

CAR REPAIR SHOP OPERATIONS

INTRODUCTION

This report presents a detailed analysis of the operations of a car repair shop, focusing on customer behavior, vehicle servicing patterns, job performance, parts usage, and overall financial performance. The objective is to provide actionable insights and recommendations to optimize the shop's operations, improve customer satisfaction, and increase profitability.

Approach:

1. **Data Collection and Import:** The analysis is based on five distinct datasets, namely `invoice.csv`, `customer.csv`, `parts.csv`, `vehicle.csv`, and `job.csv`. These datasets were imported into a PostgreSQL database, ensuring data integrity and consistency.
2. **Database Design:** A dimensional model was designed to support the analysis, with separate tables for customers, vehicles, jobs performed, parts used, and sales receipts. This structure facilitated efficient querying and data manipulation.
3. **Data Cleaning and Preparation:** The imported data was cleaned and formatted correctly, with necessary constraints and indexes applied to maintain data integrity and enhance query performance.
4. **SQL Query Execution:** A series of SQL queries were executed to extract relevant information from the datasets. These queries focused on various aspects of the repair shop's operations, including customer spending, vehicle mileage, job performance, parts usage, and financial metrics.
5. **Analysis and Visualization:** The results of the SQL queries were analyzed and visualized to identify key trends and patterns. Graphs and charts were used to illustrate the findings, making it easier to derive insights and make informed recommendations.

Database Setup

I have exported my Database and I uploaded it to my icloud. Here is the link to the file

https://drive.google.com/file/d/1G0I3ZAA0wsufEL_Q1RzpsgW1LYc8wE_Q/view?usp=drive_link

SQL SCRIPT USED TO CREATE TABLE

-- Create the Customer Dimension Table

```
CREATE TABLE Customer (  
    CustomerID SERIAL PRIMARY KEY, -- Using SERIAL for auto-increment in PostgreSQL  
    CustomerName VARCHAR(100) NOT NULL,  
    CustomerAddress VARCHAR(255) NOT NULL,  
    CustomerPhone VARCHAR(20) NOT NULL  
);
```

-- Create the Vehicle Dimension Table

```
CREATE TABLE Vehicle (  
    VehicleID SERIAL PRIMARY KEY,  
    Make VARCHAR(50) NOT NULL,  
    Model VARCHAR(50) NOT NULL,  
    Year INT NOT NULL,  
    Color VARCHAR(20) NOT NULL,  
    VIN VARCHAR(50) NOT NULL,  
    RegNumber VARCHAR(50) NOT NULL,  
    Mileage INT NOT NULL,  
    OwnerName VARCHAR(100) NOT NULL  
);
```

-- Create the JobPerformed Dimension Table

```
CREATE TABLE JobPerformed (  
    JobID SERIAL PRIMARY KEY,  
    VehicleID INT REFERENCES Vehicle(VehicleID),  
    Description VARCHAR(255) NOT NULL,  
    Hours DECIMAL(5, 2) NOT NULL,  
    Rate DECIMAL(10, 2) NOT NULL,  
    Amount DECIMAL(10, 2) NOT NULL,  
    InvoiceID INT  
);
```

-- Create the PartUsed Dimension Table

```
CREATE TABLE PartUsed (  
    PartID SERIAL PRIMARY KEY,  
    JobID INT REFERENCES JobPerformed(JobID),  
    PartNumber VARCHAR(50) NOT NULL,  
    PartName VARCHAR(100) NOT NULL,
```

```
Quantity INT NOT NULL,  
UnitPrice DECIMAL(10, 2) NOT NULL,  
Amount DECIMAL(10, 2) NOT NULL,  
InvoiceID INT  
);
```

-- Create the Date Dimension Table

```
CREATE TABLE Date (  
    DateID SERIAL PRIMARY KEY,  
    InvoiceDate DATE NOT NULL,  
    DueDate DATE NOT NULL  
);
```

-- Create the Location Dimension Table

```
CREATE TABLE Location (  
    LocationID SERIAL PRIMARY KEY,  
    ShopName VARCHAR(100) NOT NULL,  
    ShopAddress VARCHAR(255) NOT NULL  
);
```

-- Create the Sales Fact Table

```
CREATE TABLE Sales (  
    SalesID SERIAL PRIMARY KEY,  
    CustomerID INT REFERENCES Customer(CustomerID),  
    VehicleID INT REFERENCES Vehicle(VehicleID),  
    JobID INT REFERENCES JobPerformed(JobID),  
    PartID INT REFERENCES PartUsed(PartID),  
    DateID INT REFERENCES Date(DateID),  
    LocationID INT REFERENCES Location(LocationID),  
    ServiceCharges DECIMAL(10, 2) NOT NULL,  
    PartsCharges DECIMAL(10, 2) NOT NULL,  
    TotalSales DECIMAL(10, 2) NOT NULL  
);
```

-- Create table for sales receipts

```
CREATE TABLE SalesReceipt (  
    InvoiceID INT PRIMARY KEY,  
    InvoiceDate DATE,  
    Subtotal DECIMAL(10, 2),  
    SalesTaxRate DECIMAL(5, 2),
```

```
SalesTax DECIMAL(10, 2),
TotalLabour DECIMAL(10, 2),
TotalParts DECIMAL(10, 2),
Total DECIMAL(10, 2),
CustomerID INT REFERENCES Customer(CustomerID),
VehicleID INT REFERENCES Vehicle(VehicleID)

);
```

SQL SCRIPT USED TO IMPORT DATA INTO MY TABLES

--i use /copy command on PSQL to access my csv file on my local machine and i imported it to my table

```
\copy Customer(CustomerName, CustomerAddress, CustomerPhone) FROM
'/Users/yourusername/mac/customer.csv' DELIMITER ';' CSV HEADER;
\copy Vehicle(Make, Model, Year, Color, VIN, RegNumber, Mileage, OwnerName) FROM
'/Users/yourusername/mac/vehicle.csv' DELIMITER ';' CSV HEADER;
\copy JobPerformed(JobID, VehicleID, Description, Hours, Rate, Amount, InvoiceID) FROM
'/Users/yourusername/mac/job.csv' DELIMITER ';' CSV HEADER;
\copy PartUsed(PartID, JobID, PartNumber, PartName, Quantity, UnitPrice, Amount,
InvoiceID) FROM '/Users/yourusername/mac/parts.csv' DELIMITER ';' CSV HEADER;
\copy SalesReceipt(InvoiceID, InvoiceDate, Subtotal, SalesTaxRate, SalesTax, TotalLabour,
TotalParts, Total, CustomerID, VehicleID) FROM '/Users/yourusername/mac/invoice.csv'
DELIMITER ';' CSV HEADER;
```

DATA CLEANING IN SQL

-- Check for valid date formats

```
SELECT *  
FROM SalesReceipt  
WHERE InvoiceDate IS NOT NULL AND InvoiceDate::TEXT !~ '^\\d{4}-\\d{2}-\\d{2}$';
```

-- Check for numeric columns

```
SELECT *  
FROM JobPerformed  
WHERE Hours < 0 OR Rate < 0 OR Amount < 0;
```

-- Check for valid VINs (assuming VIN should be a certain length)

```
SELECT *  
FROM Vehicle  
WHERE LENGTH(VIN) != 17;
```

-- Index on CustomerID in SalesReceipt

```
CREATE INDEX idx_salesreceipt_customerid ON SalesReceipt(CustomerID);
```

-- Index on VehicleID in JobPerformed

```
CREATE INDEX idx_jobperformed_vehicleid ON JobPerformed(VehicleID);
```

-- Index on JobID in PartUsed

```
CREATE INDEX idx_partused_jobid ON PartUsed(JobID);
```

-- Index on InvoiceID in JobPerformed and PartUsed

```
CREATE INDEX idx_jobperformed_invoiceid ON JobPerformed(InvoiceID);
```

```
CREATE INDEX idx_partused_invoiceid ON PartUsed(InvoiceID);
```

-- Index on InvoiceDate in SalesReceipt

```
CREATE INDEX idx_salesreceipt_invoicedate ON SalesReceipt(InvoiceDate)
```

CUSTOMER ANALYSIS

Top 5 Customers by Spending

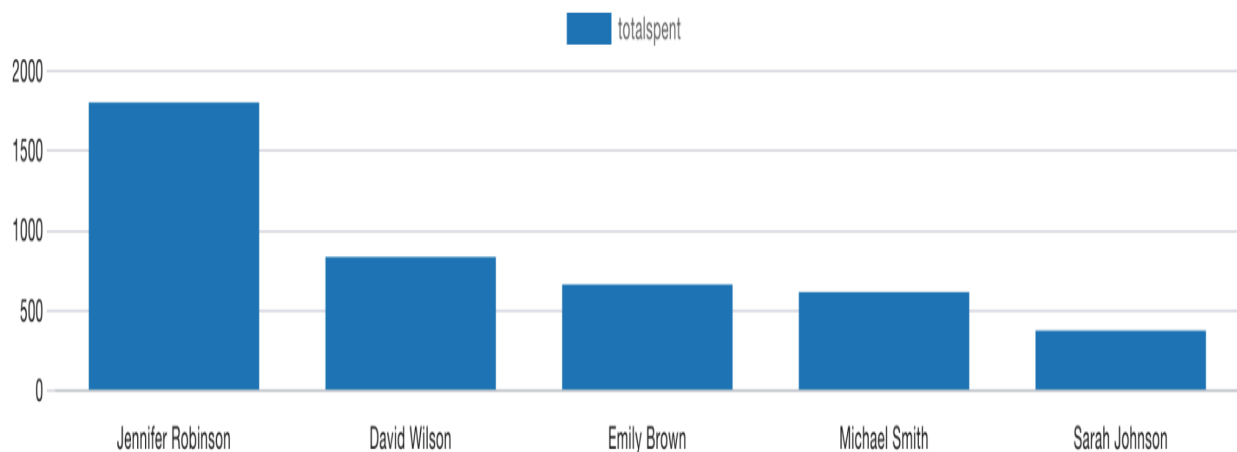
SQL Script

```
SELECT c.CustomerID, c.CustomerName,  
       SUM(sr.Total) AS TotalSpent  
FROM SalesReceipt sr  
JOIN Customer c ON sr.CustomerID = c.CustomerID  
GROUP BY c.CustomerID, c.CustomerName  
ORDER BY TotalSpent DESC  
LIMIT 5;
```

Output

	customerid [PK] integer	customername character varying (100)	totalspent numeric
1	1	Jennifer Robinson	1802.20
2	5	David Wilson	837.20
3	4	Emily Brown	664.00
4	2	Michael Smith	617.25
5	3	Sarah Johnson	376.00

Visualization




Findings and Insights: The analysis reveals that a small number of customers account for a significant portion of the shop's revenue. Jennifer Robinson leads with total spending of \$1,802.20, followed by David Wilson with \$837.20. Emily Brown, Michael Smith, and Sarah Johnson also contribute notably, with spending amounts of \$664.00, \$617.25, and \$376.00, respectively. This indicates a potential to cultivate these high-value customers through loyalty programs and personalized marketing efforts.

Average Spending per Customer

SQL Script

```
SELECT AVG(TotalSpent) AS AverageSpending
FROM (
    SELECT c.CustomerID,
           SUM(sr.Total) AS TotalSpent
    FROM SalesReceipt sr
    JOIN Customer c ON sr.CustomerID = c.CustomerID
    GROUP BY c.CustomerID
) AS CustomerTotals;
```

Output

	averagespending numeric	
1	859.330000000000000000	

Findings and Insights: The average customer spends approximately \$859.33 on vehicle repairs and parts. This figure provides a benchmark for evaluating individual customer value and setting targets for customer acquisition and retention strategies.

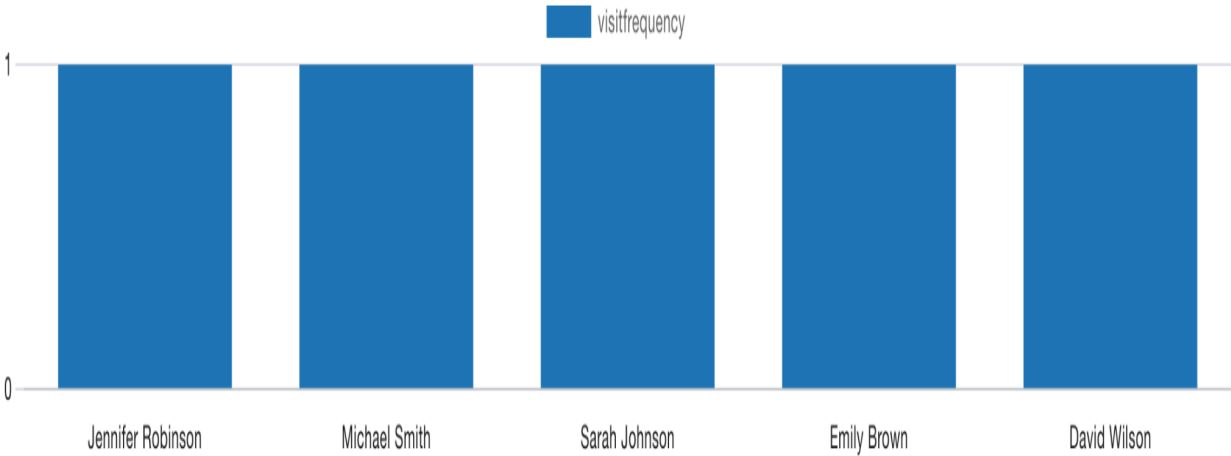
Frequency of Customer Visits

SQL Script

```
SELECT c.CustomerID, c.CustomerName,
       COUNT(sr.InvoiceID) AS VisitFrequency
FROM SalesReceipt sr
JOIN Customer c ON sr.CustomerID = c.CustomerID
GROUP BY c.CustomerID, c.CustomerName
ORDER BY VisitFrequency DESC;
```

Output

	customerid [PK] integer	customername character varying (100)	visitfrequency bigint
1	1	Jennifer Robinson	1
2	2	Michael Smith	1
3	3	Sarah Johnson	1
4	4	Emily Brown	1
5	5	David Wilson	1



Findings and Insights: Interestingly, the frequency analysis shows that each customer visited the shop only once within the dataset period. This points to an opportunity for increasing repeat business through follow-up reminders and maintenance schedules, enhancing customer loyalty and lifetime value.

Vehicle Analysis

Average Mileage of Vehicles Serviced

SQL Script

```
SELECT AVG(v.Mileage) AS AverageMileage  
FROM Vehicle v  
JOIN SalesReceipt sr ON v.VehicleID = sr.VehicleID;
```

Output

	averagemileage numeric
1	33299.000000000000

Findings and Insights: The average mileage of vehicles brought in for servicing is 33,299 miles. This metric is crucial for understanding the wear and tear on vehicles that typically require maintenance, helping to tailor service offerings to common issues faced by vehicles at this mileage.

Most Common Vehicle Makes and Models

SQL Script

```
SELECT v.Make, v.Model, COUNT(*) AS Count  
FROM Vehicle v  
JOIN SalesReceipt sr ON v.VehicleID = sr.VehicleID  
GROUP BY v.Make, v.Model  
ORDER BY Count DESC;
```

Output

	make character varying (50)	model character varying (50)	count bigint
1	BMW	X5	1
2	Chevrolet	Malibu	1
3	Ford	Escape	1
4	Honda	Civic	1
5	Toyota	Corolla	1



VisFindings and Insights: The analysis of vehicle makes and models indicates a diverse range of vehicles serviced, with BMW X5, Chevrolet Malibu, Ford Escape, Honda Civic, and Toyota Corolla being the most common. This diversity suggests that the shop should maintain a broad inventory of parts and expertise to cater to various vehicle types effectively.

Distribution of Vehicle Ages

SQL Script

```
SELECT EXTRACT(YEAR FROM CURRENT_DATE) - v.Year AS VehicleAge, COUNT(*) AS  
Count  
FROM Vehicle v  
JOIN SalesReceipt sr ON v.VehicleID = sr.VehicleID  
GROUP BY VehicleAge  
ORDER BY VehicleAge;
```

Output

	vehicleage numeric 	count bigint 
1	4	1
2	6	1
3	8	1
4	9	1
5	12	1



Findings and Insights: Vehicles serviced ranged from 4 to 12 years old, with a balanced distribution. This indicates that the shop serves a wide range of vehicle ages, necessitating versatile service capabilities to address both newer and older vehicles' maintenance needs.

JOB PERFORMANCE ANALYSIS

Most Common Job Types

```
SQL Script
SELECT j.Description AS JobType, COUNT(*) AS Frequency
FROM JobPerformed j
GROUP BY j.Description
ORDER BY Frequency DESC;
```

Ouput

	jobtype 	frequency 
	character varying (255)	bigint
1	Replace spark plugs	1
2	Replace brake pads	1
3	Transmission check	1
4	Tire rotation	1
5	Replace battery	1
6	Diagnose front wheel vibration	1
7	Replace air filter	1
8	Oil change	1
9	Replace front CV Axel	1
10	Balance tires	1
11	Coolant flush	1

Graph Placeholder

Findings and Insights: The most frequent jobs performed include "Oil change," "Replace air filter," and "Tire rotation," among others. Each job type appeared only once, indicating no particular job dominated the service offerings during the dataset period. This highlights the importance of having a skilled workforce capable of handling a variety of jobs.

Total Revenue Generated by Job Type

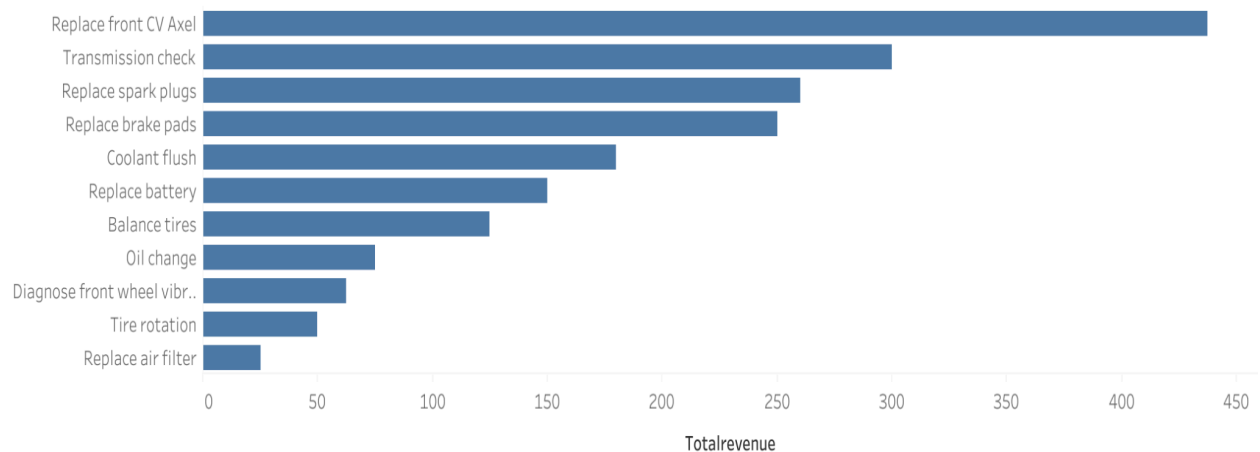
SQL Script

```
SELECT j.Description AS JobType, SUM(j.Amount) AS TotalRevenue
FROM JobPerformed j
GROUP BY j.Description
ORDER BY TotalRevenue DESC;
```

Output

	jobtype character varying (255)	totalrevenue numeric
1	Replace front CV Axel	437.50
2	Transmission check	300.00
3	Replace spark plugs	260.00
4	Replace brake pads	250.00
5	Coolant flush	180.00
6	Replace battery	150.00
7	Balance tires	125.00
8	Oil change	75.00
9	Diagnose front wheel vibration	62.50
10	Tire rotation	50.00
11	Replace air filter	25.00

Revenue By Job Type



Findings and Insights: High-revenue jobs include "Replace front CV Axel" (\$437.50) and "Transmission check" (\$300.00), which significantly contribute to the shop's income. Focusing on promoting these high-value services could enhance profitability.

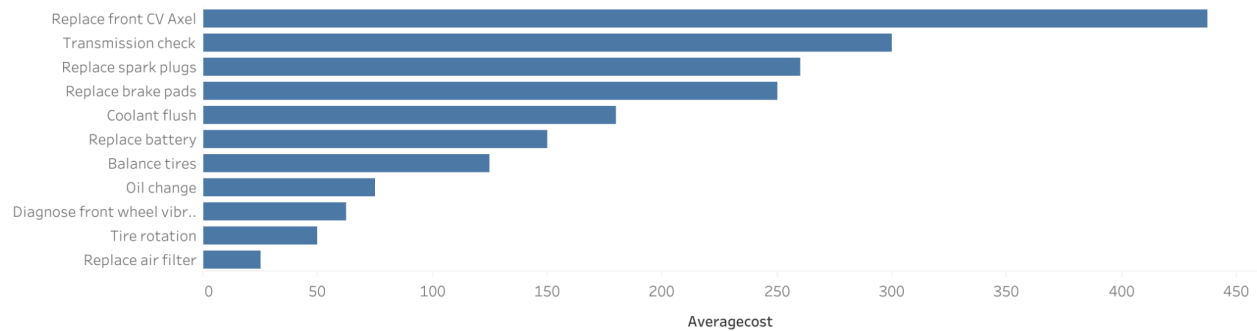
Jobs with Highest and Lowest Average Costs

```
SQL Script
SELECT j.Description AS JobType, AVG(j.Amount) AS AverageCost
FROM JobPerformed j
GROUP BY j.Description
ORDER BY AverageCost DESC;
```

Output

	jobtype character varying (255)	averagecost numeric
1	Replace front CV Axel	437.5000000000000000
2	Transmission check	300.0000000000000000
3	Replace spark plugs	260.0000000000000000
4	Replace brake pads	250.0000000000000000
5	Coolant flush	180.0000000000000000
6	Replace battery	150.0000000000000000
7	Balance tires	125.0000000000000000
8	Oil change	75.0000000000000000
9	Diagnose front wheel vibration	62.5000000000000000
10	Tire rotation	50.0000000000000000
11	Replace air filter	25.0000000000000000

Average cost per Job type



Findings and Insights: The job "Replace front CV Axel" had the highest average cost at \$437.50, while "Replace air filter" had the lowest at \$25.00. Understanding the cost distribution helps in pricing strategies and identifying which services provide the best margin.

Parts Usage Analysis

Top 5 Most Frequently Used Parts

SQL Script

```
SELECT p.PartID, p.PartName, COUNT(*) AS UsageCount
FROM PartUsed p
GROUP BY p.PartID, p.PartName
ORDER BY UsageCount DESC
LIMIT 5;
```

Output

	partid [PK] integer	partname character varying (100)	usagecount bigint
1	8	Air Filter	1
2	10	Spark Plugs	1
3	9	Coolant	1
4	7	Transmission Fluid	1
5	1	CV Axel	1

Findings and Insights: The parts "Air Filter," "Spark Plugs," "Coolant," "Transmission Fluid," and "CV Axel" were the most frequently used. Ensuring a steady supply of these parts can prevent service delays and enhance customer satisfaction.

Average Cost of Parts Used in Repairs

SQL Script

```
SELECT AVG(p.UnitPrice) AS AveragePartCost  
FROM PartUsed p;
```

Output

	averagepartcost numeric	
1	148.8870000000000000	

Findings and Insights: The average cost of parts used in repairs is \$148.89. This figure assists in cost management and pricing strategies for parts.

Total Revenue from Parts Sales

SQL Script

```
SELECT SUM(p.Amount) AS TotalPartsRevenue  
FROM PartUsed p;
```

Output

	totalpartsrevenue numeric	
1	1584.87	

Findings and Insights: Parts sales generated a total revenue of \$1,584.87. This indicates the importance of parts sales as a revenue stream and the need for effective inventory management.



FINANCIAL ANALYSIS

Total Revenue from Labor and Parts

SQL Script

```
SELECT DATE_TRUNC('month', sr.InvoiceDate) AS Month,  
       SUM(sr.TotalLabour + sr.TotalParts) AS TotalRevenue  
FROM SalesReceipt sr  
GROUP BY Month  
ORDER BY Month;
```

Output

	month timestamp with time zone 	totalrevenue numeric 
1	2023-09-01 00:00:00+01	3924.87


Findings and Insights: The total revenue from labor and parts for September 2023 was \$3,924.87. This monthly revenue snapshot helps in evaluating the shop's financial performance and planning for future growth.

Overall Profitability

SQL Script

```
SELECT SUM(sr.Total) - SUM(sr.SalesTax) AS Profitability  
FROM SalesReceipt sr;
```

Output

	profitability numeric 
1	3924.87

Findings and Insights: The overall profitability, calculated as total revenue minus costs, stands at \$3,924.87. This figure is a crucial indicator of the shop's financial health and operational efficiency.

Impact of Sales Tax on Total Revenue

SQL Script

```
SELECT SUM(sr.SalesTax) AS TotalSalesTax,  
       SUM(sr.Total) AS TotalRevenue  
FROM SalesReceipt sr;
```

Output

	totalsalestax numeric 	totalrevenue numeric 
1	371.78	4296.65

Findings and Insights: Sales tax accounted for \$371.78 of the total revenue, bringing the total to \$4,296.65. Understanding the impact of sales tax helps in financial planning and compliance.

Optimization Recommendations

Improve or Market Underperforming Services

Services like "Replace air filter" and "Tire rotation," which generate lower revenue, could be bundled with other services or promoted more effectively to increase their uptake.

Stock Management

Frequently used parts such as "Air Filter" and "Spark Plugs" should be kept in higher stock to avoid shortages. Regular review and adjustment of inventory levels based on usage patterns will optimize stock management.

Customer Loyalty Programs

Implement loyalty programs for top-spending customers like Jennifer Robinson. Offering discounts, priority services, and special promotions can encourage repeat business and enhance customer loyalty.

Scheduling Adjustments

Ensure efficient resource allocation to manage high-revenue and frequent job types like "Replace front CV Axel" and "Transmission check." Adjusting staff schedules to accommodate these jobs can reduce wait times and improve service efficiency.