PHASE 5 Expected Survival × Conditional Value (CLV Emerges)

```
# STEP 5.1 — Load Phase 4 Artifacts

import pandas as pd
import numpy as np

person_period_df =
pd.read_parquet("phase4_person_period_dataset.parquet")
print(person_period_df.shape)
person_period_df.head()
```

```
(133690, 8)
```

{"type":"dataframe","variable_name":"person_period_df"}

```
# STEP 5.1.1 — Refit Hazard Model

from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression

features = [
    "recency_days",
    "frequency",
    "monetary_avg",
    "delta_revenue",
    "delta_recency",
    "time_bin"
]

X = person_period_df[features]
y = person_period_df["event"]

# Impute missing values
imputer = SimpleImputer(strategy="median")
X_imputed = imputer.fit_transform(X)

# Refit hazard model
hazard_model = LogisticRegression(max_iter=1000)
hazard_model.fit(X_imputed, y)
```

```
LogisticRegression(max_iter=1000)
```

```
# STEP 5.2 — Recompute Hazard Probabilities (Explicitly)

# Recreate feature matrix
features = [
    "recency_days",
    "frequency",
    "monetary_avg",
```

```
    "delta_revenue",
    "delta_recency",
    "time_bin"
]

X = person_period_df[features]

from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
X_imputed = imputer.fit_transform(X)

# Predict hazard (P(event at t | alive until t))
hazard_prob = hazard_model.predict_proba(X_imputed)[:, 1]

person_period_df["hazard"] = hazard_prob
person_period_df[["Customer ID", "time_bin", "hazard"]].head()
```


```
{"summary":"{\n  \"name\": \"person_period_df[[\\\"Customer ID\\\",
\\\"time_bin\\\", \\\"hazard\\\"]]\",\n  \"rows\": 5,\n  \"fields\":
[\n    {\n      \"column\": \"Customer ID\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0.0,\n
\"min\": 12346.0,\n        \"max\": 12346.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          12346.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"time_bin\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 0,\n        \"max\": 4,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          1\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"hazard\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.003972437602158006,\n        \"min\": 0.005629960910436029,\n
\"max\": 0.015632019038142958,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          0.00727380896858972\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
# STEP 5.3 — Convert Hazard → Survival Probability

# Code (group-wise cumulative product)
person_period_df = person_period_df.sort_values(
    by=["Customer ID", "time_bin"]
)

person_period_df["survival_prob"] = (
    person_period_df.groupby("Customer ID")["hazard"]
    .transform(lambda x: (1 - x).cumprod())
)
```

```
# Quick sanity checks (run these)
# Survival prob range
person_period_df["survival_prob"].describe()
```

```
count    1.336900e+05
mean     6.977414e-01
std      3.162008e-01
min      3.749170e-49
25%      5.166298e-01
50%      8.402659e-01
75%      9.475724e-01
max      9.965548e-01
Name: survival_prob, dtype: float64
```

```
# Monotonic decrease per customer
(
    person_period_df
    .groupby("Customer ID")["survival_prob"]
    .apply(lambda x: x.is_monotonic_decreasing)
    .value_counts()
)
```

```
survival_prob
True    5881
Name: count, dtype: int64
```

```
# STEP 5.4 — Define Expected Conditional Revenue

person_period_df["expected_revenue"] =
person_period_df["monetary_avg"]

# STEP 5.5 — Choose Discount Factor & Horizon

DISCOUNT_RATE = 0.95   # monthly
MAX_HORIZON = 12       # months

# STEP 5.6 — Compute Expected CLV Contribution per Period

person_period_df = person_period_df[
    person_period_df["time_bin"] < MAX_HORIZON
].copy()

person_period_df["discount"] = DISCOUNT_RATE **
person_period_df["time_bin"]

person_period_df["clv_contribution"] = (
    person_period_df["survival_prob"]
    * person_period_df["expected_revenue"]
    * person_period_df["discount"]
)

# STEP 5.7 — Aggregate to Customer-Level Expected CLV
```

```
clv_df = (
    person_period_df.groupby("Customer ID")["clv_contribution"]
    .sum()
    .reset_index()
    .rename(columns={"clv_contribution": "expected_clv"})
)

clv_df.describe()
```


```
{"summary":"{\n  \"name\": \"clv_df\",\n  \"rows\": 8,\n  \"fields\":
[\n    {\n      \"column\": \"Customer ID\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 5728.060958005437,\n
\"min\": 1715.4297590182248,\n        \"max\": 18287.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
15314.674205067166,\n          15313.0,\n          5881.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"expected_clv\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
140319.46700920907,\n        \"min\": 0.0,\n        \"max\":
401760.5653631753,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          5772.733497240519,\n
2877.5321429331143,\n          5881.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
# STEP 5.8 — Sanity & Intuition Checks

# High-frequency customers → higher CLV
clv_df.merge(
    person_period_df[["Customer ID", "frequency"]].drop_duplicates(),
    on="Customer ID"
).corr()
```


```
{"summary":"{\n  \"name\": \")\",\n  \"rows\": 3,\n  \"fields\": [\n
{\n      \"column\": \"Customer ID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.591646259921615,\n
\"min\": -0.06426683799035594,\n        \"max\": 1.0,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          1.0,\n
0.01994028818064243,\n          -0.06426683799035594\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"expected_clv\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.4958331769269017,\n        \"min\": 0.01994028818064243,\n
\"max\": 1.0,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          0.01994028818064243,\n          1.0,\n
0.37895930875335004\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"frequency\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.5346034137232604,\n        \"min\": -
0.06426683799035594,\n        \"max\": 1.0,\n
```

\"num_unique_values\": 3,\n   \"samples\": [\n   -0.06426683799035594,\n   0.37895930875335004,\n   1.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
# Survival decreases over horizon (spot check)
(
    person_period_df
    .groupby("time_bin")["survival_prob"]
    .mean()
    .head(10)
)
```

```
time_bin
0    0.914805
1    0.900121
2    0.850939
3    0.795134
4    0.738548
5    0.685236
6    0.628153
7    0.568123
8    0.504880
9    0.436036
Name: survival_prob, dtype: float64
```

```
# STEP 5.9 — Save Phase 5 Artifact

clv_df.to_parquet("phase5_expected_clv.parquet", index=False)
```