# Smart Car Wash Sales & Profit Analysis System

Name: Mahi Ayub

Institution: Presidency University

Date: 17 July 2025

# 1. Problem Statement

Like any other business, the car wash sector needs data-driven insights to boost profitability, retain valuable clients, and optimize operations. The goal of this project is to examine sales and spending data from a car wash company in order to extract valuable information that can help with decision-making. It is not enough to rely solely on intuition in a competitive market. Managing the cost of consumables like water and cleaning supplies, determining which service packages provide the best return on investment, and adjusting demand based on weather and seasonality are some of the particular operational challenges faced by a car wash.

The goal of this analysis is to directly address these issues. We will analyze historical financial data to extract important business insights by utilizing Python's data analysis libraries, including Matplotlib/Seaborn for visualization and Pandas for data manipulation.

# 2. Dataset Description

## a. Sales Data (sales.csv)

This dataset contains 2000+ records for a full year, representing customer visits to the car wash.

Key Columns:
- id
- cust_id
- name
- service_id
- service_name
- date
- time
- amount

## b. Expenditure Data (expenditure.csv)

Contains monthly fixed and variable expenses incurred by the business.

Key Columns:
- month
- year
- salaries
- water_cost
- consumables
- electricity
- rent

- maintenance
- total_expense

# 3. Design


# 4. Implementation

The project is implemented in Python, leveraging the Pandas library for data manipulation and Matplotlib for visualization. The entire system is wrapped in a command-line interface (CLI) to ensure it is modular and user-friendly.

## Part 1 & 2: High-Value + Churned Customers

This module focuses on customer segmentation to identify two critical groups: the most loyal spenders and those at risk of leaving.

- **High-Value Customers:** The system processes the sales data by grouping all transactions by customer_id and summing their total purchase amounts. The resulting list is then sorted in descending order to rank customers from most to least valuable. This allows management to easily spot top-tier clients for appreciation and loyalty programs.
- **Churn Detection:** To identify churned customers, the system first determines the last visit date for every customer. It then calculates the number of days between their last visit and the present day (or the latest date in the dataset). Any customer who has not visited in 60 or more days is flagged as 'churned'. The system then outputs a list of these customers, for whom targeted re-engagement offers, like discount coupons, can be automatically generated.

## Part 3: Time Slot Analysis

This feature analyzes sales data to determine the most profitable times of the day, helping to optimize staffing and resource allocation.

- **Categorization**: Each transaction's timestamp is categorized into one of three distinct time slots: Morning (e.g., 6:00 AM - 11:59 AM), Afternoon (12:00 PM - 4:59 PM), and Evening (5:00 PM onwards).

- **Aggregation**: Using Pandas' groupby() function, the system aggregates the total revenue for each of these time slots.
- **Analysis**: The final output clearly displays the total earnings for Morning, Afternoon, and Evening, making it simple to identify which part of the day is most lucrative.

## Part 4: Profit Analysis

The core of the financial analysis, this module provides a clear overview of the business's monthly financial performance by consolidating revenue and expenses.

- **Data Integration**: The system loads two separate files: sales.csv for all revenue transactions and expenditure.csv for all monthly operational costs (e.g., rent, salaries, supplies).
- **Calculation**: Total monthly revenue is calculated by summing up all sales for each month. The corresponding monthly expenditure is then subtracted from this total to compute the net profit using the formula:

  **Profit=Total Revenue−Total Expenditure**

- **Visualization**: The results are rendered as a bar chart using Matplotlib, providing an intuitive, at-a-glance view of monthly profitability. This visualization makes it easy to spot trends, compare performance across months, and identify periods of high profit or potential loss.

## main.py: CLI Integration

To make the system accessible and easy to use, all analytical functions are integrated into a simple command-line interface built in main.py. Instead of running separate scripts, the user can execute simple commands to access each feature

# 5. Code & Explanation

## Part 1 & 2: High Value + Churned Customers

**Purpose:**

- Identify top spending customers.
- Detect high value customers who haven't visited in over 60 days.
- Generate re-engagement coupons for these churned customers.

**Code with comments:**

```
# Group by customer and calculate total spent + last visit
customer_stats = df.groupby(['cust_id', 'name']).agg(
    total_spent=('amount', 'sum'),
    visit_count=('date', 'count'),
    last_visit=('date', 'max')
).reset_index()

# Sort to get top 10 high-value customers
top_customers = customer_stats.sort_values(by='total_spent', ascending=False).head(10)

# Detect churned customers (not visited in last 60 days)
today = df['date'].max()
churn_threshold = today - timedelta(days=60)
churned_customers = customer_stats[
    (customer_stats['last_visit'] < churn_threshold) &
    (customer_stats['total_spent'] > 1000)
]

# Generate coupon code for each churned customer
def generate_coupon(length=8):
    return ''.join(random.choices(string.ascii_uppercase + string.digits, k=length))

churned_customers = churned_customers.copy()
churned_customers['coupon_code'] = churned_customers.apply(lambda x: generate_coupon(), axis=1)
```

**Explanation:**

- Customers are analyzed based on spending and visits.
- The top 10 are printed.
- Customers inactive for 60+ days and who have spent more than ₹1000 are flagged.
- A re-engagement coupon is saved to a user-defined file.

## Part 3: Time Slot Analysis

**Purpose:**

Divide the day into 4 time slots and find which slot contributes the most revenue.

```
# Combine date and time into a single datetime column
df["datetime"] = pd.to_datetime(df["date"].astype(str) + " " + df["time"])

# Define time-of-day parts
def get_day_part(hour):
    if 6 <= hour < 12:
        return "Morning (6–12)"
    elif 12 <= hour < 16:
        return "Afternoon (12–4)"
    elif 16 <= hour < 20:
        return "Evening (4–8)"
    else:
        return "Unknown"

# Categorize time of day
df["day_part"] = df["datetime"].dt.hour.apply(get_day_part)

# Group by time of day and calculate total revenue
earnings_by_part = df.groupby("day_part")["amount"].sum().reset_index()
earnings_by_part = earnings_by_part.sort_values("amount", ascending=False)
```

**Explanation:**

- The script extracts hours from the sale time.
- It categorizes each sale into time slots (Morning, Afternoon, Evening).
- It then aggregates revenue by slot and displays it using a pie chart.

# Part 4: Profit Analysis

**Purpose:**

Compare monthly revenue and expenses to calculate and visualize profits.

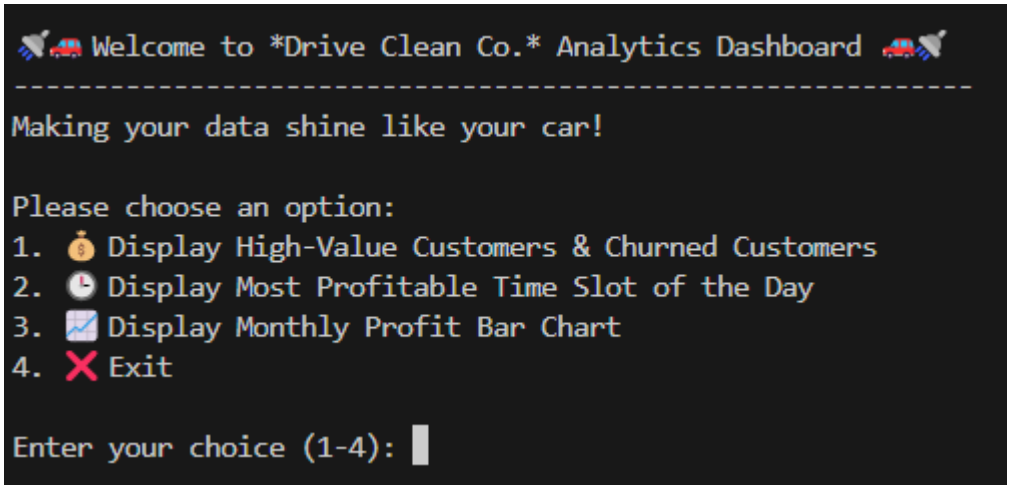
```
# Prepare sales data
sales["month"] = sales["date"].dt.month
sales["year"] = sales["date"].dt.year
monthly_revenue = sales.groupby(["month", "year"])["amount"].sum().reset_index()
monthly_revenue.rename(columns={"amount": "total_revenue"}, inplace=True)

# Ensure numeric expenses
expense_columns = ["salaries", "water_cost", "consumables", "electricity", "rent", "maintenance"]
expenses[expense_columns] = expenses[expense_columns].apply(pd.to_numeric, errors="coerce")
expenses["total_expense"] = expenses[expense_columns].sum(axis=1)

# Merge revenue and expense
profit_df = pd.merge(monthly_revenue, expenses, on=["month", "year"], how="inner")
profit_df["profit"] = profit_df["total_revenue"] - profit_df["total_expense"]

# --- Show bar chart ---
plt.figure(figsize=(10, 6))
bars = plt.bar(
    profit_df["month"],
    profit_df["profit"],
    color=["green" if p >= 0 else "red" for p in profit_df["profit"]],
    tick_label=profit_df["month"]
)
plt.title("Monthly Profit (₹)")
plt.xlabel("Month")
plt.ylabel("Profit (₹)")
plt.grid(axis="y", linestyle="--", alpha=0.7)
```

**Explanation:**

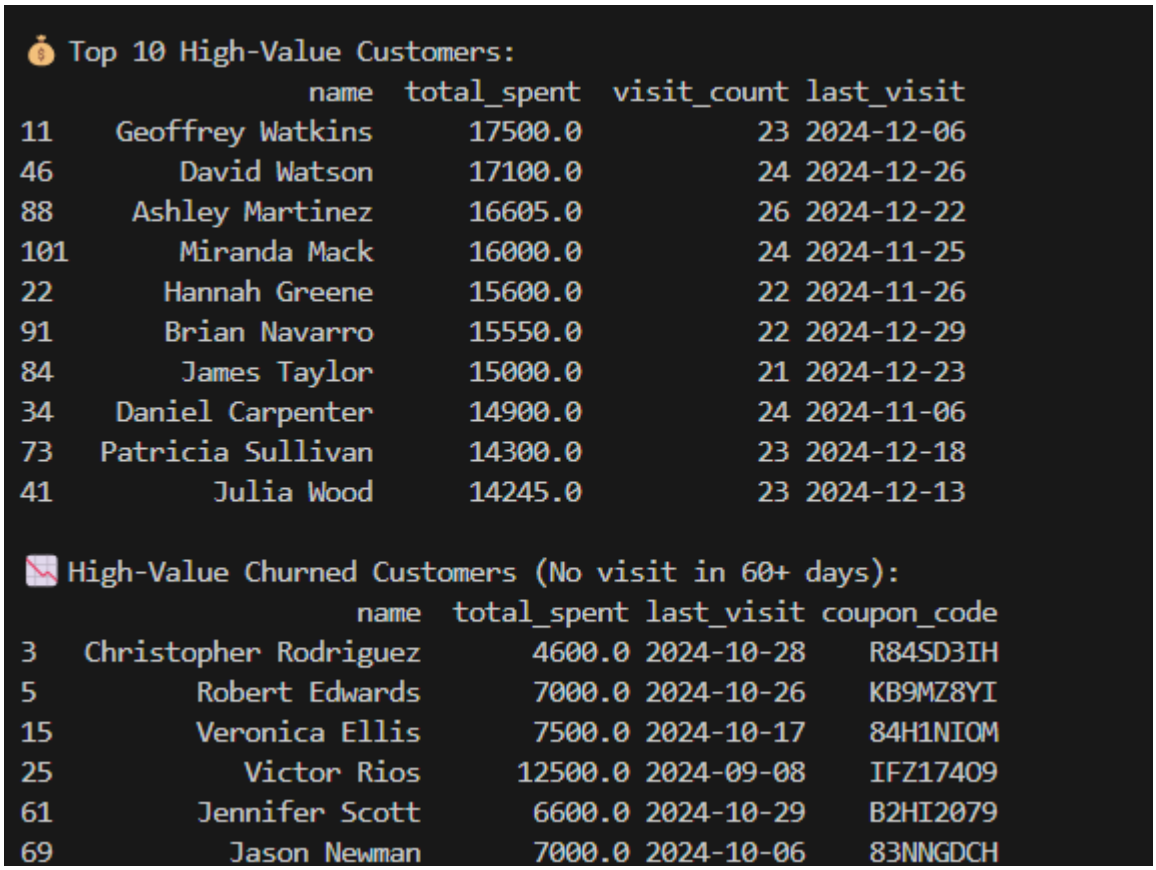
- Sales and expense data are merged month-wise.
- Profit is calculated as revenue - expense.
- A bar chart visualizes monthly profits.
- Output is saved to a CSV file as entered by the user.

# 6. Screenshots

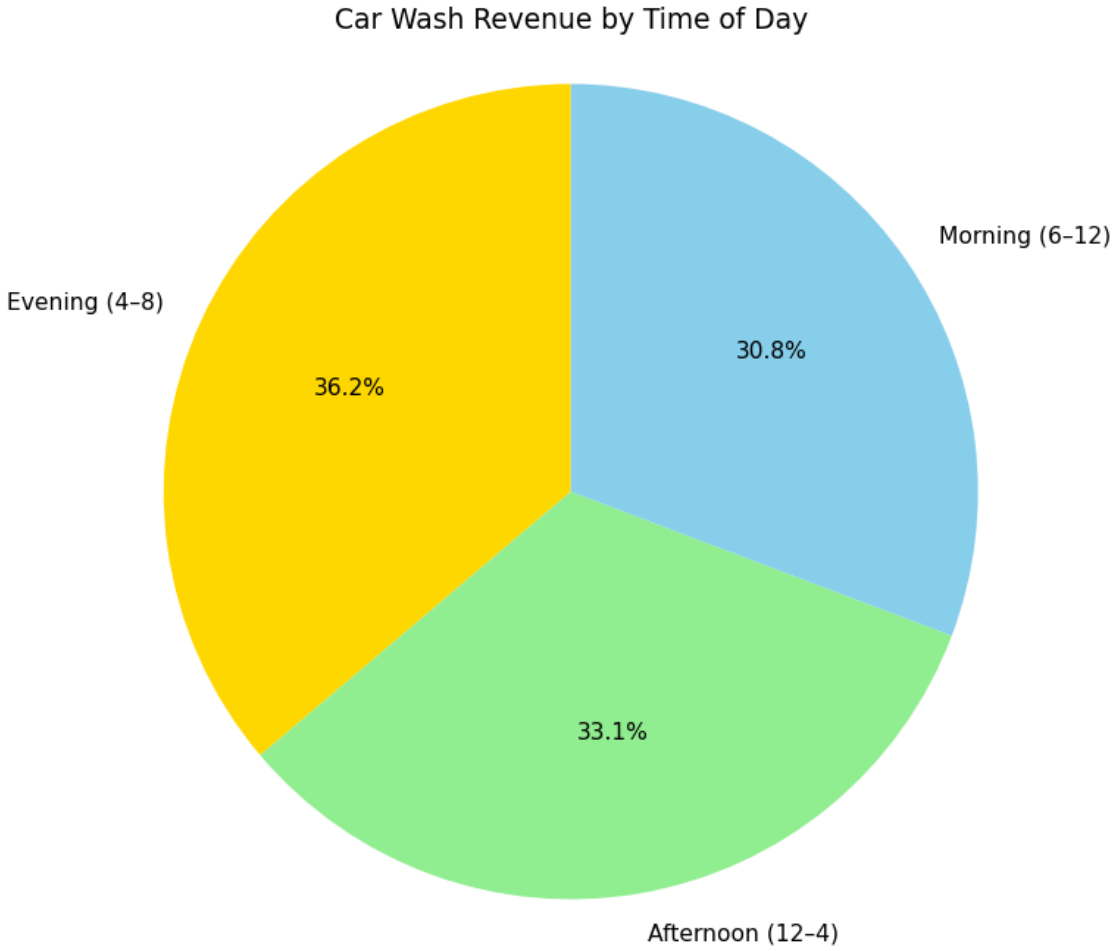
## 1: Menu



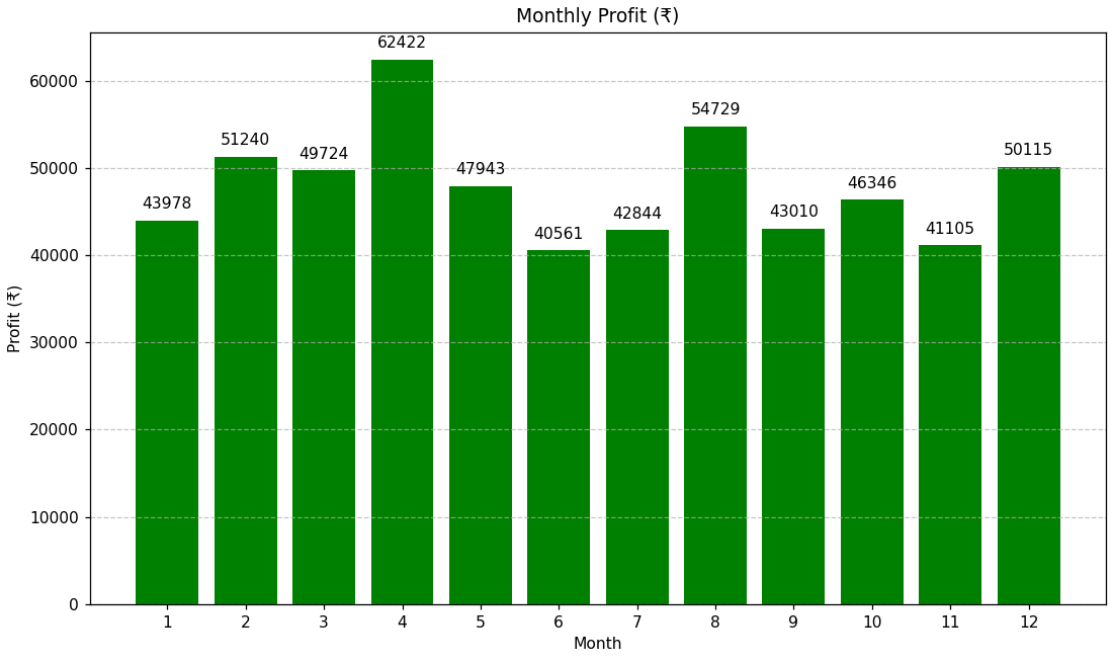

## 2: High Value & Churned Customers



## 3: Time Slot Analysis




## 4: Profit Analysis



# 6. Conclusion

This project effectively illustrates how data analysis can turn a conventional car wash into a cutting-edge, data-driven company. We have developed a potent tool that transforms decision-making from gut feeling to well-informed strategy by methodically examining sales and spending data.

- The implementation's primary goals were accomplished by:
  Finding high-value and churned customers offers a clear route to enhancing client retention and starting focused advertising campaigns.

- By analysing time slot performance, management is able to maximise revenue during peak hours, optimise staffing, and effectively manage resources.

- Calculating Monthly Profitability provides a succinct and straightforward overview of the company's financial situation, highlighting prosperous months and pointing out times that require attention.

This analysis establishes a strong proof-of-concept, despite the fact that it was conducted on synthetic data. For upcoming improvements, the modular framework provides a great starting point. Possible next actions consist of:

- Creating a Real-Time Dashboard: To give management access to real-time KPIs so they can make decisions quickly.

- By incorporating predictive analytics, operations can be further optimised by forecasting demand based on past patterns and outside variables like the weather.

- Automating Customer Engagement: Setting up a system that sends loyalty rewards or re-engagement coupons by email or SMS on a regular basis.

To sum up, this project demonstrates that using data effectively can give even small and medium-sized enterprises a major competitive edge. The information provided here is more than just a report; it is a set of practical guidelines for boosting revenue, improving client retention, and guaranteeing long-term expansion.

# 7. Bibliography


1. Python Software Foundation. (2024). *Python (Version 3.x)*. Retrieved from https://www.python.org
2. McKinney, W. (2024). *pandas: Python Data Analysis Library*. Retrieved from https://pandas.pydata.org
3. Hunter, J. D., et al. (2024). *matplotlib: Visualization Library in Python*. Retrieved from https://matplotlib.org
4. Waskom, M. L. (2024). *seaborn: Statistical Data Visualization Library*. Retrieved from https://seaborn.pydata.org
5. Stack Overflow. (2024). *Community Discussions and Code Solutions*. Retrieved from https://stackoverflow.com
6. Real Python. (2024). *Python Tutorials and Examples*. Retrieved from https://realpython.com
7. Self-Generated Dataset. (2025). *Car Wash Sales and Expenditure Dataset*. Created using custom Python scripts.
8. GeeksforGeeks. (2024). *Python Programming Examples and Tutorials*. Retrieved from https://www.geeksforgeeks.org
9. matplotlib documentation. (2024). *Creating Plots and Visualizations*. Retrieved from https://matplotlib.org/stable/contents.html
10. Official CSV Format Specification. Retrieved from https://tools.ietf.org/html/rfc4180