PHASE 2 Customer State Construction (RFM + Temporal Signals)

```
# STEP 2.1 — Load Phase 1 Artifact (Immutable)

import pandas as pd

df = pd.read_parquet("phase1_clean_transactions.parquet")
print(df.shape)
df.head()
```

```
(824364, 11)
```


```
{"type":"dataframe","variable_name":"df"}
```


```
# STEP 2.2 — Define the Decision Timeline (CRITICAL)

# State is defined using history strictly before current invoice
# Current invoice revenue is NOT used to influence its own state

# STEP 2.3 — Invoice-Level Aggregation (Minimal)

invoice_df = (
    df.groupby(["Customer ID", "Invoice", "InvoiceDate"])
      .agg(
          total_revenue=("revenue", "sum"),
          total_quantity=("Quantity", "sum"),
          is_cancelled=("is_cancelled", "max")
      )
      .reset_index()
)

invoice_df.head()
```


```
{"summary":"{\n  \"name\": \"invoice_df\",\n  \"rows\": 44941,\n
\"fields\": [\n    {\n      \"column\": \"Customer ID\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1719.7948354078635,\n        \"min\": 12346.0,\n        \"max\":
18287.0,\n        \"num_unique_values\": 5942,\n        \"samples\":
[\n          17112.0,\n          14477.0,\n          13746.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Invoice\",\n      \"properties\": {\n        \"dtype\": \"string\",\
n        \"num_unique_values\": 44876,\n        \"samples\": [\n
\"500658\",\n          \"C514384\",\n          \"C516934\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"InvoiceDate\",\n      \"properties\": {\n        \"dtype\":
\"date\",\n        \"min\": \"2009-12-01 07:45:00\",\n        \"max\":
\"2011-12-09 12:50:00\",\n        \"num_unique_values\": 41439,\n
\"samples\": [\n          \"2011-04-28 16:40:00\",\n          \"2010-
10-04 14:11:00\",\n          \"2011-01-13 11:05:00\"\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"total_revenue\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1574.1393018535339,\n   \"min\": -168469.6,\n   \"max\":
168469.6,\n   \"num_unique_values\": 29791,\n   \"samples\":
[\n   679.1500000000001,\n   407.38,\n   524.4\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"total_quantity\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1315,\n   \"min\": -87167,\n   \"max\": 87167,\n
\"num_unique_values\": 2088,\n   \"samples\": [\n   1736,\
n   1035,\n   892\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"is_cancelled\",\n
\"properties\": {\n   \"dtype\": \"boolean\",\n
\"num_unique_values\": 2,\n   \"samples\": [\n   true,\n
false\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"invoice_df"}

```
invoice_df = invoice_df[~invoice_df["is_cancelled"]].copy()

# STEP 2.4 — Build Basic RFM (Time-Causal)

# Step 2.4.1 — Sort correctly (again, no trust)

invoice_df = invoice_df.sort_values(
    by=["Customer ID", "InvoiceDate"]
).reset_index(drop=True)

# Step 2.4.2 — Recency (days since last purchase)

invoice_df["prev_invoice_date"] = (
    invoice_df.groupby("Customer ID")["InvoiceDate"]
              .shift(1)
)

invoice_df["recency_days"] = (
    (invoice_df["InvoiceDate"] - invoice_df["prev_invoice_date"])
    .dt.days
)

invoice_df.head()
```

{"summary":"{\n \"name\": \"invoice_df\",\n \"rows\": 37039,\n
\"fields\": [\n   {\n   \"column\": \"Customer ID\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1721.1264193434051,\n   \"min\": 12346.0,\n   \"max\":
18287.0,\n   \"num_unique_values\": 5881,\n   \"samples\":
[\n   17776.0,\n   17703.0,\n   12546.0\
n   ],\n   \"semantic_type\": \"\",\n

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Invoice\",\n   \"properties\": {\n   \"dtype\": \"string\",\
n   \"num_unique_values\": 36975,\n   \"samples\": [\n
\"571901\",\n   \"525298\",\n   \"549286\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"InvoiceDate\",\n
\"properties\": {\n   \"dtype\": \"date\",\n   \"min\":
\"2009-12-01 07:45:00\",\n   \"max\": \"2011-12-09 12:50:00\",\n
\"num_unique_values\": 34591,\n   \"samples\": [\n
\"2010-09-20 17:32:00\",\n   \"2011-04-15 08:45:00\",\n
\"2010-01-20 09:50:00\"\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"total_revenue\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 1373.6952611714842,\n
\"min\": 0.0,\n   \"max\": 168469.6,\n
\"num_unique_values\": 26941,\n   \"samples\": [\n
165.95,\n   395.59,\n   399.15\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"total_quantity\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1236,\n   \"min\": 1,\n   \"max\": 87167,\n
\"num_unique_values\": 1814,\n   \"samples\": [\n   677,\n
1366,\n   533\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"is_cancelled\",\n   \"properties\": {\n   \"dtype\":
\"boolean\",\n   \"num_unique_values\": 1,\n   \"samples\":
[\n   false\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"prev_invoice_date\",\n   \"properties\": {\n   \"dtype\":
\"date\",\n   \"min\": \"2009-12-01 07:45:00\",\n   \"max\":
\"2011-12-09 12:23:00\",\n   \"num_unique_values\": 29411,\n
\"samples\": [\n   \"2010-09-03 13:13:00\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"recency_days\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
75.7097797248008,\n   \"min\": 0.0,\n   \"max\": 714.0,\n
\"num_unique_values\": 515,\n   \"samples\": [\n   384.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"invoice_df"}

```
# Step 2.4.3 — Frequency (purchase count so far)

invoice_df["frequency"] = (
    invoice_df.groupby("Customer ID")
              .cumcount()
)

# Step 2.4.4 — Monetary (historical average spend)
```

```
invoice_df["cum_revenue"] = (
    invoice_df.groupby("Customer ID")["total_revenue"]
              .cumsum()
)

invoice_df["monetary_avg"] = (
    invoice_df["cum_revenue"] /
    (invoice_df["frequency"] + 1)
)

# STEP 2.5 — Temporal Dynamics (Small but Powerful)

invoice_df["prev_revenue"] = (
    invoice_df.groupby("Customer ID")["total_revenue"]
              .shift(1)
)

invoice_df["delta_revenue"] = (
    invoice_df["total_revenue"] - invoice_df["prev_revenue"]
)

# Purchase acceleration (Δ recency)

invoice_df["prev_recency"] = (
    invoice_df.groupby("Customer ID")["recency_days"]
              .shift(1)
)

invoice_df["delta_recency"] = (
    invoice_df["recency_days"] - invoice_df["prev_recency"]
)

# STEP 2.6 — Define the Customer State Vector

state_cols = [
    "recency_days",
    "frequency",
    "monetary_avg",
    "delta_revenue",
    "delta_recency"
]

state_df = invoice_df[
    ["Customer ID", "InvoiceDate"] + state_cols
].copy()

state_df.head()
```


```
{"summary":"{\n  \"name\": \"state_df\",\n  \"rows\": 37039,\n
\"fields\": [\n    {\n      \"column\": \"Customer ID\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
```

```
1721.1264193434051,\n        \"min\": 12346.0,\n        \"max\":
18287.0,\n        \"num_unique_values\": 5881,\n        \"samples\":
[\n          17776.0,\n          17703.0,\n          12546.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"InvoiceDate\",\n      \"properties\": {\n        \"dtype\":
\"date\",\n        \"min\": \"2009-12-01 07:45:00\",\n        \"max\":
\"2011-12-09 12:50:00\",\n        \"num_unique_values\": 34591,\n
\"samples\": [\n          \"2010-09-20 17:32:00\",\n          \"2011-
04-15 08:45:00\",\n          \"2010-01-20 09:50:00\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"recency_days\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
75.7097797248008,\n        \"min\": 0.0,\n        \"max\": 714.0,\n
\"num_unique_values\": 515,\n        \"samples\": [\n          384.0,\
n          682.0,\n          690.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"frequency\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
37,\n        \"min\": 0,\n        \"max\": 399,\n
\"num_unique_values\": 400,\n        \"samples\": [\n          209,\n
280,\n          33\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"monetary_avg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 777.8735726422763,\n        \"min\":
0.0,\n        \"max\": 84236.25,\n        \"num_unique_values\":
35842,\n        \"samples\": [\n          261.8607692307692,\n
443.9271428571429,\n          339.4130612244898\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"delta_revenue\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1635.9432289056133,\n        \"min\": -42409.1,\n        \"max\":
168466.7,\n        \"num_unique_values\": 28759,\n        \"samples\":
[\n          159.18,\n          80.27999999999997,\n          -
516.1199999999999\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"delta_recency\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 75.6244236542911,\n        \"min\": -
618.0,\n        \"max\": 701.0,\n        \"num_unique_values\": 796,\n
\"samples\": [\n          387.0,\n          492.0,\n          -166.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"state_df"}
```


```
# STEP 2.7 — Sanity Checks (Must Pass)

state_df.isna().mean()
```

```
Customer ID      0.000000
InvoiceDate      0.000000
recency_days     0.158779
```

```
frequency         0.000000
monetary_avg      0.000000
delta_revenue     0.158779
delta_recency     0.273738
dtype: float64
```

```
# ✔ Frequency grows monotonically

check_freq = (
    state_df.groupby("Customer ID")["frequency"]
            .apply(lambda x: x.is_monotonic_increasing)
)

check_freq.all()
```

```
np.True_
```

```
# STEP 2.8 — Save Phase 2 Artifact

state_df.to_parquet("phase2_customer_state.parquet", index=False)
```