🧹 Part A — Data Cleaning Solution (Using Pandas & NumPy Only)

```
# ------------------------------
# PART A – DATA CLEANING
# ------------------------------

import pandas as pd
import numpy as np

# Q1 – Load dataset and basic info
df = pd.read_csv("unclean_sales_data_100.csv")

print("First 10 rows:\n", df.head(10))
print("\nLast 10 rows:\n", df.tail(10))
print("\nShape:", df.shape)
print("\nData types:\n")
df.info()
```

Q2 — Missing values & duplicates

```
print("\nMissing values per column:\n", df.isna().sum())
print("\nDuplicate records:", df.duplicated().sum())

# Drop exact duplicates
df.drop_duplicates(inplace=True)
print("After removing duplicates:", df.shape)
```

Q3 — Clean Customer_Name & Gender

```
# Remove spaces & normalize names
df['Customer_Name'] =
df['Customer_Name'].astype(str).str.strip().str.title()

# Clean gender column
df['Gender'] = df['Gender'].astype(str).str.strip().str.title()
```

```
# Standardize gender values
gender_map = {
    'M': 'Male', 'F': 'Female', 'Femle': 'Female', 'Femlae': 'Female',
    'Femael': 'Female', 'FeMale': 'Female', 'Male ': 'Male', '':
np.nan
}
df['Gender'] = df['Gender'].replace(gender_map)
df['Gender'] = df['Gender'].replace({'Femael': 'Female'})
df['Gender'].fillna('Unknown', inplace=True)
```

Q4 — Clean Age column

```
# Convert textual ages to numbers
text_age_map = {'twenty': 20, 'thirty': 30, 'forty': 40}
df['Age'] = df['Age'].replace(text_age_map)

# Convert to numeric and handle errors
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# Remove negative or unrealistic ages
df.loc[(df['Age'] < 0) | (df['Age'] > 100), 'Age'] = np.nan

# Fill missing with median
df['Age'].fillna(df['Age'].median(), inplace=True)
```

Q5 — Clean Product_Category

```
df['Product_Category'] = (
    df['Product_Category'].astype(str)
    .str.strip()
    .str.title()
)

category_corrections = {
    'Electornics': 'Electronics', 'Elecronics': 'Electronics',
    'Eletronics': 'Electronics', 'Grocey': 'Groceries',
```

```
    'Clothng': 'Clothing', 'Cloth': 'Clothing', 'Furnitre':
'Furniture',
    'Homedecor': 'Home Decor'
}
df['Product_Category'] =
df['Product_Category'].replace(category_corrections)
df['Product_Category'].fillna('Other', inplace=True)
```

Q6 — Clean Quantity & Price

```
# Convert Quantity textual values to numbers
quantity_map = {'one': 1, 'two': 2, 'Three': 3, 'four': 4}
df['Quantity'] = df['Quantity'].replace(quantity_map)
df['Quantity'] = pd.to_numeric(df['Quantity'], errors='coerce')

# Convert Price to numeric
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')

# Handle invalid values
df.loc[df['Price'] <= 0, 'Price'] = np.nan
df['Quantity'].fillna(df['Quantity'].median(), inplace=True)
df['Price'].fillna(df['Price'].median(), inplace=True)
```

Q7 — Recalculate Total

```
df['Total'] = df['Quantity'] * df['Price']
```

Q8 — Clean Order_Date

```
df['Order_Date'] = (
    df['Order_Date'].astype(str).str.strip().replace('invalid',
np.nan)
)

# Convert to datetime, handle errors
df['Order_Date'] = pd.to_datetime(df['Order_Date'], errors='coerce')
```

```
# Replace missing with median date
df['Order_Date'].fillna(df['Order_Date'].median(), inplace=True)
```

Q9 — Clean Payment_Mode

```
df['Payment_Mode'] = (
    df['Payment_Mode'].astype(str)
    .str.strip()
    .str.title()
    .replace({
        'Credit  Card': 'Credit Card', 'Debit': 'Debit Card',
        'Upi': 'UPI', '': np.nan
    })
)
df['Payment_Mode'].fillna('Unknown', inplace=True)
```

Q10 — Final check & save cleaned dataset

```
print("\nCleaned Data Info:")
df.info()

print("\nDescriptive Summary:")
print(df.describe(include='all'))

# Show number of changes
print("\nFinal Shape:", df.shape)

# Save cleaned file
df.to_csv("cleaned_sales_data.csv", index=False)
print("\n✅ Cleaned dataset saved as cleaned_sales_data.csv")
```

Part B — Code Answers (Step-by-Step Using Pandas & NumPy)

🔹 Setup

```
import pandas as pd
import numpy as np

# Load cleaned dataset
df = pd.read_csv("cleaned_sales_data.csv")

# Quick check
print(df.info())
print(df.head())
```

🔹 Q1. Basic Statistics

```
# Total records and columns
print("Total Records:", df.shape[0])
print("Total Columns:", df.shape[1])

# Unique values
print("Unique Customers:", df["Customer_Name"].nunique())
print("Unique Product Categories:", df["Product_Category"].nunique())
```

🔹 Q2. Customer Demographics

```
# Basic age stats
print("Average Age:", df["Age"].mean())
print("Min Age:", df["Age"].min())
print("Max Age:", df["Age"].max())

# Gender counts
print(df["Gender"].value_counts())

# Average age by gender
print(df.groupby("Gender")["Age"].mean())
```

🔹 Q3. Sales Performance

```
# Total revenue
print("Total Sales Revenue:", df["Total"].sum())

# Average / highest / lowest order value
print("Average Order Value:", df["Total"].mean())
print("Highest Order Value:", df["Total"].max())
print("Lowest Order Value:", df["Total"].min())

# Check for invalid totals before cleaning (optional if you kept a
copy)
# print("Invalid Total count:", df["Total"].isna().sum())
```

🔹 Q4. Top-Performing Categories

```
# Group by category
category_summary = df.groupby("Product_Category").agg(
    Total_Sales=("Total", "sum"),
    Orders=("Total", "count"),
    Avg_Price=("Price", "mean")
).sort_values(by="Total_Sales", ascending=False)
print(category_summary)
```

🔹 Q5. Payment Preferences

```
# Frequency and revenue share by payment mode
payment_summary = df.groupby("Payment_Mode").agg(
    Frequency=("Total", "count"),
    Revenue=("Total", "sum")
).sort_values(by="Revenue", ascending=False)

payment_summary["Revenue_Share_%"] = (payment_summary["Revenue"] /
df["Total"].sum() * 100).round(2)
print(payment_summary)
```

🔹 Q6. Temporal (Order Date Analysis)

```
# Ensure Order_Date is datetime
df["Order_Date"] = pd.to_datetime(df["Order_Date"], errors="coerce")

print("Earliest Order Date:", df["Order_Date"].min())
print("Latest Order Date:", df["Order_Date"].max())

# Orders per month
print(df["Order_Date"].dt.month.value_counts().sort_index())

# Average daily sales
daily_sales = df.groupby(df["Order_Date"].dt.date)["Total"].sum()
print("Average Daily Sales:", daily_sales.mean())
```

🔹 Q7. Correlations & Outliers

```
# Correlation
print(df[["Quantity", "Price", "Total"]].corr())

# Outlier detection
mean_total = df["Total"].mean()
std_total = df["Total"].std()
outliers = df[df["Total"] > mean_total + 2 * std_total]
print("Outlier Count:", len(outliers))

# Percentage of cleaned records (if original file is available)
# original_len = len(df_original)
# cleaned_len = len(df)
# print("Cleaned Percentage:", (original_len - cleaned_len) /
original_len * 100)
```

🔹 Q8. Text-Based Summary (Programmatic)

You can print a mini-summary like this:

```
summary = f"""
Clean dataset contains {df.shape[0]} records.
Total revenue: ₹{df['Total'].sum():,.0f}
Top Category: {df.groupby('Product_Category')['Total'].sum().idxmax()}
Most Used Payment Mode: {df['Payment_Mode'].mode()[0]}
Average Customer Age: {df['Age'].mean():.1f} years
Strong positive correlation between Quantity, Price, and Total.
"""
print(summary)
```