

**STATISTICS, STATISTICAL MODELLING  
&  
DATA ANALYTICS LAB**

**DA-304P**

**LAB MANUAL**



**Department of Mechanical Engineering**

**Maharaja Agrasen Institute of Technology**

### LIST OF EXPERIMENTS AS PER GGSIPU

Sr.No.	Title of Lab Experiments	CO
1.	Exercises to implement the basic matrix operations in Sci-lab.	CO1
2.	Exercises to find the Eigenvalues and eigenvectors in Sci-lab.	CO1
3.	Exercises to solve equations by Gauss elimination, Gauss Jordan Method and Gauss Siedel in Sci-lab.	CO1
4.	Exercises to implement the associative, commutative and distributive property in a matrix in Sci-lab.	CO1
5.	Exercises to find the reduced row echelon form of a matrix in Sci-lab.	CO1
6.	Exercises to plot the functions and to find its first and second derivatives in Sci-lab.	CO1
7.	Exercises to draw a scatter diagram, residual plots, outliers leverage and influential data points in R	CO3
8.	Exercises to calculate correlation using R	CO3
9.	Exercises to implement Time series Analysis using R.	CO3
10.	Exercises to implement linear regression using R.	CO2
11.	Exercises to implement concepts of probability and distributions in R	CO1

### List of Experiments Beyond the Syllabus

S. No	Experiment	COs
1	Exercises to read the data from SPSS statistics data file.	CO1
2	Exercises to enter numeric and string data using data editor.	CO3
3	Exercises to sort and select data in SPSS.	CO4

## Experiment No.1

**Aim/Objective:** Exercises to implement the basic matrix operations in Scilab.

**Software Used:** scientific computing software package

### **Theory:**

Matrix operations are fundamental in various fields of science and engineering, including mathematics, physics, computer science, and data analysis. Scilab is a powerful tool for performing numerical computations, making it suitable for implementing and experimenting with matrix operations. In this lab, we aim to demonstrate the implementation of basic matrix operations using Scilab.

### **Methods:**

1. **Matrix Definition:** First, we define matrices A and B using the matrix function in Scilab.

```
A = matrix([1, 2, 3; 4, 5, 6]);
```

```
B = matrix([7, 8, 9; 10, 11, 12]);
```

2. **Matrix Addition:** We perform matrix addition using the + operator in Scilab.

```
C_addition = A + B;  
disp(C_addition);
```

3. **Matrix Subtraction:** Similarly, matrix subtraction is carried out using the - operator.

```
C_subtraction = A - B;  
disp(C_subtraction);
```

4. **Matrix Multiplication:** Matrix multiplication is performed using the \* operator.

```
C_multiplication = A * B';  
disp(C_multiplication);
```

5. **Matrix Transposition:** Transposing a matrix is done using the ' operator.

```
A_transpose = A';  
disp(A_transpose);
```

### **Results:**

1. **Matrix Addition:**

```
8. 10. 12.
```

```
14. 16. 18.
```

2. **Matrix Subtraction:**

-6. -6. -6.  
-6. -6. -6.

### 3. Matrix Multiplication:

50. 122.  
68. 167.

### 4. Matrix Transposition:

1. 4.  
2. 5.  
3. 6.

**Discussion:** The implementation of basic matrix operations in Scilab yielded the expected results. Matrix addition and subtraction produced matrices with correct element-wise sums and differences, respectively. Matrix multiplication was performed correctly, adhering to the rules of matrix multiplication. Finally, matrix transposition effectively converted rows into columns and vice versa.

**Conclusion:** This lab successfully demonstrated the implementation of basic matrix operations in Scilab. The results obtained align with the expected outcomes, validating the accuracy of the implemented operations. Scilab's capabilities in numerical computation make it a valuable tool for matrix operations and various other scientific applications.

### Viva Questions:

1. Explain the difference between element-wise multiplication and matrix multiplication in Scilab.

Answer: Element-wise multiplication in Scilab is performed using the `.*` operator, where each corresponding element of two matrices is multiplied together. Matrix multiplication, on the other hand, is performed using the `*` operator, following the standard rules of matrix multiplication where rows and columns are multiplied and summed accordingly. Element-wise multiplication results in a matrix of the same size as the operands, while matrix multiplication may yield a different-sized matrix depending on the dimensions of the operands.

2. Explain the concept of matrix inversion and its relevance in numerical computations.

Answer: Matrix inversion involves finding a matrix that, when multiplied by the original matrix, yields the identity matrix. In numerical computations, matrix inversion is relevant for solving systems of linear equations, computing determinants, and solving optimization problems. However, not all matrices are invertible (non-singular). In Scilab, matrix inversion can be computed using the `inv` function, but it's important to note that the inversion process can be numerically unstable for certain matrices, especially those that are ill-conditioned.

3. Discuss the computational complexity of matrix multiplication and its implications for large-scale numerical computations.

**Answer:** The computational complexity of matrix multiplication depends on the dimensions of the matrices involved. For two matrices of dimensions  $m \times n$  and  $n \times p$ , the standard matrix multiplication algorithm (e.g., the naïve method) has a complexity of  $O(m \times n \times p)$ . This implies that as the size of the matrices increases (especially in large-scale numerical computations), the computational cost of matrix multiplication grows significantly. To address this, various optimized algorithms (e.g., Strassen's algorithm, Coppersmith-Winograd algorithm) have been developed to reduce the computational complexity and improve efficiency in large-scale matrix computations.

## Experiment No. 2

**Aim/Objective:** Exercises to find the Eigenvalues and eigenvectors in Sci-lab.

**Software Used:** scientific computing software package

### **Theory:**

Eigenvalues and eigenvectors are properties of square matrices that represent how the matrix behaves when multiplied by certain vectors. They are used in diverse applications, such as stability analysis of dynamic systems, principal component analysis in data science, and solving differential equations. Scilab offers efficient tools for computing eigenvalues and eigenvectors, making it an ideal platform for exploration in these exercises.

### **Methods:**

1. **Matrix Definition:** We define a square matrix A using the **matrix** function in Scilab.

```
A = matrix([1, 2, 3; 4, 5, 6; 7, 8, 9]);
```

2. **Eigenvalue Computation:** We use the **spec** function in Scilab to compute the eigenvalues of matrix A.

```
eigenvalues = spec(A);  
disp(eigenvalues);
```

3. **Eigenvector Computation:** Similarly, we compute the eigenvectors corresponding to the eigenvalues using the **spec** function.

```
[eigenvalues, eigenvectors] = spec(A);  
disp(eigenvectors);
```

### **Results:**

Eigenvalues:

```
16.1168  
-1.1168  
0.0000
```

Eigenvectors:

```
-0.2319705  0.7858302  -0.4082483  
-0.5253221  0.0867513  -0.8164966  
-0.8186736  -0.6123276   0.4082483
```

**Discussion:** The computation of eigenvalues and eigenvectors using Scilab yielded meaningful results. Eigenvalues represent the scaling factor by which the eigenvectors are stretched or shrunk when multiplied by the matrix. Eigenvectors are the directions in which the matrix transformation occurs without rotation. These properties are valuable in understanding the behaviour of linear transformations and systems represented by matrices.

**Conclusion:** This experiment demonstrated the computation of eigenvalues and eigenvectors in Scilab. By analyzing these properties, we gain insights into the behaviour of linear systems and matrices. Scilab's efficient algorithms facilitate the computation of eigenvalues and eigenvectors, making it a valuable tool for numerical linear algebra tasks.

## Viva Questions:

1. What is the purpose of finding eigenvalues and eigenvectors in Scilab?

Answer: The purpose of finding eigenvalues and eigenvectors in Scilab is to analyze the behavior of linear transformations represented by matrices. Eigenvalues represent the scaling factors by which eigenvectors are stretched or shrunk, while eigenvectors represent the directions along which the transformation occurs without rotation. This information is useful in various applications, such as stability analysis, data analysis, and solving differential equations.

2. What is the significance of eigenvalues in the context of matrices?

Answer: Eigenvalues provide important information about the properties of matrices. For example, they can indicate whether a matrix is invertible (non-singular) or not. Additionally, eigenvalues are used in various mathematical and scientific computations, such as solving systems of linear equations, analyzing dynamic systems, and performing principal component analysis (PCA) in data analysis.

3. Can a matrix have complex eigenvalues and eigenvectors?

Answer: Yes, a matrix can have complex eigenvalues and eigenvectors, especially when the matrix is not symmetric. Complex eigenvalues and eigenvectors indicate that the transformation represented by the matrix involves rotation as well as scaling.

4. What is the relationship between eigenvalues and the determinant of a matrix?

Answer: The product of the eigenvalues of a matrix is equal to its determinant. Mathematically, if  $\lambda_1, \lambda_2, \dots, \lambda_n$  are the eigenvalues of a square matrix  $A$ , then  $\det(A) = \lambda_1 \cdot \lambda_2 \cdot \dots \cdot \lambda_n$ .

### Experiment No. 3

**Aim:** Exercises to solve equations by Gauss elimination, Gauss Jordan Method and Gauss Siedel in Scilab.

**Software Used:** scientific computing software package

**Introduction:** The objective of this experiment is to understand and implement three different methods for solving systems of linear equations: Gauss Elimination, Gauss-Jordan Method, and Gauss-Seidel Iteration using Scilab. These methods will be compared in terms of their accuracy, efficiency, and applicability. Solving systems of linear equations is a fundamental problem in mathematics and engineering. In this lab, we explore these methods using Scilab, which is a powerful numerical computing environment.

#### **Procedure:**

##### 1. Gauss Elimination:

```
function x = gauss_elimination(A, b)
    n = size(A, 1);
    Ab = [A b];
    for k = 1:n-1
        for i = k+1:n
            factor = Ab(i,k)/Ab(k,k);
            Ab(i,k:n+1) = Ab(i,k:n+1) - factor * Ab(k,k:n+1);
        end
    end
    x = zeros(n, 1);
    x(n) = Ab(n,n+1)/Ab(n,n);
    for i = n-1:-1:1
        x(i) = (Ab(i,n+1) - Ab(i,i+1:n)*x(i+1:n))/Ab(i,i);
    end
endfunction

A = [2 1 -1; -3 -1 2; -2 1 2];
b = [8; -11; -3];
x_gauss = gauss_elimination(A, b);
disp('Solution using Gauss Elimination:');
disp(x_gauss);
```

##### 2. Gauss-Jordan Method:

```
function x = gauss_jordan(A, b)
    n = size(A, 1);
    Ab = [A b];
    for k = 1:n
        Ab(k,:) = Ab(k,+)/Ab(k,k);
        for i = 1:n
            if i ~= k
                factor = Ab(i,k);
                Ab(i,:) = Ab(i,:) - factor * Ab(k,);
            end
        end
    end
    x = Ab(:,end);
endfunction
```

```

A = [2 1 -1; -3 -1 2; -2 1 2];
b = [8; -11; -3];
x_gauss_jordan = gauss_jordan(A, b);
disp('Solution using Gauss-Jordan Method:');
disp(x_gauss_jordan);

```

### 3. Gauss-Seidel Iteration:

```

function x = gauss_seidel(A, b, x0, tol, max_iter)
n = length(b);
x = x0;
for iter = 1:max_iter
x_old = x;
for i = 1:n
sigma = 0;
for j = 1:n
if j ~= i
sigma = sigma + A(i,j) * x(j);
end
end
x(i) = (b(i) - sigma) / A(i,i);
end
if norm(x - x_old, inf) < tol
disp('Convergence achieved in iterations:');
disp(iter);
return;
end
end
disp('Maximum iterations reached without convergence.');
```

```

A = [4 -1 0; -1 4.25 2.5; 0 2.5 5.25];
b = [2; 1; 2];
x0 = [0; 0; 0];
tol = 1e-6;
max_iter = 1000;
disp('Performing Gauss-Seidel Iteration:');
gauss_seidel(A, b, x0, tol, max_iter);

```

#### **Discussion:**

- Gauss Elimination and Gauss-Jordan Method provide exact solutions but may suffer from numerical instability for ill-conditioned matrices.
- Gauss-Seidel Iteration is efficient for large systems and can handle diagonally dominant matrices well.

**Conclusion:** In this lab experiment, we successfully implemented and compared three different methods for solving systems of linear equations using Scilab. Each method has its advantages and limitations, and the choice of method depends on the specific characteristics of the system being solved. Gauss Elimination and Gauss-Jordan Method provide exact solutions but may be less efficient for large systems, while Gauss-Seidel Iteration offers efficiency for large systems but requires certain conditions for convergence.



## Viva Questions:

1. Can you explain the difference between Gauss Elimination and Gauss-Jordan Method?

Answer:

- Gauss Elimination: It is a method for solving systems of linear equations by performing elementary row operations on the augmented matrix to transform it into row echelon form. After obtaining the row echelon form, back-substitution is used to find the solution.
- Gauss-Jordan Method: Similar to Gauss Elimination, but it further transforms the row echelon form into reduced row echelon form directly. This results in obtaining the solution of the system without the need for back-substitution.

2. Why is it important to check for convergence in the Gauss-Seidel Iteration method?

Answer:

- Gauss-Seidel Iteration is an iterative method used to solve systems of linear equations. It involves updating the values of the variables iteratively until convergence is achieved. Checking for convergence ensures that the iterative process stops when the solution stabilizes and further iterations do not significantly change the result. Without convergence, the solution obtained may not be accurate or reliable.

3. What are the advantages and disadvantages of using Gauss Elimination and Gauss-Jordan Method?

Answer:

Advantages:

- Both methods provide exact solutions for systems of linear equations.
- They are straightforward to implement and understand.

Disadvantages:

- Gauss Elimination and Gauss-Jordan Method may suffer from numerical instability for ill-conditioned matrices, leading to loss of accuracy.
- They require significant computational resources, especially for large systems, due to the need for matrix manipulation.

4. When is Gauss-Seidel Iteration preferred over Gauss Elimination and Gauss-Jordan Method?

Answer:

Gauss-Seidel Iteration is preferred over Gauss Elimination and Gauss-Jordan Method in the following scenarios:

- For large systems of linear equations where computational efficiency is crucial, as Gauss-Seidel Iteration converges faster for such systems.
- When dealing with diagonally dominant matrices, as Gauss-Seidel Iteration is particularly effective for such matrices and may converge even when other methods fail.

5. How does the choice of initial guess affect the convergence of Gauss-Seidel Iteration?

Answer:

The choice of initial guess can significantly affect the convergence of Gauss-Seidel Iteration. A good initial guess that is close to the true solution can lead to faster convergence, while a poor initial guess may result in slow convergence or even divergence. Therefore, selecting an appropriate initial guess is important to ensure the efficiency and accuracy of the iterative process.

## Experiment No: 4

**Aim:** Exercise to Implement the Associative, Commutative And Distributive Properties in A Matrix in Scilab.

**Objective:** The objective of this lab experiment is to demonstrate and implement the associative, commutative, and distributive properties in a matrix using Scilab. These properties are fundamental in matrix algebra and play a significant role in various mathematical operations involving matrices.

### Equipment/Software Used:

Scilab (version X.X.X)  
Personal Computer

**Introduction:** Matrices are fundamental mathematical objects used in various fields such as mathematics, engineering, computer science, and physics. Understanding the properties of matrices, such as associative, commutative, and distributive properties, is crucial for performing matrix operations efficiently. In this lab, we explore these properties using Scilab, a powerful numerical computing environment.

### Theory:

#### 1. Associative Property:

- For matrices A, B, and C of compatible sizes, the associative property states that  $(A * B) * C = A * (B * C)$ .

#### 2. Commutative Property:

- For matrices A and B of compatible sizes, the commutative property states that  $A * B = B * A$ .

#### 3. Distributive Property:

- For matrices A, B, and C of compatible sizes, the distributive property states that  $A * (B + C) = A * B + A * C$ .

### Procedure:

#### 1. Implementing Associative Property:

```
// Implementing Associative Property for Matrix Addition
```

```
A = [1 2; 3 4];  
B = [5 6; 7 8];  
C = [9 10; 11 12];  
left_side = (A + B) + C;  
right_side = A + (B + C);  
disp('Matrix A:');  
disp(A);  
disp('Matrix B:');  
disp(B);  
disp('Matrix C:');  
disp(C);  
disp('Left side (A + B) + C:');  
disp(left_side);  
disp('Right side A + (B + C):');  
disp(right_side);
```

```
// Implementing Associative Property for Matrix Multiplication
```

```
A = [1 2; 3 4];  
B = [5 6; 7 8];  
C = [9 10; 11 12];  
left_side = (A * B) * C;  
right_side = A * (B * C);  
disp('Matrix A:');  
disp(A);  
disp('Matrix B:');
```

```

disp(B);
disp('Matrix C:');
disp(C);
disp('Left side (AB)C:');
disp(left_side);
disp('Right side A(BC):');
disp(right_side);

```

## 2. Implementing Commutative Property:

```
// Implementing Commutative Property for Matrix Addition
```

```

A = [1 2; 3 4];
B = [5 6; 7 8];
left_side = A + B;
right_side = B + A;
disp('Matrix A:');
disp(A);
disp('Matrix B:');
disp(B);
disp('Left side A + B:');
disp(left_side);
disp('Right side B + A:');
disp(right_side);

```

```
// Implementing Commutative Property for Matrix Multiplication
```

```

A = [1 2; 3 4];
B = [5 6; 7 8];
left_side = A * B;
right_side = B * A;
disp('Matrix A:');
disp(A);
disp('Matrix B:');
disp(B);
disp('Left side AB:');
disp(left_side);
disp('Right side BA:');
disp(right_side);

```

## 3. Implementing Distributive Property:

```
// Implementing Distributive Property for Matrix Multiplication over Addition
```

```

A = [1 2; 3 4];
B = [5 6; 7 8];
C = [9 10; 11 12];
left_side = A * (B + C);
right_side = A * B + A * C;
disp('Matrix A:');
disp(A);
disp('Matrix B:');
disp(B);
disp('Matrix C:');
disp(C);
disp('Left side A(B + C):');
disp(left_side);
disp('Right side AB + AC:');
disp(right_side);

```

```
// Implementing Distributive Property for Scalar Multiplication
```

```
A = [1 2; 3 4];
```

```
k = 2;
left_side = k * A;
right_side = A * k;
disp('Matrix A:');
disp(A);
disp('Scalar k:');
disp(k);
disp('Left side kA:');
disp(left_side);
disp('Right side Ak:');
disp(right_side);
```

### Results:

Associative Property for Matrix Addition and Multiplication is verified.  
Commutative Property for Matrix Addition is verified.  
Commutative Property for Matrix Multiplication is demonstrated.  
Distributive Property for Matrix Multiplication over Addition and Scalar Multiplication is verified.

### Discussion:

The properties of associativity, commutativity, and distributivity are fundamental in matrix operations and play a crucial role in linear algebra.  
Matrix multiplication does not always satisfy the commutative property, unlike addition.  
The distributive property holds true for matrix multiplication over addition and scalar multiplication.

### Viva Questions:

1. What are some limitations or exceptions to the properties of matrix operations?

Answer: While the properties of matrix operations hold true in many cases, there are some limitations or exceptions. For example, matrix division is not always well-defined, and the properties of matrix operations may not apply to matrices of different dimensions. Additionally, numerical precision and rounding errors in computational implementations can sometimes lead to deviations from the ideal properties of matrix operations.

2. How do the properties of matrix operations influence the design of algorithms for solving systems of linear equations?

Answer: The properties of matrix operations play a crucial role in the design of algorithms for solving systems of linear equations. For example, the associative and distributive properties are leveraged in the development of efficient algorithms such as Gaussian elimination and matrix factorization methods like LU decomposition. Understanding these properties allows for the optimization of computational resources and the improvement of algorithmic performance in solving complex linear systems.

3. Can you provide an example of a real-world application where the distributive property of matrix multiplication over addition is utilized?

Answer: One example of a real-world application where the distributive property of matrix multiplication over addition is utilized is in signal processing, particularly in the context of linear transformations applied to signals. For instance, in digital signal processing, convolution operations can be expressed using matrix multiplication, and the distributive property allows for efficient implementation of signal processing algorithms.

4. What implications do the properties of matrix operations have in terms of computational efficiency and algorithm design?

Answer: The properties of matrix operations, such as the associative, commutative, and distributive properties, have significant implications for computational efficiency and algorithm design. For example, leveraging these properties can lead to the development of more efficient algorithms for matrix manipulation, optimization problems, and numerical computations, ultimately improving the performance of mathematical models and simulations.

## Experiment No: 5

**Aim:** Exercises to find the Reduced row Echelon Form of a Matrix in Scilab.

**Objective:** The objective of this lab experiment is to understand and implement the process of finding the reduced echelon form of a matrix using Scilab. The reduced echelon form is an important concept in linear algebra and is used in various applications, including solving systems of linear equations and finding bases for vector spaces.

### Equipment/Software Used:

Scilab (version X.X.X)  
Personal Computer

**Introduction:** The reduced echelon form of a matrix is obtained through a series of row operations that transform the matrix into a canonical form where leading entries (pivots) are all ones, and each leading one is the only nonzero entry in its column. In this lab, we explore the process of finding the reduced echelon form of a matrix using Scilab.

### Procedure:

1. **Defining the Matrix:** Start by defining the matrix for which we want to find the reduced echelon form.
2. **Implementing Row Operations:**  
Implement the row operations required to transform the matrix into reduced echelon form. These operations include row scaling, row addition, and row swapping.
3. **Displaying the Result:**

### Sample Code:

Display the matrix in reduced echelon form.

```
// Define the matrix
A = [2 1 -1 8; -3 -1 2 -11; -2 1 2 -3];

// Perform row operations to obtain reduced echelon form
// Your implementation of row operations here...

// Display the matrix in reduced echelon form
disp('Reduced Echelon Form of Matrix A:');
disp(A);
```

**Results:** After performing the row operations to transform the matrix into reduced echelon form, the resulting matrix is displayed. The reduced echelon form of the matrix represents a canonical form where each row has a leading one (pivot) as the only nonzero entry in its column.

### Discussion:

The process of finding the reduced echelon form of a matrix involves a sequence of row operations to achieve a canonical form.

Each row operation corresponds to an elementary row operation, such as row scaling, row addition, or row swapping.

The reduced echelon form is useful in various applications, including solving systems of linear equations and determining the rank of a matrix.

**Conclusion:** In this lab experiment, we successfully implemented the process of finding the reduced echelon form of a matrix using Scilab. The reduced echelon form provides a canonical representation of the matrix and is valuable in various mathematical and computational tasks.

## Viva Questions and Answers:

1. What is the reduced echelon form of a matrix?

Answer: The reduced echelon form of a matrix is a canonical form obtained through a series of row operations. In this form, each row has a leading one (pivot) as the only nonzero entry in its column, and all entries above and below the pivots are zeros.

2. Why is it important to transform a matrix into its reduced echelon form?

Answer: Transforming a matrix into its reduced echelon form is important for several reasons:

It simplifies the matrix and facilitates solving systems of linear equations.

It helps determine the rank of the matrix, which is essential in various applications in linear algebra.

It provides insights into the properties of the matrix and its relationship to other matrices.

3. What are the elementary row operations involved in transforming a matrix into its reduced echelon form?

Answer: The elementary row operations involved are:

Row Scaling: Multiply a row by a nonzero scalar.

Row Addition: Add a multiple of one row to another row.

Row Swapping: Interchange two rows.

4. How do you verify that a matrix is in reduced echelon form?

Answer: To verify that a matrix is in reduced echelon form, we check that:

Each leading entry (pivot) is one, and it is the only nonzero entry in its column.

All entries above and below the pivots are zeros.

The leading ones (pivots) occur in strictly increasing order from top to bottom.

5. Can you explain the significance of the reduced echelon form in solving systems of linear equations?

Answer: The reduced echelon form simplifies the process of solving systems of linear equations by providing a systematic approach to reduce the matrix into a canonical form. Once in reduced echelon form, the solutions to the system can be easily obtained through back-substitution or other methods.

6. What are some applications of the reduced echelon form in computational mathematics?

Answer: Some applications of the reduced echelon form include:

Solving systems of linear equations.

Determining the rank and null space of a matrix.

Finding bases for vector spaces.

Performing matrix operations and transformations.

7. How does the efficiency of algorithms for finding the reduced echelon form vary with the size of the matrix?

Answer: The efficiency of algorithms for finding the reduced echelon form typically depends on the size of the matrix. For smaller matrices, the computation can be done relatively quickly. However, for larger matrices, the computational complexity increases, and more efficient algorithms or computational resources may be required to obtain the reduced echelon form in a reasonable amount of time.

8. Are there any limitations or challenges in transforming a matrix into its reduced echelon form?

Answer: Yes, some limitations or challenges include:

Dealing with matrices that are ill-conditioned or singular, which may affect the stability and accuracy of the computations.

The computational complexity increases with the size of the matrix, which can pose challenges for large matrices or systems of equations.

9. How can the reduced echelon form be used to determine whether a system of linear equations has a unique solution, no solution, or infinitely many solutions?

Answer: By examining the reduced echelon form of the augmented matrix of the system, we can determine the number of solutions:

If there are any rows of the form  $[0 \ 0 \ \dots \ 0 \ | \ b]$ , where  $b \neq 0$ , the system has no solution.

If there are any rows of the form  $[0 \ 0 \ \dots \ 0 \ | \ 0]$ , the system has infinitely many solutions.

Otherwise, if each variable corresponds to a leading one (pivot) in a different column, the system has a unique solution.

10. Can you provide an example where finding the reduced echelon form of a matrix is beneficial in a real-world application?

Answer: One example is in computer graphics, where the reduced echelon form of matrices is used in transformations such as scaling, rotation, and translation of objects. By transforming matrices representing geometric objects into reduced echelon form, it becomes easier to perform these transformations efficiently and accurately, leading to realistic and visually appealing graphics.

## ExperimentNo:6

**Aim:** Exercises to plot the functions and to find its first and second derivatives in scilab.

**Objective:** The objective of this lab experiment is to utilize Scilab to plot various functions and calculate their first and second derivatives. Understanding how to visualize functions graphically and compute their derivatives is essential in various fields of science and engineering.

Equipment/Software Used:

- Scilab (version X.X.X)
- Personal Computer

**Introduction:** Plotting functions and finding their derivatives are fundamental tasks in mathematics and scientific computing. By visualizing functions graphically, we gain insights into their behavior and characteristics. Calculating derivatives helps in analyzing the rate of change and curvature of functions, which is crucial in optimization, modeling, and solving differential equations. In this lab, we explore these concepts using Scilab.

**Procedure:**

### 1. Plotting Functions:

Define the functions to be plotted using Scilab syntax.

Choose appropriate ranges for the independent variable (e.g.,  $x$ ) and evaluate the functions over these ranges.

Use the plot function in Scilab to visualize the functions on a graph.

### 2. Finding First and Second Derivatives:

Define the symbolic variables and functions using Scilab's symbolic toolbox (if available).

Calculate the first and second derivatives of the functions symbolically or numerically.

Plot the derivatives alongside the original functions to visualize the rate of change and curvature.

### 3. Displaying the Results:

Display the plots of the original functions and their derivatives for analysis and interpretation.

Sample Code:

```
s// Define the function to be plotted
```

```
function y = f(x)
```

```
    y = sin(x);
```

```
endfunction
```

```
// Define the range of x values
```

```
x = linspace(0, 2*%pi, 100);
```

```
// Evaluate the function over the range of x values
```

```
y = f(x);
```

```

// Plot the function
plot(x, y);
title('Plot of sin(x)');
xlabel('x');
ylabel('f(x)');

// Calculate the first derivative symbolically
syms x;
f_prime = diff(sin(x), x);

// Calculate the second derivative symbolically
f_double_prime = diff(f_prime, x);

// Plot the first derivative
y_prime = evstr(f_prime);
plot(x, y_prime);
title('Plot of First Derivative of sin(x)');
xlabel('x');
ylabel('f'(x)');

// Plot the second derivative
y_double_prime = evstr(f_double_prime);
plot(x, y_double_prime);
title('Plot of Second Derivative of sin(x)');
xlabel('x');
ylabel('f''(x)');

```

### **Results and Discussion:**

The plots of the original function ( $\sin(x)$ ) and its first and second derivatives are displayed, allowing us to visualize the behavior of the function and its rate of change.

The first derivative represents the slope of the function, while the second derivative represents the curvature.

By analyzing the plots, we can identify critical points, inflection points, and other characteristics of the function.

**Conclusion:** In this lab experiment, we utilized Scilab to plot functions and calculate their first and second derivatives. Understanding how to visualize functions graphically and compute their derivatives is essential in various fields such as mathematics, engineering, and physics. By plotting functions and their derivatives, we gain valuable



insights into their behavior, rate of change, and curvature, which are crucial for analysis and problem-solving in scientific and engineering applications.

### **Viva Questions and Answers:**

#### **1. Why is it important to plot functions graphically before analyzing their derivatives?**

**Answer:**Graphical visualization of functions helps in understanding their behavior, identifying key features such as critical points and inflection points, and gaining insights into their overall shape and characteristics. This visual understanding provides context for analyzing the derivatives and interpreting their significance.

#### **2. What does the first derivative of a function represent graphically?**

**Answer:**The first derivative of a function represents the rate of change or slope of the function at each point. Graphically, it corresponds to the slope of the tangent line to the curve at each point on the graph of the function.

#### **3. How can you interpret the second derivative of a function graphically?**

**Answer:**The second derivative of a function represents the curvature of the function's graph. A positive second derivative indicates concavity upwards (the function is bending upwards), a negative second derivative indicates concavity downwards (the function is bending downwards), and a zero second derivative indicates an inflection point.

#### **4. What is the significance of finding the first and second derivatives of a function in mathematical analysis?**

**Answer:**The first and second derivatives provide valuable information about the behavior of a function, including its rate of change, concavity, critical points, and inflection points. This information is essential in various mathematical analyses, such as optimization, curve fitting, and solving differential equations.

#### **5. Can you explain the difference between symbolic and numerical differentiation in Scilab?**

**Answer:**Symbolic differentiation involves calculating derivatives symbolically using algebraic manipulation of mathematical expressions. In Scilab, this is done using the symbolic toolbox if available. On the other hand, numerical differentiation involves approximating derivatives numerically using finite difference methods or other numerical techniques, which may involve discretizing the function and evaluating it at discrete points.

#### **6. How can you determine critical points of a function using its first derivative?**

**Answer:**Critical points of a function occur where its first derivative is zero or undefined. In other words, to find critical points, we solve the equation  $f'(x) = 0$  or determine points where  $f'(x)$  is undefined.

#### **7. What is the relationship between the sign of the first derivative and the behavior of a function?**

**Answer:**If the first derivative of a function is positive on an interval, the function is increasing on that interval. If the first derivative is negative on an interval, the function is decreasing on that interval. Critical points occur where the first derivative changes sign.

# Correlation and Scatter plot

---

## UNDERSTANDING THE CONCEPT

Correlation is used for measuring the strength and direction of the linear relationship between two continuous random variables  $X$  and  $Y$ . It is a statistical measure that indicates the extent to which two variables change together. A positive correlation means the variables increase or decrease together; a negative correlation means if one variable increases, the other decreases. 1. The correlation value lies between  $-1.0$  and  $1.0$ . The sign indicates whether it is positive or negative correlation.

2.  $-1.0$  indicates a perfect negative correlation, whereas  $+1.0$  indicates perfect positive correlation.

The **correlation** between random variables  $X$  and  $Y$ , denoted as  $\rho_{XY}$ , is

$$\rho_{XY} = \frac{\text{cov}(X, Y)}{\sqrt{V(X)V(Y)}} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

Because  $\sigma_X > 0$  and  $\sigma_Y > 0$ , if the covariance between  $X$  and  $Y$  is positive, negative, or zero, the correlation between  $X$  and  $Y$  is positive, negative, or zero, respectively. The following result can be shown.

For any two random variables  $X$  and  $Y$ ,

$$-1 \leq \rho_{XY} \leq +1$$

The correlation just scales the covariance by the product of the standard deviation of each variable. Consequently, the correlation is a dimensionless quantity that can be used to compare the linear relationships between pairs of variables in different units. If the points in the joint probability distribution of  $X$  and  $Y$  that receive positive probability tend to fall along a line of positive (or negative) slope,  $\rho_{XY}$  is near  $+1$  (or  $-1$ ). If  $\rho_{XY}$  equals  $+1$  or  $-1$ , it can be shown that the points in the joint probability distribution that receive positive probability fall exactly along a straight line. Two random variables with nonzero correlation are said to be **correlated**. Similar to covariance, the correlation is a measure of the linear relationship between random variables.

# EXPERIMENT-6

---

AIM: To write a python program for correlation with scatter plot.

STUDY:

## ALGORITHM

Step 1: Start the Program

Step 2: Create variable y1, y2

Step 3: Create variable x, y3 using random function

Step 4: plot the scatter plot

Step 5: Print the result

Step 6: Stop the process

## Program:

```
# Scatterplot and Correlations
```

```
# Data
```

```
x=np.random.randn(100)
```

```
y1=x*5+9
```

```
y2=-5*x
```

```
y3=np.random.randn(100)
```

```
#Plot
```

```
plt.rcParams.update({'figure.figsize': (10,8), 'figure.dpi':100})
```

```
plt.scatter(x, y1, label=f'y1', Correlation = (np.round(np.corrcoef(x,y1)[0,1], 2)))
```

```
plt.scatter(x, y2, label=f'y2', Correlation = (np.round(np.corrcoef(x,y2)[0,1], 2)))
```

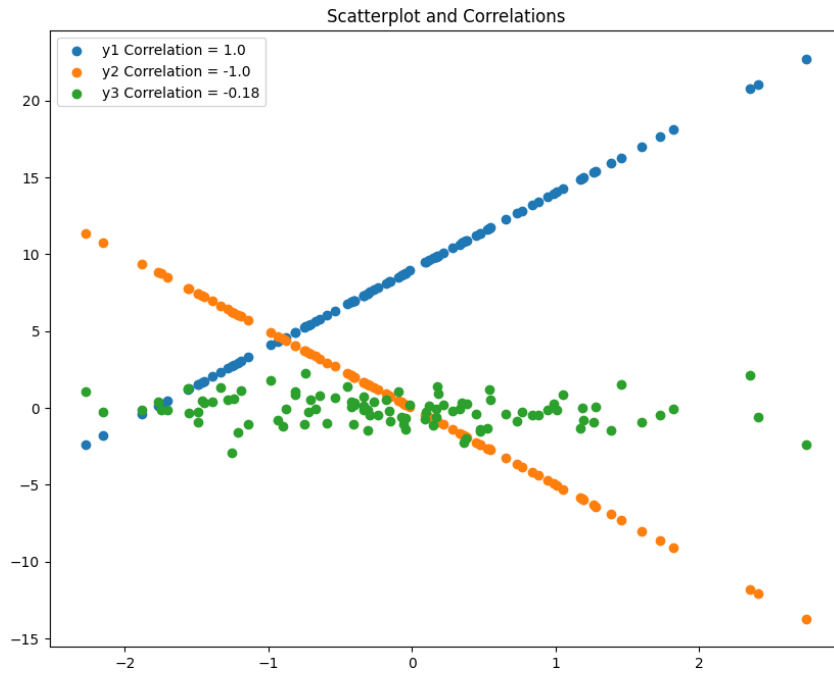
```
plt.scatter(x, y3, label=f'y3', Correlation = (np.round(np.corrcoef(x,y3)[0,1], 2)))
```

```
# Plot
```

```
plt title('Scatterplot and Correlations') plt(legend)
```

```
plt(show)
```

## Output



## Result:

Thus the Correlation and scatter plots using python program was successfully completed.

# Correlation Coefficient

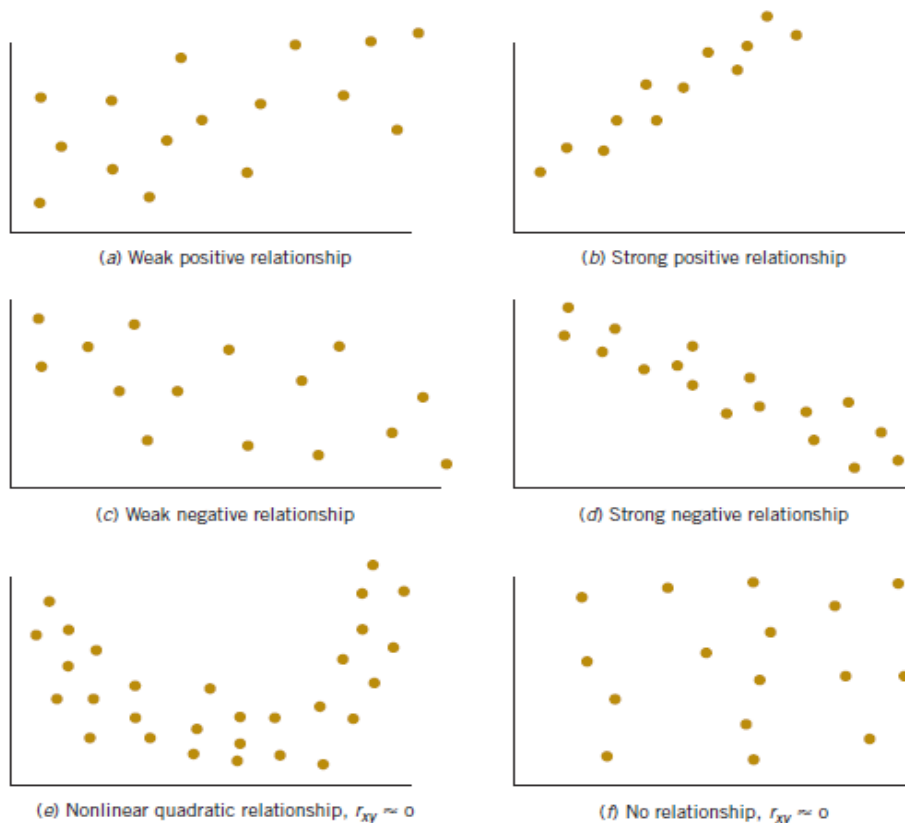
---

## UNDERSTANDING THE CONCEPT

The sample correlation coefficient  $r_{xy}$  is a quantitative measure of the strength of the linear relationship between two random variables  $x$  and  $y$ . The sample correlation coefficient is defined as

$$r_{xy} = \frac{\sum_{i=1}^n y_i(x_i - \bar{x})}{\left[ \sum_{i=1}^n (y_i - \bar{y})^2 \sum_{i=1}^n (x_i - \bar{x})^2 \right]^{1/2}}$$

If the two variables are perfectly linearly related with a positive slope  $r_{xy} = 1$  and if they are perfectly linearly related with a negative slope, then  $r_{xy} = -1$ . If no linear relationship between the two variables exists, then  $r_{xy} = 0$ . The simple correlation coefficient is also sometimes called the Pearson correlation coefficient after Karl Pearson, one of the giants of the fields of statistics in the late 19th and early 20th centuries. See below Figure for several examples of scatter diagrams exhibiting possible relationships between two variables. Parts (e) and (f) of the figure deserve special attention; in part (e), a probable quadratic relationship exists between  $y$  and  $x$ , but the sample correlation coefficient is close to zero because the correlation coefficient is a measure of linear association, but in part (f), the correlation is approximately zero because no association exists between the two variables.



# EXPERIMENT-7

---

AIM: To write a python program to compute correlation coefficient.

STUDY:

ALGORITHM

Step 1: Start the Program

Step 2: Import math package

Step 3: Define correlation coefficient function

Step 4: Calculate correlation using formula

Step 5: Print the result

Step 6: Stop the process

**Program:**

```
# Python Program to find correlation coefficient. import math
# function that returns correlation coefficient. def correlationCoefficient(X, Y, n) :
sum_X = 0
sum_Y = 0
sum_XY = 0
squareSum_X = 0
squareSum_Y = 0
i = 0
while i < n :
# sum of elements of array X.
sum_X = sum_X + X[i]
# sum of elements of array Y.
sum_Y = sum_Y + Y[i]
# sum of X[i] * Y[i].
sum_XY = sum_XY + X[i] * Y[i]
# sum of square of array elements. squareSum_X = squareSum_X + X[i] * X[i]
squareSum_Y = squareSum_Y + Y[i] * Y[i]
```

```
i = i + 1
```

```
# use formula for calculating correlation
# coefficient.
corr = (float)(n * sum_XY -
          sum_X * sum_Y)/
          (float)(math.sqrt((n *
          squareSum_X - sum_X *
          sum_X)* (n * squareSum_Y -
          sum_Y * sum_Y)))
return corr

# Driver function
X = [15, 18, 21, 24, 27]
Y = [25, 25, 27, 31, 32]

# Find the size of array.
n = len(X)

# Function call to correlation Coefficient.
print ('{0:.6f}'.format(correlationCoefficient(X, Y, n)))
```

**Output:**

```
0.953463
```

**Result:**

Thus the computation for correlation coefficient was successfully completed.

# SIMPLE LINEAR REGRESSION

---

## UNDERSTANDING THE CONCEPT

The case of simple linear regression considers a single regressor variable or predictor variable  $x$  and a dependent or response variable  $Y$ . Suppose that the true relationship between  $Y$  and  $x$  is a straight line and that the observation  $Y$  at each level of  $x$  is a random variable. As noted previously, the expected value of  $Y$  for each value of  $x$  is

$$E(Y|x) = \beta_0 + \beta_1 x$$

where the intercept  $\beta_0$  and the slope  $\beta_1$  are unknown regression coefficients. We assume that each observation,  $Y$ , can be described by the model

$$Y = \beta_0 + \beta_1 x + \epsilon$$

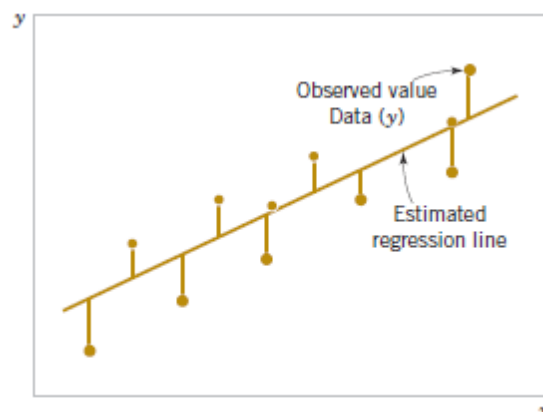
where  $\epsilon$  is a random error with mean zero and (unknown) variance  $\sigma^2$ . The random errors corresponding to different observations are also assumed to be uncorrelated random variables.

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad i = 1, 2, \dots, n$$

The least squares estimates of the intercept and slope in the simple linear regression model are

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$
$$\hat{\beta}_1 = \frac{\sum_{i=1}^n y_i x_i - \frac{\left(\sum_{i=1}^n y_i\right)\left(\sum_{i=1}^n x_i\right)}{n}}{\sum_{i=1}^n x_i^2 - \frac{\left(\sum_{i=1}^n x_i\right)^2}{n}}$$

where  $\bar{y} = (1/n)\sum_{i=1}^n y_i$  and  $\bar{x} = (1/n)\sum_{i=1}^n x_i$ .





# EXPERIMENT-8

---

AIM: To write a python program for Simple Linear Regression

STUDY:

ALGORITHM

Step 1: Start the Program

Step 2: Import numpy and matplotlib package

Step 3: Define coefficient function

Step 4: Calculate cross-deviation and deviation about x

Step 5: Calculate regression coefficients

Step 6: Plot the Linear regression and define main function

Step 7: Print the result

Step 8: Stop the process

**Program:**

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

def estimate_coef(x, y):
    # number of observations/points
    n = np.size(x)

    # mean of x and y vector
    m_x = np.mean(x)
    m_y = np.mean(y)

    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)
```

```
def plot_regression_line(x, y, b):
```

```
    # plotting the actual points as scatter plot
```

```
    plt.scatter(x, y, color = "m", marker = "o", s = 30)
```

```
    # predicted response vector
```

```
    y_pred = b[0] + b[1]*x
```

```
    # plotting the regression line
```

```
    plt.plot(x, y_pred, color = "g")
```

```
    # putting labels
```

```
    plt.xlabel('x')
```

```
    plt.ylabel('y')
```

```
    # function to show plot plt.show()
```

```
def main():
```

```
    # observations / data
```

```
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])
```

```
    # estimating coefficients
```

```
    b = estimate_coef(x, y)
```

```
    print("Estimated coefficients:\nb_0 = {} \nb_1 = {}".format(b[0], b[1]))
```

```
    # plotting regression line
```

```
    plot_regression_line(x, y, b)
```

```
if __name__ == "__main__": main()
```

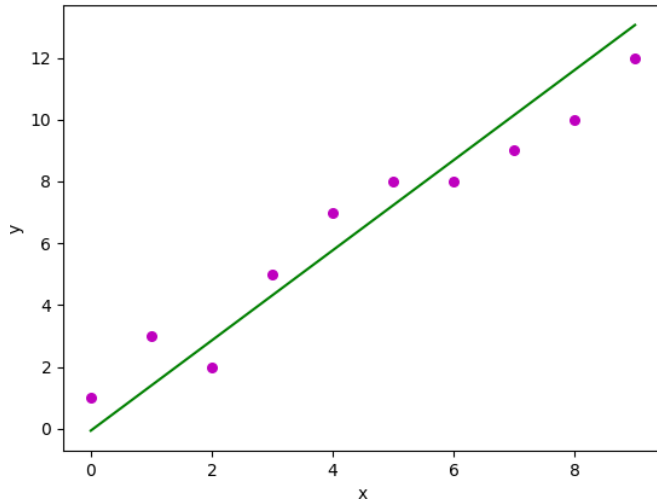
**Output :**

Estimated coefficients:

$$b_0 = -0.0586206896552$$

$$b_1 = 1.45747126437$$

**Graph:**



**Result:**

Thus the computation for Simple Linear Regression was successfully completed.

# Viva-Voice Questions and Answers

## 1. What is Data Science ?

**Ans.** Data Science is a field of computer science that explicitly deals with turning data into information and extracting meaningful insights out of it. The reason why Data Science is so popular is that the kind of insights it allows us to draw from the available data has led to some major innovations in several products and companies. Using these insights, we are able to determine the taste of a particular customer, the likelihood of a product succeeding in a particular market, etc.

## 2. Differentiate between Data Analytics and Data Science ?

**Ans.**

Data Analytics	Data Science
Data Analytics is a subset of Data Science.	Data Science is a broad technology that includes various subsets such as Data Analytics, Data Mining, Data Visualization, etc.
The goal of data analytics is to illustrate the precise details of retrieved insights.	The goal of data science is to discover meaningful insights from massive datasets and derive the best possible solutions to resolve business issues.
Requires just basic programming languages.	Requires knowledge in advanced programming languages.
It focuses on just finding the solutions.	Data Science not only focuses on finding the solutions but also predicts the future with past patterns or insights.
A data analyst's job is to analyse data in order to make decisions.	A data scientist's job is to provide insightful data visualizations from raw data that are easily understandable.

## 3. What do you understand about linear regression?

**Ans.** Linear regression helps in understanding the linear relationship between the dependent and the independent variables. Linear regression is a supervised learning algorithm, which helps in finding the linear relationship between two variables. One is the predictor or the independent variable and the other is the response or the dependent variable. In Linear Regression, we try to understand how the dependent variable changes w.r.t the independent variable. If there is only one independent variable, then it is called simple linear regression, and if there is more than one independent variable then it is known as multiple linear regression.

## 4. Why is R used in Data Visualization?

**Ans.** R provides the best ecosystem for data analysis and visualization with more than 12,000

packages in Open-source repositories. It has huge community support, which means you can easily find the solution to your problems on various platforms like StackOverflow. It has better data management and supports distributed computing by splitting the operations between multiple tasks and nodes, which eventually decreases the complexity and execution time of large datasets.

### **5. What is a normal distribution?**

**Ans.** Data distribution is a visualization tool to analyze how data is spread out or distributed. Data can be distributed in various ways. For instance, it could be with a bias to the left or the right, or it could all be jumbled up. Data may also be distributed around a central value, i.e., mean, median, etc. This kind of distribution has no bias either to the left or to the right and is in the form of a bell-shaped curve. This distribution also has its mean equal to the median. This kind of distribution is called a normal distribution.

### **6. What is variance in Data Science?**

**Ans.** Variance is a type of error that occurs in a Data Science model when the model ends up being too complex and learns features from data, along with the noise that exists in it. This kind of error can occur if the algorithm used to train the model has high complexity, even though the data and the underlying patterns and trends are quite easy to discover. This makes the model a very sensitive one that performs well on the training dataset but poorly on the testing dataset, and on any kind of data that the model has not yet seen. Variance generally leads to poor accuracy in testing and results in overfitting.

### **7. What are the popular libraries used in Data Science?**

**Ans.** Below are the popular libraries used for data extraction, cleaning, visualization, and deploying DS models:

- **TensorFlow:** Supports parallel computing with impeccable library management backed by Google.
- **SciPy:** Mainly used for solving differential equations, multidimensional programming, data manipulation, and visualization through graphs and charts.
- **Pandas:** Used to implement the ETL(Extracting, Transforming, and Loading the datasets) capabilities in business applications.
- **Matplotlib:** Being free and open-source, it can be used as a replacement for MATLAB, which results in better performance and low memory consumption.
- **PyTorch:** Best for projects which involve Machine Learning algorithms and Deep Neural Networks.

### **8. Between Python and R, which one you will choose for analyzing the text, and why?**

**Ans.** Due to the following factors, Python will outperform R for text analytics:

- Python's Pandas module provides high-performance data analysis capabilities as well as simple-to-use data structures.
- Python does all sorts of text analytics more quickly.

### **9. Explain the purpose of data cleaning.**

**Ans.** Data cleaning's primary goal is to rectify or eliminate inaccurate, corrupted, improperly formatted, duplicate, or incomplete data from a dataset. This often yields better outcomes and a higher return on investment for marketing and communications efforts.

### **10. Why is Python used for Data Cleaning in DS?**

**Ans.** Data Scientists have to clean and transform the huge data sets in a form that they can work with. It is important to deal with the redundant data for better results by removing nonsensical outliers, malformed records, missing values, inconsistent formatting, etc.

Python libraries such as Matplotlib, Pandas, Numpy, Keras, and SciPy are extensively used for Data cleaning and analysis. These libraries are used to load and clean the data and do effective analysis. For example, a CSV file named "Student" has information about the students of an institute like their names, standard, address, phone number, grades, marks, etc.



**List of Experiments  
beyond  
the Syllabus**

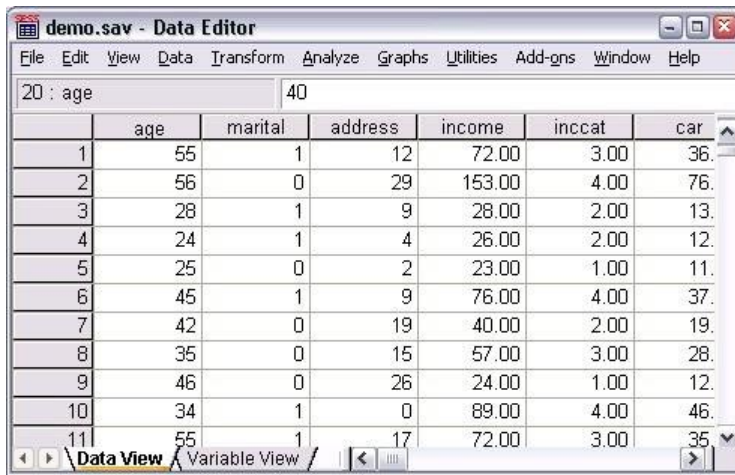


## Experiment No. 1

**Aim:** To read the data from SPSS statistics data file.

**Theory:** Data can be entered directly, or it can be imported from a number of different sources. The processes for reading data stored in IBM SPSS Statistics data files; spreadsheet applications, such as Microsoft Excel; database applications, such as Microsoft Access; and text files are all discussed in this chapter.

### Basic Structure of IBM SPSS Statistics Data Files



	age	marital	address	income	inccat	car
1	55	1	12	72.00	3.00	36.
2	56	0	29	153.00	4.00	76.
3	28	1	9	28.00	2.00	13.
4	24	1	4	26.00	2.00	12.
5	25	0	2	23.00	1.00	11.
6	45	1	9	76.00	4.00	37.
7	42	0	19	40.00	2.00	19.
8	35	0	15	57.00	3.00	28.
9	46	0	26	24.00	1.00	12.
10	34	1	0	89.00	4.00	46.
11	55	1	17	72.00	3.00	35.

Figure. Data Editor

IBM SPSS Statistics data files are organized by cases (rows) and variables (columns). In this data file, cases represent individual respondents to a survey. Variables represent responses to each question asked in the survey.

### Reading IBM SPSS Statistics Data Files

IBM SPSS Statistics data files, which have a *.sav* file extension, contain your saved data.

1. From the menus choose:

**File > Open > Data...**

2. Browse to and open *demo.sav*. See the topic [Chapter 10, "Sample Files,"](#) on page 73 for more information.

The data are now displayed in the Data Editor.

The screenshot shows the SPSS Data Editor window titled "demo.sav - Data Editor". The menu bar includes File, Edit, View, Data, Transform, Analyze, Graphs, Utilities, Add-ons, Window, and Help. The current view is "Data View". The data table is as follows:

	age	marital	address	income	inccat	car
1	55	1	12	72.00	3.00	36.
2	56	0	29	153.00	4.00	76.
3	28	1	9	28.00	2.00	13.
4	24	1	4	26.00	2.00	12.
5	25	0	2	23.00	1.00	11.
6	45	1	9	76.00	4.00	37.
7	42	0	19	40.00	2.00	19.
8	35	0	15	57.00	3.00	28.
9	46	0	26	24.00	1.00	12.
10	34	1	0	89.00	4.00	46.
11	55	1	17	72.00	3.00	35.

Figure. Opened data file

## Reading Excel Data

Rather than typing all of your data directly into the Data Editor, you can read data from applications such as Microsoft Excel. You can also read column headings as variable names.

1. From the menus choose:

**File > Import Data > Excel**

2. Go to the `Samples\English` folder and select `demo.xlsx`.

The **Read Excel File** dialog displays a preview of the data file. The contents of the first sheet in the file are displayed. If the file has multiple sheets, you can select the sheet from the list.

You can see that some of the string values for *Gender* have leading spaces. Some of the values for *Marital Status* are displayed as periods (.).

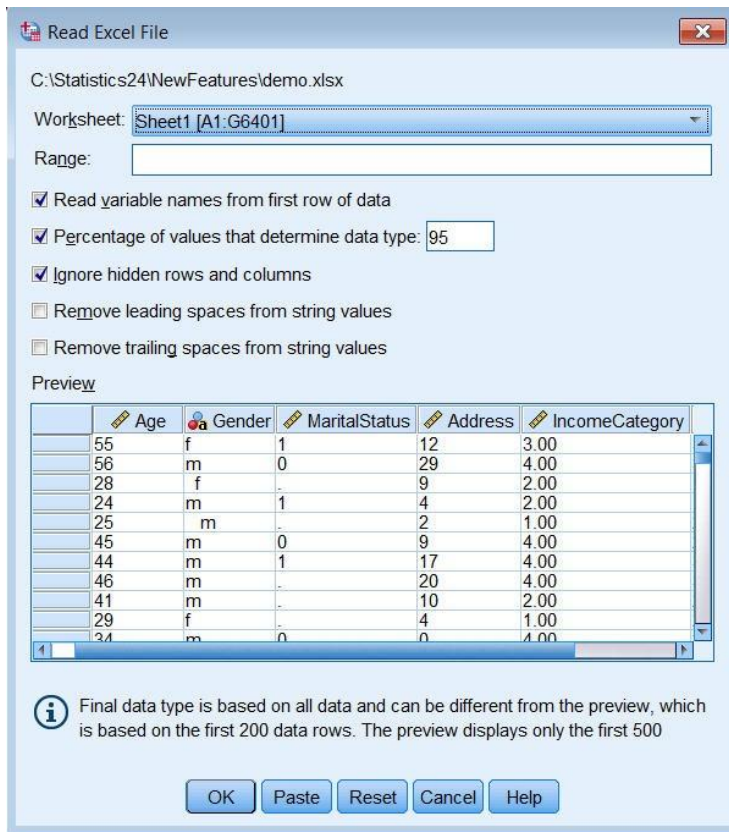
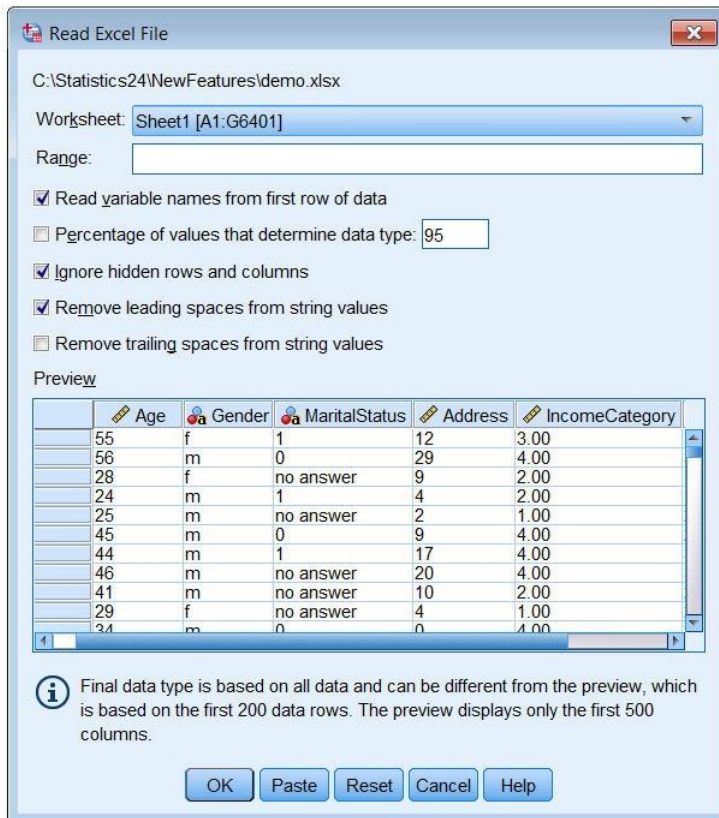


Figure. Read Excel File dialog

3. Make sure **Read variable names from the first row of data** is selected. If the column headings do not conform to variable name rules, they are converted to valid variable names. The original column headings are saved as variable labels.
4. Select **Remove leading spaces from string values**.
5. Deselect **Percentage of values that determine data type**.



The string value "no answer" is now displayed in the cells that were system-missing. If there is no percentage of values parameter and the column contains a mix of data type, the variable is read as a string data type. All values are preserved, but numeric values are treated as string values.

6. Select (check) **Percentage of values that determine data type** to treat *MaritalStatus* as a numeric variable.
7. Click **OK** to read the Excel file.

The data now appear in the Data Editor, with the column headings used as variable names. Since variable names can't contain spaces, the spaces from the original column headings are removed. For example, the column heading "Marital Status" is converted to the variable *MaritalStatus*. The original column heading is retained as a variable label.

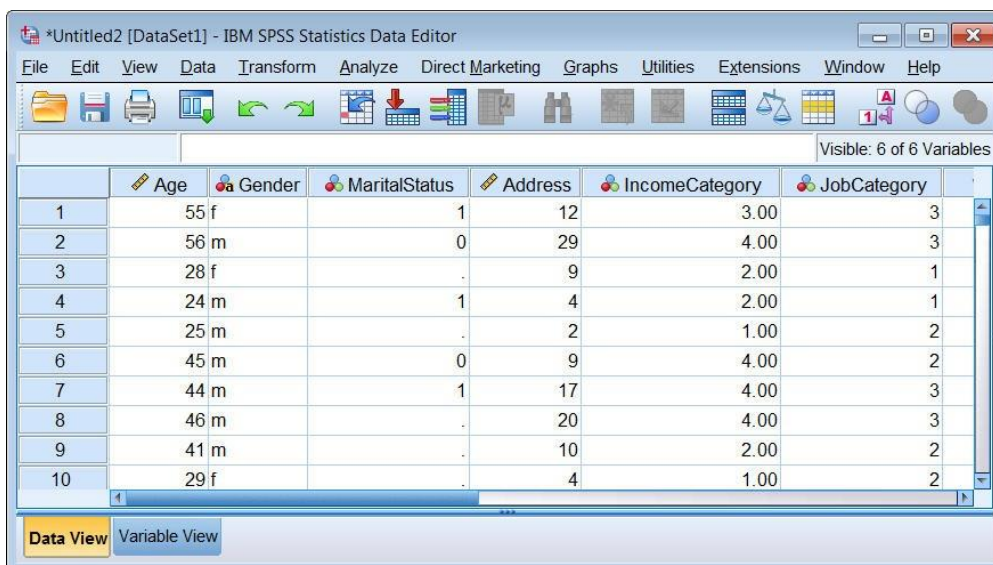


Figure. Imported Excel data

## Related information

[“Sample Files” on page 73](#)

## Reading Data from a Database

Data from database sources are easily imported using the Database Wizard. Any database that uses ODBC(Open Database Connectivity) drivers can be read directly after the drivers are installed. ODBC drivers for many database formats are supplied on the installation CD. Additional drivers can be obtained from third-party vendors. One of the most common database applications, Microsoft Access, is discussed in this example.

*Note:* This example is specific to Microsoft Windows and requires an ODBC driver for Access. The Microsoft Access ODBC driver only works with the 32-bit version of IBM SPSS Statistics. The steps are similar on other platforms but may require a third-party ODBC driver for Access.

1. From the menus choose:

**File > Import Data > Database > New Query...**

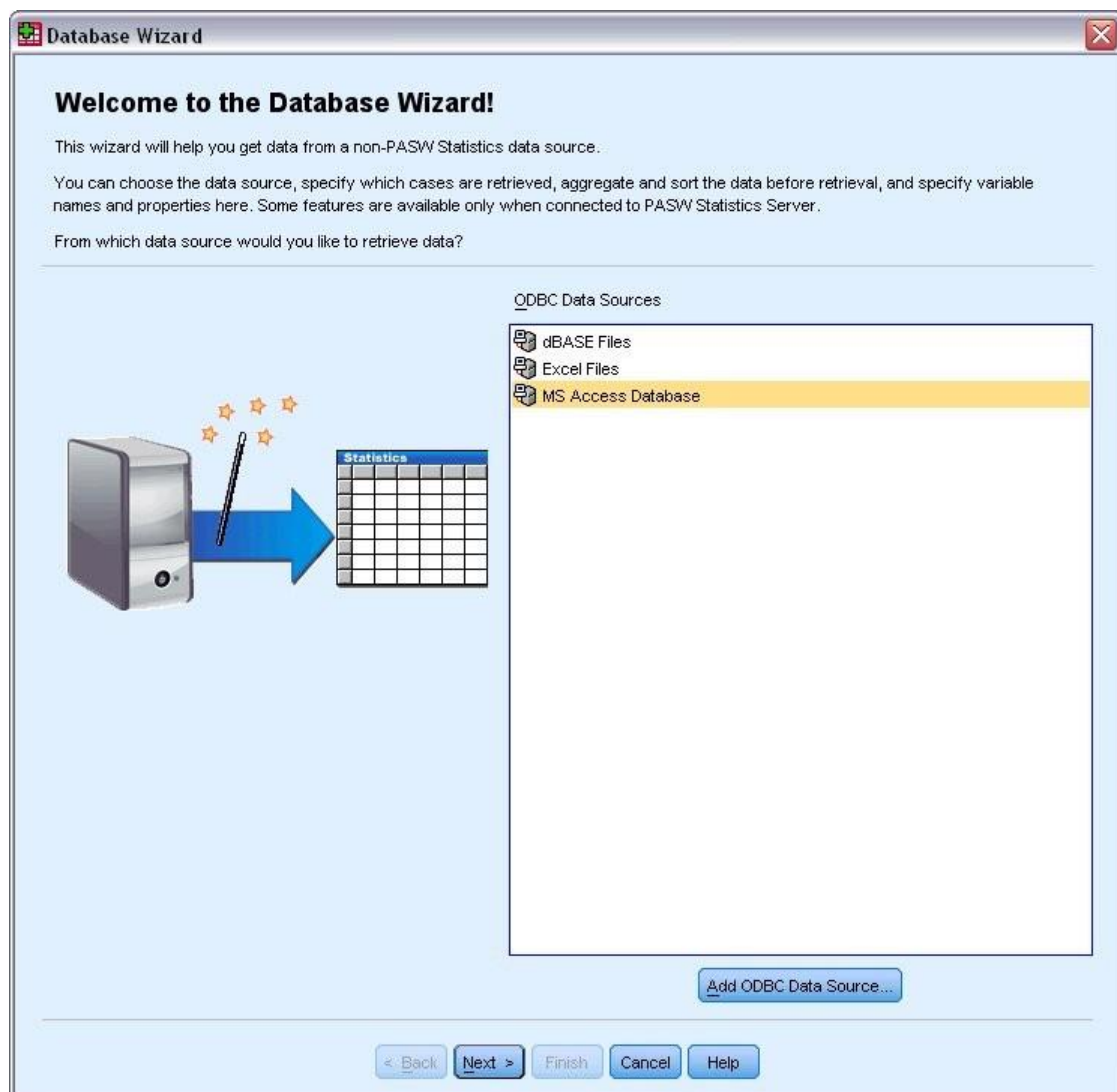


Figure. Database Wizard Welcome dialog box

2. Select **MS Access Database** from the list of data sources and click **Next**.

*Note:* Depending on your installation, you may also see a list of OLEDB data sources on the left side of the wizard (Windows operating systems only), but this example uses the list of ODBC data sources displayed on the right side.

3. Click **Browse** to navigate to the Access database file that you want to open.
4. Open *demo.mdb*. See the topic [Chapter 10, "Sample Files,"](#) on page 73 for more information.
5. Click **OK** in the login dialog box.

In the next step, you can specify the tables and variables that you want to import.

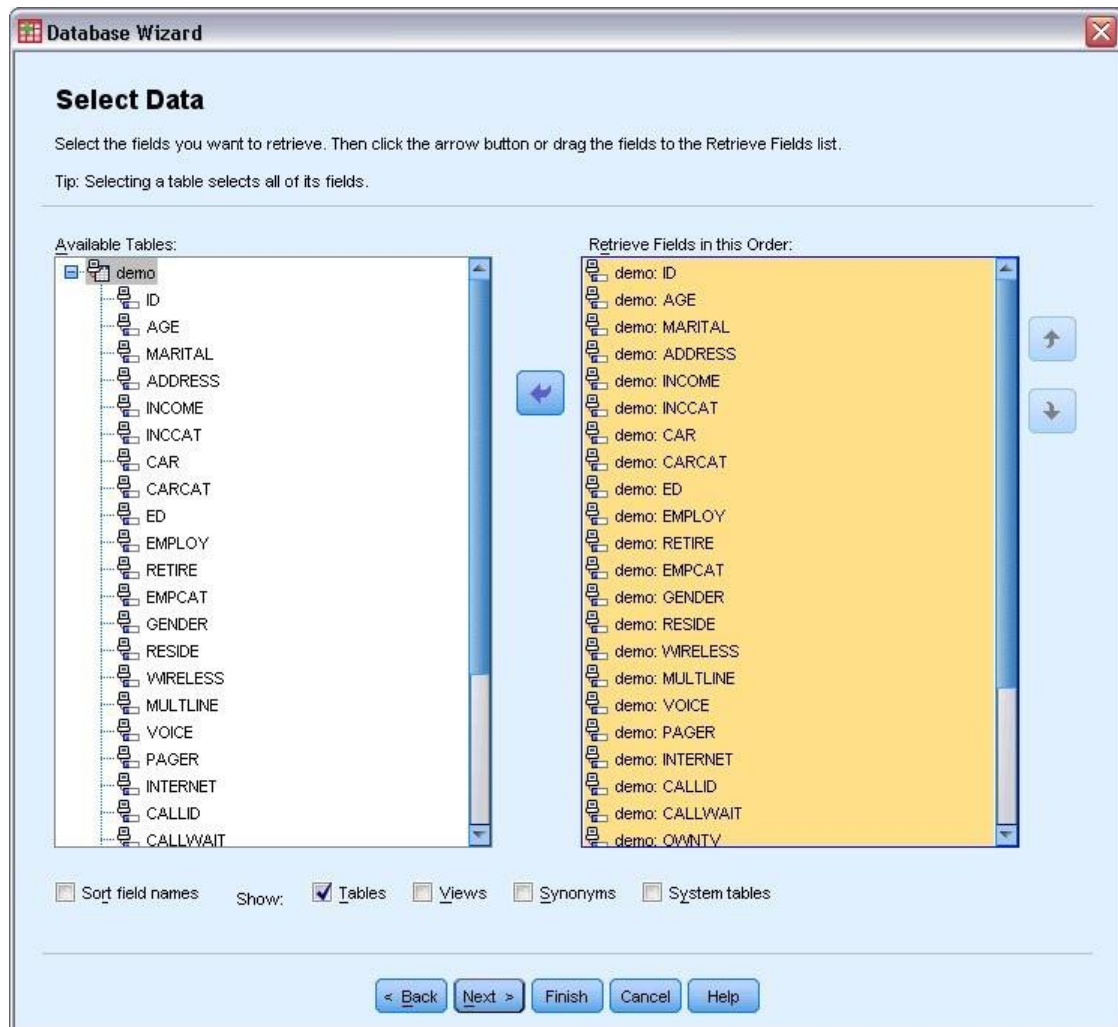


Figure. Select Data step

6. Drag the entire **demo** table to the Retrieve Fields In This Order list.
7. Click **Next**.

In the next step, you can select which records (cases) to import.

If you do not want to import all cases, you can import a subset of cases (for example, males older than 30), or you can import a random sample of cases from the data source. For large data sources, you may want to limit the number of cases to a small, representative sample to reduce the processing time.

8. Click **Next** to continue.

Field names are used to create variable names. If necessary, the names are converted to valid variable names. The original field names are preserved as variable labels. You can also change the variable names before importing the database.



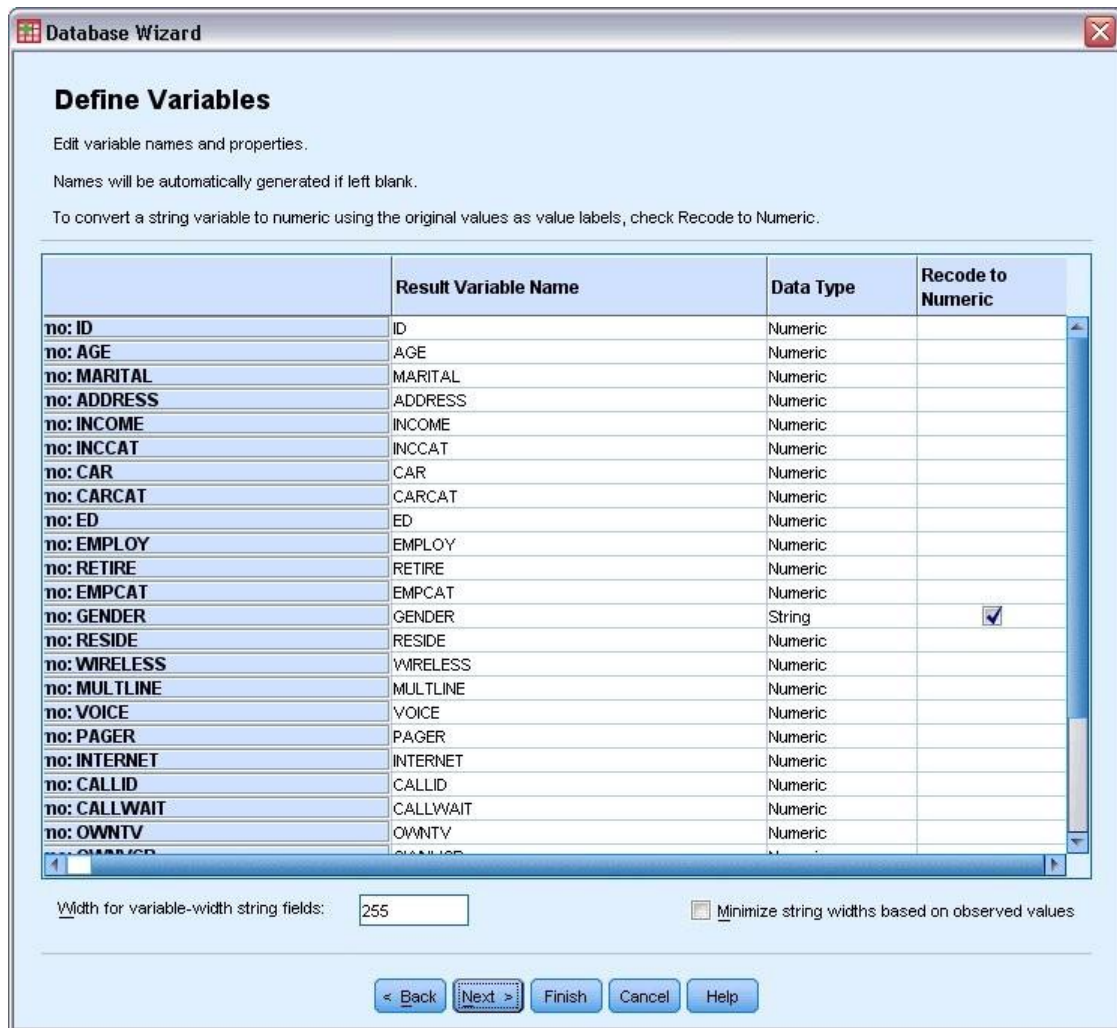


Figure. Define Variables step

9. Click the **Recode to Numeric** cell in the Gender field. This option converts string variables to integer variables and retains the original value as the value label for the new variable.
10. Click **Next** to continue.

The SQL statement created from your selections in the Database Wizard appears in the Results step. This statement can be executed now or saved to a file for later use.

11. Click **Finish** to import the data.

All of the data in the Access database that you selected to import are now available in the Data Editor.

### Reading Data from a Text File

Text files represent another common source of data. Many spreadsheet programs and databases can save their contents in one of many text file formats. Comma- or tab-delimited files refer to rows of data that use commas or tabs to indicate each variable. In this example, the data is tab-delimited.

1. From the menus choose:

**File > Import Data > Text Data**

2. Go to the `Samples\English` folder and select `demo.txt`.

The Text Import Wizard guides you through the process of defining how the specified text file is interpreted.

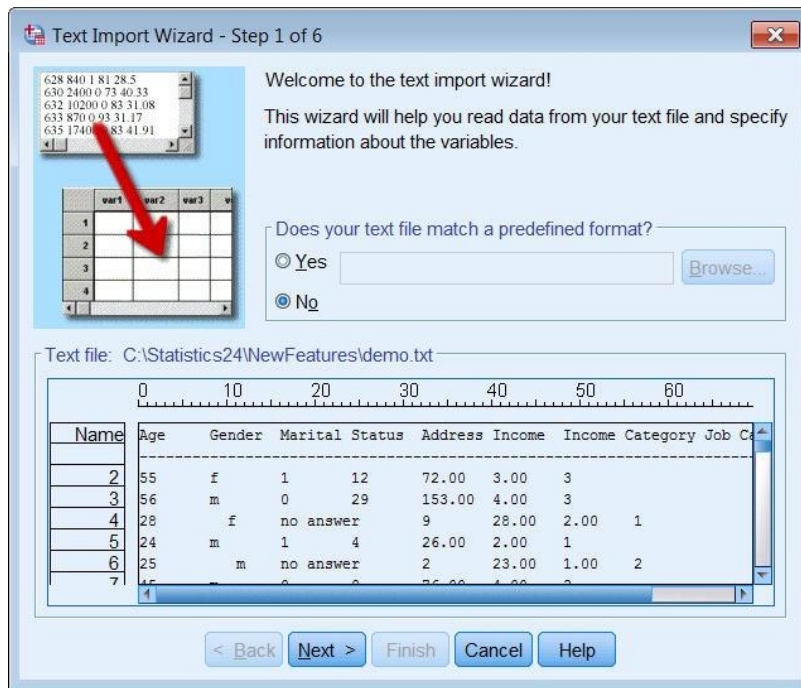


Figure. Text Import Wizard: Step 1 of 6

- In Step 1, you can choose a predefined format or create a new format in the wizard. Select **No**.
- Click **Next** to continue.

As stated earlier, this file uses tab-delimited formatting. Also, the variable names are defined on the top line of this file.

- In step 2 of the wizard, select **Delimited** to indicate that the file has a delimited formatting structure.
- Select **Yes** to indicate that the file includes variable names at the top of the file.
- Click **Next** to continue.
- In step 3, enter 2 for the line number where the first case of data begins (because variable names are on the first line).
- Keep the default values for the remainder of this step, and click **Next** to continue.

The Data preview in Step 4 provides you with a quick way to ensure that the file is read correctly

- Select **Tab** and deselect the other options for delimiters. **Space** is selected by default because the file contains spaces. For this file, spaces are part of the data values, not delimiters. You need to deselect **Space** to read the file correctly.
- Select **Remove leading spaces for string values**. Spaces at the start of string values affect how string values are evaluated in expressions. In this file, some values for *Gender* have leading spaces that are not part of the value. If you do not remove those spaces, a value of " f" is treated as a different value than "f".



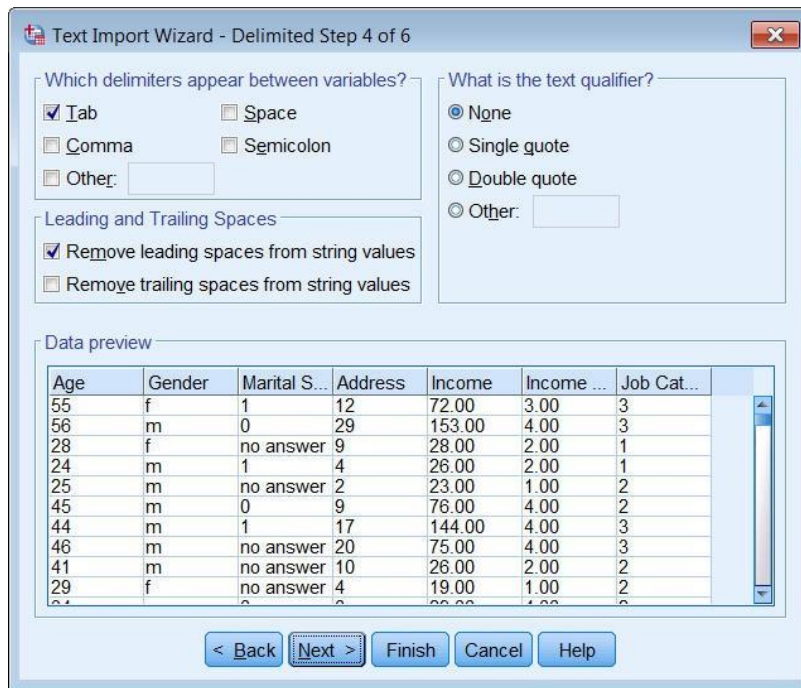


Figure. Text Import Wizard: Step 4 of 6

- Click **Next** to continue.

Because the variable names are **modified** to conform to naming rules, step 5 gives you the opportunity to edit any undesirable names.

Data types can be **defined** here as well. For example, you can change *Income* to dollar currency format.

To change a data type:

- In the **Data preview**, select *Income*.
- Select **Dollar** from the Data format drop-down list.

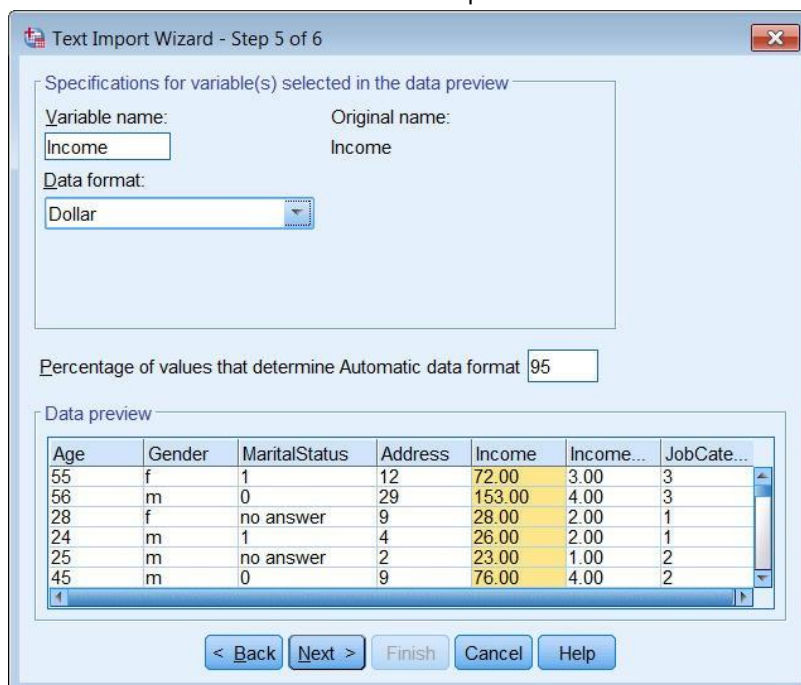


Figure. Change the data type

The variable *MaritalStatus* contains both string and numeric values. Less than five percent of the values are strings. With the default setting of 95% for **Percentage of values that determine the Automatic data format**, the variable is treated as numeric and the string values are set to system-missing. If no data format meets the percentage value, the variable is treated as a string variable. If you change the setting to 100, all values are preserved, but all numeric values are treated as strings.

15. Click **Next** to continue.
16. Leave the default selections in the last step, and click **Finish** to import the data.

## Experiment No. 2

**Aim:** To enter numeric and string data using data editor.

**Theory:** The Data Editor displays the contents of the active data file. The information in the Data Editor consists of variables and cases.

- In Data View, columns represent variables, and rows represent cases (observations).
- In Variable View, each row is a variable, and each column is an attribute that is associated with that variable.

Variables are used to represent the different types of data that you have compiled. A common analogy is that of a survey. The response to each question on a survey is equivalent to a variable. Variables come in many different types, including numbers, strings, currency, and dates.

### Entering Numeric Data

Data can be entered into the Data Editor, which may be useful for small data files or for making minor edits to larger data files.

1. Click the **Variable View** tab at the bottom of the Data Editor window.

You need to define the variables that will be used. In this case, only three variables are needed: *age*, *marital status*, and *income*.

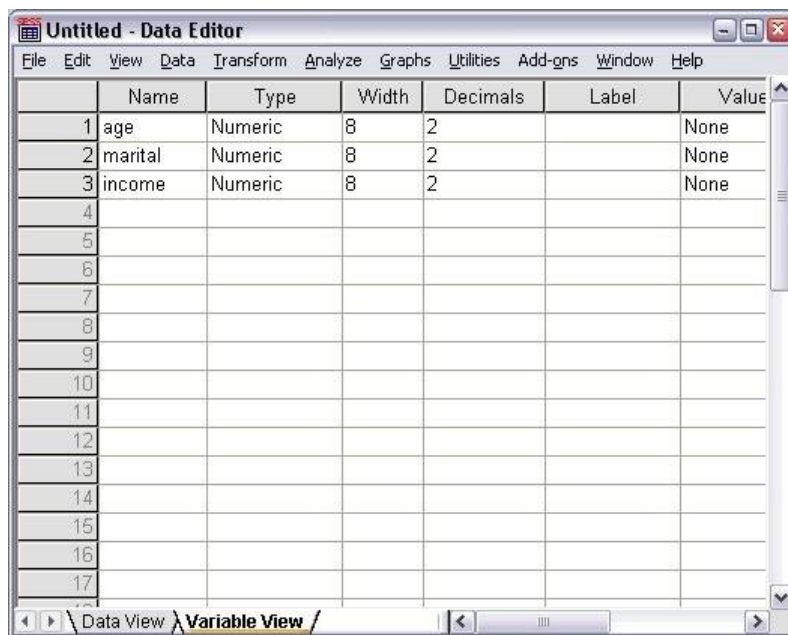


Figure. Variable names in Variable View

2. In the first row of the first column, type *age*.
3. In the second row, type *marital*.
4. In the third row, type *income*.

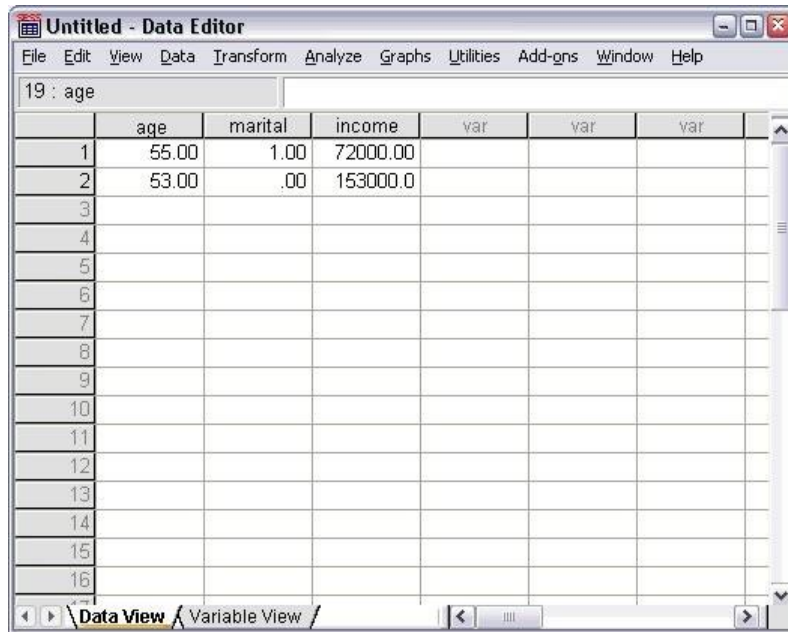
New variables are automatically given a Numeric data type.

If you don't enter variable names, unique names are automatically created. However, these names are not descriptive and are not recommended for large data files.

5. Click the **Data View** tab to continue entering the data.

The names that you entered in Variable View are now the headings for the first three columns in Data View.

Begin entering data in the first row, starting at the first column.



The screenshot shows the SPSS Data Editor window titled "Untitled - Data Editor". The menu bar includes File, Edit, View, Data, Transform, Analyze, Graphs, Utilities, Add-ons, Window, and Help. The variable list at the top shows "19 : age". The data grid has columns for "age", "marital", "income", and three "var" columns. The first two rows contain data: Row 1 has age=55.00, marital=1.00, income=72000.00; Row 2 has age=53.00, marital=.00, income=153000.0. The bottom of the window shows tabs for "Data View" (selected) and "Variable View".

	age	marital	income	var	var	var
1	55.00	1.00	72000.00			
2	53.00	.00	153000.0			
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						

Figure. Values entered in Data View

6. In the *age* column, type 55.
7. In the *marital* column, type 1.
8. In the *income* column, type 72000.
9. Move the cursor to the second row of the first column to add the next subject's data.
10. In the *age* column, type 53.
11. In the *marital* column, type 0.
12. In the *income* column, type 153000.

Currently, the *age* and *marital* columns display decimal points, even though their values are intended to be integers. To hide the decimal points in these variables:

13. Click the **Variable View** tab at the bottom of the Data Editor window.
14. In the *Decimals* column of the *age* row, type 0 to hide the decimal.
15. In the *Decimals* column of the *marital* row, type 0 to hide the decimal.

### Entering String Data

Non-numeric data, such as strings of text, can also be entered into the Data Editor.

1. Click the **Variable View** tab at the bottom of the Data Editor window.
2. In the first cell of the first empty row, type *sex* for the variable name.
3. Click the *Type* cell next to your entry.
4. Click the button on the right side of the *Type* cell to open the Variable Type dialog box.
5. Select **String** to specify the variable type.
6. Click **OK** to save your selection and return to the Data Editor.

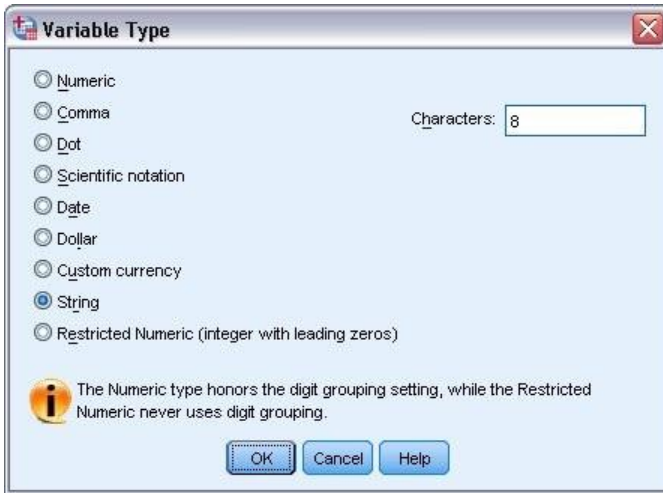


Figure. Variable Type dialog box

## Defining Data

In addition to defining data types, you can also define descriptive variable labels and value labels for variable names and data values. These descriptive labels are used in statistical reports and charts.

### Adding Variable Labels

Labels are meant to provide descriptions of variables. These descriptions are often longer versions of variable names. Labels can be up to 255 bytes. These labels are used in your output to identify the different variables.

1. Click the **Variable View** tab at the bottom of the Data Editor window.
2. In the *Label* column of the *age* row, type Respondent's Age.
3. In the *Label* column of the *marital* row, type Marital Status.
4. In the *Label* column of the *income* row, type Household Income.
5. In the *Label* column of the *sex* row, type Gender.

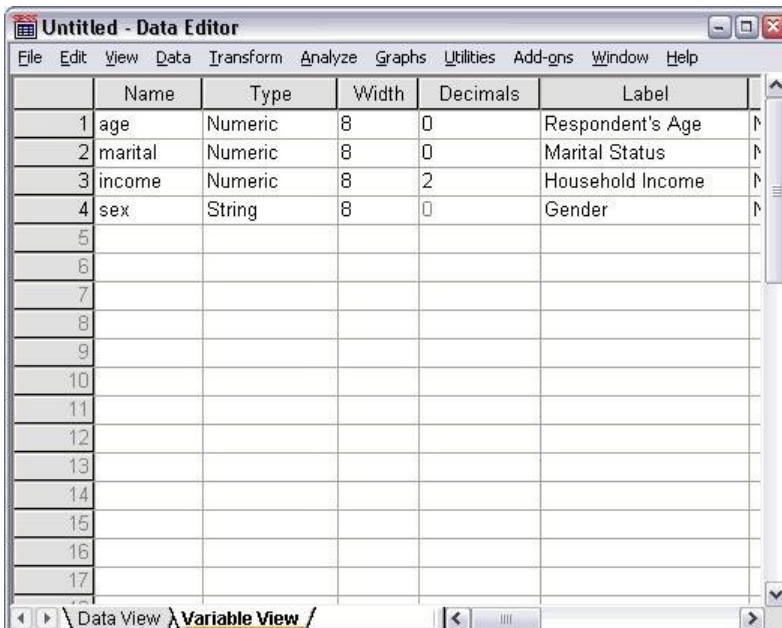


Figure. Variable labels entered in Variable View

## Changing Variable Type and Format

The *Type* column displays the current data type for each variable. The most common data types are numeric and string, but many other formats are supported. In the current data file, the *income* variable is defined as a numeric type.

1. Click the *Type* cell for the *income* row, and then click the button on the right side of the cell to open the Variable Type dialog box.
2. Select **Dollar**.

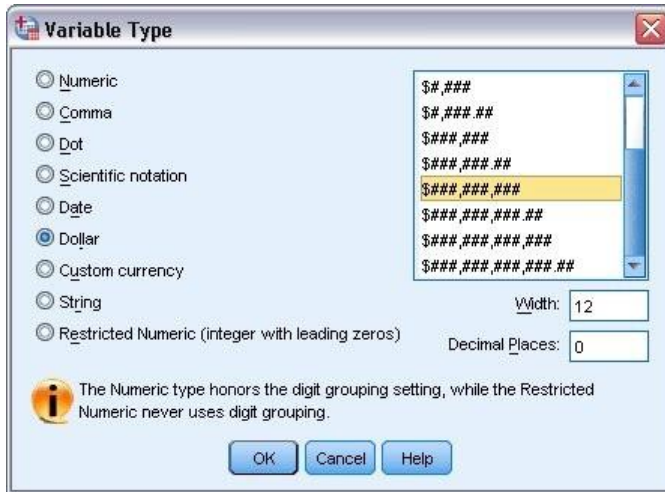


Figure. Variable Type dialog box

The formatting options for the currently selected data type are displayed.

3. For the format of the currency in this example, select **\$###,###,###**.
4. Click **OK** to save your changes.

## Adding Value Labels

Value labels provide a method for mapping your variable values to a string label. In this example, there are two acceptable values for the *marital* variable. A value of 0 means that the subject is single, and a value of 1 means that he or she is married.

1. Click the *Values* cell for the *marital* row, and then click the button on the right side of the cell to open the Value Labels dialog box.

The **value** is the actual numeric value.

The **value label** is the string label that is applied to the specified numeric value.

2. Type 0 in the Value field.
3. Type `single` in the Label field.
4. Click **Add** to add this label to the list.

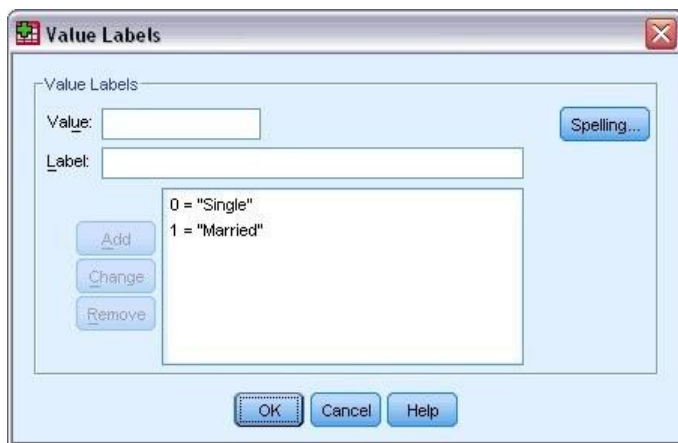


Figure. Value Labels dialog box

5. Type 1 in the Value field, and type `Married` in the Label field.
6. Click **Add**, and then click **OK** to save your changes and return to the Data Editor.

These labels can also be displayed in Data View, which can make your data more readable.

7. Click the **Data View** tab at the bottom of the Data Editor window.
8. From the menus choose:

#### **View > Value Labels**

The labels are now displayed in a list when you enter values in the Data Editor. This setup has the benefit of suggesting a valid response and providing a more descriptive answer.

If the Value Labels menu item is already active (with a check mark next to it), choosing **Value Labels** again will turn *off* the display of value labels.

### **Handling Missing Data**

Missing or invalid data are generally too common to ignore. Survey respondents may refuse to answer certain questions, may not know the answer, or may answer in an unexpected format. If you don't filter or identify these data, your analysis may not provide accurate results.

For numeric data, empty data fields or fields containing invalid entries are converted to system-missing, which is identifiable by a single period.

The reason a value is missing may be important to your analysis. For example, you may find it useful to distinguish between those respondents who refused to answer a question and those respondents who didn't answer a question because it was not applicable.

### **Missing Values for a Numeric Variable**

1. Click the **Variable View** tab at the bottom of the Data Editor window.
2. Click the *Missing* cell in the *age* row, and then click the button on the right side of the cell to open the Missing Values dialog box.

In this dialog box, you can specify up to three distinct missing values, or you can specify a range of values plus one additional discrete value.



Figure. Missing Values dialog box

3. Select **Discrete missing values**.
4. Type 999 in the first text box and leave the other two text boxes empty.
5. Click **OK** to save your changes and return to the Data Editor.

Now that the missing data value has been added, a label can be applied to that value.

6. Click the *Values* cell in the *age* row, and then click the button on the right side of the cell to open the Value Labels dialog box.
7. Type 999 in the Value field.
8. Type No Response in the Label field.
9. Click **Add** to add this label to your data file.
10. Click **OK** to save your changes and return to the Data Editor.

### Missing Values for a String Variable

Missing values for string variables are handled similarly to the missing values for numeric variables. However, unlike numeric variables, empty **fields** in string variables are not designated as system-missing. Rather, they are interpreted as an empty string.

1. Click the **Variable View** tab at the bottom of the Data Editor window.
2. Click the *Missing* cell in the *sex* row, and then click the button on the right side of the cell to open the Missing Values dialog box.
3. Select **Discrete missing values**.
4. Type NR in the first text box.

Missing values for string variables are case sensitive. So, a value of *nr* is not treated as a missing value.

5. Click **OK** to save your changes and return to the Data Editor.

Now you can add a label for the missing value.

6. Click the *Values* cell in the *sex* row, and then click the button on the right side of the cell to open the Value Labels dialog box.
7. Type NR in the Value field.
8. Type No Response in the Label field.
9. Click **Add** to add this label to your project.
10. Click **OK** to save your changes and return to the Data Editor.



## Experiment No. 3

**Aim:** To sort and select data in SPSS

**Theory:** Data files are not always organized in the ideal form for your specific needs. To prepare data for analysis, you can select from a wide range of file transformations, including the ability to:

- **Sort data.** You can sort cases based on the value of one or more variables.
- **Select subsets of cases.** You can restrict your analysis to a subset of cases or perform simultaneous analyses on different subsets.

The examples in this chapter use the data file *demo.sav*. See the topic [Chapter 10, “Sample Files,” on page 73](#) for more information.

### Sorting Data

Sorting cases (sorting rows of the data file) is often useful and sometimes necessary for certain types of analysis.

To reorder the sequence of cases in the data file based on the value of one or more sorting variables:

1. From the menus choose:

**Data > Sort Cases...**

The Sort Cases dialog box is displayed.

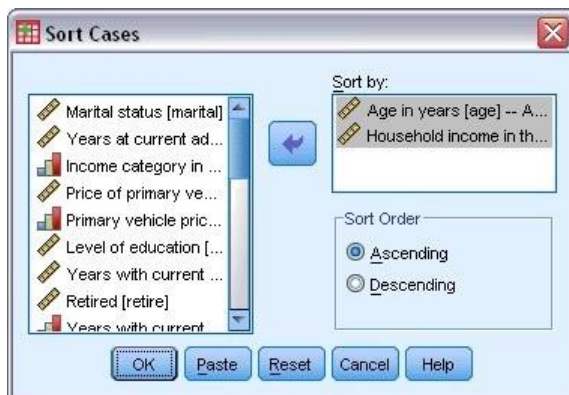


Figure. Sort Cases dialog box

2. Add the *Age in years [age]* and *Household income in thousands [income]* variables to the Sort by list.

If you select multiple sort variables, the order in which they appear on the Sort by list determines the order in which cases are sorted. In this example, based on the entries in the Sort by list, cases will be sorted by the value of *Household income in thousands [income]* within categories of *Age in years [age]*. For string variables, uppercase letters precede their lowercase counterparts in sort order (for example, the string value *Yes* comes before *yes* in the sort order).

### Split-File Processing

To split your data file into separate groups for analysis:

1. From the menus choose:

**Data > Split File...**

The Split File dialog box is displayed.

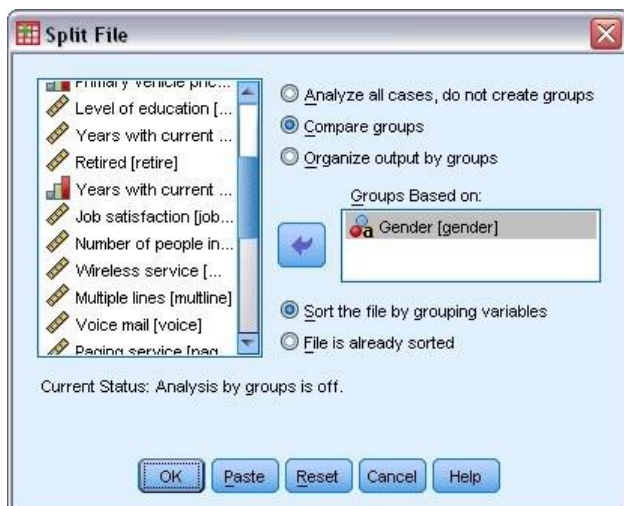


Figure. Split File dialog box

2. Select **Compare groups** or **Organize output by groups**. (The examples following these steps show the differences between these two options.)
3. Select *Gender [gender]* to split the file into separate groups for these variables.

You can use numeric, short string, and long string variables as grouping variables. A separate analysis is performed for each subgroup that is defined by the grouping variables. If you select multiple grouping variables, the order in which they appear on the Groups Based on list determines the manner in which cases are grouped.

If you select **Compare groups**, results from all split-file groups will be included in the same table(s), as shown in the following table of summary statistics that is generated by the Frequencies procedure.

**Statistics**

Household income in thousands

Female	N	Valid	3179
		Missing	0
	Mean		68.7798
	Median		44.0000
	Std. Deviation		75.73510
Male	N	Valid	3221
		Missing	0
	Mean		70.1608
	Median		45.0000
	Std. Deviation		81.56216

Figure. Split-file output with single pivot table

If you select **Organize output by groups** and run the Frequencies procedure, two pivot tables are created: one table for females and one table for males.

**Statistics<sup>a</sup>**

Household income in thousands

N	Valid	3179
	Missing	0
Mean		68.7798
Median		44.0000
Std. Deviation		75.73510

a. Gender = Female

Figure Split-file output with pivot table for females

**Statistics<sup>a</sup>**

Household income in thousands

N	Valid	3221
	Missing	0
Mean		70.1608
Median		45.0000
Std. Deviation		81.56216

a. Gender = Male

Figure. Split-file output with pivot table for males

## Sorting Cases for Split-File Processing

The Split File procedure creates a new subgroup each time it encounters a different value for one of the grouping variables. Therefore, it is important to sort cases based on the values of the grouping variables before invoking split-file processing.

By default, Split File automatically sorts the data file based on the values of the grouping variables. If the file is already sorted in the proper order, you can save processing time if you select **File is already sorted**.

## Turning Split-File Processing On and Off

After you invoke split-file processing, it remains in effect for the rest of the session unless you turn it off.

- **Analyze all cases.** This option turns split-file processing off.
- **Compare groups** and **Organize output by groups.** This option turns split-file processing on.

If split-file processing is in effect, the message **Split File on** appears on the status bar at the bottom of the application window.

## Selecting Subsets of Cases

You can restrict your analysis to a **specific** subgroup based on criteria that include variables and complex expressions. You can also select a random sample of cases. The criteria used to **define** a subgroup can include:

- Variable values and ranges
- Date and time ranges
- Case (row) numbers
- Arithmetic expressions
- Logical expressions
- Functions

To select a subset of cases for analysis:

1. From the menus choose:

**Data > Select Cases...**

This opens the Select Cases dialog box.

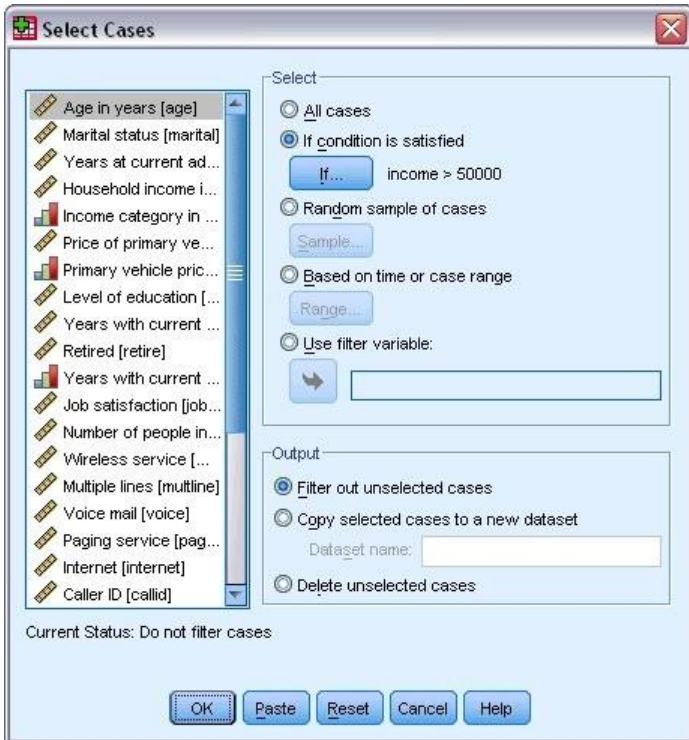
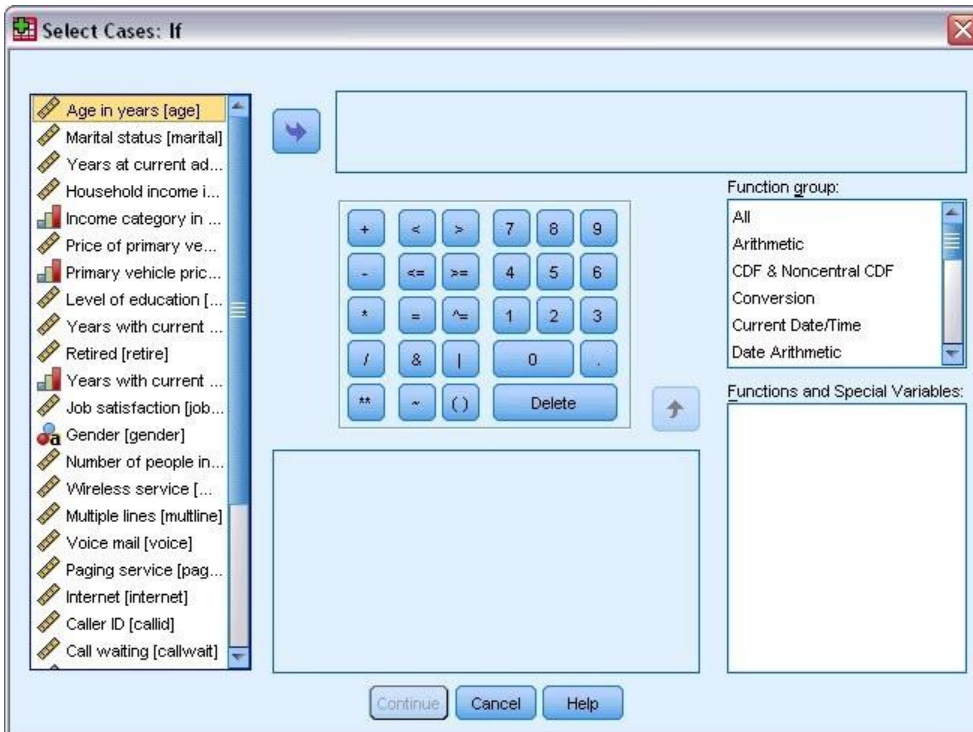


Figure. Select Cases dialog box

## Selecting Cases Based on Conditional Expressions

To select cases based on a conditional expression:

1. Select **If condition is satisfied** and click **If** in the Select Cases dialog box. This opens



the Select Cases If dialog box.

Figure. Select Cases If dialog box

The conditional expression can use existing variable names, constants, arithmetic operators, logical operators, relational operators, and functions. You can type and edit the expression in the text box just like text in an output window. You can also use the calculator pad, variable list, and function list to paste elements into the expression. See the topic [“Using Conditional Expressions”](#) on page 63 for more information.

## Selecting a Random Sample

To obtain a random sample:

1. Select **Random sample of cases** in the Select Cases dialog box.
2. Click **Sample**.

This opens the Select Cases Random Sample dialog box.

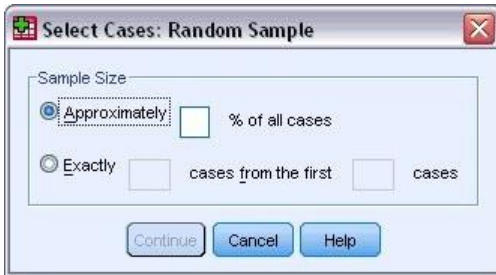


Figure. Select Cases Random Sample dialog box

You can select one of the following alternatives for sample size:

- **Approximately.** A user-specified percentage. This option generates a random sample of approximately the specified percentage of cases.
- **Exactly.** A user-specified number of cases. You must also specify the number of cases from which to generate the sample. This second number should be less than or equal to the total number of cases in the data file. If the number exceeds the total number of cases in the data file, the sample will contain proportionally fewer cases than the requested number.

## Selecting a Time Range or Case Range

To select a range of cases based on dates, times, or observation (row) numbers:

1. Select **Based on time or case range** and click **Range** in the Select Cases dialog box.

This opens the Select Cases Range dialog box, in which you can select a range of observation (row) numbers.

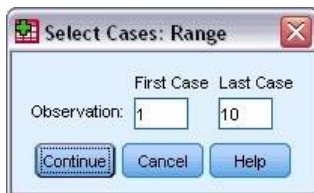


Figure. Select Cases Range dialog box

- **First Case.** Enter the starting date and/or time values for the range. If no date variables are defined, enter the starting observation number (row number in the Data Editor, unless Split File is on). If you do not specify a Last Case value, all cases from the starting date/time to the end of the time series are selected.
- **Last Case.** Enter the ending date and/or time values for the range. If no date variables are defined, enter the ending observation number (row number in the Data Editor, unless Split File is on). If you

do not specify a First Case value, all cases from the beginning of the time series up to the endingdate/time are selected.

For time series data with **defined** date variables, you can select a range of dates and/or times based on the **defined** date variables. Each case represents observations at a different time, and the file is sorted in chronological order.



Figure. Select Cases Range dialog box (time series)

To generate date variables for time series data:

2. From the menus choose:

**Data > Define Dates...**

## Treatment of Unselected Cases

You can choose one of the following alternatives for the treatment of unselected cases:

- **Filter out unselected cases.** Unselected cases are not included in the analysis but remain in the dataset. You can use the unselected cases later in the session if you turn filtering off. If you select a random sample or if you select cases based on a conditional expression, this generates a variablenamed *filter\_\$* with a value of 1 for selected cases and a value of 0 for unselected cases.
- **Copy selected cases to a new dataset.** Selected cases are copied to a new dataset, leaving the original dataset unaffected. Unselected cases are not included in the new dataset and are left in their original state in the original dataset.
- **Delete unselected cases.** Unselected cases are deleted from the dataset. Deleted cases can be recovered only by exiting from the file without saving any changes and then reopening the file. The deletion of cases is permanent if you save the changes to the data file.

*Note:* If you delete unselected cases and save the file, the cases cannot be recovered.

## Case Selection Status

If you have selected a subset of cases but have not discarded unselected cases, unselected cases are marked in the Data Editor with a diagonal line through the row number.

1
2
3
4
5
6
7
8
9
10
11

Figure. Case selection status