PHASE 3 Latent Churn & Survival Modeling (Non-Contractual Setting)

```
# STEP 3.1 — Load Phase 2 Artifact (Immutable)

import pandas as pd

state_df = pd.read_parquet("phase2_customer_state.parquet")
print(state_df.shape)
state_df.head()
```

```
(37039, 7)
```


```
{"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 3.2 — Define "Observation End" (Critical Concept)

END_DATE = state_df["InvoiceDate"].max()
END_DATE
```

```
Timestamp('2011-12-09 12:50:00')
```

```
# STEP 3.3 — Define Inactivity Threshold (Censoring Rule)

INACTIVITY_THRESHOLD_DAYS = 180  # 6 months

# STEP 3.4 — Compute "Time Since Last Purchase"

last_purchase = (
    state_df.groupby("Customer ID")["InvoiceDate"]
            .max()
            .reset_index()
            .rename(columns={"InvoiceDate": "last_invoice_date"})
)

last_purchase["days_since_last_purchase"] = (
    END_DATE - last_purchase["last_invoice_date"]
).dt.days

last_purchase.head()
```

{"summary":"{\n \"name\": \"last_purchase\",\n \"rows\": 5881,\n
\"fields\": [\n   {\n   \"column\": \"Customer ID\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1715.4297590182248,\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\":
\"last_invoice_date\",\n   \"properties\": {\n   \"dtype\":
\"date\",\n   \"min\": \"2009-12-01 09:55:00\",\n   \"max\":
\"2011-12-09 12:50:00\",\n   \"num_unique_values\": 5731,\n
\"samples\": [\n   \"2011-11-20 10:15:00\",\n   \"2010-
11-04 09:06:00\",\n   \"2011-02-04 14:03:00\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"days_since_last_purchase\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
209,\n   \"min\": 0,\n   \"max\": 738,\n

\"num_unique_values\": 676,\n   \"samples\": [\n   586,\n
492,\n   460\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"last_purchase"}

```
# STEP 3.5 — Define Survival Status (Alive vs Censored)

last_purchase["is_alive"] = (
    last_purchase["days_since_last_purchase"]
    <= INACTIVITY_THRESHOLD_DAYS
)

last_purchase["is_alive"].value_counts(normalize=True)
```

```
is_alive
True     0.591906
False    0.408094
Name: proportion, dtype: float64
```

```
# STEP 3.6 — Create Survival Table (Customer-Level)

survival_df["event"] = (
    survival_df["days_since_last_purchase"]
    > INACTIVITY_THRESHOLD_DAYS
).astype(int)

# STEP 3.7 — Merge Survival Target Back to State (Time-Indexed)

state_survival_df = state_df.merge(
    survival_df[["Customer ID", "is_alive", "duration", "event"]],
    on="Customer ID",
    how="left"
)

state_survival_df.head()
```

{"summary":"{\n \"name\": \"state_survival_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      \"column\": \"is_alive\",\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      \"column\":
\"duration\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 145,\n        \"min\": 0,\n
\"max\": 738,\n        \"num_unique_values\": 676,\n
\"samples\": [\n          586,\n          492\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"event\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\
n}","type":"dataframe","variable_name":"state_survival_df"}
```


```
# STEP 3.8 — Sanity Checks (Must Pass)
```

```python
#  No missing survival labels
state_survival_df[["is_alive", "duration", "event"]].isna().sum()
```

```
is_alive    0
duration    0
event       0
dtype: int64
```

```python
#  Reasonable alive ratio
state_survival_df["is_alive"].mean()
```

```
np.float64(0.8323118874699641)
```

```python
# STEP 3.9 — Save Phase 3 Artifact

state_survival_df.to_parquet(
    "phase3_state_with_survival.parquet",
    index=False
)
```