

# data-analyst-internship-task1-1

November 3, 2025

## 0.0.1 Summary of Changes

- Removed duplicate rows
- Filled missing Age values with mean
- Standardized Gender column to lowercase
- Renamed columns to lowercase and underscores
- Converted data types for Age, Income, and Score
- Exported final cleaned dataset

```
[1]: #Loading the data
import pandas as pd
file_path = "D:\\intern\\Mall_Customers.csv"
df=pd.read_csv(file_path)
print(df.head())
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
[2]: #Checking null values
df.isnull().sum()
```

```
[2]: CustomerID          0
Gender                0
Age                  0
Annual Income (k$)    0
Spending Score (1-100) 0
dtype: int64
```

```
[3]: #remove duplicate data
df.drop_duplicates(inplace=True)
print("ok")
```

ok

```
[4]: #Standardizing
df['Gender'] = df['Gender'].str.lower().str.strip()
print("ok")
```

ok

```
[5]: #Data checking
print(df.columns)
```

```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
      'Spending Score (1-100)'],
      dtype='object')
```

```
[7]: # Handle missing values
df['Age'] = df['Age'].fillna(df['Age'].mean())

# Rename columns
df.columns = df.columns.str.lower().str.replace(' ', '_')
print("ok")
```

ok

```
[8]: #Date fixing
print(df.columns)
```

```
Index(['customerid', 'gender', 'age', 'annual_income_(k$)',
      'spending_score_(1-100)'],
      dtype='object')
```

```
[9]: #Verifying Data Type
df.dtypes
```

```
[9]: customerid          int64
gender                object
age                  int64
annual_income_(k$)    int64
spending_score_(1-100) int64
dtype: object
```

```
[14]: import warnings
warnings.filterwarnings("ignore") # Suppress harmless warnings
```

```

# Check for date-like columns
for col in df.select_dtypes(include=['object']).columns:
    try:
        converted = pd.to_datetime(df[col], errors='coerce',
        infer_datetime_format=True)
        if converted.notna().sum() / len(df) > 0.8:
            df[col] = converted
            print(f"Converted '{col}' to datetime format.")
    except Exception:
        pass

# Final check
date_cols = df.select_dtypes(include=['datetime64[ns]']).columns
if len(date_cols) == 0:
    print("No valid date columns found - date conversion not applicable for
    this dataset.")
else:
    print(f"Date columns converted: {list(date_cols)}")

```

No valid date columns found - date conversion not applicable for this dataset.

```

[15]: #Downloading the data
df.to_csv('cleanned_dataset.csv', index = False)
print("sucesfully downloaded")

```

sucesfully downloaded

```
[ ]:
```