Below process explains about how data is loaded for data profiling and is focused on data quality, reliability, accuracy, consistency, and identifying any gaps along with suggestions for fixing them,

Step 1: Data Profiling with a Focus on Quality, Reliability, Accuracy, and Consistency

1. Load the Data:

```
  import pandas as pd

# Load datasets
Data_customer = pd.read_csv('customers.csv')
Data_order = pd.read_csv('orders.csv')
data_shipping = pd.read_csv('shipping.csv')


  # Load the data into DataFrames
  df_customer = pd.DataFrame(data_customer)
  df_order = pd.DataFrame(data_order)
  df_shipping = pd.DataFrame(data_shipping)
```

2. Check for Missing Values (Nulls):

Identifying missing values is crucial to assess data quality and completeness.

```
# Check for missing values
  print("Missing values in Customer data:")
  print(df_customer.isnull().sum())
  print("\nMissing values in Order data:")
  print(df_order.isnull().sum())
  print("\nMissing values in Shipping data:")
```

```
print(df_shipping.isnull().sum())
```

3. Check for Duplicates:

Duplicate records can affect the reliability and accuracy of analysis.

```
# Check for duplicates
print("\nDuplicate records in Customer data:", df_customer.duplicated().sum())
print("Duplicate records in Order data:", df_order.duplicated().sum())
print("Duplicate records in Shipping data:", df_shipping.duplicated().sum())
```

4. Check for Consistency in Data Relationships:

This involves ensuring that keys match across related tables.

```
# Consistency check between Customer and Order tables
customer_ids_in_orders = df_order['Customer_ID'].isin(df_customer['Customer_ID'])
print("\nOrders with invalid Customer_IDs:", df_order[~customer_ids_in_orders])

# Consistency check between Customer and Shipping tables
customer_ids_in_shipping = df_shipping['Customer_ID'].isin(df_customer['Customer_ID'])
print("Shippings with invalid Customer_IDs:", df_shipping[~customer_ids_in_shipping])
```

5. Data Quality Metrics:
You can calculate simple metrics such as:
- Percentage of missing values
- Ratio of invalid relationships (e.g., orders without corresponding customers)

```
# Calculate percentages
missing_customer_values = df_customer.isnull().mean() * 100
missing_order_values = df_order.isnull().mean() * 100
missing_shipping_values = df_shipping.isnull().mean() * 100

print("\nPercentage of missing values in Customer data:")
print(missing_customer_values)

print("\nPercentage of missing values in Order data:")
print(missing_order_values)

print("\nPercentage of missing values in Shipping data:")
print(missing_shipping_values)


# Ratio of invalid relationships
invalid_orders_ratio = (~customer_ids_in_orders).mean() * 100
invalid_shipping_ratio = (~customer_ids_in_shipping).mean() * 100
```

```
print(f"\nPercentage of invalid Customer_IDs in Order data: {invalid_orders_ratio:.2f}%")
print(f"Percentage of invalid Customer_IDs in Shipping data: {invalid_shipping_ratio:.2f}%")
```

Step 2: Assessing Reliability, Accuracy, and Consistency

1. Reliability and Accuracy:
   Reliability can be assessed by looking at the data entry errors, outliers, and potential mismatches.

```
# Check for outliers in Age (Customer Data)
print("\nOutliers in Age:")
print(df_customer[df_customer['Age'] < 0])  # Negative ages would be an error

# Check for possible inconsistencies in customer names
print("\nInconsistent names (e.g., special characters):")
print(df_customer[df_customer['First'].str.contains(r'[!@#$%^&*(),.?":{}|<>]')])
```

2. Consistency of Data:
   Check whether data is consistently formatted and logically sound.

```
# Check for consistency in Country (should be 'USA')
print("\nInconsistent countries in Customer data:")
print(df_customer[~df_customer['Country'].str.match("USA")])
```

```
# Check consistency in Status (Shipping data should only contain 'Pending')
print("\nInconsistent statuses in Shipping data:")
print(df_shipping[~df_shipping['Status'].str.match("Pending")])
```

Step 3: Identifying Gaps and Fixing Them

Based on the checks, you can identify potential gaps and suggest fixes:

1. Fixing Missing Data:
   - Approach: Depending on the significance, you can fill missing values with defaults, averages, or drop the records if they are irrelevant.

```
# Fill missing names with a placeholder
df_customer['First'].fillna('Unknown', inplace=True)
```

2. Fixing Duplicates:
   - Approach: Remove duplicate rows to ensure each entity is uniquely represented.

```
df_customer.drop_duplicates(inplace=True)
```

3. Fixing Inconsistent IDs:
   - Approach: Align IDs across datasets or flag invalid entries for review.

```
# Remove orders with invalid Customer_IDs
df_order_cleaned = df_order[customer_ids_in_orders]
```

4. Fixing Data Entry Errors:
  - Approach: Correct obvious typos, normalize inconsistent entries, or flag for manual review.
    # Replace special characters in names
    df_customer['First'] = df_customer['First'].str.replace(r'[!@#$%^&*(),.?":{}|<>]', '', regex=True)

Step 4: *Generate a Report on Data Quality*

Finally, summarize your findings and corrections in a report format:

report = f"""

Data Quality Report:

1. Missing Values:
- Customer Data: {missing_customer_values[missing_customer_values > 0].to_dict()}
- Order Data: {missing_order_values[missing_order_values > 0].to_dict()}
- Shipping Data: {missing_shipping_values[missing_shipping_values > 0].to_dict()}

2. Duplicates:
- Customer Data: {df_customer.duplicated().sum()} duplicates found and removed.
- Order Data: {df_order.duplicated().sum()} duplicates found and removed.
- Shipping Data: {df_shipping.duplicated().sum()} duplicates found and removed.

3. Consistency and Accuracy:
- Orders with invalid Customer_IDs: {invalid_orders_ratio:.2f}%
- Shippings with invalid Customer_IDs: {invalid_shipping_ratio:.2f}%

- Inconsistent Customer Names: {len(df_customer[df_customer['First'].str.contains(r'[!@#$%^&*(),.?":{}|<>]')])} names with special characters corrected.
- Age Outliers: {df_customer[df_customer['Age'] < 0].shape[0]} invalid age records flagged.

Recommendations:
- Review and correct the Customer_ID mismatches between tables.
- Ensure all names follow a consistent format.
- Address any outliers or incorrect values in age and other key fields.
"""

print(report)