Adventure Works 2019 Data Analytics Report

Data Analytics Interim Project Proposal

Overview

Our team used the AdventureWorks2019 dataset to extract actionable business insights by addressing six key analytical questions. Utilising SQL for data retrieval and Python for visualisation, we uncovered important trends and relationships involving sales, employee performance, and store metrics.

Proposed by

- Zahra Noury

Timeframe

- Completion by 17 July 2025
- Presentation on 21 July 2025

What are the regional sales in the best performing country?

Analysis, Insights & Action Plan

Methodology

Two-stage CTE query

- Step 1 – rank every country by total revenue.
- Step 2 – isolate the top country (US) and sum sales by StateProvince.

Validation

- Repeated analysis with an address-based join to confirm state-level accuracy.

Visualisation

- Produced three complementary charts
    - Figure 1 – bar chart of the five leading US sales territories.
    - Figure 2 – full 27-state bar chart, sorted by revenue.
    - Figure 3 – choropleth map for an instant geographic view.

Findings

Southwest territory is #1 at $27.15 M; Northeast is last at $7.82 M (Figure 1).

California, Washington, and Texas together deliver = 55 % of US revenue (Figure 2).

Coastal dominance is clear; most central states cluster in the mid-tier at roughly $9 M.

Insight

US revenue is heavily concentrated in three coastal states. Incremental spend in these markets should yield the quickest return on investment.

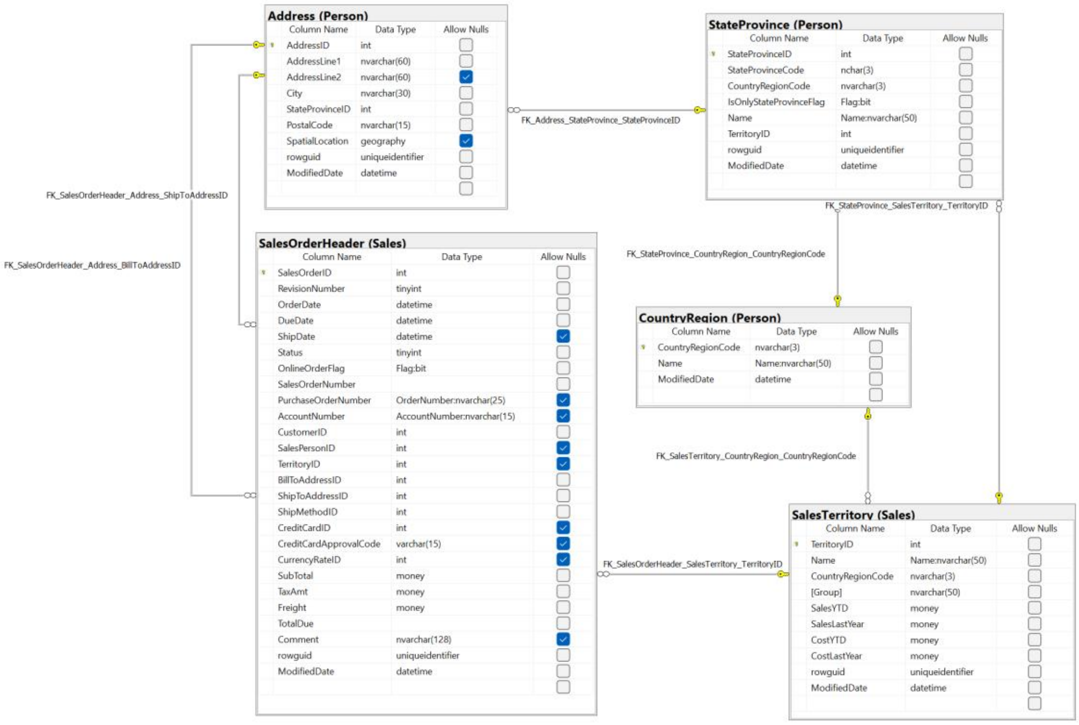
Action Plan

Concentrate marketing in CA, WA, TX over the next two quarters.

Diagnose under-performing territories (Northeast & central) — examine store footprint, product assortment, and local demand drivers.

Analytical Workflow & Findings

Schema Diagram



SQL Extraction Query & Results

```
WITH CountryTotalSales AS (
        SELECT Person.CountryRegion.Name AS Country
        , SUM(Sales.SalesOrderHeader.TotalDue) AS CountrySales
        FROM Sales.SalesOrderHeader
        INNER JOIN Person.Address
        ON Sales.SalesOrderHeader.BillToAddressID = Person.Address.AddressID
        INNER JOIN Person.StateProvince
ON Person.Address.StateProvinceID = Person.StateProvince.StateProvinceID
        INNER JOIN Person.CountryRegion
ON Person.StateProvince.CountryRegionCode = Person.CountryRegion.CountryRegionCode
        GROUP BY Person.CountryRegion.Name
),
BestPerformingCountry AS (
        SELECT TOP(1) Country
        FROM CountryTotalSales
        ORDER BY CountrySales DESC
),
RegionalTotalSales AS (
        SELECT Person.StateProvince.Name AS Region
                , Person.StateProvince.StateProvinceCode AS StateCode
                , SUM(Sales.SalesOrderHeader.TotalDue) AS RegionalSales
        FROM Sales.SalesOrderHeader
                INNER JOIN Person.Address
ON Sales.SalesOrderHeader.BillToAddressID = Person.Address.AddressID
                INNER JOIN Person.StateProvince
ON Person.Address.StateProvinceID = Person.StateProvince.StateProvinceID
                INNER JOIN Person.CountryRegion
                        ON Person.StateProvince.CountryRegionCode =
Person.CountryRegion.CountryRegionCode
```

```
            WHERE Person.CountryRegion.Name IN (SELECT* FROM BestPerformingCountry)
            GROUP BY Person.StateProvince.Name,
            Person.StateProvince.StateProvinceCode
)
SELECT Region
            , StateCode
            , ROUND(RegionalSales / 1000000, 2) AS RegionalSales
FROM RegionalTotalSales
ORDER BY RegionalSales DESC;
```

	Region	StateCode	RegionalSales
1	California	CA	17.34
2	Washington	WA	10.57
3	Texas	TX	7.50
4	Oregon	OR	3.00
5	Colorado	CO	2.70
6	Florida	FL	2.60
7	Tennessee	TN	2.36
8	New Hampshire	NH	2.04
9	Missouri	MO	2.03
10	Utah	UT	1.98

```
WITH CountryTotals AS (
   SELECT Sales.SalesTerritory.CountryRegionCode
       , SUM(Sales.SalesOrderHeader.TotalDue) AS CountrySales
   FROM  Sales.SalesOrderHeader
      INNER JOIN  Sales.SalesTerritory
ON Sales.SalesTerritory.TerritoryID = Sales.SalesOrderHeader.TerritoryID
   GROUP BY Sales.SalesTerritory.CountryRegionCode
),
BestCountry AS (
   SELECT TOP (1) CountryRegionCode
   FROM   CountryTotals
   ORDER  BY CountrySales DESC
),
RegionalTotals AS (
   SELECT Sales.SalesTerritory.Name AS Region
       , SUM(Sales.SalesOrderHeader.TotalDue) AS RegionalSales
   FROM  Sales.SalesOrderHeader
   INNER JOIN  Sales.SalesTerritory
         ON Sales.SalesTerritory.TerritoryID = Sales.SalesOrderHeader.TerritoryID
   WHERE Sales.SalesTerritory.CountryRegionCode
      = (SELECT CountryRegionCode FROM BestCountry)
   GROUP BY Sales.SalesTerritory.Name
)
SELECT
   Region, RegionalSales
FROM   RegionalTotals
ORDER  BY RegionalSales DESC;
```

	Region	RegionalSales
1	Southwest	27150594.5893
2	Northwest	18061660.371
3	Central	8913299.2473
4	Southeast	8884099.3669
5	Northeast	7820209.6285

Python Processing Script & Visualisation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

region_data= {
"Region": ["Southwest", "Northwest", "Central", "Southeast", "Northeast"],
"RegionalSales": [27150594.5893, 18061660.371, 8913299.2473, 8884099.3669, 7820209.6285]}

region = pd.DataFrame(region_data)
region["RegionalSales"] = region["RegionalSales"] / 1e6 # Convert to millions

sns.set(style="whitegrid")
sns.catplot(x='Region', y='RegionalSales', data=region, kind='bar', height=6, aspect=2)
plt.title('Best Performing Country by Regions')
plt.xlabel('Region')
plt.ylabel('Regional Sales (in USD Millions)')
plt.show()
```

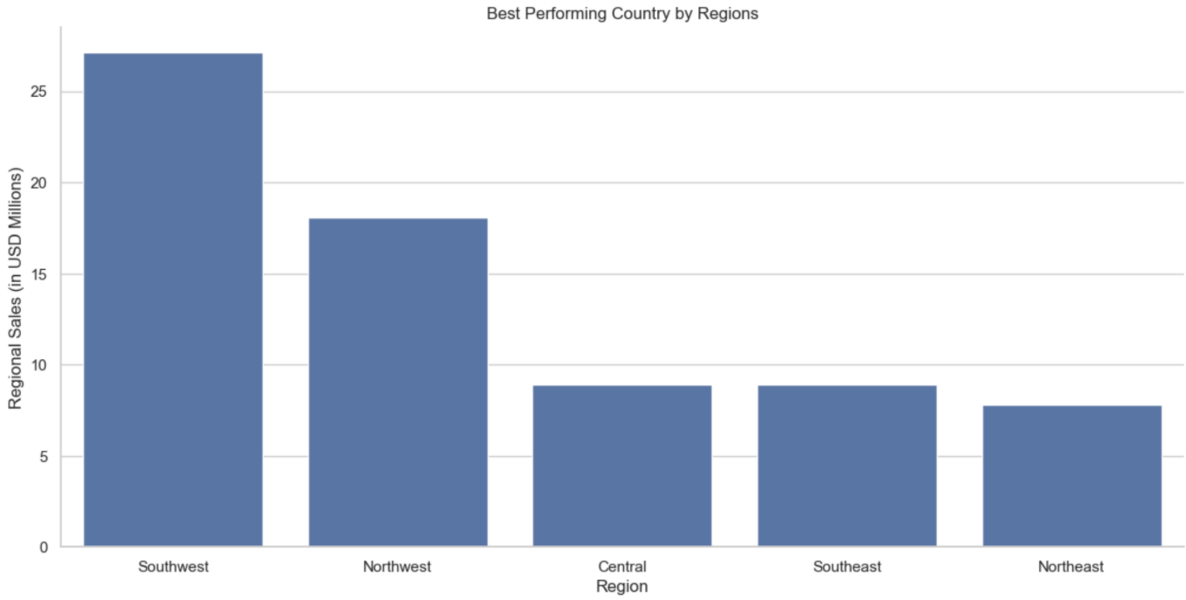


Figure 1 – bar chart of the five leading US sales territories.

```
states['RegionalSalesM'] = states['RegionalSales']/1e6 # Convert to millions

sns.set(style="whitegrid")
sns.catplot(x='Region', y='RegionalSalesM', data=states, kind='bar', height=6, aspect=2)
plt.title('Best Performing Country by Regions')
plt.xlabel('Region')
plt.ylabel('Regional Sales (in millions)')
plt.xticks(rotation=60)
plt.show()
```

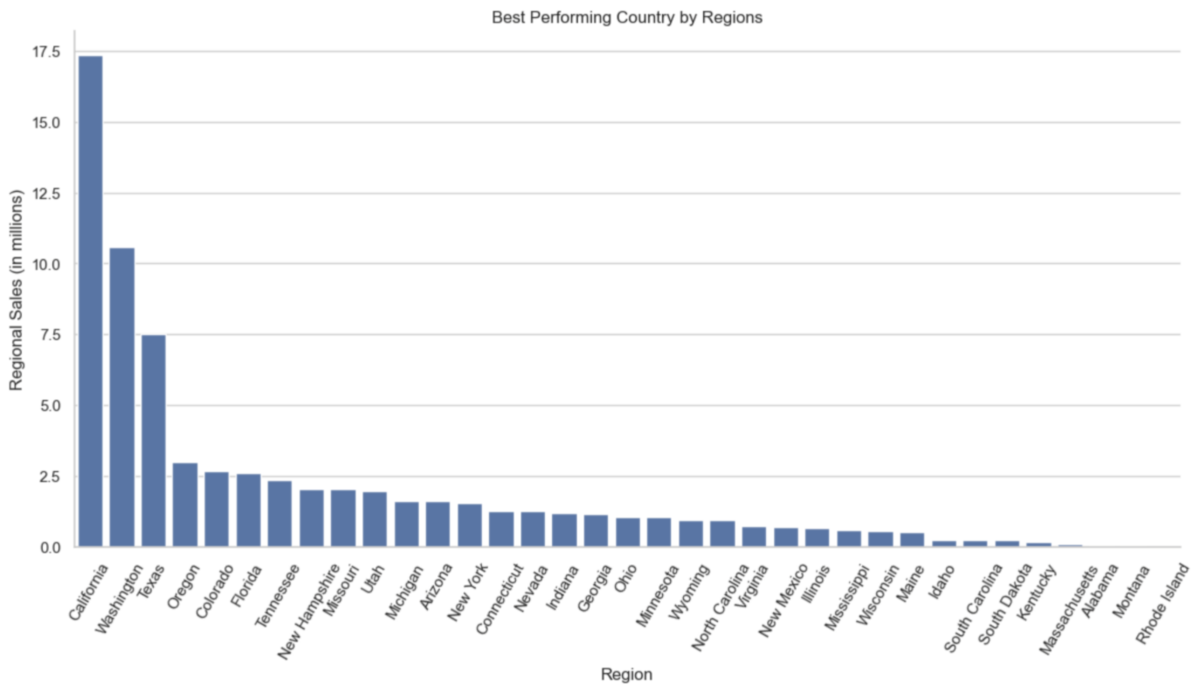


Figure 2 – full 27-state bar chart, sorted by revenue.

```
states = pd.read_csv("Q1.csv")
# Convert to millions
states["SalesM"] = round(states["RegionalSales"] / 1e6, 2)
# Custom hover text with millions
states["hover_text"] = states["Region"] + "<br>" + states["SalesM"].round(2).astype(str) + " M"
fig = px.choropleth(states, locations="StateCode",
locationmode="USA-states", color="SalesM", scope="usa",
color_continuous_scale="Blues", hover_name="Region",
hover_data={"SalesM": False, "StateCode": False, "hover_text": True},
labels={"SalesM": "Sales (Million USD)"},
title="Regional Sales by U.S. State (in Millions of USD)",
custom_data=["hover_text"])

fig.update_traces(hovertemplate="%{customdata[0]}<extra></extra>")
fig.show()
```

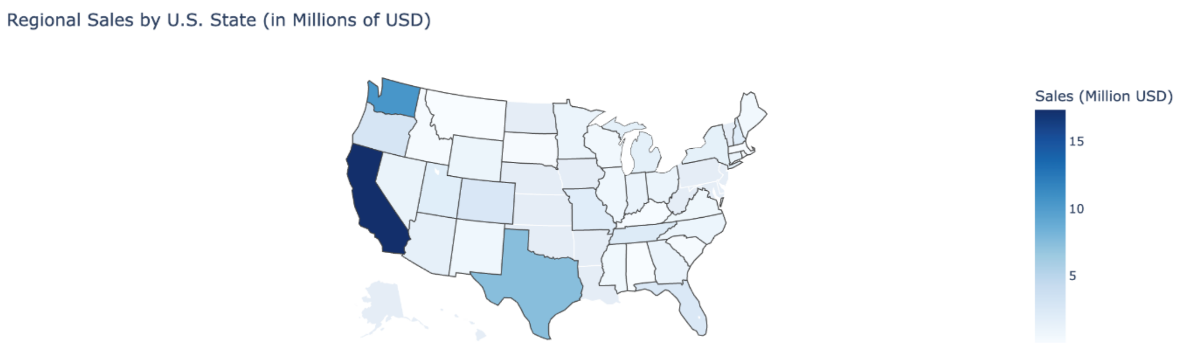


Figure 3 – choropleth map for an instant geographic view.

What is the relationship between annual leave taken and bonus?

Analysis, Insights & Action Plan

Methodology

SQL Extraction

- Joined HumanResources.Employee with Sales.SalesPerson on BusinessEntityID; filtered to rows where Bonus > 0

Statistical Test

- Calculated Correlation between VacationHours and Bonus in Python
- Result r = -0.041.

Visualisation

- Figure 4 – bar chart of bonus by vacation-hour bucket.
- Figure 5 – scatter-plot with linear-regression line (flat red trend).

Findings

Correlation is effectively zero; the regression line is flat (Figure 5).

Highest bonus ($6.7 k) occurs at 29 vacation hours; lowest ($75) at 35 vacation hours has no consistent pattern.

Visual inspection confirms broad vertical spread at each vacation level (Figure 4).

Insight

Vacation time does not impact sales bonuses; performance appears driven by factors other than days taken off.

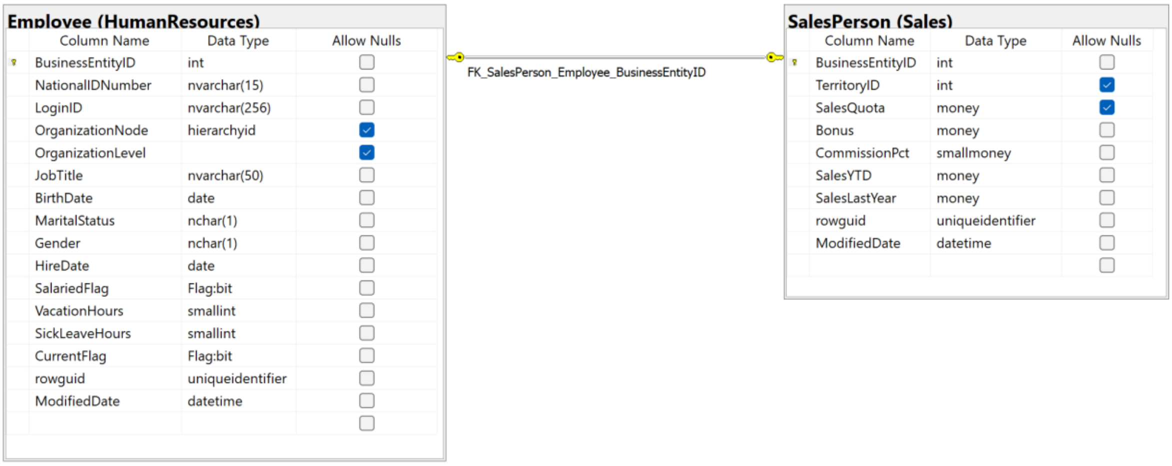
Action Plan

Maintain flexible-leave policy – reassure staff that taking vacation will not hurt their bonus.

Continue monitoring annually to confirm whether the relationship remains consistent as staff numbers and bonus structures change. Current data is limited, so further analysis over time is needed to draw stronger conclusions.

Analytical Workflow & Findings

Schema Diagram



SQL Extraction Query & Results

```
SELECT HumanResources.Employee.VacationHours
        , Sales.SalesPerson.Bonus
FROM HumanResources.Employee
        INNER JOIN Sales.SalesPerson
                ON HumanResources.Employee.BusinessEntityID = Sales.SalesPerson.BusinessEntityID
WHERE Sales.SalesPerson.Bonus > 0
ORDER BY HumanResources.Employee.VacationHours;
```

	VacationHours	Bonus
1	22	5000.00
2	23	3500.00
3	24	2500.00
4	26	3550.00
5	27	2000.00
6	29	6700.00
7	31	5000.00
8	33	500.00
9	34	985.00
10	35	75.00
11	36	5650.00
12	37	5150.00
13	38	4100.00
14	39	3900.00

Python Processing Script & Visualisation

```
bonus = pd.read_csv("Q2.csv")
sns.set(style="whitegrid")
sns.catplot(x='VacationHours', y='Bonus', data=bonus, kind='bar', height=6, aspect=2)
plt.title('Vacation Hours vs Bonus')
plt.show()
```

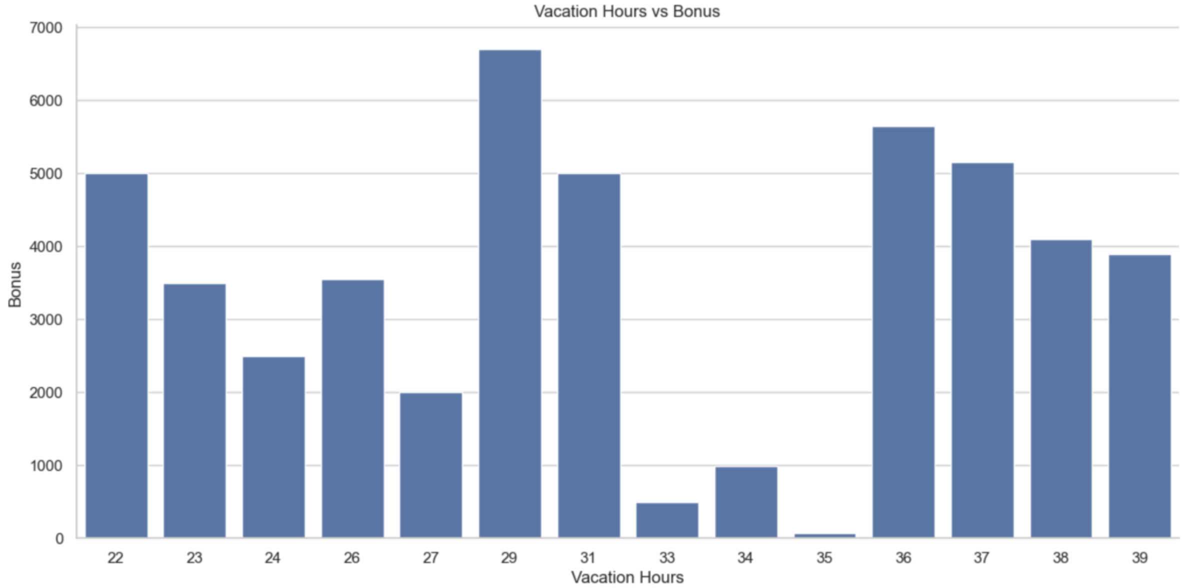


Figure 4 – bar chart of bonus by vacation-hour bucket.

```
sns.set(style="whitegrid")
sns.relplot(x='VacationHours', y='Bonus', data=bonus, kind='scatter', height=6, aspect=2, s=220)
sns.regplot(x='VacationHours', y='Bonus', data=bonus, scatter_kws={'alpha':0.5}, line_kws={'color':'red'}, ci =None)
plt.title('Vacation Hours vs Bonus')
plt.xlabel('Vacation Hours')
plt.show()
```

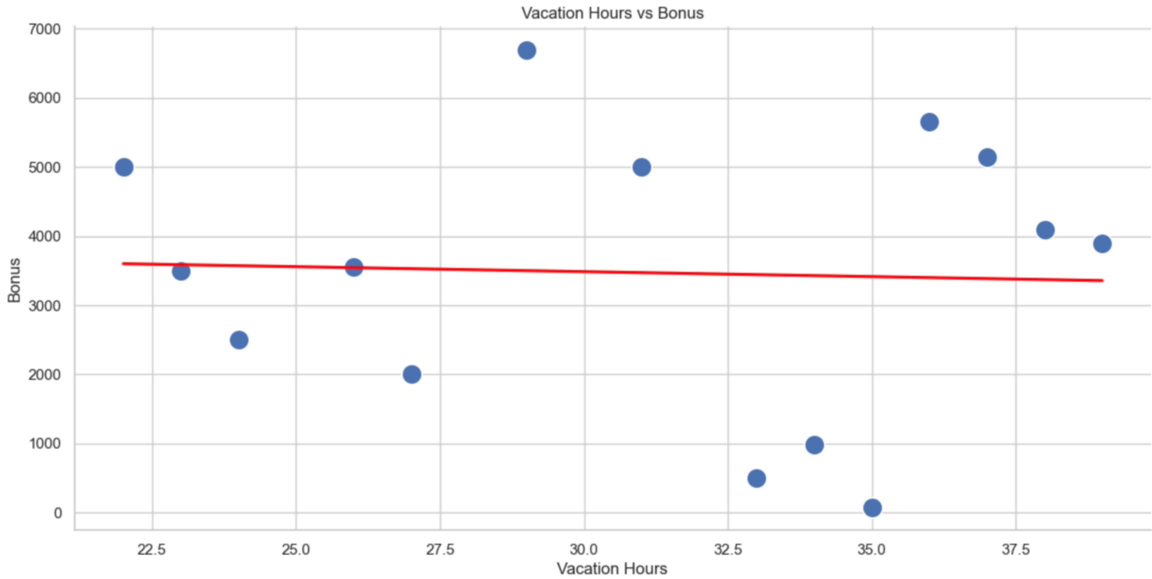


Figure 5 – scatter-plot with linear-regression line (flat red trend).

What is the relationship between Country and Revenue?

Analysis, Insights & Action Plan

Methodology

SQL Query

- Joined SalesOrderHeader with Address with StateProvince with CountryRegion
- Summed TotalDue and counted customers per country.

Normalisation

- In Python calculated Average Revenue per Customer (Revenue / NumberOfCustomers).

Visualisation

- Figure 6 – bar chart of total revenue by country.
- Figure 7 – bar chart of average revenue per customer.

Findings

On absolute sales, Australia (AU) sits third—just behind the US, and Canada—and ahead of the UK, France and Germany.

However, when divided each country's revenue by its active-customer count, Australia drops to the very bottom. Its average revenue per customer is the lowest in the entire dataset.

USA and Canada contribute 72.5% of the total world revenue.

Insight

Australia's revenue bulk comes from many low-spend customers, whereas European markets (FR, DE, UK) have fewer but higher-spending buyers. Marketing tactics must be tailored accordingly.

Action Plan

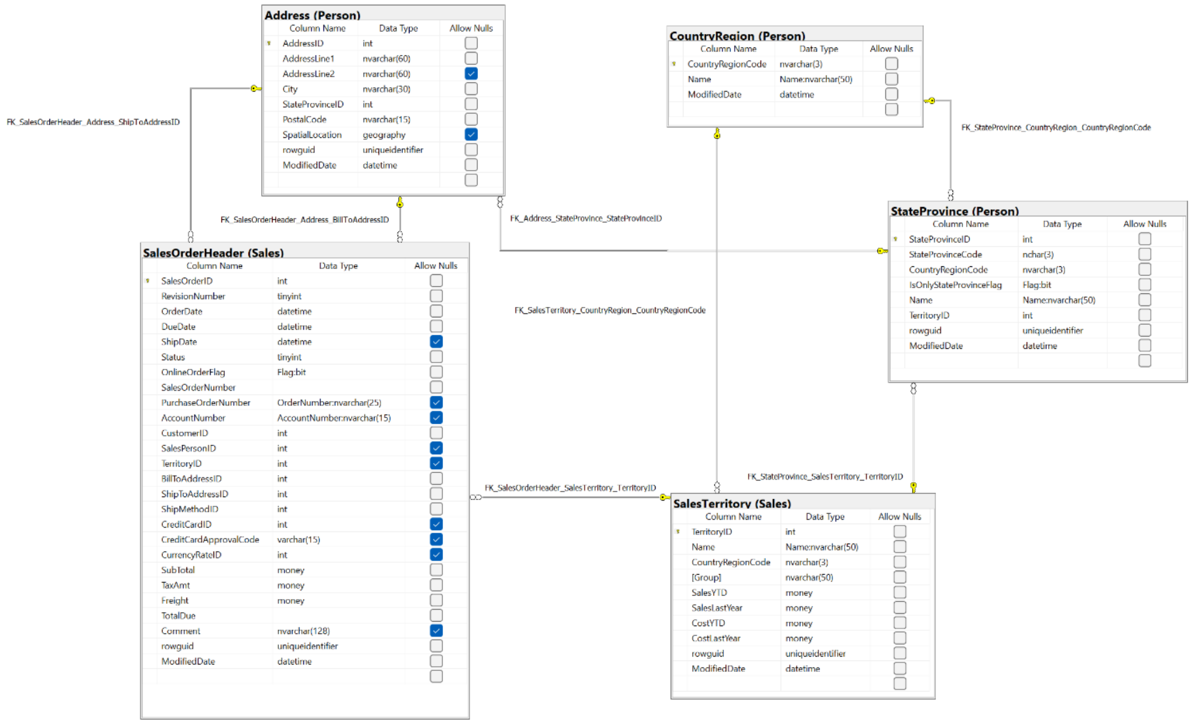
Upsell & bundle in Australia to lift basket size (loyalty offers, add-ons).

Continue broad campaigns; USA and Canada countries drive nearly three-quarters of revenue.

Focus on high-value product lines for Germany, France, UK.

Analytical Workflow & Findings

Schema Diagram



SQL Extraction Query & Results

```
SELECT Person.CountryRegion.Name AS Country
            , SUM(Sales.SalesOrderHeader.TotalDue) AS Revenue
            , COUNT(Sales.SalesOrderHeader.CustomerID) AS NumberOfCustomers
FROM Sales.SalesOrderHeader
            INNER JOIN Person.Address
                        ON Sales.SalesOrderHeader.ShipToAddressID = Person.Address.AddressID
            INNER JOIN Person.StateProvince
                        ON Person.Address.StateProvinceID = Person.StateProvince.StateProvinceID
            INNER JOIN Person.CountryRegion
                        ON Person.StateProvince.CountryRegionCode = Person.CountryRegion.CountryRegionCode
GROUP BY Person.CountryRegion.Name
ORDER BY Revenue DESC;
```

	Country	Revenue	NumberOfCustomers
1	United States	70829863.203	12041
2	Canada	18398929.188	4067
3	Australia	11814376.0952	6843
4	United Kingdom	8574048.7082	3219
5	France	8119749.346	2672
6	Germany	5479819.5755	2623

Python Processing Script & Visualisation

```
totalrevenue = pd.read_csv("Q3.csv")
totalrevenue["number of customer"] = [12041, 4067, 6843, 3219, 2672, 2623]
totalrevenue['Revenue'] = totalrevenue['Revenue']/1000000 # Convert to millions
sns.catplot(x='Country', y='Revenue', data=totalrevenue, kind='bar', height=6, aspect=2)
plt.title('Revenue by Country')
```

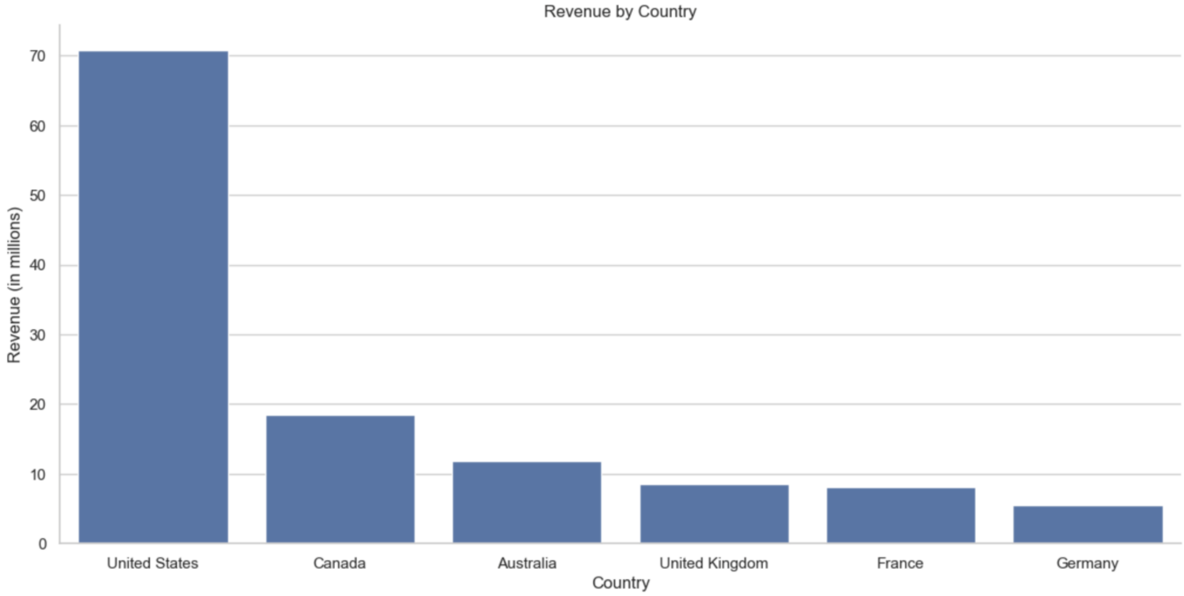


Figure 6 – total revenue by country

```
totalrevenue["percustomer"] = totalrevenue["Revenue"] / totalrevenue["number of customer"]
totalrevenue.sort_values = totalrevenue.sort_values(by='percustomer', ascending=False)
sns.catplot(x='Country', y='percustomer', data=totalrevenue.sort_values, kind='bar', height=6, aspect=2)
plt.title('Average Revenue by Country')
plt.show()
```

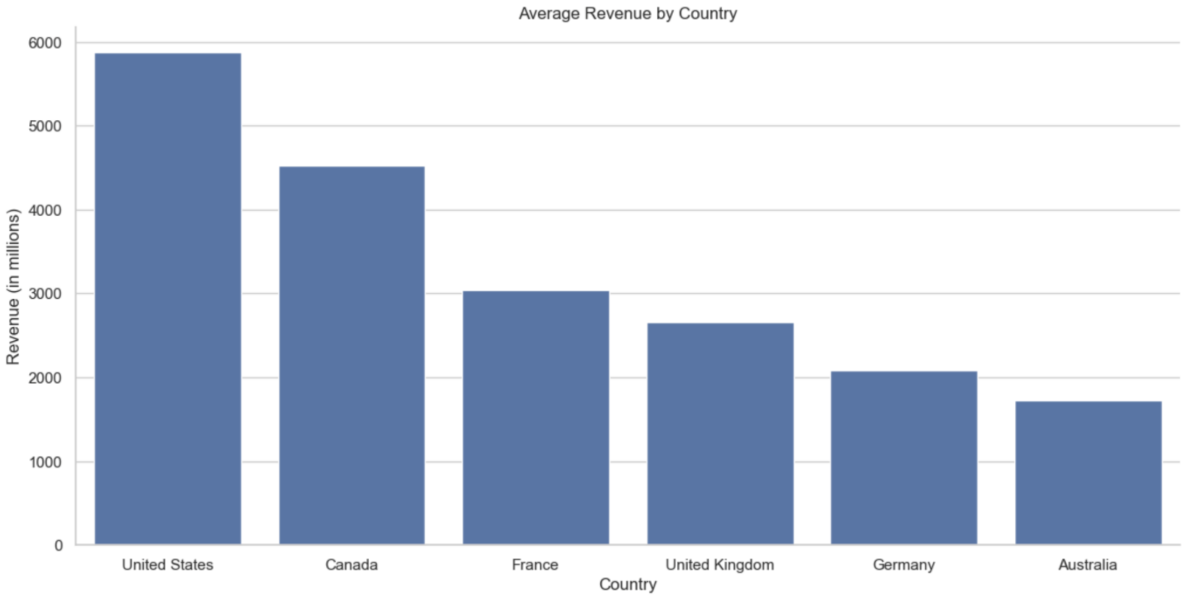


Figure 7 – average revenue per customer

What is the relationship between sick leave and Job Title (Person Type)?

Analysis, Insights & Action Plan

Methodology

SQL Queries

- Total sick-leave hours and head-count by Department
  (Employee → EmployeeDepartmentHistory → Department).
- Sick leave by PersonType (corporate-employee EM vs salesperson SP).
- Sick leave by Job Title
- Sick leave by Shift (day, evening, night).

Normalisation

- In Python calculated averages (TotalSickLeave / HeadCount) and generated visuals.

Visualisation

- Figure 8 – bar chart:
  - (a) total sick-leave by head-count by department.
  - (b) average sick-leave vs head-count by department.
- Figure 9 – bar chart:
  - (a) total sick leave by Job Title.
  - (b) average sick leave per employee by Job Title.
- Figure 10 – pie charts:
  - (a) total sick leave by PersonType
  - (b) average sick leave by PersonType
- Figure 11 – pie charts:
  - (a) Total sick leave hours by Shift
  - (b) Average sick leave per employee by Shift

Findings

Departments

Shipping and Receiving had the highest average sick leave at 67 hours.

Engineering had the lowest average sick leave at 29 hours.

Production recorded the highest departmental sick leave at 7 971 hours, because it employs 180 staff, compared with 110 across all other departments combined.

Job Titles

Highest average per person - CEO (69hours) , Stocker (68 hours)

Lowest average per person - CFO (20hours), VP Engineering (20 hours)

Production Technicians WC40-WC20 record the highest total (1 260 hours), however their per-person average is mid-pack (55 hours).

CFO and VP Engineering record the fewest total hours (20 h each) because there is only one person in each role.

Person Type

Corporate staff (EM) generate 95.6 % of total sick-leave hours, versus 4.4 % for Sales staff (SP)—largely because there are far more corporate employees.

Even on a per-employee basis, corporate workers still take about 15 % more sick leave than their sales counterparts.

Shift

Night shift has the highest average sick leave per employee (35 %), edging out Day and Evening shifts.

Day shift accounts for the majority of total sick-leave hours (60.8 %) simply because it has the largest head-count.

Insight

Sick-leave impact is concentrated in the Production department primarily because of its size, while night-shift patterns and higher per-capita leave among sales staff highlight targeted wellness gaps.

Action Plan

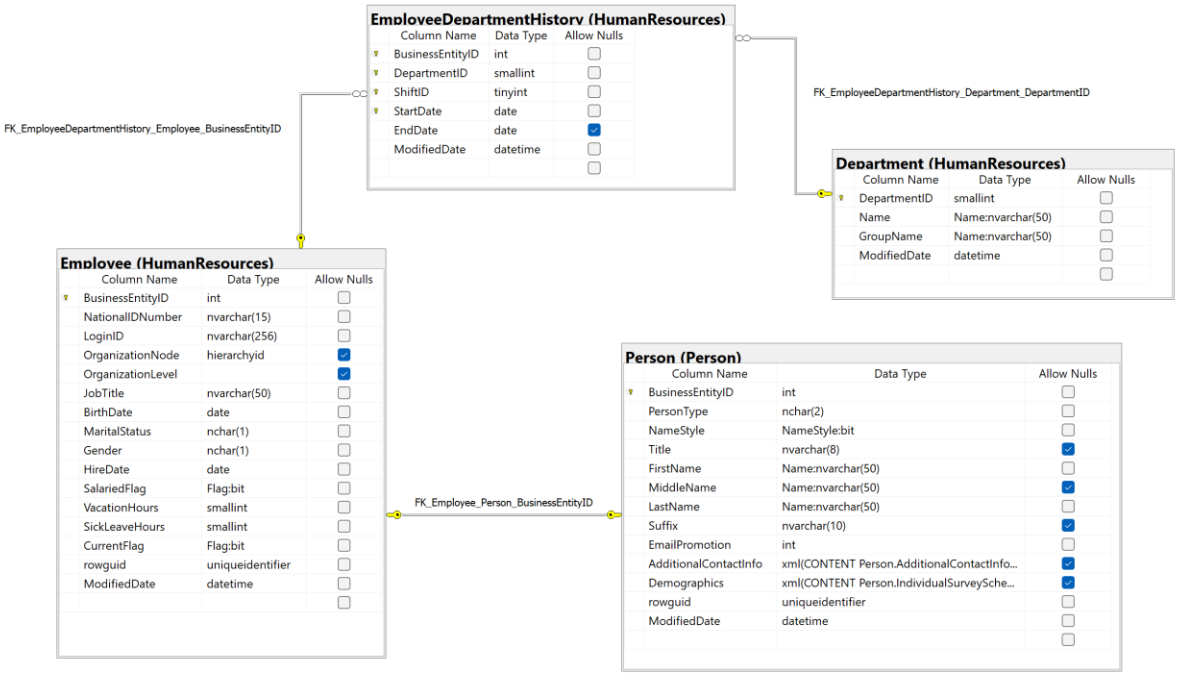
Roll out ergonomic & wellness programmes; pilot staggered shift rotations to reduce fatigue.

Introduce health-monitoring and flexible scheduling to address the highest average leave segment.

Investigate underlying causes (travel strain, commission stress) and offer preventative support.

Analytical Workflow & Findings

Schema Diagram



SQL Extraction Query & Results

```
SELECT SUM(HumanResources.Employee.SickLeaveHours) AS TotalSickLeave
        , HumanResources.Department.Name AS DepartmentName
        ,count(HumanResources.Employee.BusinessEntityID) AS NumberOfEmployees
FROM HumanResources.Employee
        INNER JOIN HumanResources.EmployeeDepartmentHistory
                ON HumanResources.EmployeeDepartmentHistory.BusinessEntityID
                        = HumanResources.Employee.BusinessEntityID
        INNER JOIN HumanResources.Department
                ON HumanResources.EmployeeDepartmentHistory.DepartmentID
                        = HumanResources.Department.DepartmentID
GROUP BY HumanResources.Department.Name
ORDER BY TotalSickLeave;
```

	TotalSickLeave	DepartmentName	NumberOfEmployees
1	89	Executive	2
2	151	Tool Design	4
3	209	Engineering	7
4	216	Research and Development	4
5	255	Production Control	6
6	273	Human Resources	6
7	291	Document Control	5
8	405	Shipping and Receiving	6
9	410	Quality Assurance	7
10	420	Marketing	10

```
SELECT SUM(HumanResources.Employee.SickLeaveHours) AS TotalSickLeave
, HumanResources.Employee.JobTitle
, COUNT(HumanResources.Employee.BusinessEntityID) As NumberOfEmployee
FROM HumanResources.Employee
INNER JOIN Person.Person
ON HumanResources.Employee.BusinessEntityID = Person.Person.BusinessEntityID
GROUP BY HumanResources.Employee.JobTitle
ORDER BY TotalSickLeave;
```

	TotalSickLeave	JobTitle	NumberOfEmployee
1	20	Chief Financial Officer	1
2	20	Vice President of Engineering	1
3	21	Senior Design Engineer	1
4	21	Engineering Manager	1
5	25	Vice President of Sales	1
6	27	North American Sales Manager	1
7	30	Pacific Sales Manager	1
8	30	European Sales Manager	1
9	40	Marketing Manager	1
10	41	Production Control Manager	1

```
SELECT SUM(HumanResources.Employee.SickLeaveHours) AS TotalSickLeave
, HumanResources.Employee.JobTitle
, COUNT(HumanResources.Employee.BusinessEntityID) AS NumberOfEmployee
FROM HumanResources.Employee
INNER JOIN Person.Person
ON HumanResources.Employee.BusinessEntityID = Person.Person.BusinessEntityID
WHERE Person.Person.PersonType = 'SP'
GROUP BY HumanResources.Employee.JobTitle
ORDER BY TotalSickLeave;
```

	TotalSickLeave	JobTitle	NumberOfEmployee
1	27	North American Sales Manager	1
2	30	Pacific Sales Manager	1
3	30	European Sales Manager	1
4	493	Sales Representative	14

```
SELECT SUM(HumanResources.Employee.SickLeaveHours) AS TotalSickLeave
, COUNT(HumanResources.Employee.BusinessEntityID) AS NumberOfEmployee
, Person.Person.PersonType
FROM HumanResources.Employee
INNER JOIN Person.Person
ON HumanResources.Employee.BusinessEntityID = Person.Person.BusinessEntityID
GROUP BY Person.Person.PersonType
ORDER BY TotalSickLeave;
```

	TotalSickLeave	NumberOfEmployee	PersonType
1	580	17	SP
2	12559	273	EM

```
SELECT HumanResources.Shift.Name AS ShiftName
 , Sum(HumanResources.employee.SickLeaveHours) AS TotalSickLeaveHours
 , COUNT(HumanResources.Employee.BusinessEntityID) AS NumberOfEmployee
FROM HumanResources.EmployeeDepartmentHistory
INNER JOIN HumanResources.Shift
ON HumanResources.EmployeeDepartmentHistory.ShiftID
= HumanResources.Shift.ShiftID
INNER JOIN HumanResources.Employee
ON EmployeeDepartmentHistory.BusinessEntityID
= HumanResources.Employee.BusinessEntityID
Group BY HumanResources.Shift.Name
```

	ShiftName	TotalSickLeaveHours	NumberOfEmployee
1	Day	8153	182
2	Evening	2758	62
3	Night	2498	52

Python Processing Script & Visualisation

```
department = pd.read_csv("Q4 - Department.csv")
department = department.sort_values(by='TotalSickLeave', ascending=False)
sns.set(style="whitegrid")
sns.catplot(x='DepartmentName', y='TotalSickLeave', data=department, kind='bar', height=6, aspect=2)
plt.title('Total Sick Leave by Department')
plt.xlabel('Department Name')
plt.ylabel('Total Sick Leave')
plt.xticks(rotation=45)
plt.show()
```

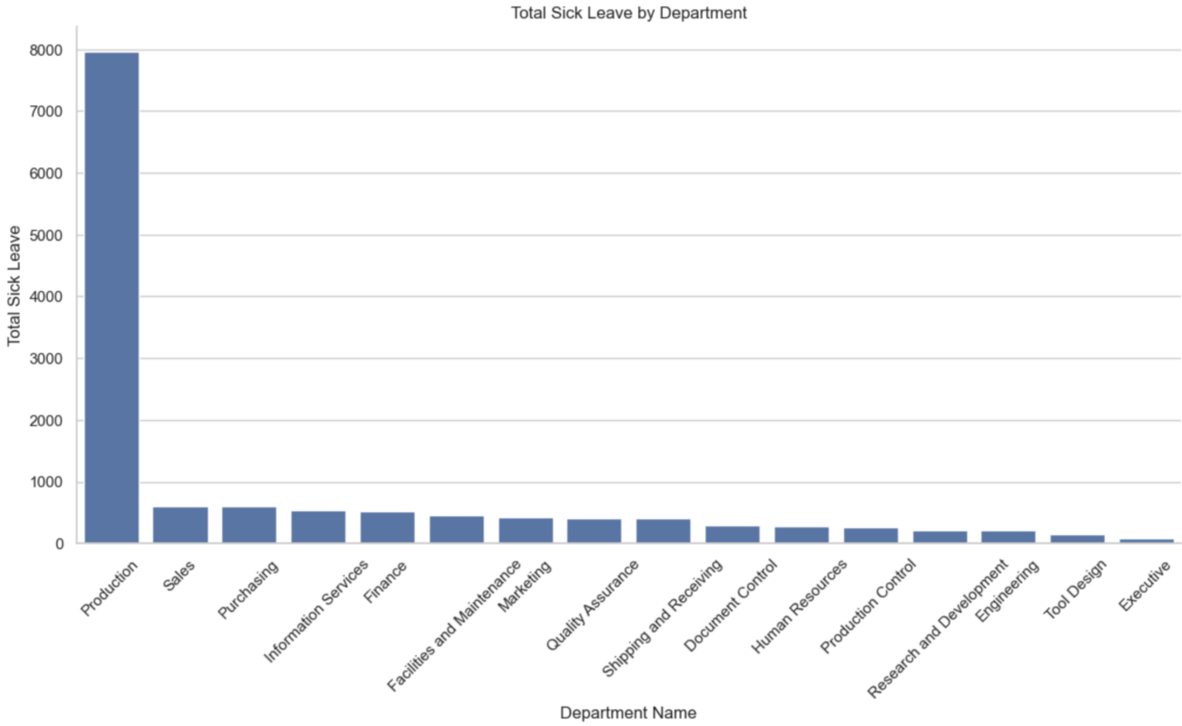


Figure 8 (a) total sick-leave by head-count by department.

```
department['average sickleave'] = department['TotalSickLeave'] / department['Number of Employees']
department1 = department.sort_values(by='average sickleave', ascending=False)

sns.set(style="whitegrid")
sns.catplot(x='DepartmentName', y='average sickleave', data=department1, kind='bar', height=6, aspect=2)
plt.title('Average Sick Leave by Department')
plt.xlabel('Department Name')
plt.ylabel('Total Sick Leave')
plt.xticks(rotation=45)
plt.show()
```

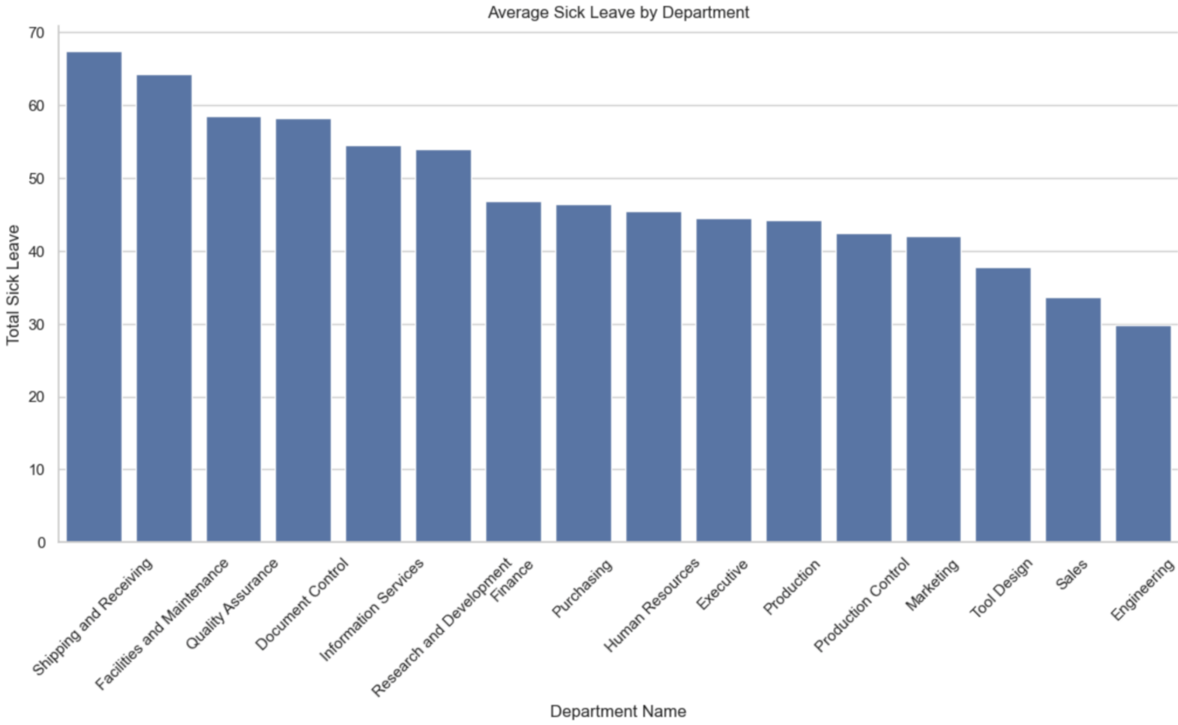


Figure 8 (b) average sick-leave vs head-count by department.

```
jobtitle = pd.read_csv("Q4 - JobTitle.csv")
sns.set(style="whitegrid")
sns.catplot(x='JobTitle', y='TotalSickLeave', data=jobtitle, kind='bar', height=6, aspect=2)
plt.title('Total Sick Leave by Job Title')
plt.xlabel('Job Title')
plt.ylabel('Total Sick Leave')
plt.xticks(rotation=90)
plt.show()
```

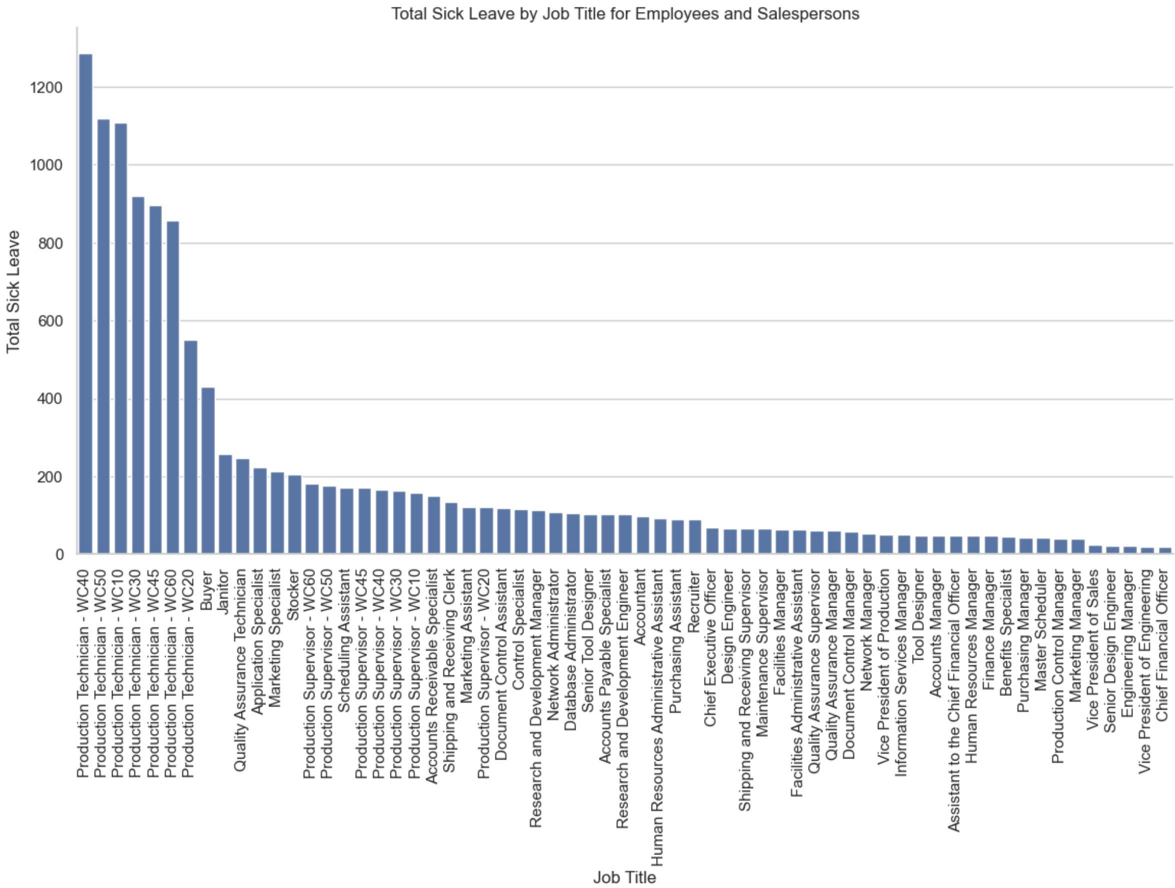


Figure 9 (a) total sick leave by Job Title.

```
jobtitle["per"]= jobtitle["TotalSickLeave"] / jobtitle["Number of Employee"]
jobtitle = jobtitle.sort_values(by='per', ascending=False)

sns.set(style="whitegrid")
sns.catplot(x='JobTitle', y='per', data=jobtitle, kind='bar', height=6, aspect=2)
plt.title('Average Sick Leave per Employee by Job Title')
plt.xlabel('Job Title')
plt.ylabel('Average Sick Leave per Employee')
plt.xticks(rotation=90)
plt.show()
```

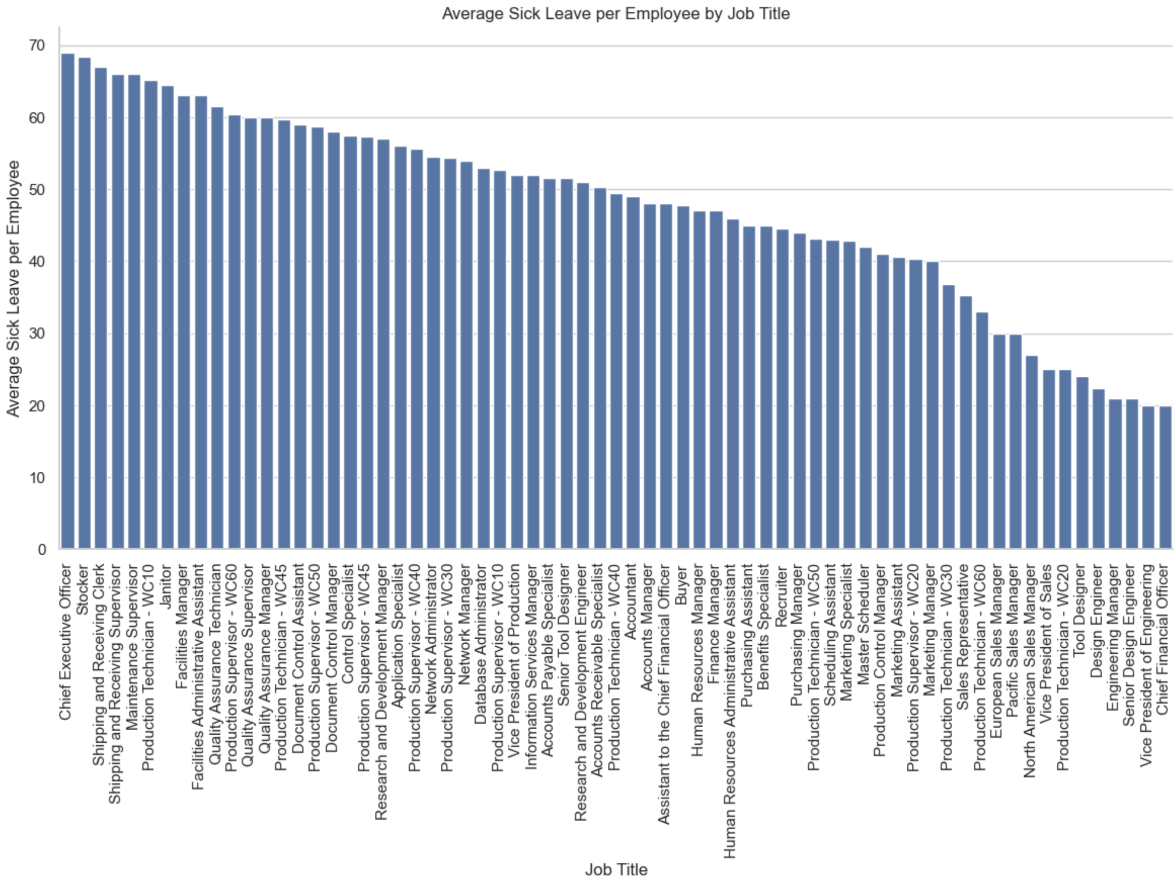


Figure 9 (b) average sick leave per employee by Job Title.

```
persontype = pd.read_csv("Q4 - PersonType.csv")
labels = persontype["PersonType"].replace({"EM": "Cooperate","SP": "Salesperson"})
plt.pie(persontype['TotalSickLeave'], labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Proportion of Sick Leave by Person Type', fontsize=14)
plt.axis('equal')
plt.show()
```

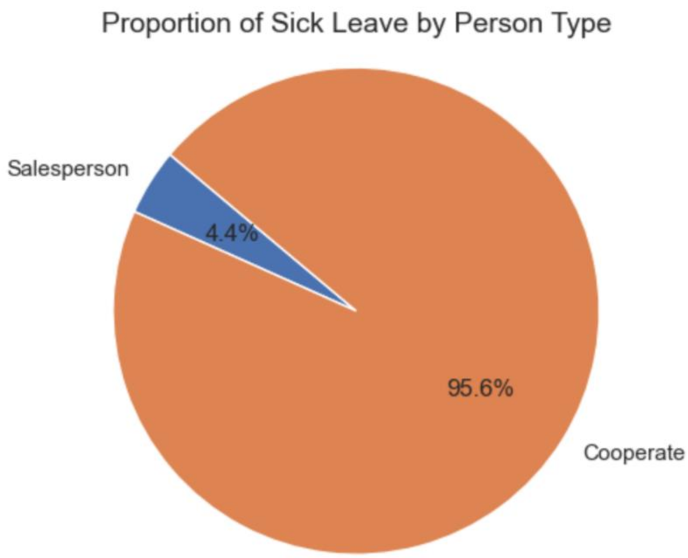


Figure 10 (a) total sick leave by PersonType

```
# Calculate average sick leave
persontype["AverageSickLeave"] = persontype["TotalSickLeave"] / persontype["PersonCount"]
plt.pie(persontype["AverageSickLeave"],
labels=persontype["PersonType"],
autopct='%1.1f%%', startangle=140)
plt.title("Average Sick Leave by Person Type", fontsize=14, y=1.08)
plt.axis("equal")
plt.show()
```

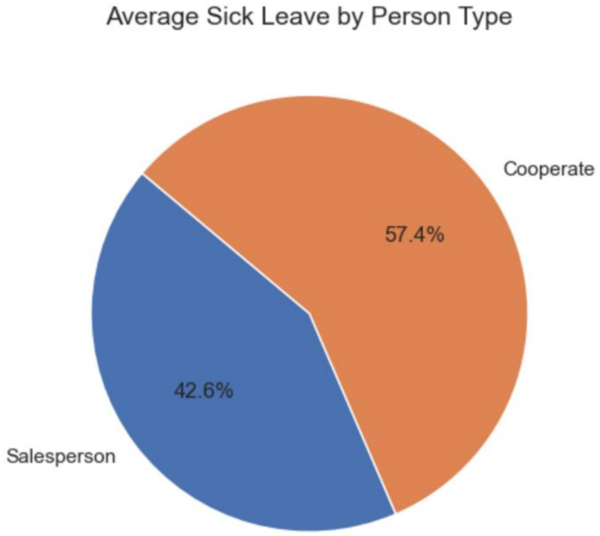


Figure 10 (b) average sick leave by PersonType

```
shift_data = pd.DataFrame({
"ShiftName": ["Day", "Evening", "Night"],
"TotalSickLeaveHours": [8153, 2758, 2498].
"TotalSickLeaveHours": [44.796703, 44.483871, 48.038462]})
plt.pie(shift_data["TotalSickLeaveHours"],
labels=shift_data["ShiftName"], autopct="%1.1f%%",startangle=140)
plt.title("Total Sick Leave Hours by Shift", fontsize=14, y=1.08)
plt.axis("equal")
plt.show()
```

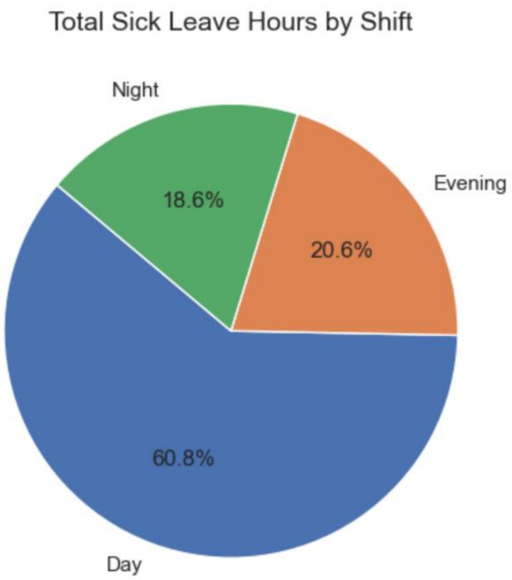


Figure 11 (a) Total sick leave hours by Shift

```python
# Plot the pie chart
plt.pie(
shift_data["TotalSickLeaveHours"],
labels=shift_data["ShiftName"],
autopct="%1.1f%%",
startangle=140)
plt.title("Average Sick Leave Hours by Shift", fontsize=14, y=1.08)
plt.axis("equal") # Make the pie chart circular
plt.show()
```

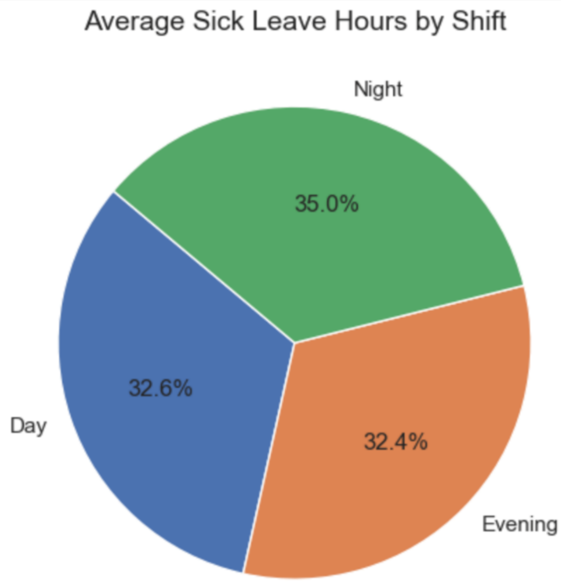


Figure 11 (b) Average sick leave per employee by Shift

What is the relationship between store trading duration and revenue?

Analysis, Insights & Action Plan

Methodology

Two-step CTE query

- For every StoreID compute months trading = DATEDIFF(MONTH, MIN(OrderDate), MAX(OrderDate)).
- CustomerRevenue = sum TotalDue per store.
- Join the two CTEs to get a (TradingMonths, Revenue, StoreID) table.

Validation

- Cross-checked that StoreID in Sales.Customer is NOT NULL
- Tested alternative granularity (days vs months) – pattern unchanged.

Visualisation

- Figure 15 – bar chart of revenue totals by trading-month bucket (easier to see peaks).
- Figure 16 – scatter-plot of all 635 stores with an OLS trend-line & 95 % CI band (red).
    - Correlation = 0.41 (moderate, positive).
    - Shaded band shows model uncertainty – narrower in mid-range, wider at extremes (the regression band widens at the edges because we have fewer very young/very old stores – predictions there are less certain)

Findings

There is a moderate positive relationship between trading duration and revenue, with r = 0.41 across 635 stores.

Revenue leaders are stores that have traded for 32, 33, and 34 months; each of these locations exceeds $0.30 million in sales.

A few early over-achievers—specifically 8, 20, 21, and 22-month-old stores—already generate more than $0.10 million in revenue.

Insight

Although longer trading time boosts revenue, several stores that are less than 24 months old still out-earn older outlets.

The scatter-plot trend line confirms this upward relationship, and the wider confidence band at the youngest and oldest durations highlights greater prediction uncertainty where data are sparse.

Action Plan

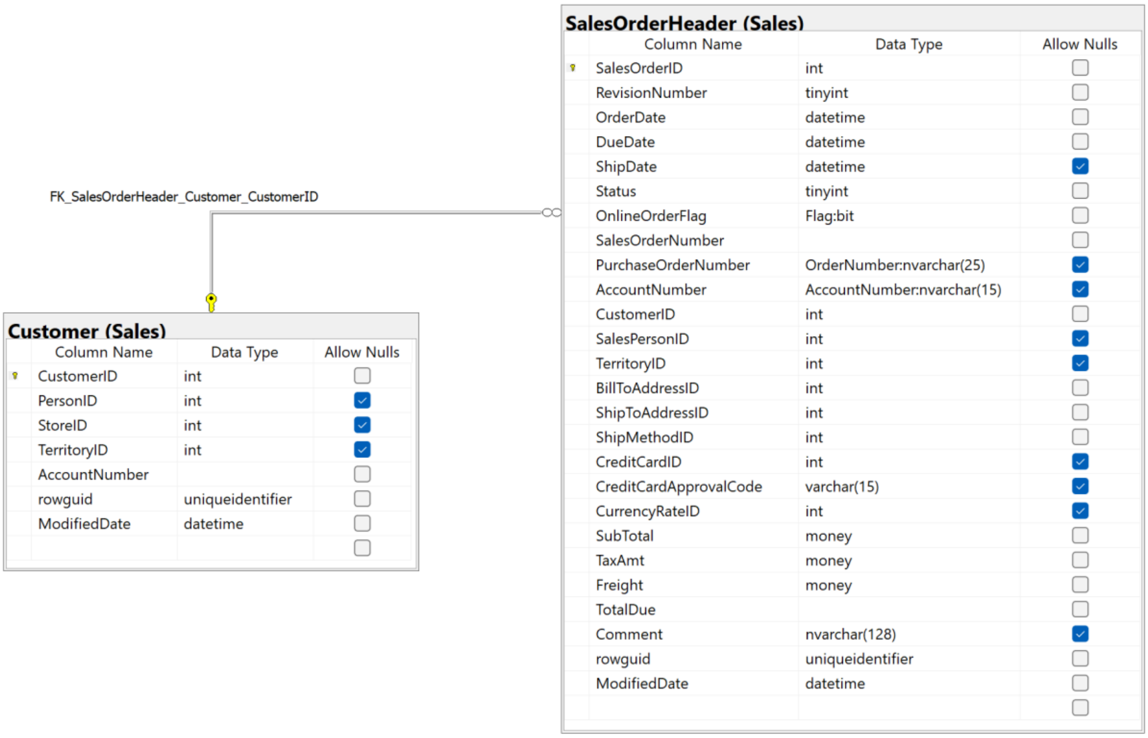
Profile the “rising-stars” (20 to 22-month-old stores) by interviewing their managers about product mix, staffing, and local-marketing tactics.

Create an onboarding playbook that applies those best practices to accelerate the revenue ramp-up of all new stores.

Keep a list of stores older than 30 months that still sit in the bottom 25 % for sales; when one shows up, run a quick audit of its products, location, and management, or potential exit the store.

Analytical Workflow & Findings

Schema Diagram




```
WITH StoreTrading AS (

        SELECT Sales.Customer.StoreID AS StoreID,
        DATEDIFF (MONTH,MIN(Sales.SalesOrderHeader.OrderDate) ,MAX(Sales.SalesOrderHeader.OrderDate)) AS
TradingDurationDays
        FROM Sales.SalesOrderHeader
        INNER JOIN Sales.Customer
        ON Sales.SalesOrderHeader.CustomerID = Sales.Customer.CustomerID
        WHERE Sales.Customer.StoreID IS NOT NULL
        GROUP BY Sales.Customer.StoreID
),
CustomerRevenue AS (
        SELECT Sales.Customer.StoreID AS StoreID,
        SUM(Sales.SalesOrderHeader.TotalDue) AS Revenue
        FROM Sales.SalesOrderHeader
        INNER JOIN Sales.Customer
        ON Sales.SalesOrderHeader.CustomerID = Sales.Customer.CustomerID
        WHERE Sales.Customer.StoreID IS NOT NULL
        GROUP BY Sales.Customer.StoreID
)
SELECT StoreTrading.TradingDurationDays
        , Revenue AS TotalRevenue , CustomerRevenue.StoreID AS Store
FROM StoreTrading
INNER JOIN CustomerRevenue ON StoreTrading.StoreID = CustomerRevenue.StoreID
```

	TradingDurationDays	TotalRevenue	Store
1	20	147804.9208	292
2	10	127379.7919	294
3	33	584949.1308	296
4	33	77585.195	298
5	10	249804.8673	300

Python Processing Script & Visualisation

```
storeduratrion = pd.read_csv("Q5 - Month.csv")
storeduratrion["TotalRevenue"] = round(storeduratrion["TotalRevenue"]/1e6,2)
sns.catplot(x='TradingDurationMonth', y='TotalRevenue', data=storeduratrion, kind='bar', height=6, aspect=2,
errorbar=None)
plt.title('Total Revenue by Trading Duration (Months)')
plt.ylabel('Total Revenue (Millions)')
```

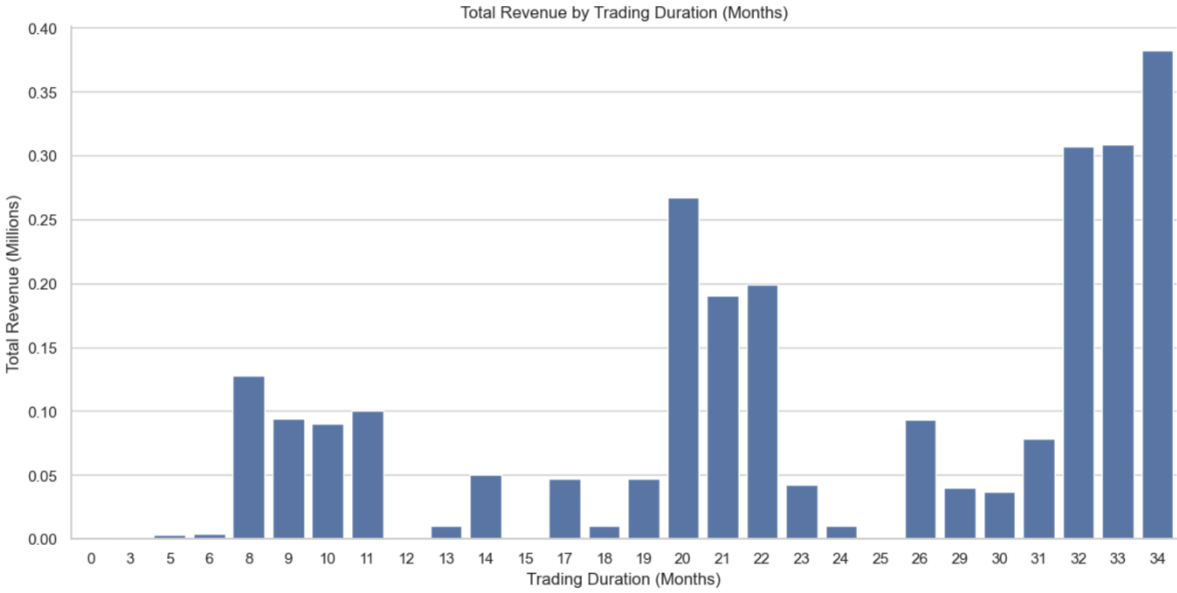


Figure 15 – bar chart of revenue totals by trading-month bucket.

```
storeduratrion = storeduratrion.sort_values(by='TradingDurationMonth', ascending=True)
sns.lmplot(x='TradingDurationMonth', y='TotalRevenue', data=storeduratrion, height=6, aspect=2,
scatter_kws={'alpha': 0.5}, line_kws={'color': 'red'})
plt.title("Trading Duration (Months) vs Total Revenue with Trend Line")
plt.xlabel("Trading Duration (Months)")
plt.ylabel("Total Revenue (Millions)")
plt.show()
```

Figure 16 – scatter-plot of all 635 stores with an OLS trend-line & 95 % CI band (red)

What is the relationship between the size of the stores, number of employees and revenue?

Analysis, Insights & Action Plan

Methodology

Data pull

- Parsed XML inside Sales.Store.Demographics to expose SquareFeet & NumberEmployees, then joined to SalesOrderHeader.TotalDue.

Binning

- Size bands: Small ≤ 20 k ft^2, Medium 21-60 k ft^2, Large ≥ 61 k ft^2.
- Employee bands: Few ≤ 30, Average 31-60, Many ≥ 61.

Statistics

- Pair-wise Pearson correlation coefficient, box plots, matched-group bar chart, 3-D scatter.

Visualisation

- Figure17 – Scatter with Store Size vs Revenue and Employees vs Revenue
- Figure18 – Bar chart of Average revenue by Store Size and Employee Count
- Figure19 - Scatter size vs employee vs revenue
- Figure20 - 3-D scatter (size, employees, revenue) highlights the two high-performing clusters.
- Figure21 - Heat-map with correlation coefficients.
- Figure22 - Pair-plot with correlation coefficients.
- Figure23 - Average-revenue bars Store Size and Employee Count category
- Figure24 - box-plots showing spread and outliers.
- Figure25 - Side-by-side bar – matched groups (Small/Few, Medium/Average, Large/Many)
- Figure26 - Scatter size vs employee vs Annual Revenue
- Figure27 - Scatter: Store Size vs Revenue and Employees vs AnnualRevenue

Findings

There is a very strong relationship between store size and number of employees, with a correlation coefficient of $r = 0.97$, indicating that as store size increases, staffing levels rise almost proportionally.

However, the relationship between store size and revenue is weak, with a correlation of $r = 0.10$, showing that having a larger store does not necessarily lead to higher sales.

Similarly, the correlation between number of employees and revenue is also weak (r = 0.9), meaning that having more staff does not guarantee increased revenue.

Looking at performance by store size category, large stores lead with an average revenue of $192k, although only a small number exceed $800k in total sales.

Small stores perform surprisingly well, averaging $172k, and demonstrate strong efficiency relative to their size and staffing.

In contrast, medium-sized stores perform poorly, averaging just $53k and showing the widest range of results across the group.

Insight

While larger stores require more space and more staff, this increase in capacity does not automatically generate proportional increases in revenue.

In fact, the data shows that store size alone is not a reliable driver of performance—only certain combinations of store size and team structure deliver strong results.

The most successful store types fall into two categories: large stores with many employees, and small stores with lean, efficient teams.

Medium-sized stores are consistently under-monetised and should be carefully reviewed for layout, assortment, and operational efficiency.

To address this, medium stores should undergo performance audits to identify specific issues and areas for optimisation.

Action Plan

Successful small stores often have focused product ranges and flexible staff who manage multiple roles—these layouts and scheduling models should be documented and replicated elsewhere.

Going forward, any proposal for a new medium-sized store must be supported by a business case that projects at least 30% more revenue than the current average for that category.

Analytical Workflow & Findings

Schema Diagram


```
WITH Store AS(
SELECT
Sales.Customer.StoreID AS StoreID
              , Store.Demographics.value('declare default element namespace
         "http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/StoreSurvey";
      (/StoreSurvey/SquareFeet)[1]', 'int') AS SquareFeet
   , Store.Demographics.value(
      'declare default element namespace
         "http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/StoreSurvey";
      (/StoreSurvey/NumberEmployees)[1]', 'int') AS NumEmployees
   , Sales.SalesOrderHeader.TotalDue AS Total
FROM Sales.SalesOrderHeader
INNER JOIN Sales.Customer
ON Sales.SalesOrderHeader.CustomerID = Sales.Customer.CustomerID
INNER JOIN Sales.Store
ON Sales.Customer.StoreID = Sales.Store.BusinessEntityID
)
SELECT StoreID, SquareFeet, NumEmployees, SUM(Total) AS TotalDue
FROM Store
GROUP BY StoreID, SquareFeet, NumEmployees
```

	StoreID	SquareFeet	NumEmployees	TotalDue
4	298	18000	16	77585.195
5	300	21000	17	249804.8673
6	302	9000	8	428350.5326
7	304	7000	9	7431.0704
8	306	17000	10	98273.5468
9	308	72000	66	158025.1722
10	310	39000	40	6387.5291

```
SELECT
   Sales.vStoreWithDemographics.BusinessEntityID AS StoreID
```

```
, Sales.vStoreWithDemographics.SquareFeet
  , Sales.vStoreWithDemographics.NumberEmployees
  , Sales.vStoreWithDemographics.AnnualRevenue
FROM Sales.vStoreWithDemographics
```

	StoreID	SquareFeet	NumberEmployees	AnnualRevenue
1	292	21000	13	80000.00
2	294	18000	14	80000.00
3	296	21000	15	80000.00
4	298	18000	16	80000.00
5	300	21000	17	80000.00
6	302	9000	8	30000.00
7	304	7000	9	30000.00
8	306	17000	10	80000.00
9	308	72000	66	300000.00
10	310	39000	40	150000.00

Python Processing Script & Visualisation

```
storesize = pd.read_csv("Q6.csv")
sns.set(style="whitegrid")
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
sns.scatterplot(data=storesize, x="SquareFeet", y="TotalDue", ax=axes[0], color="steelblue")
axes[0].set_title("Store Size vs Revenue")
axes[0].set_xlabel("Square Feet")
axes[0].set_ylabel("Revenue")

sns.scatterplot(data=storesize, x="NumEmployees", y="TotalDue", ax=axes[1], color="lightcoral")
axes[1].set_title("Number of Employees vs Revenue")
axes[1].set_xlabel("Number of Employees")
axes[1].set_ylabel("Revenue")

plt.tight_layout()
plt.show()
```

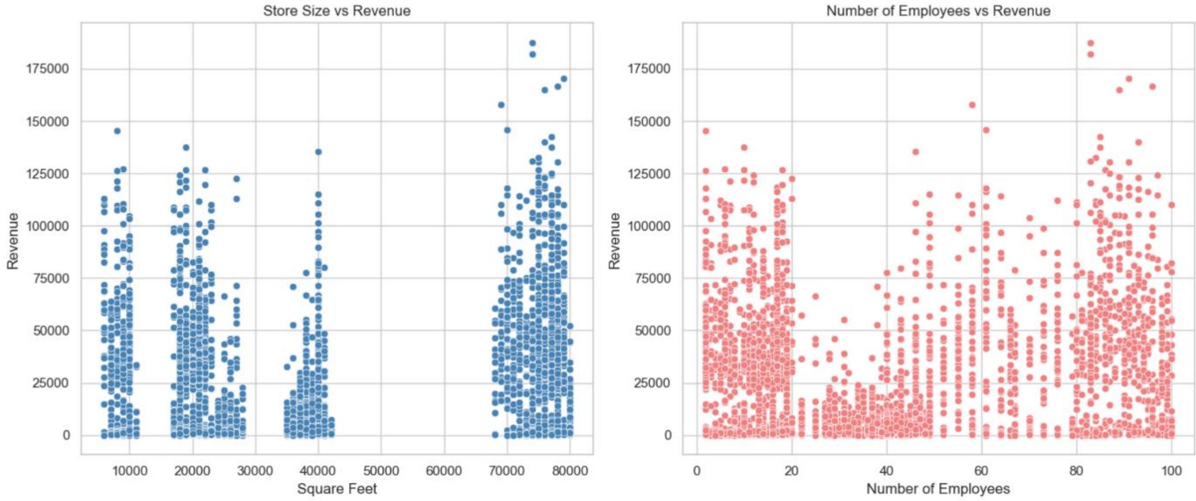


Figure17 - Scatter: Store Size vs Revenue and Employees vs Revenue

```
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
sns.set(style="whitegrid")

sns.barplot(data=storesize, x="SquareFeetBin", y="TotalDue", estimator="mean", ci=None, palette="Blues_d",
ax=axes[0])
axes[0].set_title("Average Revenue by Store Size", fontsize=13)
axes[0].set_xlabel("Square Feet",labelpad=15)
axes[0].set_ylabel("Average Revenue")
axes[0].tick_params(axis='x', rotation=90, labelsize=10)

sns.barplot(data=storesize, x="NumEmployeesBin", y="TotalDue", estimator="mean", ci=None, palette="Greens_d",
ax=axes[1])
axes[1].set_title("Average Revenue by Number of Employees", fontsize=13)
axes[1].set_xlabel("Number of Employees", labelpad=15)
axes[1].set_ylabel("Average Revenue")
axes[1].tick_params(axis='x', rotation=0)
plt.tight_layout()
plt.show()
```

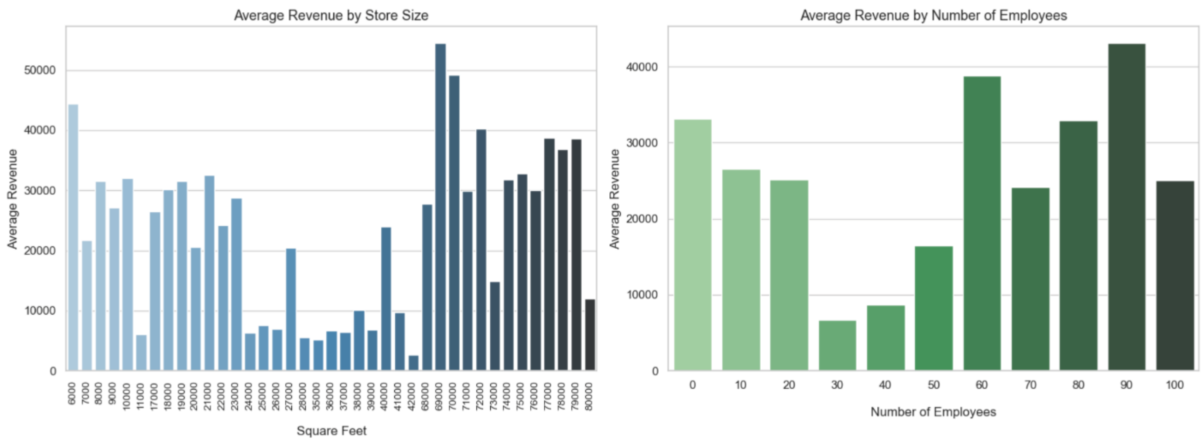


Figure18 - Average revenue by Store Size and Employee Count (bar chart)

```
plt.figure(figsize=(12, 6))
sns.set(style="whitegrid")
scatter = sns.scatterplot(data=storesize, x="SquareFeet", y="TotalDue", size="NumEmployees",
hue="NumEmployees", palette="viridis", sizes=(20, 300), alpha=0.7, legend="brief")

plt.title("Store Size vs Revenue\n(Dot Size & Color Represent Number of Employees)", fontsize=14)
plt.xlabel("Store Size (Square Feet)", fontsize=12)
plt.ylabel("Revenue (Total Due)", fontsize=12)
plt.xticks(rotation=0)

plt.legend( title="Employees", bbox_to_anchor=(1.01, 1), loc='upper left', borderaxespad=0)
plt.show()
```

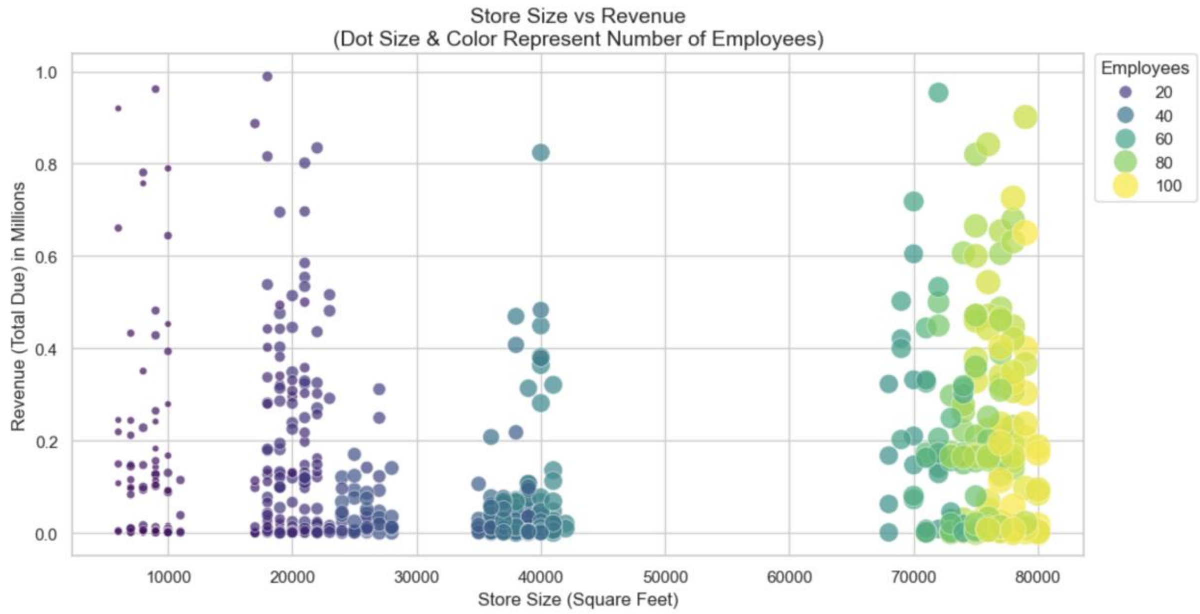


Figure19 - Scatter size vs employee vs revenue

```python
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cm
import matplotlib.colors as colors

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Normalise colour based on number of employees
norm = colors.Normalize(vmin=storesize["NumEmployees"].min(), vmax=storesize["NumEmployees"].max())
cmap = cm.get_cmap('viridis')
scatter = ax.scatter(storesize["SquareFeet"],
storesize["NumEmployees"], storesize["TotalDue"],
c=cmap(norm(storesize["NumEmployees"])), s=50, alpha=0.8)

ax.set_xlabel("Store Size (Square Feet)", fontsize=12, labelpad=12)
ax.set_ylabel("Number of Employees", fontsize=12, labelpad=12)
ax.set_zlabel("Revenue (TotalDue)", fontsize=12, labelpad=12)

mappable = cm.ScalarMappable(norm=norm, cmap=cmap)
mappable.set_array([])
cbar = fig.colorbar(mappable, ax=ax, pad=0.1, shrink=0.6)
cbar.set_label("Number of Employees")
plt.show()
```

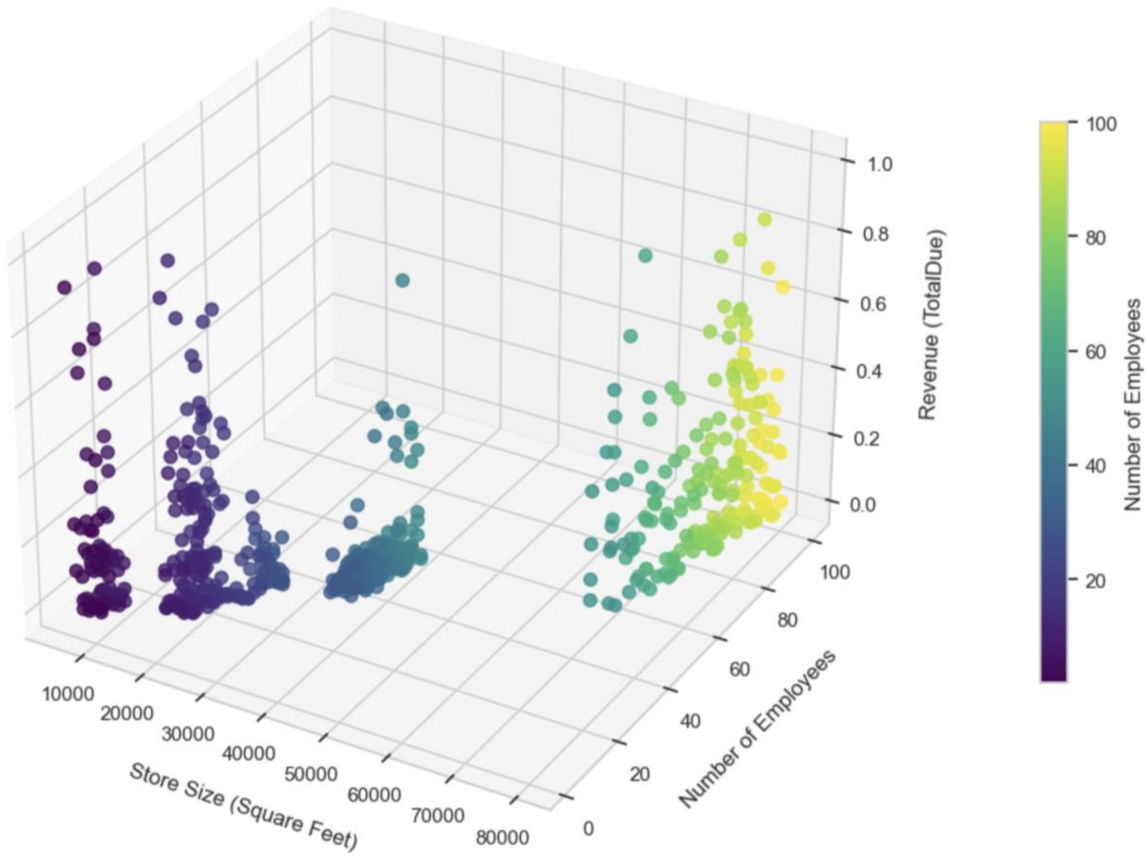


Figure20 - 3-D scatter (size, employees, revenue) highlights the two high-performing clusters.

```python
storesize_num = storesize[["SquareFeet", "NumEmployees", "TotalDue"]]
corr = storesize_num.corr()
mask = np.zeros_like(corr)
np.fill_diagonal(mask, True)
```

```
plt.figure(figsize=(6, 5))
sns.heatmap(corr, annot=True, fmt=".2f",
cmap="coolwarm", vmin=-1, vmax=1, linewidths=0.5,
linecolor='gray', mask=mask, square=True,
cbar_kws={"shrink": 0.75})
# Add diagonal manually in gray
for i in range(len(corr)):
plt.text(i + 0.5, i + 0.5, "1.00", ha='center', va='center', color='gray', fontsize=10)
plt.title("Correlation Heatmap", fontsize=13)
plt.tight_layout()
plt.show()
```

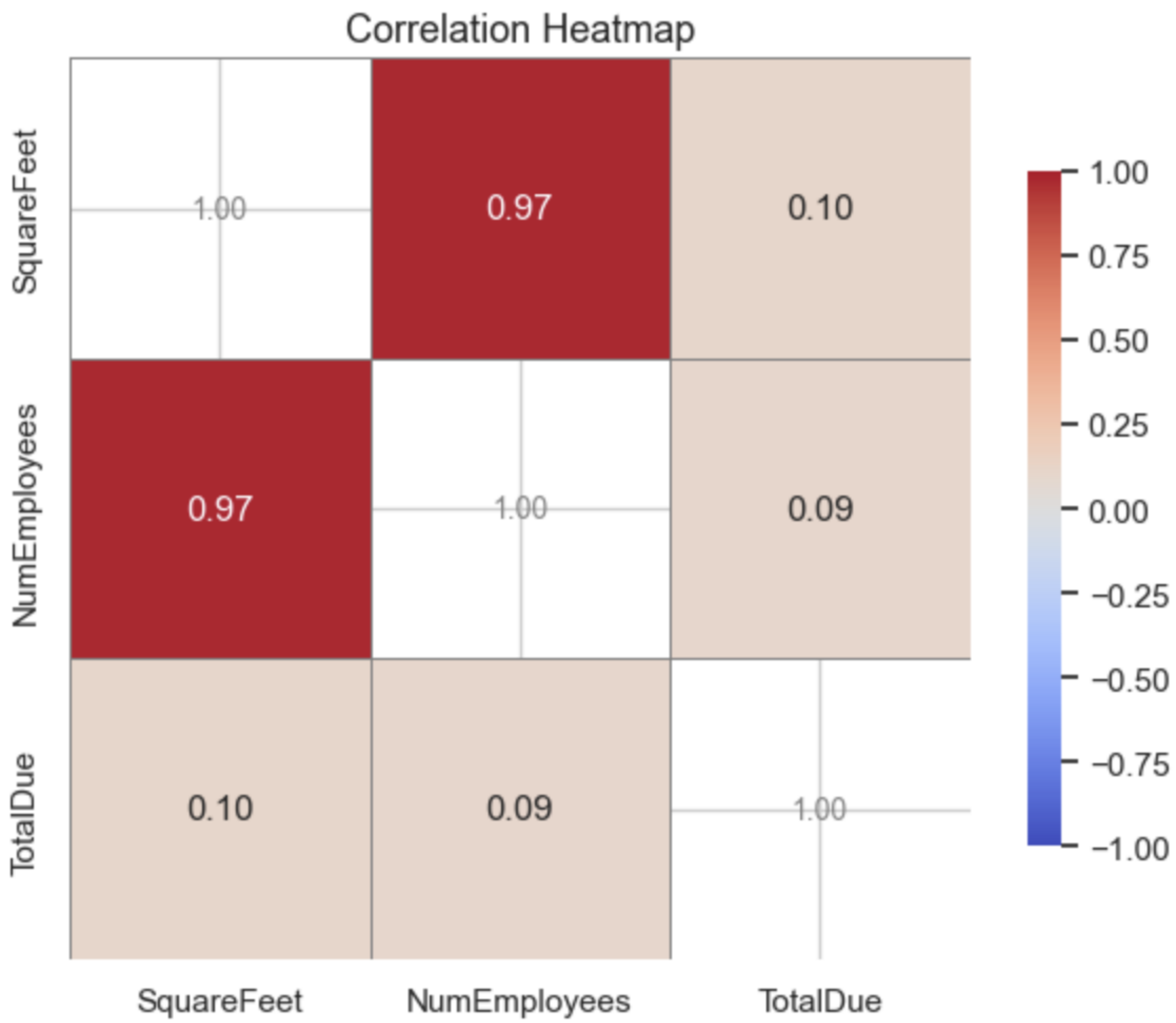


Figure21 - Heat-map with correlation coefficients.

```
# Custom function for scatter + regression + correlation
def scatter_with_trend_and_corr(x, y, **kwargs):
ax = plt.gca()
# Scatter and regression line
sns.regplot(x=x, y=y, scatter_kws={'s': 40, 'alpha': 0.6}, line_kws={'color': 'red'}, ax=ax)
r = np.corrcoef(x, y)[0, 1]
ax.annotate(f"r = {r:.2f}", xy=(0.05, 0.9),
xycoords='axes fraction',
fontsize=11, fontweight='bold', color='black')
g = sns.PairGrid(storesize_num, diag_sharey=False)
g.map_lower(scatter_with_trend_and_corr)
g.map_upper(scatter_with_trend_and_corr)
```

```
g.map_diag(sns.histplot, kde=True, color="lightblue")
plt.tight_layout()
plt.show()
```

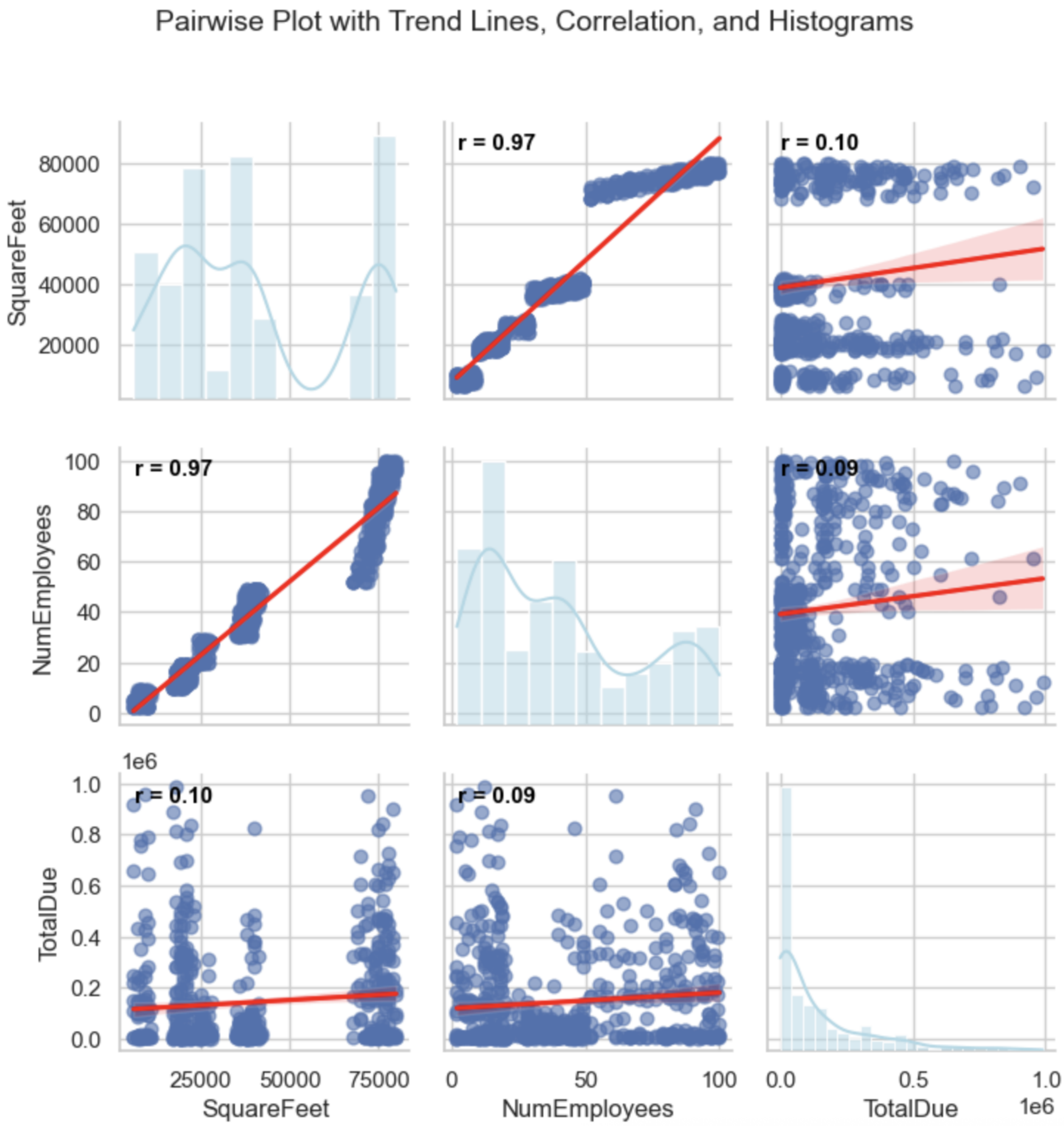


Figure22 - Pair-plot with correlation coefficients.

```
# Categorise store size and employee count
storesize["SizeCategory"] = pd.qcut(storesize["SquareFeet"], q=3, labels=["Small", "Medium", "Large"])
storesize["EmployeeCategory"] = pd.qcut(storesize["NumEmployees"], q=3, labels=["Few", "Average", "Many"])

size_group = storesize.groupby("SizeCategory")["TotalDue"].mean().reset_index()
size_group["Group"] = "Store Size"
size_group.rename(columns={"SizeCategory": "Category"}, inplace=True)
emp_group = storesize.groupby("EmployeeCategory")["TotalDue"].mean().reset_index()
emp_group["Group"] = "Employee Count"
emp_group.rename(columns={"EmployeeCategory": "Category"}, inplace=True)
combined = pd.concat([size_group, emp_group])

plt.figure(figsize=(10, 6))
sns.set(style="whitegrid")
sns.barplot(data=combined,
```

```
x="Category", y="TotalDue",
hue="Group", palette="Set2")
plt.title("Average Revenue by Store Size and Employee Count")
plt.xlabel("Category")
plt.ylabel("Average Revenue")
plt.legend(title="Group")
plt.tight_layout()
plt.show()
```

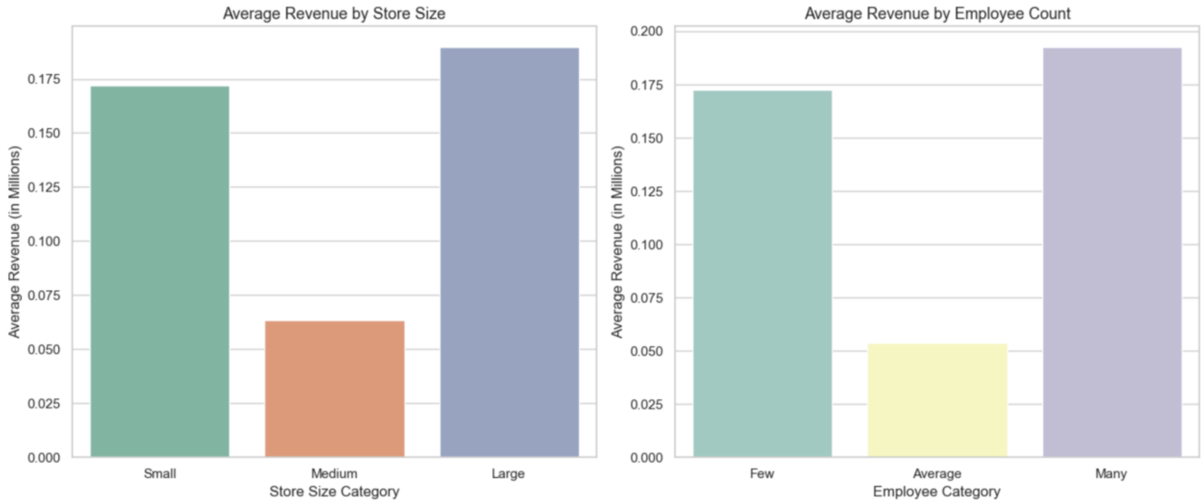


Figure23 - Average-revenue bars Store Size and Employee Count category

```
plt.figure(figsize=(12, 5))

# Boxplot by Store Size
plt.subplot(1, 2, 1)
sns.boxplot(data=storesize, x="SizeCategory", y="TotalDue", palette="Set2")
plt.title("Revenue by Store Size")
plt.xlabel("Store Size Category")
plt.ylabel("Store Revenue (in Millions)")

# Boxplot by Employee Category
plt.subplot(1, 2, 2)
sns.boxplot(data=storesize, x="EmployeeCategory", y="TotalDue", palette="Set3")
plt.title("Revenue by Employee Count")
plt.xlabel("Employee Category")
plt.ylabel("Store Revenue (in Millions)")

plt.tight_layout()
plt.show()
```

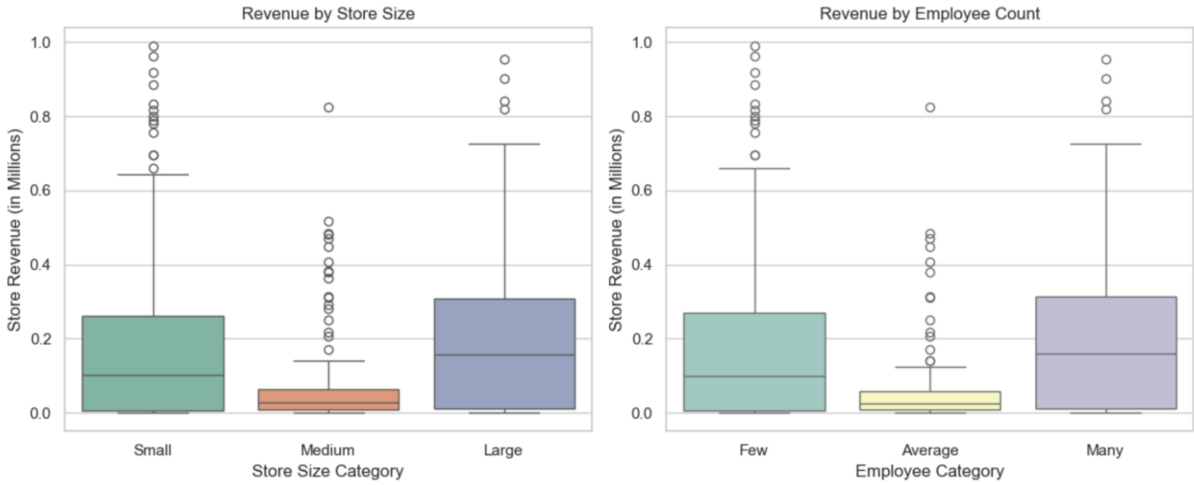


Figure24 - box-plots showing spread and outliers.

```
# Match categories: Small ↔ Few, Medium ↔ Average, Large ↔ Many
group_map = {"Small": "Few","Medium": "Average","Large": "Many"}
rows = []
for size_label, emp_label in group_map.items():
size_mean = storesize.loc[storesize["SizeCategory"] == size_label, "TotalDue"].mean()
emp_mean = storesize.loc[storesize["EmployeeCategory"] == emp_label, "TotalDue"].mean()
rows.append({"Group": size_label, "Type": "Store Size", "AvgRevenue": size_mean})
rows.append({"Group": size_label, "Type": "Employee Count", "AvgRevenue": emp_mean})
# Create DataFrame for plotting
compare_store = pd.DataFrame(rows)
plt.figure(figsize=(10, 6))
sns.set(style="whitegrid")
sns.barplot(data=compare_store, x="Group", y="AvgRevenue", hue="Type", palette="Set2")
plt.title("Average Revenue by Store Size and Employee Count (Matched Groups)")
plt.legend(title="Type")
plt.tight_layout()
plt.show()
```

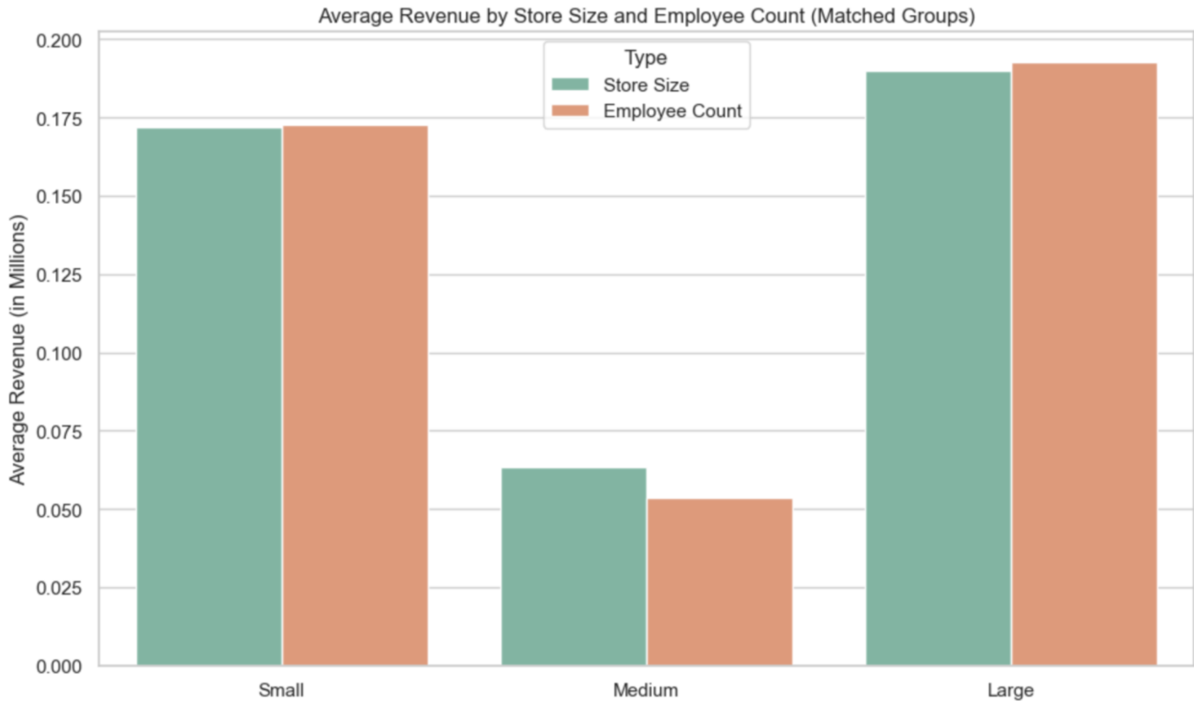


Figure25 - Side-by-side bar – matched groups (Small/Few, Medium/Average, Large/Many)

```
sns.set(style="whitegrid")
sns.set(style="whitegrid")
scatter = sns.scatterplot(
data=store_relationship,
x="SquareFeet",
y="AnnualRevenue",
size="NumberEmployees",
hue="NumberEmployees",
palette="viridis",
sizes=(20, 300),
alpha=0.7,
legend="brief"
)
plt.title("Store Size vs Revenue\n(Dot Size & Color Represent Number of Employees)", fontsize=14)
plt.xlabel("Store Size (Square Feet)", fontsize=12)
plt.ylabel("Revenue (Annual)", fontsize=12)
plt.xticks(rotation=0)
plt.legend(
title="Employees",
bbox_to_anchor=(1.01, 1),
loc='upper left',
borderaxespad=0
)
plt.tight_layout()
plt.show()
```

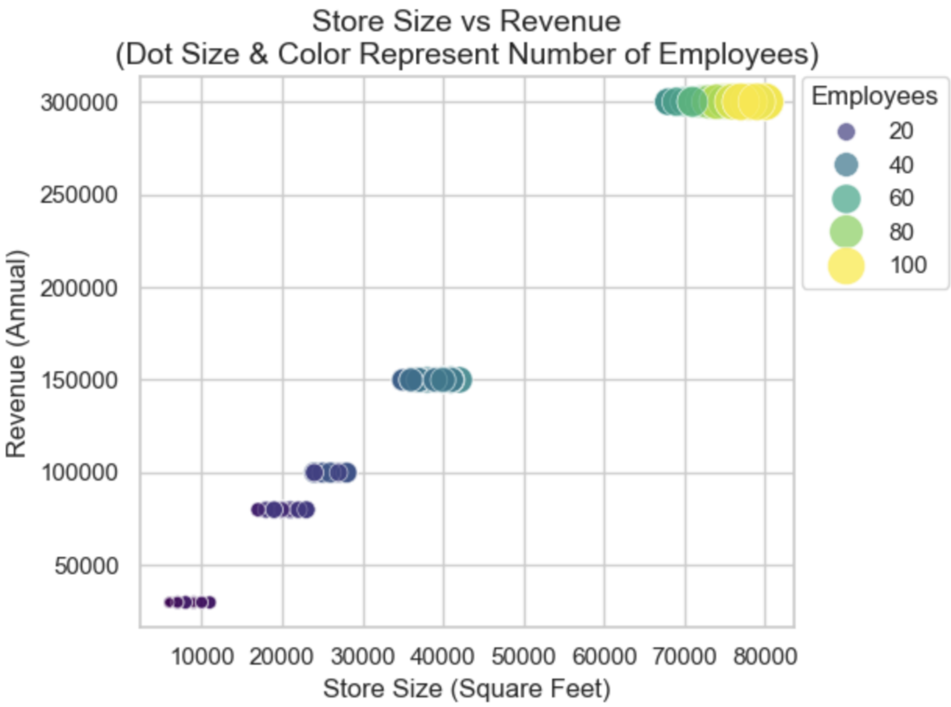


Figure26 - Scatter size vs employee vs Annual Revenue

```
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
sns.scatterplot(data=store_relationship, x="SquareFeet", y="AnnualRevenue", ax=axes[0], color="steelblue")
axes[0].set_title("Store Size vs Revenue")
axes[0].set_xlabel("Square Feet")
axes[0].set_ylabel("Revenue")

sns.scatterplot(data=store_relationship, x="NumberEmployees", y="AnnualRevenue", ax=axes[1], color="lightcoral")
axes[1].set_title("Number of Employees vs Revenue")
```

```
axes[1].set_xlabel("Number of Employees")
axes[1].set_ylabel("Revenue")
plt.tight_layout()
plt.show()
```

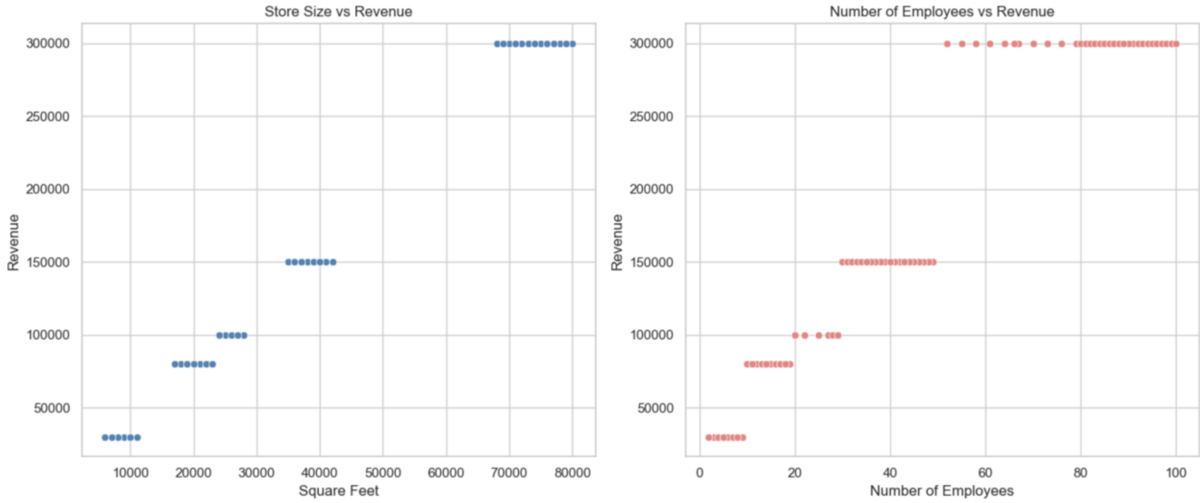


Figure27 - Scatter: Store Size vs Revenue and Employees vs Annual Revenue