

Task Solution

Verifying the accuracy/ completeness/ reliability of the datasets shared. This is done using a mix approach via python & sql. The accuracy & completeness validation is done using python [pandas & re] while reliability checks are done on SQL.

Accuracy & completeness

This python script [verify_accuracy.py] validates the dataset across three sources i.e. (Customer / Order / Shipping). The python script is intended to work with functions created.

- **load_customer_csv** : using `pd.read_csv` it loads the two datasets [Customer & Order] into respective dataframes. For shipping data sitting inside a json file `pd.read_json` was used & loaded into a df
- **check_special_chars** : returns the records where special / non-alphabet characters are present in pandas dataframe . This function finds and returns rows where a given column contains anything other than letters (A–Z, a–z)
 - numbers (0–9)
 - spaces
 - special characters (@, #, \$, etc.)

Showing Results

- Customer (dataset)

For the **First** column in the customer dataset, the following records are inaccurate, as customer names should not contain special characters. It violates expected domain rules for a customer name.

```
check_special_chars(dfl, 'First')
```

Records with special chrc					
	Customer_ID	First	Last	Age	Country
5	6	N!cole	Jones	33	USA
13	14	N!cole	Lara	77	UK
108	109	R0bert	Moore	40	UK
117	118	R0bert	Shepherd	28	UK
161	162	N!cole	Bennett	51	USA
170	171	L@rry	Cole	50	USA
197	198	R0bert	Bryan	49	UK
210	211	Al1cia	Thompson	38	USA
213	214	N!cole	Mcintyre	18	UK
235	236	Al1cia	Jensen	19	USA

Across the field **Last**, having the last_names of customers only 2 records fail the accuracy checks.

```
check_special_chars(df1, 'Last')
```

Records with special chars						
	Customer_ID	First	Last	Age	Country	
112	113	Derrick	R0berts	72	UK	
241	242	Mark	R0berts	61	USA	

- Order (dataset)

```
check_special_chars(df2, 'Item')
```

	Order_ID	Item	Amount	Customer_ID
8	9	DDR RAM	1500	119
18	19	DDR RAM	1500	5
28	29	DDR RAM	1500	33
38	39	DDR RAM	1500	99
48	49	DDR RAM	1500	194
58	59	DDR RAM	1500	164
68	69	DDR RAM	1500	136
78	79	DDR RAM	1500	249
88	89	DDR RAM	1500	193
98	99	DDR RAM	1500	86
108	109	DDR RAM	1500	8
118	119	DDR RAM	1500	184
128	129	DDR RAM	1500	107
138	139	DDR RAM	1500	96
148	149	DDR RAM	1500	20
158	159	DDR RAM	1500	172
168	169	DDR RAM	1500	236
178	179	DDR RAM	1500	67
188	189	DDR RAM	1500	195
198	199	DDR RAM	1500	229
208	209	DDR RAM	1500	40
218	219	DDR RAM	1500	97
228	229	DDR RAM	1500	57
238	239	DDR RAM	1500	223
248	249	DDR RAM	1500	176

We observe that these records are flagged due to spaces between “DDR” and “RAM” in the *Item* field. Since this formatting is expected for this dataset, these can be treated as false positives and safely ignored.

Additionally, for **data completeness** I've added a function to check for fields having nulls or 0 as values.

```
def check_nulls_or_zeroes(df, column_name):  
    mask = df[column_name].isnull() | (df[column_name] == 0)  
    issues = df[mask]  
  
    if issues.empty:  
        print(f"No NULL or ZERO values found in '{column_name}'")  
    else:  
        print(f"Records with NULL or ZERO in '{column_name}':")  
        print(issues)  
  
    return issues
```

Showing Results

No NULL or ZERO values found in 'First', 'Last', 'Age', 'Country' - Customer dataset

```
check_nulls_or_zeroes(df2, 'Customer_ID')
```

No NULL or ZERO values found in 'Item', 'Amount', 'Customer_ID' - Order dataset

The null check on the **Customer_ID** field confirms that every order is properly associated with a customer, ensuring that each **Order_ID** (primary key) has a valid **Customer_ID** mapped to it.

The Order fact table successfully passed completeness and referential integrity checks, with no NULL or zero values identified in the **Item**, **Amount**, or **Customer_ID** fields. All order records maintain valid foreign-key relationships with the customer dimension.

Reliability

Because I am doing null handling separately using na=False in the special-character check is intentional

Outline Requirements

Verification of source data for accuracy, completeness & reliability has resulted in few findings. Few **domain violations** like invalid characters in names were found.

Proposed domain model - I would go ahead with a proposal of OLAP based design where order table acts as the fact table & other tables like - customer , shipping as dimensions table. Since the dataset is not multi-dimensional & hence a denormalised(star-schema) approach would be feasible here.

While order_id uniquely identifies each order as a transaction the customer_id links these order to customer attributes.

order (Fact)

 order_id (PK)
 customer_id (FK)
 item
 Amount

Customer (Dim)

 customer_id (PK)
 first_name
 last_name
 age
 country
 is_valid

Shipping (Dim)

 shipping_id (PK)
 shipping_status
 customer_id

Story & Technical Specifications

This model will support analytical use cases like examining country-level spend analysis for pending deliveries, identification of top-selling products by country and age category. The use of foreign keys in the proposed domain model ensures the maintenance of referential integrity across related tables.

1. Create order as fact table & customer & shipping as dimension tbl
2. Separation of raw vs Cleaned Data - Ingesting of data into staging as received from the source
3. Transformation Layer - We shall retain the raw values & create set of cleaned names for first & last field for dim_customer
4. Cleansing & Validation Rules
 - Using a flag for special characters using regex (records found with non-alphabet characters in first & last field of customer dataset)
 - Allow only alphabets and spaces (spaces are ok)
 - Is_valid : If original values contains special characters then false else true


```
is_valid = CASE
  WHEN first_name_raw ~ '^[A-Za-z ]'
    OR last_name_raw ~ '^[A-Za-z ]'
  THEN false
  ELSE true
END
```

5. Load Strategy Recommendation

- Do not drop records due to invalid names
- Load all customers into dim_customer
- Use is_valid flag for:

6. Result -

For example, In the dim_customer: first_name_raw = N!cole, is_valid = False & first_name = Nicole

[flow diagram](#)

