**1. Sales Data (CSV) – sales\_data.csv**

| **transaction\_id** | **customer\_id** | **product\_id** | **quantity** | **price** | **transaction\_date** |
| --- | --- | --- | --- | --- | --- |
| T001 | C001 | P001 | 2 | 25.50 | 2025-09-01 |
| T002 | C002 | P002 | 1 | 15.00 | 2025-09-01 |
| T003 | C003 | P003 | 5 | 9.99 | 2025-09-02 |
| T004 | C001 | P002 | 3 | 15.00 | 2025-09-03 |
| T005 | C004 | P001 | 1 | 25.50 | 2025-09-03 |

**CSV Content Example:**

transaction\_id,customer\_id,product\_id,quantity,price,transaction\_date

T001,C001,P001,2,25.50,2025-09-01

T002,C002,P002,1,15.00,2025-09-01

T003,C003,P003,5,9.99,2025-09-02

T004,C001,P002,3,15.00,2025-09-03

T005,C004,P001,1,25.50,2025-09-03

**2. Inventory Data (JSON) – inventory\_data.json**

[

{"product\_id": "P001", "product\_name": "Laptop", "stock": 50, "last\_updated": "2025-09-01T10:00:00"},

{"product\_id": "P002", "product\_name": "Mouse", "stock": 150, "last\_updated": "2025-09-01T11:00:00"},

{"product\_id": "P003", "product\_name": "Keyboard", "stock": 100, "last\_updated": "2025-09-02T09:30:00"},

{"product\_id": "P004", "product\_name": "Monitor", "stock": 30, "last\_updated": "2025-09-03T14:00:00"}

]

**3. Customer Data (Parquet) – customer\_data.parquet**

| **customer\_id** | **name** | **email** | **city** | **loyalty\_points** |
| --- | --- | --- | --- | --- |
| C001 | Alice | alice@example.com | Mumbai | 120 |
| C002 | Bob | bob@example.com | Delhi | 80 |
| C003 | Charlie | charlie@example.com | Bangalore | 50 |
| C004 | Diana | diana@example.com | Pune | 200 |

You can generate this Parquet file using Python:

import pandas as pd

df = pd.DataFrame({

"customer\_id": ["C001", "C002", "C003", "C004"],

"name": ["Alice", "Bob", "Charlie", "Diana"],

"email": ["alice@example.com","bob@example.com","charlie@example.com","diana@example.com"],

"city": ["Mumbai","Delhi","Bangalore","Pune"],

"loyalty\_points":[120,80,50,200]

})

df.to\_parquet("customer\_data.parquet", engine='pyarrow', index=False)

**4. Returns Data (ORC) – returns\_data.orc**

| **return\_id** | **transaction\_id** | **product\_id** | **quantity** | **return\_date** | **reason** |
| --- | --- | --- | --- | --- | --- |
| R001 | T003 | P003 | 1 | 2025-09-04 | Defective |
| R002 | T004 | P002 | 1 | 2025-09-05 | Changed Mind |
| R003 | T001 | P001 | 1 | 2025-09-05 | Defective |

Generate ORC using Python:

df = pd.DataFrame({

"return\_id": ["R001","R002","R003"],

"transaction\_id": ["T003","T004","T001"],

"product\_id": ["P003","P002","P001"],

"quantity": [1,1,1],

"return\_date": ["2025-09-04","2025-09-05","2025-09-05"],

"reason": ["Defective","Changed Mind","Defective"]

})

df.to\_orc("returns\_data.orc")

**Usage Notes**

* All files can be uploaded to Snowflake internal stages (@stage\_name) for ingestion.
* Use **COPY INTO** for loading CSV, JSON, Parquet, and ORC tables.
* These small datasets allow testing **Snowflake features** like clustering, materialized views, streams, tasks, and Snowpark transformations.