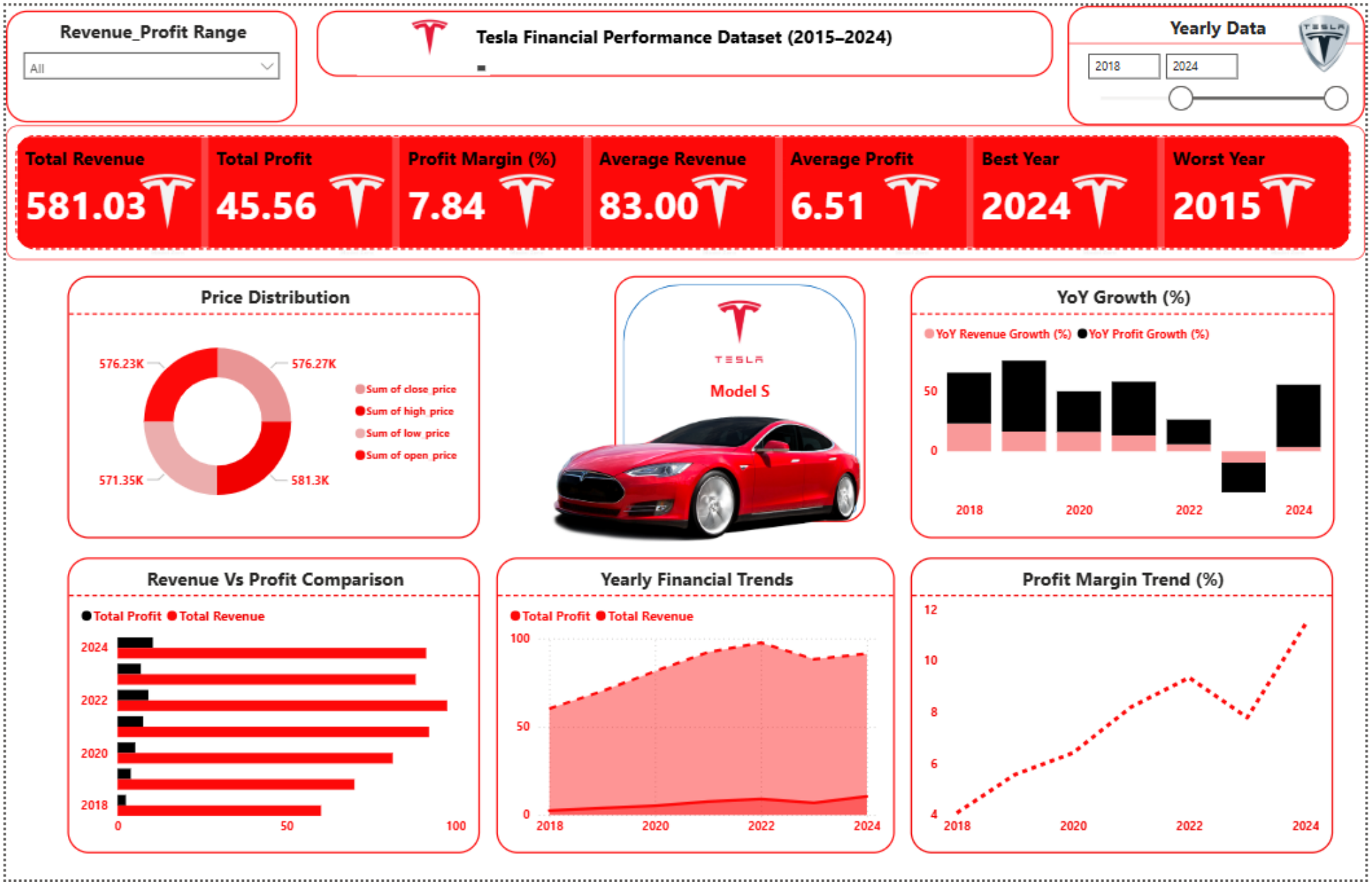
```sql
-- Calculate Total Revenue and Profit
Select sum(Revenue_billion) as Total_Revenue,
 sum(Profit_Billion) as Total_Profit
from tesla_financials;
```

```sql
-- Calculate Average Revenue and Profit
Select Round(Avg(Revenue_Billion),2) as Avg_Revenue,
Round(Avg(Profit_Billion),2) as Avg_Profit
From tesla_financials;
```

```sql
-- Year-over-Year Growth
SELECT
  Year,
  SUM(Revenue_Billion) AS Revenue_Billion,
  SUM(Profit_Billion) AS Profit_Billion,

  ROUND(
    100.0 * (SUM(Revenue_Billion) - LAG(SUM(Revenue_Billion)) OVER (ORDER BY Year))
    / NULLIF(LAG(SUM(Revenue_Billion)) OVER (ORDER BY Year), 0),
    2
  ) AS Revenue_YoY_Growth_Percent,

  ROUND(
    100.0 * (SUM(Profit_Billion) - LAG(SUM(Profit_Billion)) OVER (ORDER BY Year))
    / NULLIF(LAG(SUM(Profit_Billion)) OVER (ORDER BY Year), 0),
    2
  ) AS Profit_YoY_Growth_Percent

FROM tesla_financials
GROUP BY Year
ORDER BY Year;
```

Power BI Dashboard:

Explain Layout: Top KPIs, Filters, Visuals used (Bar, Donut, Line Charts)



Revenue_Profit Range

Multiple selections

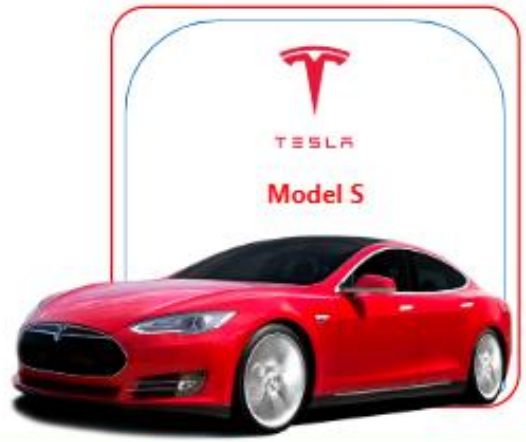
Tesla Financial Performance Dataset (2015–2024)

Yearly Data

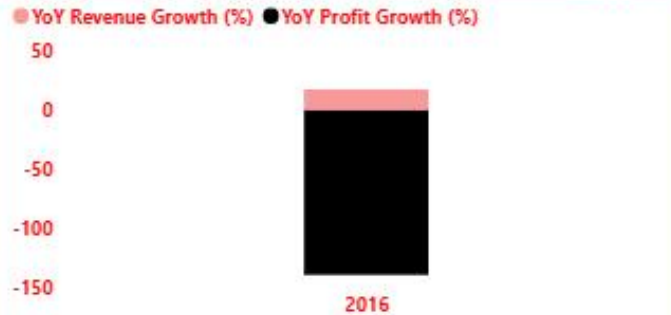
2015 2024

Total Revenue	Total Profit	Profit Margin (%)	Average Revenue	Average Profit	Best Year	Worst Year
77.89	-0.87	-1.12	38.95	-0.44	2024	2015

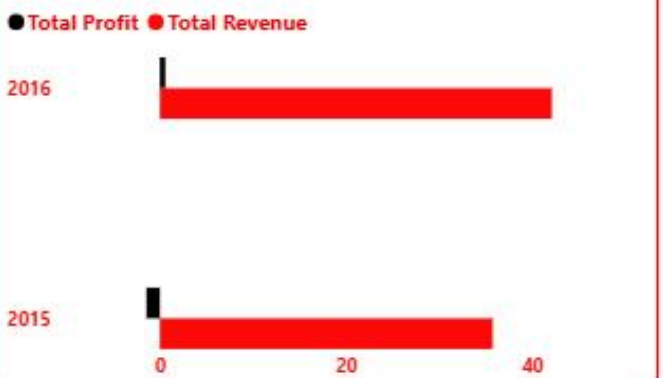
Price Distribution



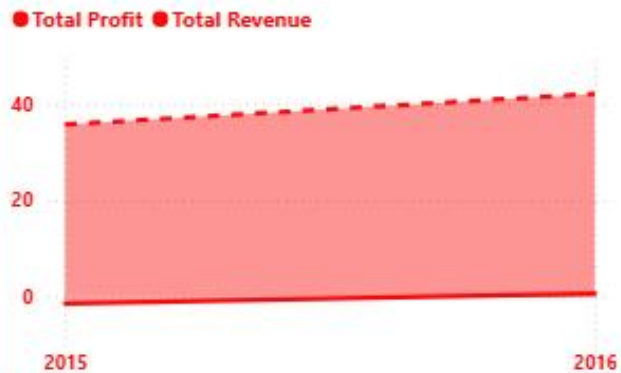
YoY Growth (%)



Revenue Vs Profit Comparison



Yearly Financial Trends



Profit Margin Trend (%)

KPIs (DAX Measures)

- **List Key KPIs:**
  - Total Revenue
  - Total Profit
  - Average Revenue & Profit
  - Best/Worst Year
  - Profit Margin (%)
- **Explain DAX:** Used to create dynamic calculations.
- **Example:** % margin logic: (Total Profit / Total Revenue) * 100

```
Best Year =
VAR MaxProfit = MAXX(ALL('tesla_financials tesla_financials'), 'tesla_financials tesla_financials'[Profit_Billion])
VAR YearWithMaxProfit =
    CALCULATE(
        Max('tesla_financials tesla_financials'[Year]),
        FILTER(
            ALL('tesla_financials tesla_financials'),
            'tesla_financials tesla_financials'[Profit_Billion] = MAXProfit
        )
    )
RETURN
    FORMAT(YearWithMaxProfit, "0")

```

```
Worst Year =
VAR MinProfit = MINX(ALL('tesla_financials tesla_financials'), 'tesla_financials tesla_financials'[Profit_Billion])
VAR YearWithMinProfit =
    CALCULATE(
        MIN('tesla_financials tesla_financials'[Year]),
        FILTER(
            ALL('tesla_financials tesla_financials'),
            'tesla_financials tesla_financials'[Profit_Billion] = MinProfit
        )
    )
RETURN
    FORMAT(YearWithMinProfit, "0")

```

Visuals Explained

Break into three sub-parts:

1.YoY Growth Chart:

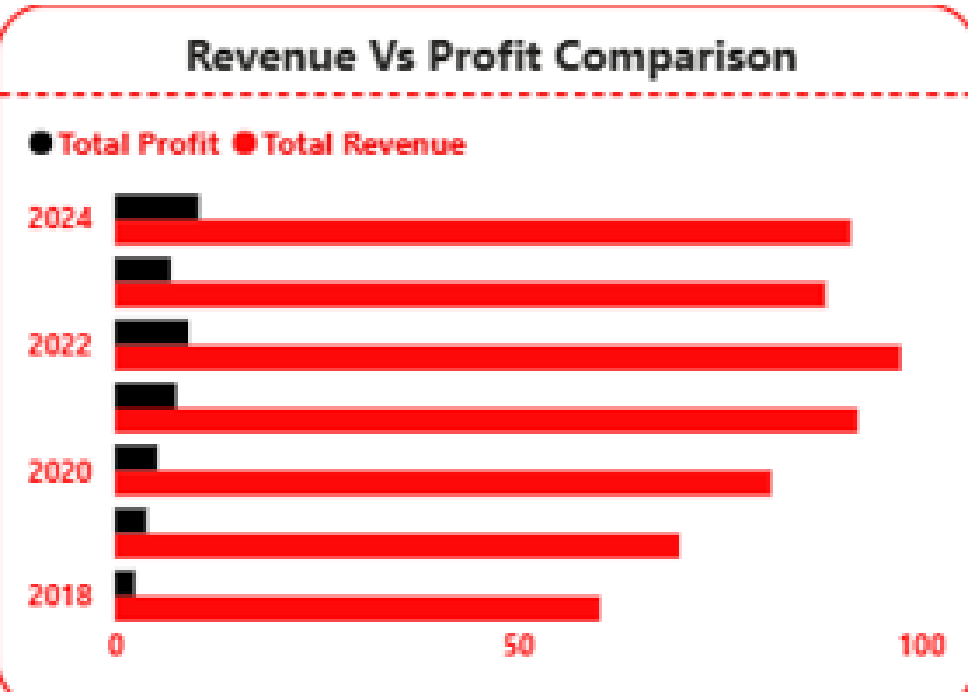
1. Shows revenue and profit percent change each year.

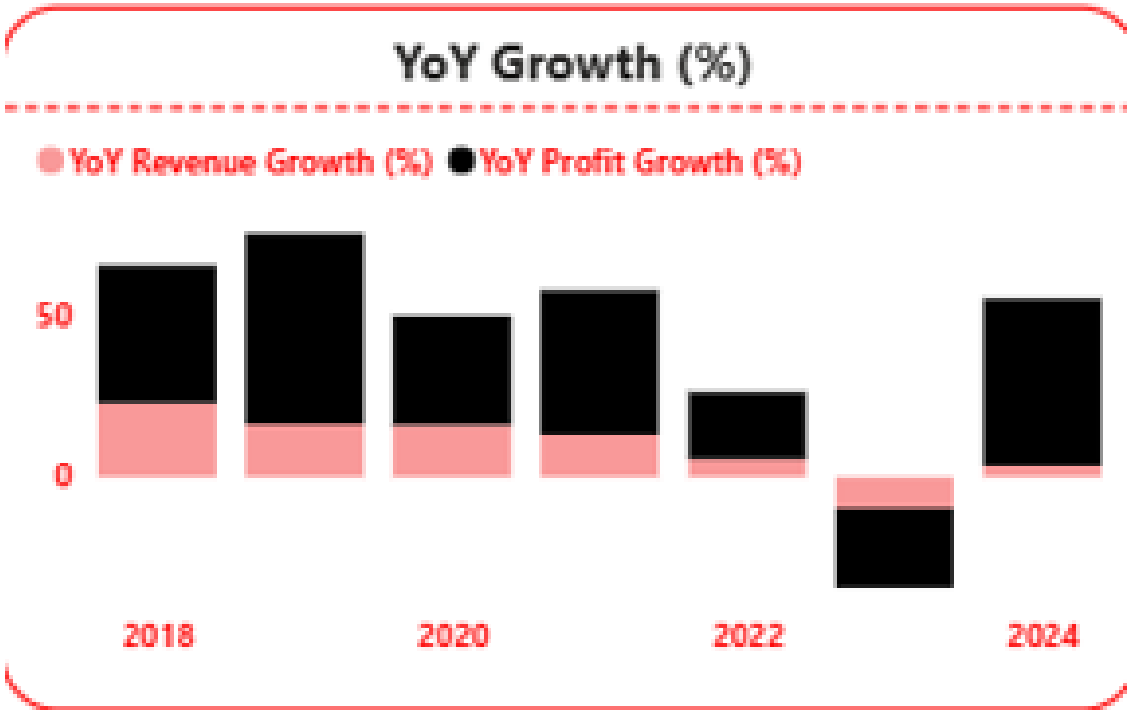
2.Revenue vs Profit Comparison (Bar):

1. Compare growth across years.

3.Profit Margin Trend (Line):

1. How efficiently Tesla turned revenue into profit over time.





Creative Touches

•3D Tesla Car Image:

- Explain how you added it as a custom image from online.
- Adds visual engagement, especially for freshers.

Challenges & Learnings

•Challenges:

- Local emulator issues with Azurite.
- HTTPS restrictions for cloud connection.
- Power BI tooltip not visible in some visuals.

•Learnings:

- Simulated full cloud-based pipeline.
- Mastered end-to-end data workflow.
- Created production-grade dashboard using only free tools.

Future Improvements:

•Use actual financial data from Tesla SEC reports.
•Host database on Azure SQL or AWS RDS (paid version).
•Implement Power BI Service (cloud sharing, refresh).
•Add forecasting and anomaly detection in Power BI.

Thank you/Let’s Connect

LinkedIn: https://www.linkedin.com/in/tanmay-sharma-800599373/

Github: https://github.com/Tanu272004/Tesla_Stock_Analysis.git