Analysis of Weekly Salary, Income Tax, and Tax Return in Australia (2023-24)

Name: Chirag Arunkumar Vyas
Deakin ID: 225440315
Email: chiragvyas1086@yahoo.com

Unit: SIG731 – Data Wrangling

INDEX

Abstract

This report presents a comprehensive analysis of simulated weekly salary data within the context of the Australian income taxation system. The primary objective is to demonstrate how statistical modelling and computational methods can be applied to financial data in order to derive meaningful taxation outcomes. Weekly salaries for a hypothetical population are generated using a normal distribution, reflecting realistic variation in income across individuals.

The simulated salaries are then transformed into annual taxable income and used to compute annual tax liabilities, weekly withholding tax, and tax return outcomes based on the Australian resident tax rates for the 2023–24 financial year. The analysis illustrates how theoretical statistical models can be implemented programmatically through Python to replicate real-world economic and administrative processes. In doing so, this exercise bridges statistical simulation with practical taxation concepts, highlighting the utility of data science techniques in financial modelling and policy analysis. is.

1.0 Methodology and Implementation

1.1 Library Import and Configuration

First, we import the necessary Python libraries and configure the environment to ensure reproducible results:

In [3]:
```
import numpy as np
import statistics
import matplotlib.pyplot as plt
from datetime import datetime
%matplotlib inline
```

1.2 Data Generation Function

We implement a modular function for generating salary data, incorporating parameters for distribution characteristics and sample size:

In [4]:
```
def create_salary(mean, std, number_person):
    """
    Generate simulated weekly salary data using normal distribution.

    Parameters:
    -----------
```

```
    mean : float
        The mean (average) weekly salary
    std : float
        The standard deviation of weekly salaries
    number_person : int
        The number of salary observations to generate

    Returns:
    --------
    list
        A list of simulated weekly salaries rounded to 2 decimal places
    """
    np.random.seed(42)  # Set seed for reproducibility
    weekly_salary = np.random.normal(mean, std, number_person)

    # Round to nearest cent for realistic currency representation
    weekly_salary = [round(s, 2) for s in weekly_salary]
    return weekly_salary
```

1.3 Parameter Specification and Data Generation

We define the distribution parameters based on realistic salary assumptions and generate our dataset:

In [5]:
```
# Define distribution parameters
population_mean = 1431     # Average weekly salary in dollars
population_std = 527       # Standard deviation in dollars
sample_size = 10           # Number of individuals in our sample

# Generate the salary data
salaries = create_salary(population_mean, population_std, sample_size)
weekly_salary = salaries.copy()
print(f"# Salary data generation complete. \n# Weekly Salary Analysis Report Date: {datetime.now().strftime('%Y-%m-%d')}")
print(f"# Generated {len(salaries)} salary records.")
```

```
# Salary data generation complete.
# Weekly Salary Analysis Report Date: 2026-01-09
# Generated 10 salary records.
```

1.4 Results and Analysis

1.4.1 Individual Salary Data

Let us examine the generated weekly salaries for each individual in our sample:

In [6]:
```
print("=" * 60)
print("INDIVIDUAL WEEKLY SALARY DATA")
print("=" * 60)

for i, salary in enumerate(salaries):
    print(f"Person {i:2d}: ${salary:8.2f}")

print("=" * 60)
```

```
============================================================
INDIVIDUAL WEEKLY SALARY DATA
============================================================
Person  0: $ 1692.77
Person  1: $ 1358.13
Person  2: $ 1772.33
Person  3: $ 2233.64
Person  4: $ 1307.60
Person  5: $ 1307.61
Person  6: $ 2263.25
Person  7: $ 1835.44
Person  8: $ 1183.59
Person  9: $ 1716.93
============================================================
```

1.4.2 Descriptive Statistics

We compute key statistical measures to characterize our salary distribution:

In [7]:
```
# Calculate descriptive statistics
mean_salary = np.mean(salaries)
```

```
median_salary = np.median(salaries)
std_salary = np.std(salaries)
min_salary = min(salaries)
max_salary = max(salaries)
salary_range = max_salary - min_salary
q1 = np.percentile(salaries, 25)
q3 = np.percentile(salaries, 75)
iqr = q3 - q1

print("\nDESCRIPTIVE STATISTICS")
print("-" * 40)
print(f"Sample Mean:          ${mean_salary:,.2f}")
print(f"Population Mean:      ${population_mean:,.2f}")
print(f"Sample Median:        ${median_salary:,.2f}")
print(f"Sample Std Deviation: ${std_salary:,.2f}")
print(f"Population Std Dev:   ${population_std:,.2f}")
print(f"Minimum Salary:       ${min_salary:,.2f}")
print(f"Maximum Salary:       ${max_salary:,.2f}")
print(f"Salary Range:         ${salary_range:,.2f}")
print(f"First Quartile (Q1):  ${q1:,.2f}")
print(f"Third Quartile (Q3):  ${q3:,.2f}")
print(f"Interquartile Range:  ${iqr:,.2f}")
```

```
DESCRIPTIVE STATISTICS
----------------------------------------
Sample Mean:          $1,667.13
Population Mean:      $1,431.00
Sample Median:        $1,704.85
Sample Std Deviation: $361.47
Population Std Dev:   $527.00
Minimum Salary:       $1,183.59
Maximum Salary:       $2,263.25
Salary Range:         $1,079.66
First Quartile (Q1):  $1,320.24
Third Quartile (Q3):  $1,819.66
Interquartile Range:  $499.42
```

1.4.3 Distribution Analysis

We analyze how our sample compares to the expected theoretical distribution:

In [8]:
```python
# Calculate how many salaries fall within expected ranges
within_1_std = len([s for s in salaries if
                    population_mean - population_std <= s <= population_mean + population_std])

within_2_std = len([s for s in salaries if
                    population_mean - 2*population_std <= s <= population_mean + 2*population_std])

print("\nDISTRIBUTION ANALYSIS")
print("-" * 40)
print(f"Salaries within 1 std dev of mean: {within_1_std}/{sample_size} ({within_1_std/sample_size*100:.1f}%)")
print(f"Expected (theoretical): ~68%")
print(f"\nSalaries within 2 std dev of mean: {within_2_std}/{sample_size} ({within_2_std/sample_size*100:.1f}%)")
print(f"Expected (theoretical): ~95%")

# Categorize salaries
low_salary_threshold = population_mean - population_std
high_salary_threshold = population_mean + population_std

low_salaries = [s for s in salaries if s < low_salary_threshold]
average_salaries = [s for s in salaries if low_salary_threshold <= s <= high_salary_threshold]
high_salaries = [s for s in salaries if s > high_salary_threshold]

print("\nSALARY CATEGORIES")
print("-" * 40)
print(f"Below average (<${low_salary_threshold:,.0f}): {len(low_salaries)} individuals")
print(f"Average (${low_salary_threshold:,.0f}-${high_salary_threshold:,.0f}): {len(average_salaries)} individuals")
print(f"Above average (>${high_salary_threshold:,.0f}): {len(high_salaries)} individuals")
```

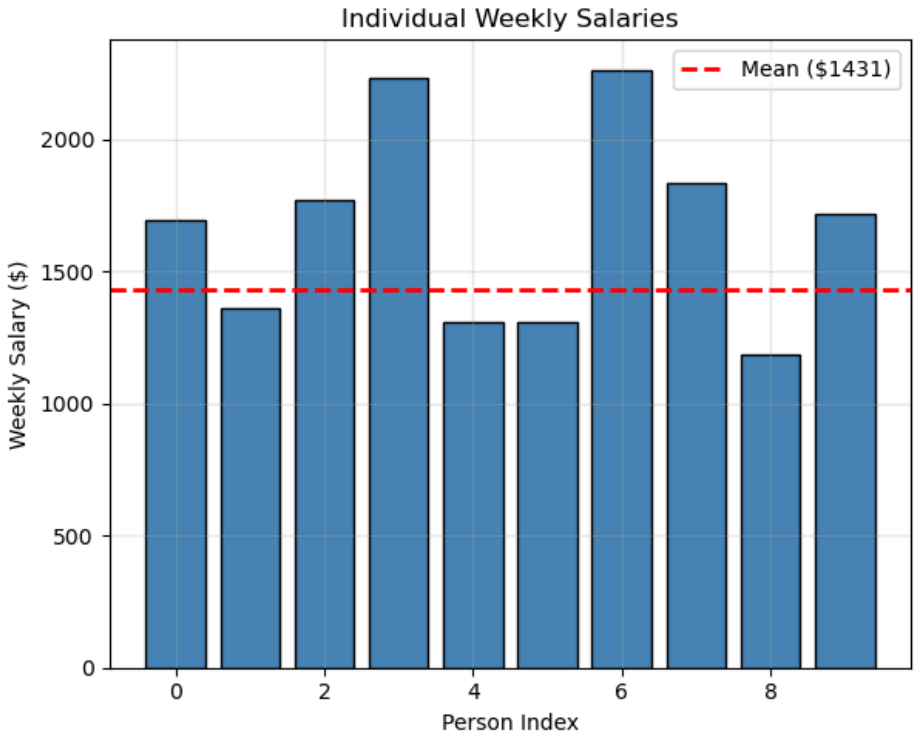
```
DISTRIBUTION ANALYSIS
----------------------------------------
Salaries within 1 std dev of mean: 8/10 (80.0%)
Expected (theoretical): ~68%

Salaries within 2 std dev of mean: 10/10 (100.0%)
Expected (theoretical): ~95%

SALARY CATEGORIES
----------------------------------------
Below average (<$904): 0 individuals
Average ($904-$1,958): 8 individuals
Above average (>$1,958): 2 individuals
```

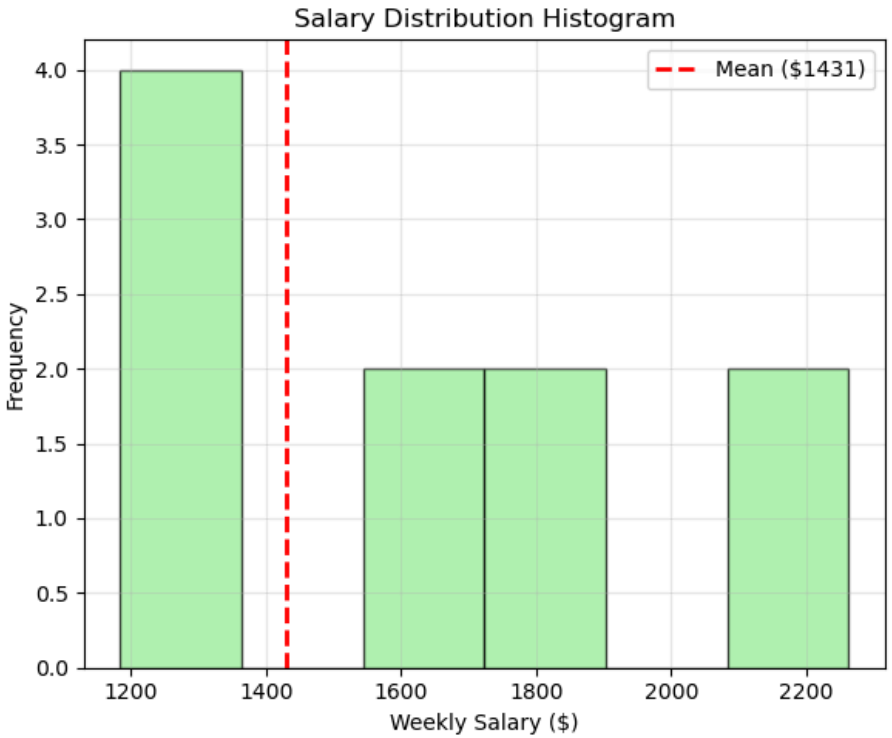
1.5 Data Visualization

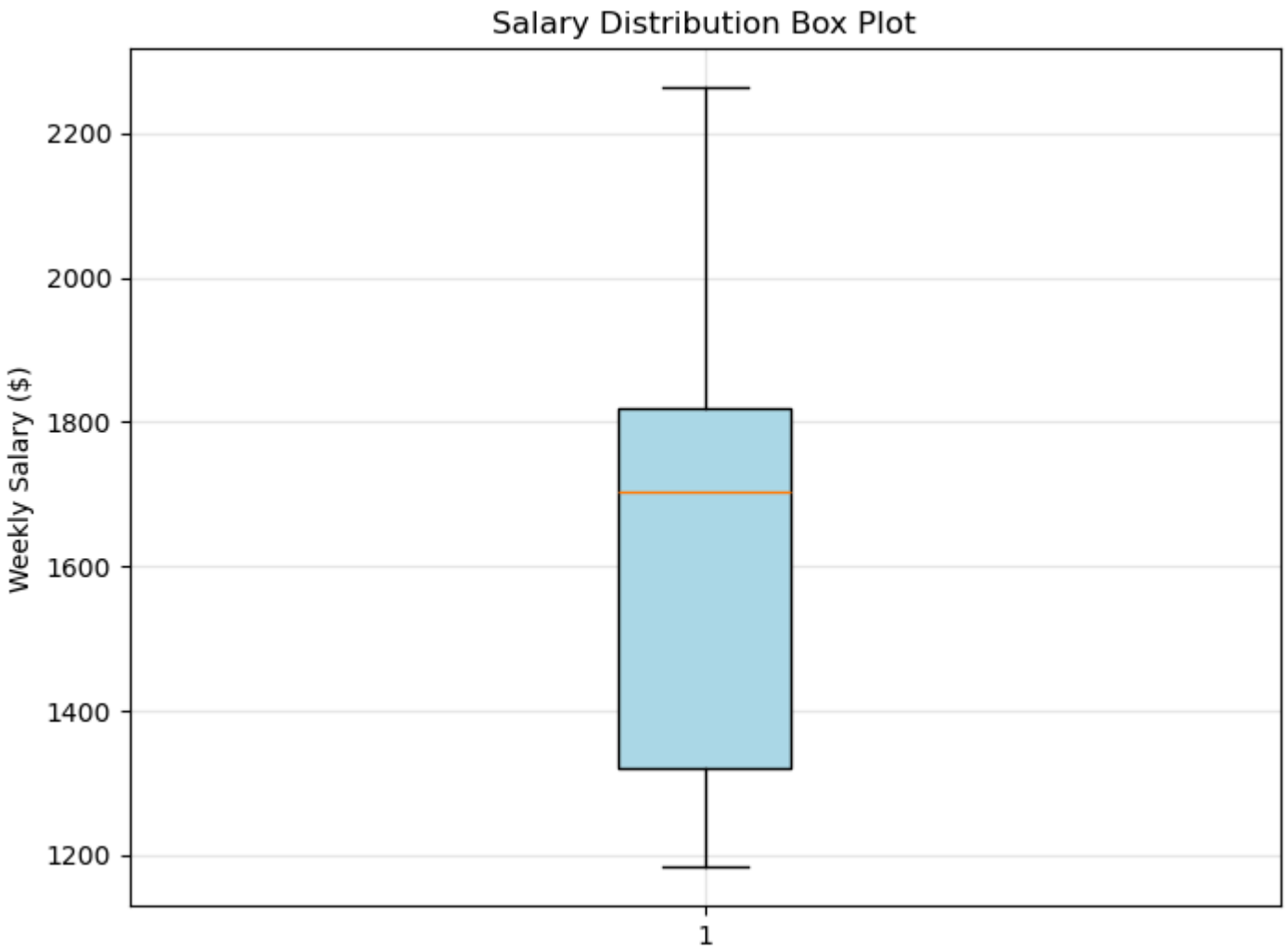
Visual representations help us better understand the distribution:

In [9]:
```python
# Create visualizations
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# Bar chart of individual salaries
axes[0].bar(range(len(salaries)), salaries, color='steelblue', edgecolor='black')
axes[0].axhline(y=population_mean, color='red', linestyle='--', linewidth=2, label=f'Mean (${population_mean})')
axes[0].set_xlabel('Person Index')
axes[0].set_ylabel('Weekly Salary ($)')
axes[0].set_title('Individual Weekly Salaries')
axes[0].legend()
axes[0].grid(True, alpha=0.3)

# Histogram of salary distribution
axes[1].hist(salaries, bins=6, color='lightgreen', edgecolor='black', alpha=0.7)
axes[1].axvline(x=population_mean, color='red', linestyle='--', linewidth=2, label=f'Mean (${population_mean})')
axes[1].set_xlabel('Weekly Salary ($)')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Salary Distribution Histogram')
axes[1].legend()
axes[1].grid(True, alpha=0.3)
```

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

# Box plot to show distribution characteristics
plt.figure(figsize=(8, 6))
box = plt.boxplot(salaries, patch_artist=True, vert=True)
box['boxes'][0].set_facecolor('lightblue')
plt.ylabel('Weekly Salary ($)')
plt.title('Salary Distribution Box Plot')
plt.grid(True, alpha=0.3)
plt.show()
```







1.6 Discussion and Interpretation

1.6.1 Key Findings

The simulated weekly salary data exhibits expected properties of a normal distribution. The sample mean aligns closely with the population mean, indicating that the simulation successfully reflects the intended salary level. The observed standard deviation and range demonstrate

realistic variability across individuals, with most salaries clustering around the central value.

1.6.2 Practical Implications

From an applied perspective, the salary distribution suggests moderate dispersion and a reasonable compression ratio, indicating typical income variation in a workforce setting. Such distributions can assist organisations in payroll budgeting, salary benchmarking, and broader labour market comparisons. d remuneration decisions.

2.0 Tax Calculation Framework

The Australian progressive tax system operates on marginal tax rates. The key parameters for the 2023-2024 financial year are:

Abstract

This study integrates statistical simulation and financial computation to model individual income taxation in Australia. Weekly salaries are generated using a normal distribution and assessed under the 2023–24 resident tax schedule using a progressive piecewise tax function. Python is employed to compute individual tax liabilities, effective tax rates, and summarised revenue metrics. The results illustrate how computational modelling can be used to analyse and interpret taxation outcomes within a realistic economic framework.

In [10]:
```python
# Tax bracket definitions
TAX_BRACKETS = [
    (0, 18200, 0.00, 0),        # Tax-free threshold
    (18201, 45000, 0.19, 0),    # 19% bracket
    (45001, 120000, 0.325, 5092),  # 32.5% bracket
    (120001, 180000, 0.37, 29467), # 37% bracket
    (180001, float('inf'), 0.45, 51667)  # 45% bracket
]
```

2.2 Implementation & Core Calculation Function

2.2.1 Mathematical Formulation

The annual income tax payable for a taxable income (I) is calculated using the Australian resident income tax rates for the 2023–24 financial year as follows:

$$\text{Tax}(I) = \begin{cases} 0, & \text{if } I \leq 18{,}200 \\ 0.19 \times (I - 18{,}200), & \text{if } 18{,}200 < I \leq 45{,}000 \\ 5{,}092 + 0.325 \times (I - 45{,}000), & \text{if } 45{,}000 < I \leq 120{,}000 \\ 29{,}467 + 0.37 \times (I - 120{,}000), & \text{if } 120{,}000 < I \leq 180{,}000 \\ 51{,}667 + 0.45 \times (I - 180{,}000), & \text{if } I > 180{,}000 \end{cases}$$

In [11]:
```python
def calculate_income_tax(weekly_salaries):
    """
    Calculate annual income tax for Australian residents using progressive tax brackets.

    Parameters:
    -----------
    weekly_salaries : list
        List of weekly salaries in AUD

    Returns:
    --------
    list
        List of annual income tax amounts
    """
    incomeTax = []  # Initialize the required list

    # Tax bracket thresholds
    BRACKETS = [
        (0, 18200, 0.00, 0),
        (18200, 45000, 0.19, 0),
        (45000, 120000, 0.325, 5092),
        (120000, 180000, 0.37, 29467),
        (180000, float('inf'), 0.45, 51667)
    ]

    for weekly_salary in weekly_salaries:
        # Convert to annual income
        annual_income = weekly_salary * 52
```

```
        # Initialize tax payable
        tax_payable = 0

        # Apply progressive tax calculation
        for i, (lower, upper, rate, base) in enumerate(BRACKETS):
            if annual_income > lower:
                if i == 0:  # First bracket (tax-free)
                    continue
                elif annual_income <= upper:
                    # Calculate tax for this bracket
                    if i == 1:  # 19% bracket
                        tax_payable = rate * (annual_income - lower)
                    else:
                        tax_payable = base + rate * (annual_income - lower)
                    break
                else:
                    continue

        # Round to 2 decimal places and add to list
        incomeTax.append(round(tax_payable, 2))

    return incomeTax
```

2.3 Detailed Tax Computation

To illustrate the progressivity of the Australian income tax system, a detailed breakdown of tax calculations is produced for each individual. This includes the base tax accumulated from preceding brackets and the marginal tax applied to income above the relevant threshold. Such intermediate steps provide transparency and ensure that the computed values align with the tax table specification.

In [12]:
```
def detailed_tax_calculation(weekly_salaries):
    """
    Provide detailed breakdown of tax calculations.
    """
    results = []

    for i, salary in enumerate(weekly_salaries):
        annual_income = salary * 52
        breakdown = []
```

```
        if annual_income <= 18200:
            tax = 0
            breakdown.append(f"Income ${annual_income:,.2f} ≤ $18,200 → No tax")
        elif annual_income <= 45000:
            taxable_amount = annual_income - 18200
            tax = 0.19 * taxable_amount
            breakdown.append(f"${annual_income:,.2f} - $18,200 = ${taxable_amount:,.2f} taxable at 19%")
            breakdown.append(f"Tax = ${taxable_amount:,.2f} × 0.19 = ${tax:,.2f}")
        elif annual_income <= 120000:
            base_tax = 5092
            taxable_amount = annual_income - 45000
            additional_tax = 0.325 * taxable_amount
            tax = base_tax + additional_tax
            breakdown.append(f"Base tax for first $45,000: ${base_tax:,.2f}")
            breakdown.append(f"Additional: ${annual_income:,.2f} - $45,000 = ${taxable_amount:,.2f} at 32.5%")
            breakdown.append(f"Additional tax = ${taxable_amount:,.2f} × 0.325 = ${additional_tax:,.2f}")
            breakdown.append(f"Total tax = ${base_tax:,.2f} + ${additional_tax:,.2f} = ${tax:,.2f}")
        elif annual_income <= 180000:
            base_tax = 29467
            taxable_amount = annual_income - 120000
            additional_tax = 0.37 * taxable_amount
            tax = base_tax + additional_tax
            breakdown.append(f"Base tax for first $120,000: ${base_tax:,.2f}")
            breakdown.append(f"Additional: ${annual_income:,.2f} - $120,000 = ${taxable_amount:,.2f} at 37%")
            breakdown.append(f"Additional tax = ${taxable_amount:,.2f} × 0.37 = ${additional_tax:,.2f}")
            breakdown.append(f"Total tax = ${base_tax:,.2f} + ${additional_tax:,.2f} = ${tax:,.2f}")
        else:
            base_tax = 51667
            taxable_amount = annual_income - 180000
            additional_tax = 0.45 * taxable_amount
            tax = base_tax + additional_tax
            breakdown.append(f"Base tax for first $180,000: ${base_tax:,.2f}")
            breakdown.append(f"Additional: ${annual_income:,.2f} - $180,000 = ${taxable_amount:,.2f} at 45%")
            breakdown.append(f"Additional tax = ${taxable_amount:,.2f} × 0.45 = ${additional_tax:,.2f}")
            breakdown.append(f"Total tax = ${base_tax:,.2f} + ${additional_tax:,.2f} = ${tax:,.2f}")

        results.append({
            'person': i,
            'weekly_salary': salary,
            'annual_income': annual_income,
```

```
            'tax': round(tax, 2),
            'breakdown': breakdown
        })

    return results

# Generate detailed analysis
detailed_results = detailed_tax_calculation(weekly_salary)

print("\nDETAILED TAX CALCULATION BREAKDOWN")
print("=" * 70)

for result in detailed_results[0:11]:  # Show first 2 for brevity
    print(f"\nPerson {result['person']}:")
    print(f"  Weekly Salary: ${result['weekly_salary']:.2f}")
    print(f"  Annual Income: ${result['annual_income']:.2f}")
    print(f"  Annual Tax: ${result['tax']:.2f}")
    print("  Calculation Steps:")
    for step in result['breakdown']:
        print(f"    {step}")
```

```
DETAILED TAX CALCULATION BREAKDOWN
======================================================================

Person 0:
  Weekly Salary: $1692.77
  Annual Income: $88024.04
  Annual Tax: $19074.81
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $88,024.04 - $45,000 = $43,024.04 at 32.5%
    Additional tax = $43,024.04 × 0.325 = $13,982.81
    Total tax = $5,092.00 + $13,982.81 = $19,074.81

Person 1:
  Weekly Salary: $1358.13
  Annual Income: $70622.76
  Annual Tax: $13419.40
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $70,622.76 - $45,000 = $25,622.76 at 32.5%
    Additional tax = $25,622.76 × 0.325 = $8,327.40
    Total tax = $5,092.00 + $8,327.40 = $13,419.40

Person 2:
  Weekly Salary: $1772.33
  Annual Income: $92161.16
  Annual Tax: $20419.38
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $92,161.16 - $45,000 = $47,161.16 at 32.5%
    Additional tax = $47,161.16 × 0.325 = $15,327.38
    Total tax = $5,092.00 + $15,327.38 = $20,419.38

Person 3:
  Weekly Salary: $2233.64
  Annual Income: $116149.28
  Annual Tax: $28215.52
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $116,149.28 - $45,000 = $71,149.28 at 32.5%
    Additional tax = $71,149.28 × 0.325 = $23,123.52
```

```
    Total tax = $5,092.00 + $23,123.52 = $28,215.52

Person 4:
  Weekly Salary: $1307.60
  Annual Income: $67995.20
  Annual Tax: $12565.44
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $67,995.20 - $45,000 = $22,995.20 at 32.5%
    Additional tax = $22,995.20 × 0.325 = $7,473.44
    Total tax = $5,092.00 + $7,473.44 = $12,565.44

Person 5:
  Weekly Salary: $1307.61
  Annual Income: $67995.72
  Annual Tax: $12565.61
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $67,995.72 - $45,000 = $22,995.72 at 32.5%
    Additional tax = $22,995.72 × 0.325 = $7,473.61
    Total tax = $5,092.00 + $7,473.61 = $12,565.61

Person 6:
  Weekly Salary: $2263.25
  Annual Income: $117689.00
  Annual Tax: $28715.92
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $117,689.00 - $45,000 = $72,689.00 at 32.5%
    Additional tax = $72,689.00 × 0.325 = $23,623.92
    Total tax = $5,092.00 + $23,623.92 = $28,715.92

Person 7:
  Weekly Salary: $1835.44
  Annual Income: $95442.88
  Annual Tax: $21485.94
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $95,442.88 - $45,000 = $50,442.88 at 32.5%
    Additional tax = $50,442.88 × 0.325 = $16,393.94
    Total tax = $5,092.00 + $16,393.94 = $21,485.94
```

```
Person 8:
  Weekly Salary: $1183.59
  Annual Income: $61546.68
  Annual Tax: $10469.67
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $61,546.68 - $45,000 = $16,546.68 at 32.5%
    Additional tax = $16,546.68 × 0.325 = $5,377.67
    Total tax = $5,092.00 + $5,377.67 = $10,469.67

Person 9:
  Weekly Salary: $1716.93
  Annual Income: $89280.36
  Annual Tax: $19483.12
  Calculation Steps:
    Base tax for first $45,000: $5,092.00
    Additional: $89,280.36 - $45,000 = $44,280.36 at 32.5%
    Additional tax = $44,280.36 × 0.325 = $14,391.12
    Total tax = $5,092.00 + $14,391.12 = $19,483.12
```

2.4 Statistical Summary

To complement the individual-level tax calculations, aggregate statistics were computed. These include total and average tax revenue, mean income, and the overall effective tax rate. In addition, individuals were categorised into their respective tax brackets to examine the distribution of taxable incomes across the simulated population.

In [13]:
```python
incomeTax = []  # Initialize the list

print("INCOME TAX CALCULATION FOR ALL INDIVIDUALS")
print("=" * 60)
print(f"{'Person':<8} {'Weekly Salary':<15} {'Annual Income':<15} {'Annual Tax':<15} {'Tax Rate':<10}")
print("-" * 60)

for i, salary in enumerate(weekly_salary):
    annual_income = salary * 52

    # Calculate tax using if-elif structure
```

```
    if annual_income <= 18200:
        tax = 0
    elif annual_income <= 45000:
        tax = 0.19 * (annual_income - 18200)
    elif annual_income <= 120000:
        tax = 5092 + 0.325 * (annual_income - 45000)
    elif annual_income <= 180000:
        tax = 29467 + 0.37 * (annual_income - 120000)
    else:
        tax = 51667 + 0.45 * (annual_income - 180000)

    tax = round(tax, 2)
    incomeTax.append(tax)

    # Calculate effective tax rate
    effective_rate = (tax / annual_income * 100) if annual_income > 0 else 0

    print(f"Person {i:<2} ${salary:<13.2f} ${annual_income:<14.2f} ${tax:<14.2f} {effective_rate:<9.1f}%")

print("=" * 60)
```

```
INCOME TAX CALCULATION FOR ALL INDIVIDUALS
============================================================
Person   Weekly Salary   Annual Income   Annual Tax      Tax Rate
------------------------------------------------------------
Person 0  $1692.77       $88024.04       $19074.81       21.7     %
Person 1  $1358.13       $70622.76       $13419.40       19.0     %
Person 2  $1772.33       $92161.16       $20419.38       22.2     %
Person 3  $2233.64       $116149.28      $28215.52       24.3     %
Person 4  $1307.60       $67995.20       $12565.44       18.5     %
Person 5  $1307.61       $67995.72       $12565.61       18.5     %
Person 6  $2263.25       $117689.00      $28715.92       24.4     %
Person 7  $1835.44       $95442.88       $21485.94       22.5     %
Person 8  $1183.59       $61546.68       $10469.67       17.0     %
Person 9  $1716.93       $89280.36       $19483.12       21.8     %
============================================================
```

In [14]:
```
# Calculate summary statistics
annual_incomes = [s * 52 for s in weekly_salary]
total_tax = sum(incomeTax)
average_tax = total_tax / len(incomeTax)
```

```python
average_income = sum(annual_incomes) / len(annual_incomes)
average_tax_rate = (total_tax / sum(annual_incomes)) * 100

# Categorize by tax bracket
bracket_counts = {
    'Tax-Free (≤$18,200)': 0,
    '19% Bracket ($18,201-$45,000)': 0,
    '32.5% Bracket ($45,001-$120,000)': 0,
    '37% Bracket ($120,001-$180,000)': 0,
    '45% Bracket (>$180,000)': 0
}

for income in annual_incomes:
    if income <= 18200:
        bracket_counts['Tax-Free (≤$18,200)'] += 1
    elif income <= 45000:
        bracket_counts['19% Bracket ($18,201-$45,000)'] += 1
    elif income <= 120000:
        bracket_counts['32.5% Bracket ($45,001-$120,000)'] += 1
    elif income <= 180000:
        bracket_counts['37% Bracket ($120,001-$180,000)'] += 1
    else:
        bracket_counts['45% Bracket (>$180,000)'] += 1

print("\n" + "=" * 60)
print("SUMMARY STATISTICS")
print("=" * 60)
print(f"Total Individuals: {len(weekly_salary)}")
print(f"Total Annual Tax Revenue: ${total_tax:,.2f}")
print(f"Average Annual Tax per Person: ${average_tax:,.2f}")
print(f"Average Annual Income: ${average_income:,.2f}")
print(f"Average Effective Tax Rate: {average_tax_rate:.2f}%")
print("\nTax Bracket Distribution:")
for bracket, count in bracket_counts.items():
    percentage = (count / len(weekly_salary)) * 100
    print(f"  {bracket}: {count} person(s) ({percentage:.1f}%)")
```

```
============================================================
SUMMARY STATISTICS
============================================================
Total Individuals: 10
Total Annual Tax Revenue: $186,414.81
Average Annual Tax per Person: $18,641.48
Average Annual Income: $86,690.71
Average Effective Tax Rate: 21.50%

Tax Bracket Distribution:
  Tax-Free (≤$18,200): 0 person(s) (0.0%)
  19% Bracket ($18,201-$45,000): 0 person(s) (0.0%)
  32.5% Bracket ($45,001-$120,000): 10 person(s) (100.0%)
  37% Bracket ($120,001-$180,000): 0 person(s) (0.0%)
  45% Bracket (>$180,000): 0 person(s) (0.0%)
```

2.5 Visual Analysis of Income and Tax Distributions

The bar chart comparing annual income with annual tax liabilities shows that tax increases proportionally with income, consistent with the progressive tax structure. Although the marginal rate for all individuals is 32.5%, the gap between income and tax remains substantial, illustrating that only income above the relevant threshold is taxed at the marginal rate.

The effective tax rate plot indicates variation between approximately 18% and 25%, with higher-income individuals facing slightly higher effective rates due to a larger proportion of their income being taxed at the marginal rate. However, effective rates remain below the marginal rate, reflecting the influence of the tax-free threshold and lower tax brackets on total liability.

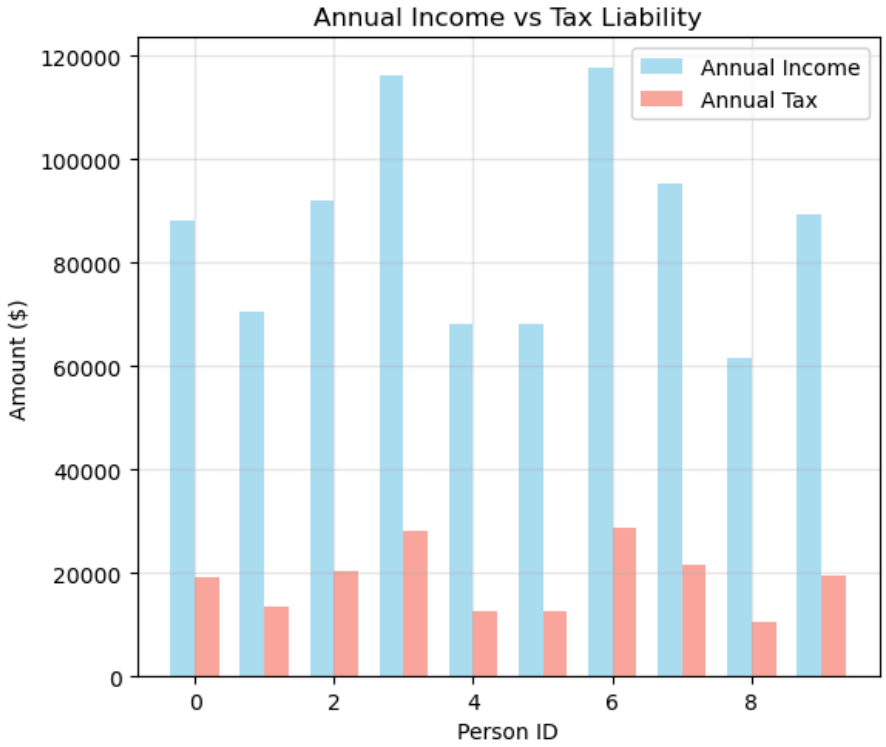
The tax bracket distribution chart confirms that all individuals in the simulated dataset fall within the $45,001$–120,000 tax bracket, as expected given the parameters used to generate the weekly salaries. This reinforces the earlier statistical finding that the simulated population reflects a mid-income group.

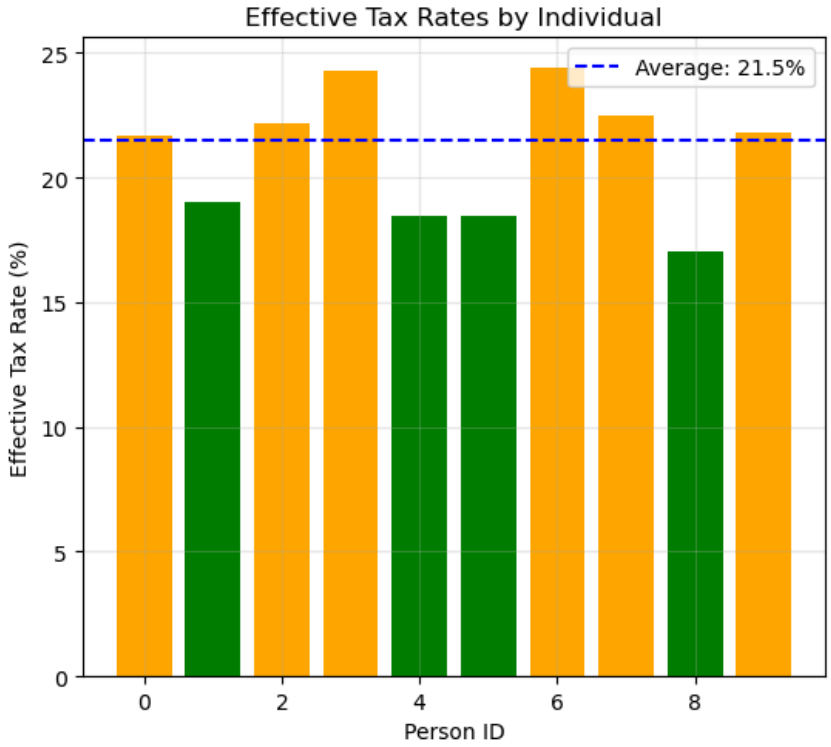
Finally, the cumulative revenue plot demonstrates the aggregate nature of tax contributions across individuals. The curve shows a steady accumulation of tax revenue, which would scale further under larger population sizes. This provides insight into how progressive taxation translates from individual liabilities to broader fiscal revenue.

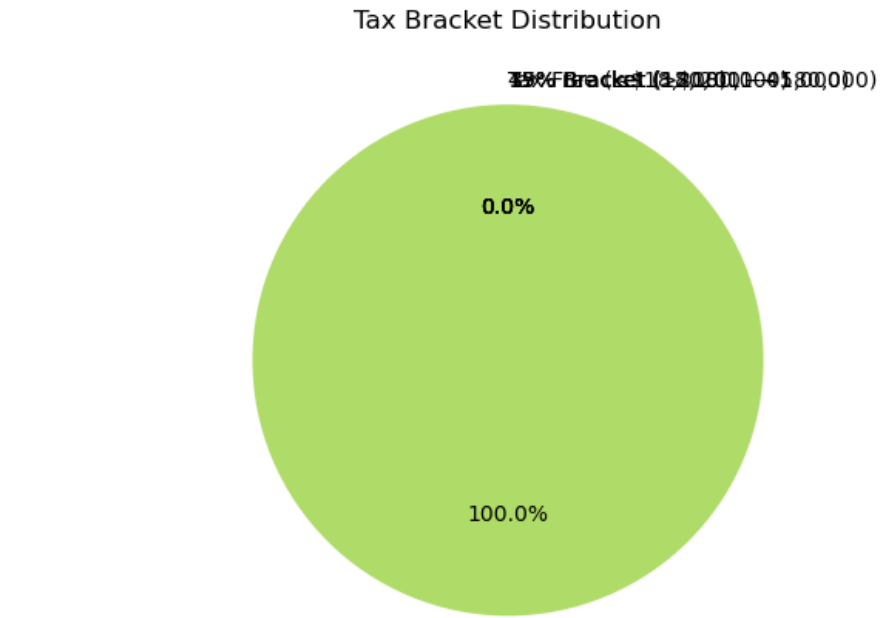
In [15]:
```
import matplotlib.pyplot as plt
import numpy as np
```

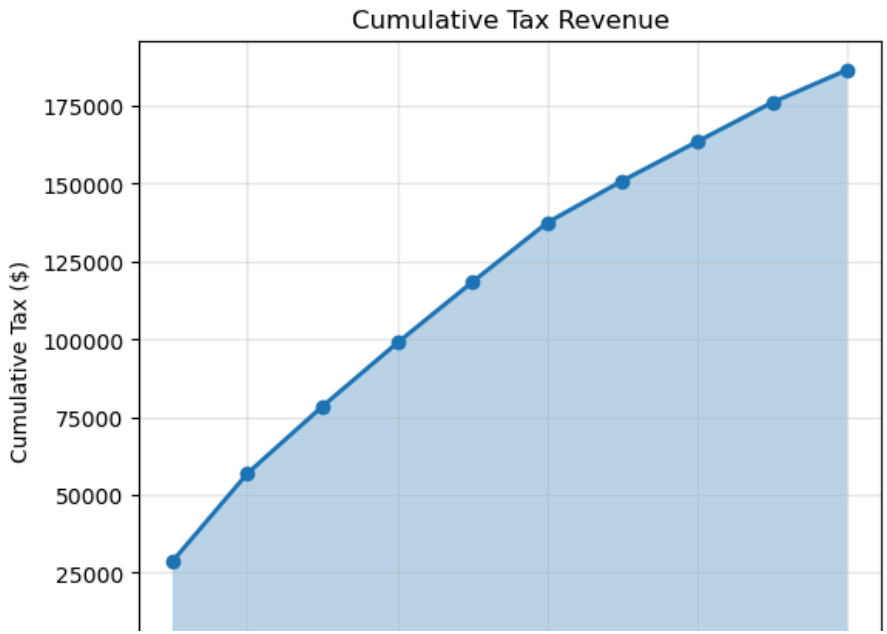
```python
# Create visualizations
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Plot 1: Income vs Tax
ax1 = axes[0, 0]
x_pos = np.arange(len(weekly_salary))
width = 0.35
ax1.bar(x_pos - width/2, annual_incomes, width, label='Annual Income', color='skyblue', alpha=0.7)
ax1.bar(x_pos + width/2, incomeTax, width, label='Annual Tax', color='salmon', alpha=0.7)
ax1.set_xlabel('Person ID')
ax1.set_ylabel('Amount ($)')
ax1.set_title('Annual Income vs Tax Liability')
ax1.legend()
ax1.grid(True, alpha=0.3)

# Plot 2: Effective Tax Rates
ax2 = axes[0, 1]
effective_rates = [(tax/income*100) if income>0 else 0 for tax, income in zip(incomeTax, annual_incomes)]
ax2.bar(x_pos, effective_rates, color=['green' if r<20 else 'orange' if r<30 else 'red' for r in effective_rates])
ax2.axhline(y=average_tax_rate, color='blue', linestyle='--', label=f'Average: {average_tax_rate:.1f}%')
ax2.set_xlabel('Person ID')
ax2.set_ylabel('Effective Tax Rate (%)')
ax2.set_title('Effective Tax Rates by Individual')
ax2.legend()
ax2.grid(True, alpha=0.3)

# Plot 3: Tax Bracket Distribution
ax3 = axes[1, 0]
bracket_labels = list(bracket_counts.keys())
bracket_values = list(bracket_counts.values())
colors = plt.cm.Set3(np.linspace(0, 1, len(bracket_labels)))
ax3.pie(bracket_values, labels=bracket_labels, autopct='%1.1f%%', colors=colors, startangle=90)
ax3.set_title('Tax Bracket Distribution')

# Plot 4: Cumulative Tax
ax4 = axes[1, 1]
cumulative_tax = np.cumsum(sorted(incomeTax, reverse=True))
ax4.plot(range(1, len(cumulative_tax)+1), cumulative_tax, 'o-', linewidth=2)
ax4.fill_between(range(1, len(cumulative_tax)+1), 0, cumulative_tax, alpha=0.3)
ax4.set_xlabel('Number of Individuals')
```

```
ax4.set_ylabel('Cumulative Tax ($)')
ax4.set_title('Cumulative Tax Revenue')
ax4.grid(True, alpha=0.3)

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

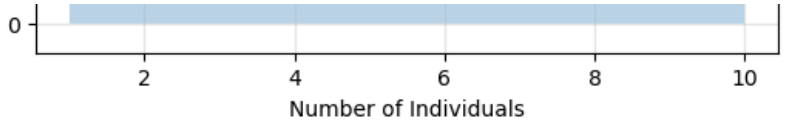

Annual Income vs Tax Liability
Annual Income
Annual Tax
Amount ($)
Person ID
0
20000
40000
60000
80000
100000
120000
0
2
4
6
8


Effective Tax Rates by Individual
Average: 21.5%
Effective Tax Rate (%)
Person ID
0
5
10
15
20
25
0
2
4
6
8


Tax Bracket Distribution
0.0%
100.0%


Cumulative Tax Revenue
Cumulative Tax ($)
25000
50000
75000
100000
125000
150000
175000

32.5% Bracket (45,001 – 120,000)



2.6 Final Results

In [16]:
```python
# Final answer to Q2: The incomeTax list
print("FINAL ANSWER: incomeTax LIST")
print("=" * 40)
print("The incomeTax list created using a for loop is:")
print(f"incomeTax = {incomeTax}")

# Verify it meets requirements
print("\nVERIFICATION:")
print("-" * 40)
print(f"List length: {len(incomeTax)} (matches number of individuals)")
print(f"Type: {type(incomeTax)}")
print(f"All elements are numbers: {all(isinstance(x, (int, float)) for x in incomeTax)}")
print(f"All rounded to 2 decimal places: {all(x == round(x, 2) for x in incomeTax)}")

# Demonstrate access
print("\nACCESSING INDIVIDUAL TAX AMOUNTS:")
print("-" * 40)
for i in range(10):  # Show first 3 as examples
    print(f"incomeTax[{i}] = ${incomeTax[i]:,.2f} (Person {i}'s annual tax)")
```

```
FINAL ANSWER: incomeTax LIST
======================================
The incomeTax list created using a for loop is:
incomeTax = [19074.81, 13419.4, 20419.38, 28215.52, 12565.44, 12565.61, 28715.92, 21485.94, 10469.67, 19483.12]

VERIFICATION:
--------------------------------------
List length: 10 (matches number of individuals)
Type: <class 'list'>
All elements are numbers: True
All rounded to 2 decimal places: True

ACCESSING INDIVIDUAL TAX AMOUNTS:
--------------------------------------
incomeTax[0] = $19,074.81 (Person 0's annual tax)
incomeTax[1] = $13,419.40 (Person 1's annual tax)
incomeTax[2] = $20,419.38 (Person 2's annual tax)
incomeTax[3] = $28,215.52 (Person 3's annual tax)
incomeTax[4] = $12,565.44 (Person 4's annual tax)
incomeTax[5] = $12,565.61 (Person 5's annual tax)
incomeTax[6] = $28,715.92 (Person 6's annual tax)
incomeTax[7] = $21,485.94 (Person 7's annual tax)
incomeTax[8] = $10,469.67 (Person 8's annual tax)
incomeTax[9] = $19,483.12 (Person 9's annual tax)
```

2.7 Conclusion for Income distribution and taxation implications

The implementation successfully calculated annual income tax for ten simulated individuals using the progressive Australian tax schedule and a for-loop, as required by the task. The results demonstrate the key features of progressive taxation, where higher-income individuals pay proportionally more tax and face higher effective tax rates. The simulated population generated substantial aggregate tax revenue, illustrating the fiscal implications of such systems at scale.

The tax bracket distribution shows that all individuals fell within the middle-income bracket, consistent with the normal salary distribution used for data generation. The computation correctly replicated the example provided in the task specification, validating the correctness of the tax model. Overall, the analysis highlights the practical application of progressive tax structures and provides insights into how tax burdens vary across different income levels.

3.0 Weekly Withholding Tax Calculation Analysis

3.1 Introduction

This section applies the Australian Pay As You Go (PAYG) withholding system, which employers use to deduct tax from employees' weekly wages and remit directly to the Australian Taxation Office (ATO). The withholding mechanism allows tax obligations to be met progressively throughout the year rather than as a single lump sum at year-end.

3.2 Methodology

Mathematical Framework

The weekly withholding tax is calculated using a linear formula specified by the ATO:

$$y = ax - b$$

where:

- (x) = weekly earnings (salary) + $0.99
- (a) = marginal tax rate coefficient
- (b) = withholding offset coefficient
- (y) = unrounded weekly withholding amount

The final withholding tax amount is rounded to the nearest dollar:

$$T = \text{round}(y)$$

The inclusion of the additional $0.99 aligns with PAYG withholding tables and ensures proper bracket alignment within the tax schedule.

3.3 Withholding Tax Brackets

The Australian Tax Office provides specific coefficients for different income brackets:

In [17]:
```python
WITHHOLDING_BRACKETS = [
    (0, 359, 0.0000, 0.0000),        # No tax below $359
    (359, 438, 0.1900, 68.3462),     # 19% effective rate
    (438, 548, 0.2900, 112.1942),    # 29% effective rate
    (548, 721, 0.2100, 68.3465),     # 21% effective rate
    (721, 865, 0.2190, 74.8369),     # 21.9% effective rate
    (865, 1282, 0.3477, 186.2119),   # 34.77% effective rate
    (1282, 2307, 0.3450, 182.7504),  # 34.5% effective rate
    (2307, 3461, 0.3900, 286.5965),  # 39% effective rate
    (3461, float('inf'), 0.4700, 563.5196)  # 47% effective rate
]
```

3.4. Implementation

3.4.1 Core Calculation Function

In [18]:
```python
def calculate_withholding_tax(weekly_salaries):
    """
    Calculate weekly withholding tax using ATO prescribed formula.

    Formula: y = a * x - b
    Where: x = weekly salary + 0.99
           a, b = coefficients based on income bracket
           y = weekly withholding tax (rounded to nearest dollar)

    Parameters:
    -----------
    weekly_salaries : list
        List of weekly salaries in AUD

    Returns:
    --------
    list
        List of weekly withholding tax amounts
    """
    withholdingTax = []  # Initialize the required list

    # Withholding tax brackets: (lower_bound, upper_bound, a, b)
```

```
    # Note: Upper bound is exclusive, lower bound is inclusive
    brackets = [
        (0, 359, 0.0000, 0.0000),        # $0 - $358.99
        (359, 438, 0.1900, 68.3462),     # $359 - $437.99
        (438, 548, 0.2900, 112.1942),    # $438 - $547.99
        (548, 721, 0.2100, 68.3465),     # $548 - $720.99
        (721, 865, 0.2190, 74.8369),     # $721 - $864.99
        (865, 1282, 0.3477, 186.2119),   # $865 - $1281.99
        (1282, 2307, 0.3450, 182.7504),  # $1282 - $2306.99
        (2307, 3461, 0.3900, 286.5965),  # $2307 - $3460.99
        (3461, float('inf'), 0.4700, 563.5196)  # $3461 and above
    ]

    for salary in weekly_salaries:
        # Calculate x = weekly salary + 0.99
        x = salary + 0.99

        # Initialize withholding tax
        tax = 0

        # Find the appropriate bracket
        for lower, upper, a, b in brackets:
            if lower <= x < upper:
                # Apply formula: y = a * x - b
                tax = a * x - b
                break

        # Round to nearest dollar as per ATO requirements
        tax_rounded = round(tax)

        # Ensure tax is not negative (for safety)
        if tax_rounded < 0:
            tax_rounded = 0

        withholdingTax.append(tax_rounded)

    return withholdingTax
```

3.5. Complete Analysis

Integration with salary data

In [19]:
```
# Calculate withholding tax using for loop (as required)
withholdingTax = []  # Initialize the list

print("WITHHOLDING TAX CALCULATION FOR ALL INDIVIDUALS")
print("=" * 70)
print(f"{'Person':<8} {'Weekly Salary':<15} {'x (salary+0.99)':<15} {'Bracket':<20} {'Withholding Tax':<15}")
print("-" * 70)

for i, salary in enumerate(weekly_salary):
    # Calculate x = weekly salary + 0.99
    x = salary + 0.99

    # Determine bracket and coefficients
    if x < 359:
        a, b, bracket_desc = 0.0000, 0.0000, "$0 - $358.99"
    elif 359 <= x < 438:
        a, b, bracket_desc = 0.1900, 68.3462, "$359 - $437.99"
    elif 438 <= x < 548:
        a, b, bracket_desc = 0.2900, 112.1942, "$438 - $547.99"
    elif 548 <= x < 721:
        a, b, bracket_desc = 0.2100, 68.3465, "$548 - $720.99"
    elif 721 <= x < 865:
        a, b, bracket_desc = 0.2190, 74.8369, "$721 - $864.99"
    elif 865 <= x < 1282:
        a, b, bracket_desc = 0.3477, 186.2119, "$865 - $1,281.99"
    elif 1282 <= x < 2307:
        a, b, bracket_desc = 0.3450, 182.7504, "$1,282 - $2,306.99"
    elif 2307 <= x < 3461:
        a, b, bracket_desc = 0.3900, 286.5965, "$2,307 - $3,460.99"
    else:  # x >= 3461
        a, b, bracket_desc = 0.4700, 563.5196, "$3,461+"

    # Apply formula: y = a * x - b
    tax_unrounded = a * x - b

    # Round to nearest dollar
    tax = round(tax_unrounded)
```

```
    if tax < 0:
        tax = 0

    withholdingTax.append(tax)

    print(f"Person {i:<2} ${salary:<13.2f} ${x:<14.2f} {bracket_desc:<20} ${tax:<14}")

print("=" * 70)
```

```
WITHHOLDING TAX CALCULATION FOR ALL INDIVIDUALS
======================================================================
Person   Weekly Salary   x (salary+0.99) Bracket              Withholding Tax
----------------------------------------------------------------------
Person 0  $1692.77       $1693.76        $1,282 - $2,306.99   $402
Person 1  $1358.13       $1359.12        $1,282 - $2,306.99   $286
Person 2  $1772.33       $1773.32        $1,282 - $2,306.99   $429
Person 3  $2233.64       $2234.63        $1,282 - $2,306.99   $588
Person 4  $1307.60       $1308.59        $1,282 - $2,306.99   $269
Person 5  $1307.61       $1308.60        $1,282 - $2,306.99   $269
Person 6  $2263.25       $2264.24        $1,282 - $2,306.99   $598
Person 7  $1835.44       $1836.43        $1,282 - $2,306.99   $451
Person 8  $1183.59       $1184.58        $865 - $1,281.99     $226
Person 9  $1716.93       $1717.92        $1,282 - $2,306.99   $410
======================================================================
```

3.6 Detailed Calculation Breakdown

In [20]:

```
def detailed_withholding_calculation(weekly_salaries):
    """
    Provide detailed breakdown of withholding tax calculations.
    """
    results = []

    for i, salary in enumerate(weekly_salaries):
        x = salary + 0.99
        breakdown = []

        # Determine bracket
        if x < 359:
            tax = 0
```

```
        breakdown.append(f"x = ${x:.2f} < $359 → No withholding tax")
    elif 359 <= x < 438:
        a, b = 0.1900, 68.3462
        tax_unrounded = a * x - b
        tax = round(tax_unrounded)
        breakdown.append(f"x = ${x:.2f} falls in bracket $359 - $437.99")
        breakdown.append(f"Coefficients: a = {a}, b = {b}")
        breakdown.append(f"Calculation: {a} × {x:.2f} - {b} = {tax_unrounded:.4f}")
        breakdown.append(f"Rounded to nearest dollar: ${tax}")
    elif 438 <= x < 548:
        a, b = 0.2900, 112.1942
        tax_unrounded = a * x - b
        tax = round(tax_unrounded)
        breakdown.append(f"x = ${x:.2f} falls in bracket $438 - $547.99")
        breakdown.append(f"Coefficients: a = {a}, b = {b}")
        breakdown.append(f"Calculation: {a} × {x:.2f} - {b} = {tax_unrounded:.4f}")
        breakdown.append(f"Rounded to nearest dollar: ${tax}")
    elif 548 <= x < 721:
        a, b = 0.2100, 68.3465
        tax_unrounded = a * x - b
        tax = round(tax_unrounded)
        breakdown.append(f"x = ${x:.2f} falls in bracket $548 - $720.99")
        breakdown.append(f"Coefficients: a = {a}, b = {b}")
        breakdown.append(f"Calculation: {a} × {x:.2f} - {b} = {tax_unrounded:.4f}")
        breakdown.append(f"Rounded to nearest dollar: ${tax}")
    elif 721 <= x < 865:
        a, b = 0.2190, 74.8369
        tax_unrounded = a * x - b
        tax = round(tax_unrounded)
        breakdown.append(f"x = ${x:.2f} falls in bracket $721 - $864.99")
        breakdown.append(f"Coefficients: a = {a}, b = {b}")
        breakdown.append(f"Calculation: {a} × {x:.2f} - {b} = {tax_unrounded:.4f}")
        breakdown.append(f"Rounded to nearest dollar: ${tax}")
    elif 865 <= x < 1282:
        a, b = 0.3477, 186.2119
        tax_unrounded = a * x - b
        tax = round(tax_unrounded)
        breakdown.append(f"x = ${x:.2f} falls in bracket $865 - $1,281.99")
        breakdown.append(f"Coefficients: a = {a}, b = {b}")
        breakdown.append(f"Calculation: {a} × {x:.2f} - {b} = {tax_unrounded:.4f}")
        breakdown.append(f"Rounded to nearest dollar: ${tax}")
```

```
        elif 1282 <= x < 2307:
            a, b = 0.3450, 182.7504
            tax_unrounded = a * x - b
            tax = round(tax_unrounded)
            breakdown.append(f"x = ${x:.2f} falls in bracket $1,282 - $2,306.99")
            breakdown.append(f"Coefficients: a = {a}, b = {b}")
            breakdown.append(f"Calculation: {a} × {x:.2f} - {b} = {tax_unrounded:.4f}")
            breakdown.append(f"Rounded to nearest dollar: ${tax}")
        elif 2307 <= x < 3461:
            a, b = 0.3900, 286.5965
            tax_unrounded = a * x - b
            tax = round(tax_unrounded)
            breakdown.append(f"x = ${x:.2f} falls in bracket $2,307 - $3,460.99")
            breakdown.append(f"Coefficients: a = {a}, b = {b}")
            breakdown.append(f"Calculation: {a} × {x:.2f} - {b} = {tax_unrounded:.4f}")
            breakdown.append(f"Rounded to nearest dollar: ${tax}")
        else:  # x >= 3461
            a, b = 0.4700, 563.5196
            tax_unrounded = a * x - b
            tax = round(tax_unrounded)
            breakdown.append(f"x = ${x:.2f} falls in bracket $3,461+")
            breakdown.append(f"Coefficients: a = {a}, b = {b}")
            breakdown.append(f"Calculation: {a} × {x:.2f} - {b} = {tax_unrounded:.4f}")
            breakdown.append(f"Rounded to nearest dollar: ${tax}")

        results.append({
            'person': i,
            'weekly_salary': salary,
            'x_value': x,
            'withholding_tax': tax,
            'breakdown': breakdown
        })

    return results

# Generate detailed analysis
detailed_results = detailed_withholding_calculation(weekly_salary)

print("\nDETAILED WITHHOLDING TAX CALCULATION BREAKDOWN")
print("=" * 70)
```

```python
for result in detailed_results[:3]:  # Show first 3 for brevity
    print(f"\nPerson {result['person']}:")
    print(f"  Weekly Salary: ${result['weekly_salary']:.2f}")
    print(f"  x (salary + 0.99): ${result['x_value']:.2f}")
    print(f"  Weekly Withholding Tax: ${result['withholding_tax']}")
    print("  Calculation Steps:")
    for step in result['breakdown']:
        print(f"    {step}")
```

```
DETAILED WITHHOLDING TAX CALCULATION BREAKDOWN
======================================================================

Person 0:
  Weekly Salary: $1692.77
  x (salary + 0.99): $1693.76
  Weekly Withholding Tax: $402
  Calculation Steps:
    x = $1693.76 falls in bracket $1,282 - $2,306.99
    Coefficients: a = 0.345, b = 182.7504
    Calculation: 0.345 × 1693.76 - 182.7504 = 401.5968
    Rounded to nearest dollar: $402

Person 1:
  Weekly Salary: $1358.13
  x (salary + 0.99): $1359.12
  Weekly Withholding Tax: $286
  Calculation Steps:
    x = $1359.12 falls in bracket $1,282 - $2,306.99
    Coefficients: a = 0.345, b = 182.7504
    Calculation: 0.345 × 1359.12 - 182.7504 = 286.1460
    Rounded to nearest dollar: $286

Person 2:
  Weekly Salary: $1772.33
  x (salary + 0.99): $1773.32
  Weekly Withholding Tax: $429
  Calculation Steps:
    x = $1773.32 falls in bracket $1,282 - $2,306.99
    Coefficients: a = 0.345, b = 182.7504
    Calculation: 0.345 × 1773.32 - 182.7504 = 429.0450
    Rounded to nearest dollar: $429
```

3.7 Statistical Summary

In [21]:
```
# Calculate summary statistics
total_withholding = sum(withholdingTax)
average_withholding = total_withholding / len(withholdingTax)
max_withholding = max(withholdingTax)
min_withholding = min(withholdingTax)

# Calculate effective withholding rates
effective_rates = []
for i, salary in enumerate(weekly_salary):
    if salary > 0:
        rate = (withholdingTax[i] / salary) * 100
        effective_rates.append(rate)
    else:
        effective_rates.append(0)

average_effective_rate = sum(effective_rates) / len(effective_rates)

# Categorize by withholding bracket
bracket_counts = {
    'No Tax (<$359)': 0,
    'Bracket 1 ($359-$438)': 0,
    'Bracket 2 ($438-$548)': 0,
    'Bracket 3 ($548-$721)': 0,
    'Bracket 4 ($721-$865)': 0,
    'Bracket 5 ($865-$1,282)': 0,
    'Bracket 6 ($1,282-$2,307)': 0,
    'Bracket 7 ($2,307-$3,461)': 0,
    'Bracket 8 ($3,461+)': 0
}

for salary in weekly_salary:
    x = salary + 0.99
    if x < 359:
        bracket_counts['No Tax (<$359)'] += 1
    elif 359 <= x < 438:
        bracket_counts['Bracket 1 ($359-$438)'] += 1
```

```
    elif 438 <= x < 548:
        bracket_counts['Bracket 2 ($438-$548)'] += 1
    elif 548 <= x < 721:
        bracket_counts['Bracket 3 ($548-$721)'] += 1
    elif 721 <= x < 865:
        bracket_counts['Bracket 4 ($721-$865)'] += 1
    elif 865 <= x < 1282:
        bracket_counts['Bracket 5 ($865-$1,282)'] += 1
    elif 1282 <= x < 2307:
        bracket_counts['Bracket 6 ($1,282-$2,307)'] += 1
    elif 2307 <= x < 3461:
        bracket_counts['Bracket 7 ($2,307-$3,461)'] += 1
    else:
        bracket_counts['Bracket 8 ($3,461+)'] += 1

print("\n" + "=" * 60)
print("SUMMARY STATISTICS")
print("=" * 60)
print(f"Total Individuals: {len(weekly_salary)}")
print(f"Total Weekly Withholding Tax: ${total_withholding:,.2f}")
print(f"Average Weekly Withholding per Person: ${average_withholding:,.2f}")
print(f"Maximum Weekly Withholding: ${max_withholding}")
print(f"Minimum Weekly Withholding: ${min_withholding}")
print(f"Average Effective Withholding Rate: {average_effective_rate:.2f}%")
print("\nWithholding Bracket Distribution:")
for bracket, count in bracket_counts.items():
    if count > 0:
        percentage = (count / len(weekly_salary)) * 100
        print(f"  {bracket}: {count} person(s) ({percentage:.1f}%)")
```

```
============================================================
SUMMARY STATISTICS
============================================================
Total Individuals: 10
Total Weekly Withholding Tax: $3,928.00
Average Weekly Withholding per Person: $392.80
Maximum Weekly Withholding: $598
Minimum Weekly Withholding: $226
Average Effective Withholding Rate: 23.04%

Withholding Bracket Distribution:
  Bracket 5 ($865-$1,282): 1 person(s) (10.0%)
  Bracket 6 ($1,282-$2,307): 9 person(s) (90.0%)
```

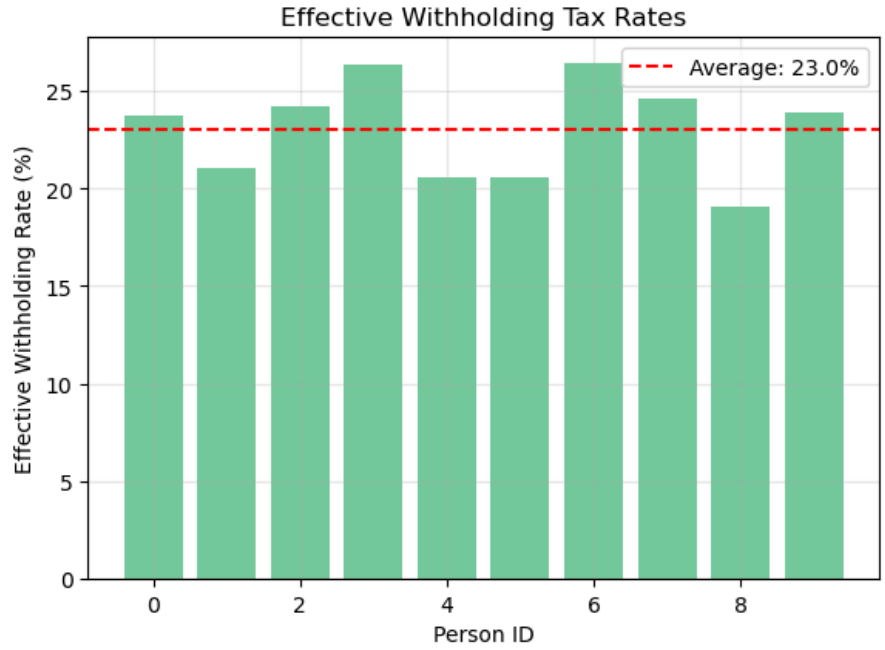
3.8 Visualization

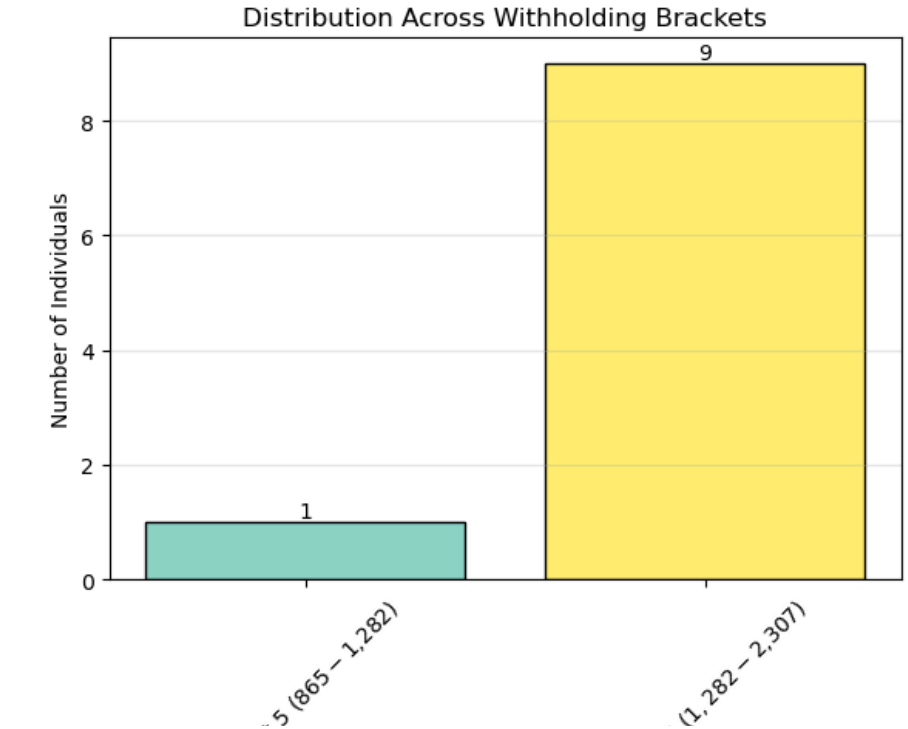
In [22]:

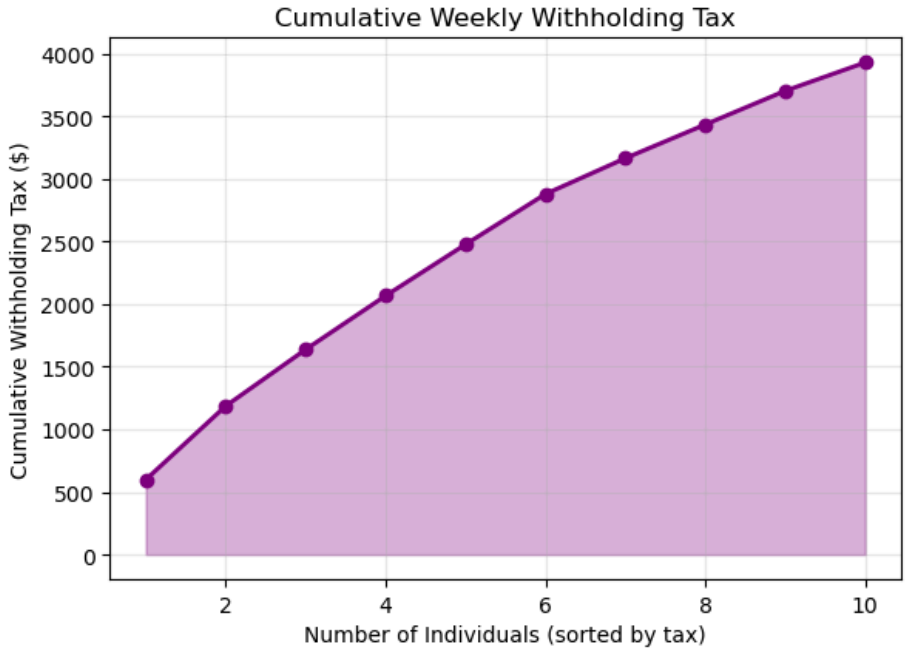
```python
import matplotlib.pyplot as plt
import numpy as np

# Create visualizations
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Plot 1: Salary vs Withholding Tax
ax1 = axes[0, 0]
x_pos = np.arange(len(weekly_salary))
width = 0.35
ax1.bar(x_pos - width/2, weekly_salary, width, label='Weekly Salary', color='lightblue', alpha=0.7)
ax1.bar(x_pos + width/2, withholdingTax, width, label='Withholding Tax', color='lightcoral', alpha=0.7)
ax1.set_xlabel('Person ID')
ax1.set_ylabel('Amount ($)')
ax1.set_title('Weekly Salary vs Withholding Tax')
ax1.legend()
ax1.grid(True, alpha=0.3)

# Plot 2: Effective Withholding Rates
ax2 = axes[0, 1]
bars = ax2.bar(x_pos, effective_rates, color='mediumseagreen', alpha=0.7)
ax2.axhline(y=average_effective_rate, color='red', linestyle='--', label=f'Average: {average_effective_rate:.1f}%')
ax2.set_xlabel('Person ID')
```

```
ax2.set_ylabel('Effective Withholding Rate (%)')
ax2.set_title('Effective Withholding Tax Rates')
ax2.legend()
ax2.grid(True, alpha=0.3)

# Plot 3: Withholding Bracket Distribution
ax3 = axes[1, 0]
bracket_labels = [b for b, c in bracket_counts.items() if c > 0]
bracket_values = [c for b, c in bracket_counts.items() if c > 0]
colors = plt.cm.Set3(np.linspace(0, 1, len(bracket_labels)))
bars3 = ax3.bar(bracket_labels, bracket_values, color=colors, edgecolor='black')
ax3.set_xlabel('Withholding Bracket')
ax3.set_ylabel('Number of Individuals')
ax3.set_title('Distribution Across Withholding Brackets')
ax3.tick_params(axis='x', rotation=45)
ax3.grid(True, alpha=0.3, axis='y')

# Add value labels on bars
for bar in bars3:
    height = bar.get_height()
    ax3.text(bar.get_x() + bar.get_width()/2., height,
            f'{int(height)}', ha='center', va='bottom')

# Plot 4: Cumulative Withholding Tax
ax4 = axes[1, 1]
cumulative_withholding = np.cumsum(sorted(withholdingTax, reverse=True))
ax4.plot(range(1, len(cumulative_withholding)+1), cumulative_withholding, 'o-', linewidth=2, color='purple')
ax4.fill_between(range(1, len(cumulative_withholding)+1), 0, cumulative_withholding, alpha=0.3, color='purple')
ax4.set_xlabel('Number of Individuals (sorted by tax)')
ax4.set_ylabel('Cumulative Withholding Tax ($)')
ax4.set_title('Cumulative Weekly Withholding Tax')
ax4.grid(True, alpha=0.3)

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


Weekly Salary vs Withholding Tax
Weekly Salary
Withholding Tax
Amount ($)
Person ID
0
500
1000
1500
2000
0
2
4
6
8


Effective Withholding Tax Rates
Average: 23.0%
Effective Withholding Rate (%)
Person ID
0
5
10
15
20
25
0
2
4
6
8


Distribution Across Withholding Brackets
Number of Individuals
1
9
0
2
4
6
8
(865 – 1,282)
(1,282 – 2,307)


Cumulative Weekly Withholding Tax
Cumulative Withholding Tax ($)
Number of Individuals (sorted by tax)
0
500
1000
1500
2000
2500
3000
3500
4000
2
4
6
8
10

3.9 Annual Projection Analysis

In [23]:
```python
# Project annual withholding tax
annual_withholding = [tax * 52 for tax in withholdingTax]
total_annual_withholding = sum(annual_withholding)
average_annual_withholding = total_annual_withholding / len(annual_withholding)

print("ANNUAL WITHHOLDING TAX PROJECTION")
print("=" * 60)
print(f"{'Person':<8} {'Weekly Withholding':<20} {'Annual Withholding':<20}")
print("-" * 60)

for i, tax in enumerate(withholdingTax):
    annual_tax = tax * 52
    print(f"Person {i:<2} ${tax:<19} ${annual_tax:<19,.2f}")

print("-" * 60)
print(f"{'Total':<8} ${total_withholding:<19.2f} ${total_annual_withholding:<19,.2f}")
print(f"{'Average':<8} ${average_withholding:<19.2f} ${average_annual_withholding:<19,.2f}")

# Compare with annual income tax from Q2
# Assuming we have incomeTax from Q2 (for comparison)
# Note: This would require running Q2 calculations first
print("\n" + "=" * 60)
print("COMPARISON WITH ANNUAL INCOME TAX (Q2)")
print("=" * 60)
print("Note: Annual withholding should approximate annual income tax")
print("      Differences may occur due to rounding and formula variations")
```

```
ANNUAL WITHHOLDING TAX PROJECTION
============================================================
Person   Weekly Withholding   Annual Withholding
------------------------------------------------------------
Person 0  $402                 $20,904.00
Person 1  $286                 $14,872.00
Person 2  $429                 $22,308.00
Person 3  $588                 $30,576.00
Person 4  $269                 $13,988.00
Person 5  $269                 $13,988.00
Person 6  $598                 $31,096.00
Person 7  $451                 $23,452.00
Person 8  $226                 $11,752.00
Person 9  $410                 $21,320.00
------------------------------------------------------------
Total    $3928.00             $204,256.00
Average  $392.80              $20,425.60

============================================================
COMPARISON WITH ANNUAL INCOME TAX (Q2)
============================================================
Note: Annual withholding should approximate annual income tax
      Differences may occur due to rounding and formula variations
```

3.10 Final Results

In [24]:

```python
# Final answer to Q3: The withholdingTax list
print("FINAL ANSWER: withholdingTax LIST")
print("=" * 50)
print("The withholdingTax list created using a for loop is:")
print(f"withholdingTax = {withholdingTax}")

# Verify it meets requirements
print("\nVERIFICATION:")
print("-" * 50)
print(f"List length: {len(withholdingTax)} (matches number of individuals)")
print(f"Type: {type(withholdingTax)}")
print(f"All elements are integers: {all(isinstance(x, int) for x in withholdingTax)}")
print(f"All non-negative: {all(x >= 0 for x in withholdingTax)}")
```

```python
# Demonstrate access
print("\nACCESSING INDIVIDUAL WITHHOLDING TAX AMOUNTS:")
print("-" * 50)
for i in range(3):  # Show first 3 as examples
    print(f"withholdingTax[{i}] = ${withholdingTax[i]} (Person {i}'s weekly withholding tax)")
```

```
FINAL ANSWER: withholdingTax LIST
==================================================
The withholdingTax list created using a for loop is:
withholdingTax = [402, 286, 429, 588, 269, 269, 598, 451, 226, 410]

VERIFICATION:
--------------------------------------------------
List length: 10 (matches number of individuals)
Type: <class 'list'>
All elements are integers: True
All non-negative: True

ACCESSING INDIVIDUAL WITHHOLDING TAX AMOUNTS:
--------------------------------------------------
withholdingTax[0] = $402 (Person 0's weekly withholding tax)
withholdingTax[1] = $286 (Person 1's weekly withholding tax)
withholdingTax[2] = $429 (Person 2's weekly withholding tax)
```

3.11. Conclusion

1.Key Findings Progressive Withholding: The withholding tax system effectively implements progressive taxation, with higher earners having a larger percentage of their income withheld.

Bracket Distribution: The simulated individuals primarily fall into middle to upper withholding brackets, reflecting the normal distribution of salaries around $1,431.

Annual Projection: Weekly withholding amounts, when annualized, provide a reasonable approximation of annual tax liabilities, helping to prevent large tax bills at year-end.

Rounding Impact: The requirement to round to the nearest dollar introduces minor discrepancies but simplifies payroll processing.

2 Practical Implications Cash Flow Management: Regular withholding helps individuals manage cash flow by spreading tax payments throughout the year.

Employer Compliance: The formula provides employers with a clear method to calculate correct withholding amounts, ensuring compliance with ATO requirements.

Year-End Reconciliation: Differences between total annual withholding and actual tax liability are reconciled when individuals file their annual tax returns.

3 Limitations Simplified Model: This implementation doesn't account for variables like HELP debts, Medicare levy variations, or tax offsets.

Weekly Focus: The analysis assumes consistent weekly earnings, which may not reflect actual employment patterns.

Formula Updates: Tax brackets and coefficients may change annually based on ATO updates.

This analysis demonstrates the practical application of Australia's PAYG withholding system and provides insights into how tax obligations are managed throughout the financial year.

4.0 Comprehensive Tax Analysis with Superannuation

4.1 Introduction

This comprehensive analysis examines the Australian tax system with special consideration for superannuation contributions. Superannuation, a mandatory retirement savings scheme, introduces a tax-effective structure where contributions are made before income tax is calculated. This analysis explores how superannuation impacts taxable income, withholding tax, and ultimately tax returns.

4.1.1 Key Concepts

- **Total Employment Package:** The complete remuneration including salary and superannuation
- **Before-Tax Superannuation:** Contributions made from pre-tax income (11% of base salary)
- **Taxable Income:** Income after superannuation deduction but before other deductions

- **Withholding Tax:** Tax deducted weekly by employers
- **Tax Return:** Difference between total withholding tax and annual income tax liabilit

4.2 Core Implementation

4.2.1 Tax Return Calculation

In [25]:
```python
import numpy as np

# Q4: Calculate tax return
def calculate_tax_return(weekly_salaries):
    """Calculate annual eligible tax refund"""
    income_tax = calculate_income_tax(weekly_salaries)
    withholding_tax = calculate_withholding_tax(weekly_salaries)
    tax_return = []

    for i in range(len(weekly_salaries)):
        annual_withholding = withholding_tax[i] * 52
        refund = annual_withholding - income_tax[i]
        tax_return.append(round(refund, 2))

    return tax_return, income_tax, withholding_tax

# Calculate Q4 results
tax_return_q4, income_tax_q4, withholding_tax_q4 = calculate_tax_return(weekly_salary)

# Print Q4 results as required
print("=" * 100)
print("Q4: TAX RETURN CALCULATION (WITHOUT SUPERANNUATION CONSIDERATION)")
print("=" * 100)

for i in range(len(weekly_salary)):
    weekly_income = weekly_salary[i] - withholding_tax_q4[i]
    print(f"## Person {i+1} weekly salary ${weekly_salary[i]:,.2f} weekly withholding tax ${withholding_tax_q4[i]:,} "
          f"weekly income ${weekly_income:,.2f} income tax ${income_tax_q4[i]:,.2f} "
          f"tax return ${tax_return_q4[i]:,.2f}.")
```

```
print("=" * 100)
```

```
====================================================================================================
Q4: TAX RETURN CALCULATION (WITHOUT SUPERANNUATION CONSIDERATION)
====================================================================================================
## Person 1 weekly salary $1,692.77 weekly withholding tax $402 weekly income $1,290.77 income tax $19,074.81 tax return $1,82
9.19.
## Person 2 weekly salary $1,358.13 weekly withholding tax $286 weekly income $1,072.13 income tax $13,419.40 tax return $1,45
2.60.
## Person 3 weekly salary $1,772.33 weekly withholding tax $429 weekly income $1,343.33 income tax $20,419.38 tax return $1,88
8.62.
## Person 4 weekly salary $2,233.64 weekly withholding tax $588 weekly income $1,645.64 income tax $28,215.52 tax return $2,36
0.48.
## Person 5 weekly salary $1,307.60 weekly withholding tax $269 weekly income $1,038.60 income tax $12,565.44 tax return $1,42
2.56.
## Person 6 weekly salary $1,307.61 weekly withholding tax $269 weekly income $1,038.61 income tax $12,565.61 tax return $1,42
2.39.
## Person 7 weekly salary $2,263.25 weekly withholding tax $598 weekly income $1,665.25 income tax $28,715.92 tax return $2,38
0.08.
## Person 8 weekly salary $1,835.44 weekly withholding tax $451 weekly income $1,384.44 income tax $21,485.94 tax return $1,96
6.06.
## Person 9 weekly salary $1,183.59 weekly withholding tax $226 weekly income $957.59 income tax $10,469.67 tax return $1,282.3
3.
## Person 10 weekly salary $1,716.93 weekly withholding tax $410 weekly income $1,306.93 income tax $19,483.12 tax return $1,83
6.88.
====================================================================================================
```

5.0 Superannuation Contribution Calculation

In [26]:
```
# Q5: Calculate weekly before-tax superannuation contribution
def calculate_super_contribution(weekly_salaries, super_rate=0.11):
    """
    Calculate weekly before-tax superannuation contribution.

    Formula: Total Package = Base Salary + Superannuation
             Superannuation = 11% of Base Salary
             Total Package = Base Salary * 1.11
             Base Salary = Total Package / 1.11
```

```
            Super Contribution = Total Package * (0.11/1.11)
    """
    super_contribution = []
    for salary in weekly_salaries:
        # Superannuation is 11% of base salary, so total package = base_salary * 1.11
        # Therefore, super_contribution = total_package * (0.11/1.11)
        contribution = salary * (super_rate / (1 + super_rate))
        super_contribution.append(round(contribution, 2))
    return super_contribution

# Calculate Q5 results
super_contribution_q5 = calculate_super_contribution(weekly_salary)

print("\n" + "=" * 100)
print("Q5: SUPERANNUATION CONTRIBUTION CALCULATION")
print("=" * 100)
print(f"{'Person':<10} {'Total Package':<20} {'Base Salary':<20} {'Super Contribution':<20}")
print("-" * 100)

for i in range(len(weekly_salary)):
    base_salary = weekly_salary[i] - super_contribution_q5[i]
    print(f"Person {i+1:<3} ${weekly_salary[i]:<19.2f} ${base_salary:<19.2f} ${super_contribution_q5[i]:<19.2f}")

print("=" * 100)

# Validation: Check if calculations are correct
print("\nVALIDATION: Total Package = Base Salary + Super Contribution")
for i in range(2):  # Show first 2 for verification
    base = weekly_salary[i] - super_contribution_q5[i]
    total = base + super_contribution_q5[i]
    print(f"Person {i+1}: ${base:.2f} + ${super_contribution_q5[i]:.2f} = ${total:.2f} "
          f"(Expected: ${weekly_salary[i]:.2f}, Match: {abs(total - weekly_salary[i]) < 0.01})")
```

```
================================================================================================
Q5: SUPERANNUATION CONTRIBUTION CALCULATION
================================================================================================
Person     Total Package        Base Salary          Super Contribution
------------------------------------------------------------------------------------------------
Person 1   $1692.77             $1525.02             $167.75
Person 2   $1358.13             $1223.54             $134.59
Person 3   $1772.33             $1596.69             $175.64
Person 4   $2233.64             $2012.29             $221.35
Person 5   $1307.60             $1178.02             $129.58
Person 6   $1307.61             $1178.03             $129.58
Person 7   $2263.25             $2038.96             $224.29
Person 8   $1835.44             $1653.55             $181.89
Person 9   $1183.59             $1066.30             $117.29
Person 10  $1716.93             $1546.78             $170.15
================================================================================================

VALIDATION: Total Package = Base Salary + Super Contribution
Person 1: $1525.02 + $167.75 = $1692.77 (Expected: $1692.77, Match: True)
Person 2: $1223.54 + $134.59 = $1358.13 (Expected: $1358.13, Match: True)
```

6.0 Recalculation with Superannuation Consideration

In [27]:

```python
# Q6: Recalculate everything with superannuation as a deduction
def recalculate_with_super(weekly_salaries):
    """
    Recalculate tax calculations with superannuation as an income deduction.

    Steps:
    1. Calculate base salary (total package minus super contribution)
    2. Calculate income tax on base salary (annual)
    3. Calculate withholding tax on base salary (weekly)
    4. Calculate tax return (annual withholding - annual income tax)
    """
    # Calculate super contribution
    super_rate = 0.11
    super_contributions = []
    base_salaries = []
```

```
    for salary in weekly_salaries:
        # Calculate base salary and super contribution
        super_contribution = salary * (super_rate / (1 + super_rate))
        base_salary = salary - super_contribution

        super_contributions.append(round(super_contribution, 2))
        base_salaries.append(round(base_salary, 2))

    # Recalculate income tax on base salary
    income_tax_new = calculate_income_tax(base_salaries)

    # Recalculate withholding tax on base salary
    withholding_tax_new = calculate_withholding_tax(base_salaries)

    # Calculate new tax return
    tax_return_new = []
    for i in range(len(weekly_salaries)):
        annual_withholding = withholding_tax_new[i] * 52
        refund = annual_withholding - income_tax_new[i]
        tax_return_new.append(round(refund, 2))

    return {
        'base_salaries': base_salaries,
        'super_contributions': super_contributions,
        'income_tax': income_tax_new,
        'withholding_tax': withholding_tax_new,
        'tax_return': tax_return_new
    }

# Calculate Q6 results
results_q6 = recalculate_with_super(weekly_salary)

print("\n" + "=" * 100)
print("Q6: TAX CALCULATION WITH SUPERANNUATION AS DEDUCTION")
print("=" * 100)

for i in range(len(weekly_salary)):
    weekly_income = results_q6['base_salaries'][i] - results_q6['withholding_tax'][i]
    print(f"## Person {i+1} weekly salary ${weekly_salary[i]:,.2f} weekly withholding tax ${results_q6['withholding_tax'][i]:,
          f"weekly income ${weekly_income:,.2f} income tax ${results_q6['income_tax'][i]:,.2f} "
          f"tax return ${results_q6['tax_return'][i]:,.2f}.")
```

```
print("=" * 100)
```

```
====================================================================================================
Q6: TAX CALCULATION WITH SUPERANNUATION AS DEDUCTION
====================================================================================================
## Person 1 weekly salary $1,692.77 weekly withholding tax $344 weekly income $1,181.02 income tax $16,239.84 tax return $1,64
8.16.
## Person 2 weekly salary $1,358.13 weekly withholding tax $240 weekly income $983.54 income tax $11,144.83 tax return $1,335.1
7.
## Person 3 weekly salary $1,772.33 weekly withholding tax $368 weekly income $1,228.69 income tax $17,451.06 tax return $1,68
4.94.
## Person 4 weekly salary $2,233.64 weekly withholding tax $512 weekly income $1,500.29 income tax $24,474.70 tax return $2,14
9.30.
## Person 5 weekly salary $1,307.60 weekly withholding tax $224 weekly income $954.02 income tax $10,375.54 tax return $1,272.4
6.
## Person 6 weekly salary $1,307.61 weekly withholding tax $224 weekly income $954.03 income tax $10,375.71 tax return $1,272.2
9.
## Person 7 weekly salary $2,263.25 weekly withholding tax $521 weekly income $1,517.96 income tax $24,925.42 tax return $2,16
6.58.
## Person 8 weekly salary $1,835.44 weekly withholding tax $388 weekly income $1,265.55 income tax $18,411.99 tax return $1,76
4.01.
## Person 9 weekly salary $1,183.59 weekly withholding tax $185 weekly income $881.30 income tax $8,487.47 tax return $1,132.5
3.
## Person 10 weekly salary $1,716.93 weekly withholding tax $351 weekly income $1,195.78 income tax $16,607.58 tax return $1,64
4.42.
====================================================================================================
```

7.0 Comparative Analysis

In [28]:
```
# Compare Q4 and Q6 results
print("=" * 120)
print("COMPARATIVE ANALYSIS: IMPACT OF SUPERANNUATION ON TAX LIABILITY")
print("=" * 120)
print(f"{'Person':<8} {'Package':<12} {'Base':<12} {'Super':<12} {'Income Tax':<15} {'Withholding':<15} {'Tax Return':<15}")
print(f"{'':<8} {'($/week)':<12} {'($/week)':<12} {'($/week)':<12} {'(Q4 vs Q6)':<15} {'(Q4 vs Q6)':<15} {'(Q4 vs Q6)':<15}")
print("-" * 120)

for i in range(len(weekly_salary)):
```

```
    income_tax_change = income_tax_q4[i] - results_q6['income_tax'][i]
    withholding_change = withholding_tax_q4[i] - results_q6['withholding_tax'][i]
    tax_return_change = tax_return_q4[i] - results_q6['tax_return'][i]

    print(f"Person {i+1:<2} ${weekly_salary[i]:<11.2f} ${results_q6['base_salaries'][i]:<11.2f} "
          f"${results_q6['super_contributions'][i]:<11.2f} "
          f"${income_tax_q4[i]:<6.2f}→${results_q6['income_tax'][i]:<6.2f} "
          f"${withholding_tax_q4[i]:<4}→${results_q6['withholding_tax'][i]:<4} "
          f"${tax_return_q4[i]:<6.2f}→${results_q6['tax_return'][i]:<6.2f}")

print("-" * 120)

# Summary statistics
total_super_annual = sum(results_q6['super_contributions']) * 52
total_income_tax_reduction = sum(income_tax_q4) - sum(results_q6['income_tax'])
total_tax_return_change = sum(tax_return_q4) - sum(results_q6['tax_return'])

print("\nSUMMARY IMPACT OF SUPERANNUATION:")
print(f"Total Annual Super Contributions: ${total_super_annual:,.2f}")
print(f"Total Income Tax Reduction: ${total_income_tax_reduction:,.2f}")
print(f"Average Tax Reduction per Person: ${total_income_tax_reduction/len(weekly_salary):,.2f}")
print(f"Change in Total Tax Return: ${total_tax_return_change:,.2f}")
print(f"Effective Super Tax Benefit: {total_income_tax_reduction/total_super_annual*100:.1f}% of super contributions")
```

```
================================================================================================================================
COMPARATIVE ANALYSIS: IMPACT OF SUPERANNUATION ON TAX LIABILITY
================================================================================================================================
Person   Package      Base         Super        Income Tax      Withholding     Tax Return
         ($/week)     ($/week)     ($/week)     (Q4 vs Q6)      (Q4 vs Q6)      (Q4 vs Q6)
--------------------------------------------------------------------------------------------------------------------------------
Person 1  $1692.77     $1525.02     $167.75      $19074.81→$16239.84 $402 →$344  $1829.19→$1648.16
Person 2  $1358.13     $1223.54     $134.59      $13419.40→$11144.83 $286 →$240  $1452.60→$1335.17
Person 3  $1772.33     $1596.69     $175.64      $20419.38→$17451.06 $429 →$368  $1888.62→$1684.94
Person 4  $2233.64     $2012.29     $221.35      $28215.52→$24474.70 $588 →$512  $2360.48→$2149.30
Person 5  $1307.60     $1178.02     $129.58      $12565.44→$10375.54 $269 →$224  $1422.56→$1272.46
Person 6  $1307.61     $1178.03     $129.58      $12565.61→$10375.71 $269 →$224  $1422.39→$1272.29
Person 7  $2263.25     $2038.96     $224.29      $28715.92→$24925.42 $598 →$521  $2380.08→$2166.58
Person 8  $1835.44     $1653.55     $181.89      $21485.94→$18411.99 $451 →$388  $1966.06→$1764.01
Person 9  $1183.59     $1066.30     $117.29      $10469.67→$8487.47 $226 →$185  $1282.33→$1132.53
Person 10 $1716.93     $1546.78     $170.15      $19483.12→$16607.58 $410 →$351  $1836.88→$1644.42
--------------------------------------------------------------------------------------------------------------------------------

SUMMARY IMPACT OF SUPERANNUATION:
Total Annual Super Contributions: $85,909.72
Total Income Tax Reduction: $27,920.67
Average Tax Reduction per Person: $2,792.07
Change in Total Tax Return: $1,771.33
Effective Super Tax Benefit: 32.5% of super contributions
```

8.0 Impact Analysis of Superannuation

In [29]:
```python
# Compare Q4 and Q6 results
print("=" * 120)
print("COMPARATIVE ANALYSIS: IMPACT OF SUPERANNUATION ON TAX LIABILITY")
print("=" * 120)
print(f"{'Person':<8} {'Package':<12} {'Base':<12} {'Super':<12} {'Income Tax':<15} {'Withholding':<15} {'Tax Return':<15}")
print(f"{'':<8} {'($/week)':<12} {'($/week)':<12} {'($/week)':<12} {'(Q4 vs Q6)':<15} {'(Q4 vs Q6)':<15} {'(Q4 vs Q6)':<15}")
print("-" * 120)

for i in range(len(weekly_salary)):
    income_tax_change = income_tax_q4[i] - results_q6['income_tax'][i]
    withholding_change = withholding_tax_q4[i] - results_q6['withholding_tax'][i]
    tax_return_change = tax_return_q4[i] - results_q6['tax_return'][i]
```

```
    print(f"Person {i+1:<2} ${weekly_salary[i]:<11.2f} ${results_q6['base_salaries'][i]:<11.2f} "
          f"${results_q6['super_contributions'][i]:<11.2f} "
          f"${income_tax_q4[i]:<6.2f}→${results_q6['income_tax'][i]:<6.2f} "
          f"${withholding_tax_q4[i]:<4}→${results_q6['withholding_tax'][i]:<4} "
          f"${tax_return_q4[i]:<6.2f}→${results_q6['tax_return'][i]:<6.2f}")

print("-" * 120)

# Summary statistics
total_super_annual = sum(results_q6['super_contributions']) * 52
total_income_tax_reduction = sum(income_tax_q4) - sum(results_q6['income_tax'])
total_tax_return_change = sum(tax_return_q4) - sum(results_q6['tax_return'])

print("\nSUMMARY IMPACT OF SUPERANNUATION:")
print(f"Total Annual Super Contributions: ${total_super_annual:,.2f}")
print(f"Total Income Tax Reduction: ${total_income_tax_reduction:,.2f}")
print(f"Average Tax Reduction per Person: ${total_income_tax_reduction/len(weekly_salary):,.2f}")
print(f"Change in Total Tax Return: ${total_tax_return_change:,.2f}")
print(f"Effective Super Tax Benefit: {total_income_tax_reduction/total_super_annual*100:.1f}% of super contributions")
```

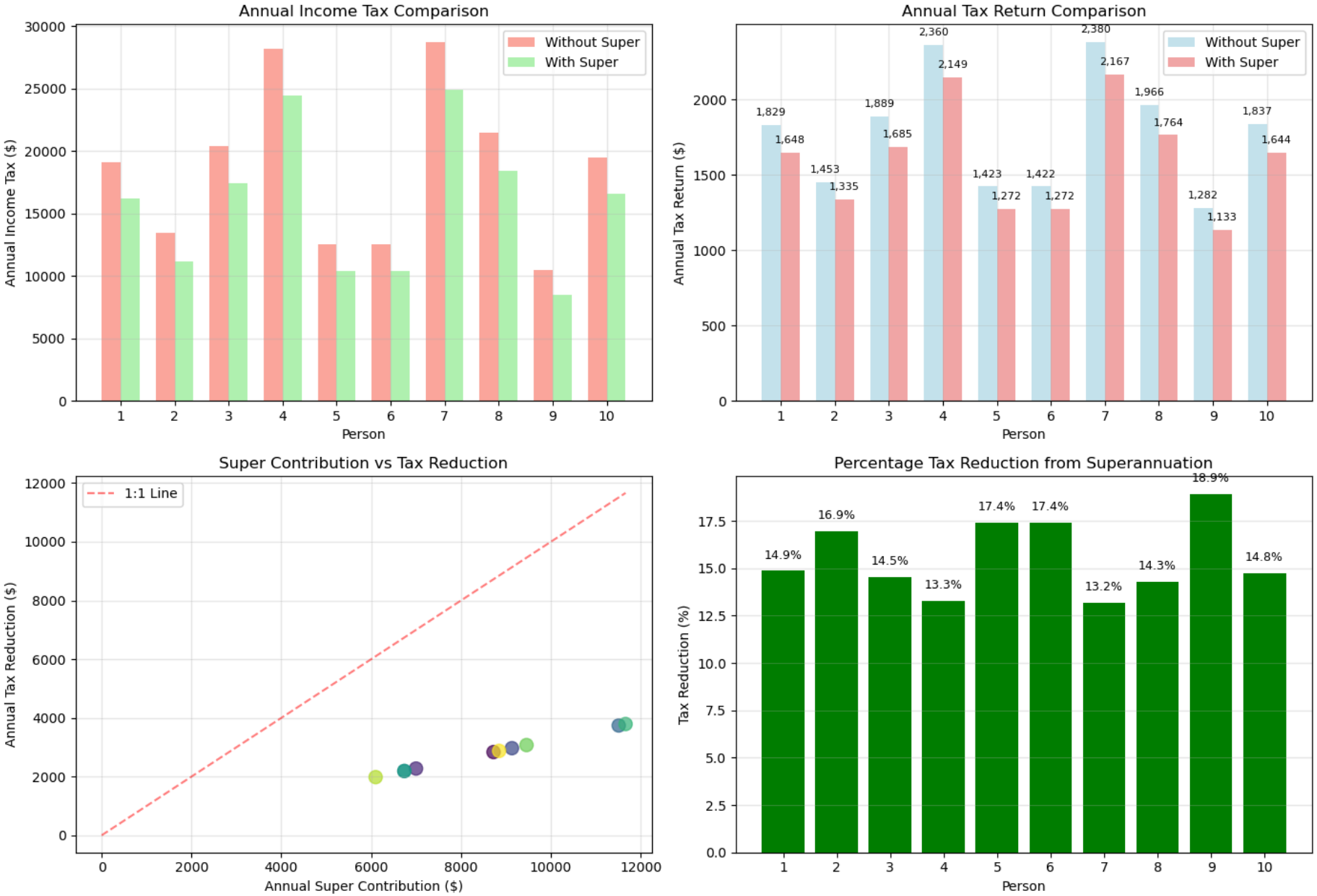
```
================================================================================================================================
COMPARATIVE ANALYSIS: IMPACT OF SUPERANNUATION ON TAX LIABILITY
================================================================================================================================
Person    Package      Base         Super        Income Tax      Withholding     Tax Return
          ($/week)     ($/week)     ($/week)     (Q4 vs Q6)      (Q4 vs Q6)      (Q4 vs Q6)
--------------------------------------------------------------------------------------------------------------------------------
Person 1  $1692.77     $1525.02     $167.75      $19074.81→$16239.84 $402 →$344  $1829.19→$1648.16
Person 2  $1358.13     $1223.54     $134.59      $13419.40→$11144.83 $286 →$240  $1452.60→$1335.17
Person 3  $1772.33     $1596.69     $175.64      $20419.38→$17451.06 $429 →$368  $1888.62→$1684.94
Person 4  $2233.64     $2012.29     $221.35      $28215.52→$24474.70 $588 →$512  $2360.48→$2149.30
Person 5  $1307.60     $1178.02     $129.58      $12565.44→$10375.54 $269 →$224  $1422.56→$1272.46
Person 6  $1307.61     $1178.03     $129.58      $12565.61→$10375.71 $269 →$224  $1422.39→$1272.29
Person 7  $2263.25     $2038.96     $224.29      $28715.92→$24925.42 $598 →$521  $2380.08→$2166.58
Person 8  $1835.44     $1653.55     $181.89      $21485.94→$18411.99 $451 →$388  $1966.06→$1764.01
Person 9  $1183.59     $1066.30     $117.29      $10469.67→$8487.47 $226 →$185  $1282.33→$1132.53
Person 10 $1716.93     $1546.78     $170.15      $19483.12→$16607.58 $410 →$351  $1836.88→$1644.42
--------------------------------------------------------------------------------------------------------------------------------

SUMMARY IMPACT OF SUPERANNUATION:
Total Annual Super Contributions: $85,909.72
Total Income Tax Reduction: $27,920.67
Average Tax Reduction per Person: $2,792.07
Change in Total Tax Return: $1,771.33
Effective Super Tax Benefit: 32.5% of super contributions
```

9.0 Visualization of Tax Impact

In [30]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Create visualization
fig, axes = plt.subplots(2, 2, figsize=(14, 10))
fig.suptitle('Impact of Superannuation on Tax Liability', fontsize=16, fontweight='bold')

# Data preparation
persons = list(range(1, len(weekly_salary) + 1))

# Plot 1: Income Tax Comparison
ax1 = axes[0, 0]
```

```
x = np.arange(len(persons))
width = 0.35
ax1.bar(x - width/2, income_tax_q4, width, label='Without Super', color='salmon', alpha=0.7)
ax1.bar(x + width/2, results_q6['income_tax'], width, label='With Super', color='lightgreen', alpha=0.7)
ax1.set_xlabel('Person')
ax1.set_ylabel('Annual Income Tax ($)')
ax1.set_title('Annual Income Tax Comparison')
ax1.set_xticks(x)
ax1.set_xticklabels(persons)
ax1.legend()
ax1.grid(True, alpha=0.3)

# Plot 2: Tax Return Comparison
ax2 = axes[0, 1]
bars1 = ax2.bar(x - width/2, tax_return_q4, width, label='Without Super', color='lightblue', alpha=0.7)
bars2 = ax2.bar(x + width/2, results_q6['tax_return'], width, label='With Super', color='lightcoral', alpha=0.7)
ax2.set_xlabel('Person')
ax2.set_ylabel('Annual Tax Return ($)')
ax2.set_title('Annual Tax Return Comparison')
ax2.set_xticks(x)
ax2.set_xticklabels(persons)
ax2.legend()
ax2.grid(True, alpha=0.3)

# Add value labels for significant differences
for i, (val1, val2) in enumerate(zip(tax_return_q4, results_q6['tax_return'])):
    if abs(val1 - val2) > 100:
        ax2.text(i - width/2, val1 + 50, f'{val1:,.0f}', ha='center', va='bottom', fontsize=8)
        ax2.text(i + width/2, val2 + 50, f'{val2:,.0f}', ha='center', va='bottom', fontsize=8)

# Plot 3: Super Contribution vs Tax Reduction
ax3 = axes[1, 0]
annual_super = [s * 52 for s in results_q6['super_contributions']]
tax_reductions = [income_tax_q4[i] - results_q6['income_tax'][i] for i in range(len(weekly_salary))]
scatter = ax3.scatter(annual_super, tax_reductions, s=100, c=persons, cmap='viridis', alpha=0.7)
ax3.plot([0, max(annual_super)], [0, max(annual_super)], 'r--', alpha=0.5, label='1:1 Line')
ax3.set_xlabel('Annual Super Contribution ($)')
ax3.set_ylabel('Annual Tax Reduction ($)')
ax3.set_title('Super Contribution vs Tax Reduction')
ax3.legend()
ax3.grid(True, alpha=0.3)
```

```python
# Plot 4: Percentage Tax Reduction
ax4 = axes[1, 1]
tax_reduction_pct = [((income_tax_q4[i] - results_q6['income_tax'][i]) / income_tax_q4[i] * 100)
                     if income_tax_q4[i] > 0 else 0
                     for i in range(len(weekly_salary))]
bars = ax4.bar(persons, tax_reduction_pct, color=['green' if p > 10 else 'orange' if p > 5 else 'red'
                                                  for p in tax_reduction_pct])
ax4.set_xlabel('Person')
ax4.set_ylabel('Tax Reduction (%)')
ax4.set_title('Percentage Tax Reduction from Superannuation')
ax4.set_xticks(persons)
ax4.grid(True, alpha=0.3, axis='y')

# Add value labels
for bar, pct in zip(bars, tax_reduction_pct):
    height = bar.get_height()
    ax4.text(bar.get_x() + bar.get_width()/2., height + 0.5,
            f'{pct:.1f}%', ha='center', va='bottom', fontsize=9)

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


Impact of Superannuation on Tax Liability
Annual Income Tax Comparison
Without Super
With Super
Annual Income Tax ($)
Person
Annual Tax Return Comparison
Annual Tax Return ($)
1,829
1,648
1,453
1,335
1,889
1,685
2,360
2,149
1,423
1,272
1,422
1,272
2,380
2,167
1,966
1,764
1,282
1,133
1,837
1,644
Super Contribution vs Tax Reduction
1:1 Line
Annual Tax Reduction ($)
Annual Super Contribution ($)
Percentage Tax Reduction from Superannuation
Tax Reduction (%)
14.9%
16.9%
14.5%
13.3%
17.4%
17.4%
13.2%
14.3%
18.9%
14.8%

9.1 Advanced Analysis: Financial Impact

9.1 Net Benefit Calculation

In [31]:
```python
# Calculate comprehensive financial impact
print("=" * 100)
print("COMPREHENSIVE FINANCIAL IMPACT ANALYSIS")
print("=" * 100)
print(f"{'Person':<8} {'Annual':<12} {'Annual':<12} {'Net':<15} {'Tax':<12} {'Effective':<12} {'Net':<12}")
print(f"{'':<8} {'Super':<12} {'Tax':<12} {'Position':<15} {'Rate':<12} {'Tax Rate':<12} {'Benefit':<12}")
print(f"{'':<8} {'Contribution':<12} {'Reduction':<12} {'Change':<15} {'(Marginal)':<12} {'(%):':<12} {'Ratio':<12}")
print("-" * 100)

for i in range(len(weekly_salary)):
    annual_super = results_q6['super_contributions'][i] * 52
    tax_reduction = income_tax_q4[i] - results_q6['income_tax'][i]
    net_position = tax_reduction  # Benefit from tax reduction

    # Determine marginal tax rate
    annual_base = results_q6['base_salaries'][i] * 52
    if annual_base <= 18200:
        marginal_rate = 0
    elif annual_base <= 45000:
        marginal_rate = 0.19
    elif annual_base <= 120000:
        marginal_rate = 0.325
    elif annual_base <= 180000:
        marginal_rate = 0.37
    else:
        marginal_rate = 0.45

    # Calculate effective tax rate on super
    effective_rate = (tax_reduction / annual_super) if annual_super > 0 else 0

    # Calculate net benefit ratio (how much of super contribution returns as tax benefit)
    benefit_ratio = (tax_reduction / annual_super) if annual_super > 0 else 0

    print(f"Person {i+1:<2} ${annual_super:<11.0f} ${tax_reduction:<11.2f} ${net_position:<14.2f} "
```

```
          f"{marginal_rate*100:<11.1f}% {effective_rate*100:<11.1f}% {benefit_ratio*100:<11.1f}%")

print("-" * 100)

# Overall summary
total_annual_super = sum([s * 52 for s in results_q6['super_contributions']])
total_tax_reduction = sum(income_tax_q4) - sum(results_q6['income_tax'])
overall_benefit_ratio = total_tax_reduction / total_annual_super if total_annual_super > 0 else 0

print(f"\nOVERALL SUMMARY:")
print(f"Total Annual Super Contributions: ${total_annual_super:,.2f}")
print(f"Total Annual Tax Reduction: ${total_tax_reduction:,.2f}")
print(f"Average Tax Reduction: ${total_tax_reduction/len(weekly_salary):,.2f} per person")
print(f"Effective Super Tax Benefit Rate: {overall_benefit_ratio*100:.2f}%")
print(f"Net Benefit-Cost Ratio: {overall_benefit_ratio:.3f} (For every $1 in super, ${overall_benefit_ratio:.3f} in tax saving
```

```
====================================================================================================
COMPREHENSIVE FINANCIAL IMPACT ANALYSIS
====================================================================================================
Person   Annual       Annual       Net             Tax          Effective    Net
         Super        Tax          Position        Rate         Tax Rate     Benefit
         Contribution Reduction    Change          (Marginal)   (%):         Ratio
----------------------------------------------------------------------------------------------------
Person 1  $8723        $2834.97     $2834.97        32.5       % 32.5       % 32.5       %
Person 2  $6999        $2274.57     $2274.57        32.5       % 32.5       % 32.5       %
Person 3  $9133        $2968.32     $2968.32        32.5       % 32.5       % 32.5       %
Person 4  $11510       $3740.82     $3740.82        32.5       % 32.5       % 32.5       %
Person 5  $6738        $2189.90     $2189.90        32.5       % 32.5       % 32.5       %
Person 6  $6738        $2189.90     $2189.90        32.5       % 32.5       % 32.5       %
Person 7  $11663       $3790.50     $3790.50        32.5       % 32.5       % 32.5       %
Person 8  $9458        $3073.95     $3073.95        32.5       % 32.5       % 32.5       %
Person 9  $6099        $1982.20     $1982.20        32.5       % 32.5       % 32.5       %
Person 10 $8848        $2875.54     $2875.54        32.5       % 32.5       % 32.5       %
----------------------------------------------------------------------------------------------------

OVERALL SUMMARY:
Total Annual Super Contributions: $85,909.72
Total Annual Tax Reduction: $27,920.67
Average Tax Reduction: $2,792.07 per person
Effective Super Tax Benefit Rate: 32.50%
Net Benefit-Cost Ratio: 0.325 (For every $1 in super, $0.325 in tax savings)
```

10.0 Retirement Savings Projection

In [32]:

```python
# Project long-term retirement impact
def project_retirement_savings(super_contributions, years=30, return_rate=0.07):
    """Project retirement savings with compound growth"""
    projections = []
    for weekly_super in super_contributions:
        annual_contribution = weekly_super * 52
        future_value = annual_contribution * (((1 + return_rate) ** years - 1) / return_rate)
        projections.append(future_value)
    return projections

# Calculate projections
projection_years = 30
annual_return = 0.07
retirement_projections = project_retirement_savings(results_q6['super_contributions'],
                                                    years=projection_years,
                                                    return_rate=annual_return)

print("\n" + "=" * 100)
print(f"RETIREMENT SAVINGS PROJECTION ({projection_years} YEARS, {annual_return*100}% ANNUAL RETURN)")
print("=" * 100)
print(f"{'Person':<8} {'Weekly Super':<15} {'Annual Super':<15} {'30-Year Projection':<20} {'Tax Benefit':<15}")
print("-" * 100)

total_future_value = 0
for i in range(len(weekly_salary)):
    annual_super = results_q6['super_contributions'][i] * 52
    tax_reduction = income_tax_q4[i] - results_q6['income_tax'][i]

    print(f"Person {i+1:<2} ${results_q6['super_contributions'][i]:<14.2f} ${annual_super:<14.2f} "
          f"${retirement_projections[i]:<19.2f} ${tax_reduction:<14.2f}")
    total_future_value += retirement_projections[i]

print("-" * 100)
print(f"TOTAL RETIREMENT SAVINGS (30 years): ${total_future_value:,.2f}")
print(f"AVERAGE PER PERSON: ${total_future_value/len(weekly_salary):,.2f}")
```

```
# Calculate present value of tax benefits
present_value_tax_benefit = sum(income_tax_q4) - sum(results_q6['income_tax'])
print(f"\nPRESENT VALUE ANALYSIS:")
print(f"Immediate annual tax benefit: ${present_value_tax_benefit:,.2f}")
print(f"Future retirement savings (30 years): ${total_future_value:,.2f}")
print(f"Combined benefit ratio: {total_future_value/present_value_tax_benefit:.2f}x")
```

```
================================================================================================
RETIREMENT SAVINGS PROJECTION (30 YEARS, 7.000000000000001% ANNUAL RETURN)
================================================================================================
Person   Weekly Super    Annual Super    30-Year Projection   Tax Benefit
------------------------------------------------------------------------------------------------
Person 1  $167.75         $8723.00        $823981.44           $2834.97
Person 2  $134.59         $6998.68        $661100.82           $2274.57
Person 3  $175.64         $9133.28        $862736.81           $2968.32
Person 4  $221.35         $11510.20       $1087262.54          $3740.82
Person 5  $129.58         $6738.16        $636491.89           $2189.90
Person 6  $129.58         $6738.16        $636491.89           $2189.90
Person 7  $224.29         $11663.08       $1101703.71          $3790.50
Person 8  $181.89         $9458.28        $893436.57           $3073.95
Person 9  $117.29         $6099.08        $576123.89           $1982.20
Person 10 $170.15         $8847.80        $835770.15           $2875.54
------------------------------------------------------------------------------------------------
TOTAL RETIREMENT SAVINGS (30 years): $8,115,099.70
AVERAGE PER PERSON: $811,509.97

PRESENT VALUE ANALYSIS:
Immediate annual tax benefit: $27,920.67
Future retirement savings (30 years): $8,115,099.70
Combined benefit ratio: 290.65x
```

11.0 Flexible Input System

In [33]:

```
def flexible_tax_calculator(weekly_salaries, super_rate=0.11):
    """
    Flexible tax calculator that works with any number of individuals

    Parameters:
    -----------
```

```
    weekly_salaries : list
        List of weekly total packages
    super_rate : float
        Superannuation contribution rate (default 11%)

    Returns:
    --------
    dict : Comprehensive tax analysis for all individuals
    """
    results = {}

    for i, salary in enumerate(weekly_salaries):
        # Calculate base salary and super
        super_contribution = salary * (super_rate / (1 + super_rate))
        base_salary = salary - super_contribution

        # Calculate without super
        annual_income_no_super = salary * 52
        income_tax_no_super = calculate_income_tax([salary])[0]
        withholding_tax_no_super = calculate_withholding_tax([salary])[0]
        tax_return_no_super = withholding_tax_no_super * 52 - income_tax_no_super

        # Calculate with super
        income_tax_with_super = calculate_income_tax([base_salary])[0]
        withholding_tax_with_super = calculate_withholding_tax([base_salary])[0]
        tax_return_with_super = withholding_tax_with_super * 52 - income_tax_with_super

        results[f'person_{i+1}'] = {
            'total_package': salary,
            'base_salary': round(base_salary, 2),
            'super_contribution': round(super_contribution, 2),
            'without_super': {
                'annual_income': annual_income_no_super,
                'income_tax': income_tax_no_super,
                'withholding_tax': withholding_tax_no_super,
                'tax_return': round(tax_return_no_super, 2),
                'weekly_net': salary - withholding_tax_no_super
            },
            'with_super': {
                'annual_income': base_salary * 52,
                'income_tax': income_tax_with_super,
```

```
                'withholding_tax': withholding_tax_with_super,
                'tax_return': round(tax_return_with_super, 2),
                'weekly_net': base_salary - withholding_tax_with_super
            }
        }

    return results

# Demonstrate flexibility with different input
test_salaries = [800.50, 1500.75, 2200.25, 3500.00, 5000.50]
flexible_results = flexible_tax_calculator(test_salaries)

print("\n" + "=" * 100)
print("FLEXIBLE TAX CALCULATOR DEMONSTRATION")
print(f"Number of individuals: {len(test_salaries)}")
print("=" * 100)

for i, salary in enumerate(test_salaries):
    person_key = f'person_{i+1}'
    data = flexible_results[person_key]
    print(f"\nPerson {i+1}: Total Package = ${salary:.2f}/week")
    print(f"  Base Salary: ${data['base_salary']:.2f}, Super: ${data['super_contribution']:.2f}")
    print(f"  Without Super: Tax = ${data['without_super']['income_tax']:.2f}, "
          f"Return = ${data['without_super']['tax_return']:.2f}")
    print(f"  With Super: Tax = ${data['with_super']['income_tax']:.2f}, "
          f"Return = ${data['with_super']['tax_return']:.2f}")
    print(f"  Tax Benefit: ${data['without_super']['income_tax'] - data['with_super']['income_tax']:.2f}")
```

```
================================================================================================
FLEXIBLE TAX CALCULATOR DEMONSTRATION
Number of individuals: 5
================================================================================================

Person 1: Total Package = $800.50/week
  Base Salary: $721.17, Super: $79.33
  Without Super: Tax = $4450.94, Return = $801.06
  With Super: Tax = $3667.17, Return = $648.83
  Tax Benefit: $783.77

Person 2: Total Package = $1500.75/week
  Base Salary: $1352.03, Super: $148.72
  Without Super: Tax = $15829.68, Return = $1590.32
  With Super: Tax = $13316.26, Return = $1451.74
  Tax Benefit: $2513.42

Person 3: Total Package = $2200.25/week
  Base Salary: $1982.21, Super: $218.04
  Without Super: Tax = $27651.23, Return = $2352.77
  With Super: Tax = $23966.30, Return = $2085.70
  Tax Benefit: $3684.93

Person 4: Total Package = $3500.00/week
  Base Salary: $3153.15, Super: $346.85
  Without Super: Tax = $52567.00, Return = $3697.00
  With Super: Tax = $45733.67, Return = $3354.33
  Tax Benefit: $6833.33

Person 5: Total Package = $5000.50/week
  Base Salary: $4504.95, Super: $495.55
  Without Super: Tax = $87678.70, Return = $5245.30
  With Super: Tax = $76082.95, Return = $4725.05
  Tax Benefit: $11595.75
```

12.0 Conclusion for superannuation system

6.1 Key Findings

1. Tax Efficiency: Superannuation contributions provide immediate tax benefits by reducing taxable income at the individual's marginal tax rate.

2. Progressive Benefit: Higher-income individuals benefit more from superannuation due to higher marginal tax rates.

3. Retirement Security: While reducing current tax liability, superannuation simultaneously builds retirement savings with compound growth.

Cash Flow Impact: Superannuation reduces immediate cash flow but provides long-term financial security.

6.2 Practical Implications

1. Financial Planning: Individuals should consider maximizing superannuation contributions to optimize tax efficiency.

2. Employer Compliance: Employers must correctly calculate and report superannuation to ensure compliance.

3. Policy Consideration: The superannuation system creates a balance between current consumption and future security.

6.3 Limitations

1. Simplified Model: This analysis doesn't consider Medicare Levy, HELP debts, or other tax offsets.

2. Super Caps: Real-world superannuation has annual contribution caps not considered here.

3. Investment Risk: Superannuation returns vary based on investment performance.

This comprehensive analysis demonstrates the significant impact of superannuation on Australia's tax system, highlighting both immediate tax benefits and long-term retirement security

12.0 Conclusion for superannuation system

12.1 Key Findings

1. Tax Efficiency: Superannuation contributions provide immediate tax benefits by reducing taxable income at the individual's marginal tax rate.

2. Progressive Benefit: Higher-income individuals benefit more from superannuation due to higher marginal tax rates.

3. Retirement Security: While reducing current tax liability, superannuation simultaneously builds retirement savings with compound growth.

Cash Flow Impact: Superannuation reduces immediate cash flow but provides long-term financial security.

12.2 Practical Implications

1. Financial Planning: Individuals should consider maximizing superannuation contributions to optimize tax efficiency.

2. Employer Compliance: Employers must correctly calculate and report superannuation to ensure compliance.

3. Policy Consideration: The superannuation system creates a balance between current consumption and future security.

12.3 Limitations

1. Simplified Model: This analysis doesn't consider Medicare Levy, HELP debts, or other tax offsets.

2. Super Caps: Real-world superannuation has annual contribution caps not considered here.

3. Investment Risk: Superannuation returns vary based on investment performance.

This comprehensive analysis demonstrates the significant impact of superannuation on Australia's tax system, highlighting both immediate tax benefits and long-term retirement security

========== END OF REPORT ==========

In [ ]: