```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import csv

ls
```

```
sample_data/           test_identity.csv     train_identity.csv
sample_submission.csv  test_transaction.csv  train_transaction.csv
```

```
# phase 1A - data loading

# PHASE 1-B — IEEE-CIS DATASET

train_txn = pd.read_csv("train_transaction.csv")
train_txn.head()
```

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

```
train_txn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 272611 entries, 0 to 272610
Columns: 394 entries, TransactionID to V339
dtypes: float64(376), int64(4), object(14)
memory usage: 819.5+ MB
```

```
train_id = pd.read_csv("train_identity.csv")
train_id.head()
```

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

```
train_id.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144233 entries, 0 to 144232
Data columns (total 41 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   TransactionID  144233 non-null  int64
 1   id_01          144233 non-null  float64
 2   id_02          140872 non-null  float64
 3   id_03          66324 non-null   float64
 4   id_04          66324 non-null   float64
 5   id_05          136865 non-null  float64
 6   id_06          136865 non-null  float64
 7   id_07          5155 non-null    float64
 8   id_08          5155 non-null    float64
 9   id_09          74926 non-null   float64
 10  id_10          74926 non-null   float64
 11  id_11          140978 non-null  float64
```

```
 12  id_12           144233 non-null  object
 13  id_13           127320 non-null  float64
 14  id_14           80044 non-null   float64
 15  id_15           140985 non-null  object
 16  id_16           129340 non-null  object
 17  id_17           139369 non-null  float64
 18  id_18           45113 non-null   float64
 19  id_19           139318 non-null  float64
 20  id_20           139261 non-null  float64
 21  id_21           5159 non-null    float64
 22  id_22           5169 non-null    float64
 23  id_23           5169 non-null    object
 24  id_24           4747 non-null    float64
 25  id_25           5132 non-null    float64
 26  id_26           5163 non-null    float64
 27  id_27           5169 non-null    object
 28  id_28           140978 non-null  object
 29  id_29           140978 non-null  object
 30  id_30           77565 non-null   object
 31  id_31           140282 non-null  object
 32  id_32           77586 non-null   float64
 33  id_33           73289 non-null   object
 34  id_34           77805 non-null   object
 35  id_35           140985 non-null  object
 36  id_36           140985 non-null  object
 37  id_37           140985 non-null  object
 38  id_38           140985 non-null  object
 39  DeviceType      140810 non-null  object
 40  DeviceInfo      118666 non-null  object
dtypes: float64(23), int64(1), object(17)
memory usage: 45.1+ MB
```

"At Phase 1-B, the analysis is restricted to structural inspection only. No semantic interpretation of features is assumed, particularly for anonymized variables (V*, id_*). All semantic assumptions are deferred to Phase 1-C."

```
# PHASE 1-C — ASSUMPTION TABLE (FORMAL, AUDITABLE)
```

Task

Select the 'decision-critical' columns `TransactionID`, `TransactionDT`, `TransactionAmt`, `ProductCD`, `card1`, `addr1`, `isFraud` from `train_txn` and `TransactionID`, `DeviceType`, `DeviceInfo` from `train_id`, then merge these selected columns into a single DataFrame on `TransactionID`, and finally display the first few rows and summary information of the merged DataFrame.

```
transaction_columns = ['TransactionID', 'TransactionDT',
'TransactionAmt', 'ProductCD', 'card1', 'addr1', 'isFraud']
train_txn_selected = train_txn[transaction_columns]
train_txn_selected.head()
```

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

```
identity_columns = ['TransactionID', 'DeviceType', 'DeviceInfo']
train_id_selected = train_id[identity_columns]
train_id_selected.head()
```

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

```
merged_df = pd.merge(train_txn_selected, train_id_selected,
on='TransactionID', how='left')
print("First 5 rows of the merged DataFrame:")
print(merged_df.head())
print("\nSummary information of the merged DataFrame:")
merged_df.info()
```

```
First 5 rows of the merged DataFrame:
   TransactionID  TransactionDT  TransactionAmt ProductCD  card1
addr1  \
0        2987000          86400            68.5         W  13926
315.0
1        2987001          86401            29.0         W   2755
325.0
2        2987002          86469            59.0         W   4663
330.0
3        2987003          86499            50.0         W  18132
476.0
4        2987004          86506            50.0         H   4497
420.0

   isFraud DeviceType                      DeviceInfo
0        0        NaN                             NaN
1        0        NaN                             NaN
2        0        NaN                             NaN
3        0        NaN                             NaN
4        0     mobile  SAMSUNG SM-G892A Build/NRD90M

Summary information of the merged DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 272611 entries, 0 to 272610
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   TransactionID   272611 non-null  int64
 1   TransactionDT   272611 non-null  int64
 2   TransactionAmt  272611 non-null  float64
 3   ProductCD       272611 non-null  object
```

```
 4   card1           272611 non-null  int64
 5   addr1           242693 non-null  float64
 6   isFraud         272611 non-null  int64
 7   DeviceType      81599 non-null   object
 8   DeviceInfo      71107 non-null   object
dtypes: float64(2), int64(4), object(3)
memory usage: 18.7+ MB
```

STEP 1-C.3 — Create the Assumption Table

Based on the 'Gold Standard Entry' examples, we'll create a DataFrame to systematically document the assumed meaning, confidence, and risk associated with each 'decision-critical' column in our `merged_df`.

```
assumption_table_data = [
    {'Column': 'TransactionID', 'Table':
'train_transaction/train_identity', 'Assumed Meaning': 'Unique
transaction identifier used for joins', 'Confidence': 'High', 'Risk if
Wrong': 'Incorrect joins, duplicated or missing records'},
    {'Column': 'TransactionDT', 'Table': 'train_transaction', 'Assumed
Meaning': 'Time offset (likely seconds) from reference point',
'Confidence': 'Low', 'Risk if Wrong': 'Temporal analysis invalid,
trends meaningless'},
    {'Column': 'TransactionAmt', 'Table': 'train_transaction',
'Assumed Meaning': 'Transaction amount (currency and scaling
unspecified)', 'Confidence': 'Medium', 'Risk if Wrong': 'Incorrect
monetary scaling leading to invalid fraud thresholds.'},
    {'Column': 'ProductCD', 'Table': 'train_transaction', 'Assumed
Meaning': 'Product code for the transaction', 'Confidence': 'Medium',
'Risk if Wrong': 'Misinterpretation of product categories affecting
fraud patterns'},
    {'Column': 'card1', 'Table': 'train_transaction', 'Assumed
Meaning': 'Masked credit card number (first block)', 'Confidence':
'Medium', 'Risk if Wrong': 'Misleading insights on card usage
patterns'},
    {'Column': 'addr1', 'Table': 'train_transaction', 'Assumed
Meaning': 'Billing address (masked)', 'Confidence': 'Medium', 'Risk if
Wrong': 'Incorrect geographical analysis'},
    {'Column': 'isFraud', 'Table': 'train_transaction', 'Assumed
Meaning': 'Target variable: 1 for fraud, 0 for legitimate',
'Confidence': 'High', 'Risk if Wrong': 'Incorrect model training and
evaluation'},
    {'Column': 'DeviceType', 'Table': 'train_identity', 'Assumed
Meaning': 'Type of device used for the transaction (e.g., mobile,
desktop)', 'Confidence': 'High', 'Risk if Wrong': 'Misunderstanding of
device-specific fraud behavior'},
    {'Column': 'DeviceInfo', 'Table': 'train_identity', 'Assumed
Meaning': 'Specific device information', 'Confidence': 'Medium', 'Risk
if Wrong': 'Loss of granular device insights, if any'}
```

```
]

assumption_df = pd.DataFrame(assumption_table_data)

print("Assumption Table for merged_df columns:")
display(assumption_df)
```

```
Assumption Table for merged_df columns:
```


```
{"summary":"{\n  \"name\": \"assumption_df\",\n  \"rows\": 9,\n
\"fields\": [\n    {\n      \"column\": \"Column\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 9,\n        \"samples\": [\n
\"DeviceType\",\n          \"TransactionDT\",\n          \"addr1\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Table\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n
\"train_transaction/train_identity\",\n
\"train_transaction\",\n          \"train_identity\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Assumed Meaning\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 9,\n        \"samples\": [\n          \"Type of
device used for the transaction (e.g., mobile, desktop)\",\n
\"Time offset (likely seconds) from reference point\",\n
\"Billing address (masked)\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Confidence\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"High\",\n          \"Low\",\n
\"Medium\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Risk if Wrong\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 9,\n        \"samples\":
[\n          \"Misunderstanding of device-specific fraud behavior\",\n
\"Temporal analysis invalid, trends meaningless\",\n
\"Incorrect geographical analysis\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"assumption_df"}
```


```
# “The column selection and merge operations are exploratory
scaffolding and are not treated as validated analytical steps at Phase
1-C.”

# PHASE 2 — VALIDATION & SILENT-ERROR DETECTION

# PHASE 2-A — SILENT ERROR TAXONOMY
```

Phase 2-A Deliverable: Silent Error Table

Error Category	Example Scenario	Severity	Why It's Silent
Schema Error	`card1` is loaded as a float type due to `NaN` values, preventing its direct use as a categorical identifier in analysis.	Major	The column loads without error, but its true categorical nature is lost, leading to incorrect aggregation or feature engineering.
Join Error	`train_transaction` is left-joined with `train_identity`, and missing `DeviceType` values are interpreted as "unknown device" rather than "no identity record exists."	Critical	The join executes successfully and produces valid rows, but the semantic meaning of missing identity data is silently misinterpreted.
Temporal Error	`TransactionDT` is treated as a real timestamp and used to generate daily fraud trends, despite being a relative offset without calendar semantics.	Major	The time series appears valid and ordered, but the derived temporal insights are meaningless.
Aggregation Error	`TransactionAmt` is aggregated across `ProductCD` categories, assuming a consistent currency and scaling, leading to invalid monetary sums or averages.	Critical	Aggregations compute without error, but the calculated totals are fundamentally incorrect due to unvalidated currency/scale, providing misleading business insights.
Confidence Error	High sparsity in `train_identity` columns like `id_02` or `DeviceInfo` is not flagged, leading to overconfidence in identity-based fraud detection despite poor data coverage.	Critical	Models built using these sparse features appear to function correctly, but their underlying weakness leads to unacknowledged bias or poor generalization in real-world scenarios.

```
# PHASE 2-B — VALIDATION RULES (ERROR → INTERCEPTION)
```

Rule ID: JE-IEEE-001 Linked Error Category: Join Error Trigger Condition: After left-joining `train_transaction` and `train_identity` on `TransactionID`, identity-related columns

(DeviceType, DeviceInfo) contain null values. Validation Check: Identify all TransactionIDs in the merged_df where either DeviceType or DeviceInfo is null. If Violated (System Action): Issue a critical alert. Halt any downstream analysis that relies on the semantic interpretation of DeviceType or DeviceInfo for these affected records. Log the count and a sample of TransactionIDs with null identity information. Human Involvement: A data steward or domain expert must explicitly clarify the semantic meaning of these nulls (e.g., 'no identity record exists' vs. 'unknown device'). They need to provide a formal decision on how these nulls should be treated before data processing can resume.

```
# PHASE 2-B — TEMPORAL ERROR VALIDATION RULE (JOINT)
```

Rule ID: TE-IEEE-001 Linked Error Category: Temporal Error Trigger Condition: Any analytical operation attempts to interpret TransactionDT as a real timestamp (e.g., conversion to date/time, calendar-based grouping, or time-of-day analysis). Validation Check: Inspect whether TransactionDT is being used in functions or logic that assume calendar semantics (such as date parsing, daily/weekly aggregation, or time-window comparisons tied to real-world clocks). If Violated (System Action): Block result presentation and label the output as “Invalid Temporal Semantics”. Suppress any time-based trends, charts, or summaries derived from TransactionDT. Human Involvement: Require the analyst to explicitly state the intended temporal interpretation (e.g., relative ordering only vs. assumed calendar mapping) and approve any downstream use of TransactionDT under that assumption.

```
# PHASE 2-B — AGGREGATION ERROR RULE
```

Rule ID: AE-IEEE-001 Linked Error Category: Aggregation Error Trigger Condition: Any analytical operation involving aggregation (e.g., sum, average, rate calculation, time-windowed statistics) on TransactionAmt or isFraud without explicit validation of underlying units or definitions. Validation Check: Inspect whether aggregations of TransactionAmt (e.g., sums across ProductCD or card1) or isFraud (e.g., fraud rates by DeviceType) are being performed without prior formal declaration of consistent units (for TransactionAmt) or consistent population/time-window definitions (for isFraud). If Violated (System Action): Block result presentation and label the output as “Invalid Aggregation Semantics”. Suppress any aggregated metrics, charts, or summaries derived from these unvalidated aggregations. Human Involvement: Require the analyst to explicitly state the assumed currency and scaling for TransactionAmt, or the specific population and time-window definitions for isFraud aggregations, and approve any downstream use of these aggregated metrics under that assumption.

```
# PHASE 2-B — CONFIDENCE ERROR VALIDATION RULE
```

Rule ID: CE-IEEE-001 Linked Error Category: Confidence Error Trigger Condition: Any analytical output (e.g., trend, correlation, model performance metric) is generated using a feature (e.g., id_02, DeviceInfo, V* columns) with high data sparsity (e.g., < 10% non-null values) without

explicit acknowledgment of low data coverage. Validation Check: Inspect whether analyses or conclusions are being presented with high confidence when derived from columns identified in the 'Assumption Table' or through data profiling to have 'Low' confidence, 'High' risk, or significant sparsity. If Violated (System Action): Block the presentation of any confidence metrics (e.g., 'model is 95% accurate'). Downgrade or prefix all related output with a 'Low Confidence' label. Suppress any interpretations implying high certainty. Human Involvement: Require the analyst to explicitly acknowledge the low data coverage/confidence for the features used. A data steward must approve any statement of confidence or downstream use of insights derived from these features.

```
# PHASE 3 — EXPLANATION & HUMAN INTERACTION LAYER

# PHASE 3-A — EXPLANATION CONTRACT REVIEW
```

Rule ID: AE-IEEE-001 Level 1 (Executive): This result relies on aggregated transaction amounts or fraud rates whose underlying units or definitions are unverified. Decisions based on this insight may be fundamentally unreliable. Level 2 (Analyst): An aggregation was attempted on `TransactionAmt` or `isFraud` without explicit declaration of currency/scaling (for amounts) or population/time-window definitions (for fraud rates). This could lead to combining incomparable values or drawing conclusions from inconsistent contexts, rendering the aggregated metrics misleading or incorrect. Level 3 (Audit): Rule Triggered: AE-IEEE-001 Reason: Aggregation performed on `TransactionAmt` or `isFraud` without explicit validation of underlying units or definitions. Affected Columns: `TransactionAmt`, `isFraud` Severity: Critical

```
# PHASE 3-B — HUMAN RESPONSE PATHS
```

Scenario: An analyst has calculated the average `TransactionAmt` per `ProductCD` and attempts to generate a chart. Rule AE-IEEE-001 is triggered because no explicit currency or scaling has been formally declared for `TransactionAmt`, leading to an 'Invalid Aggregation Semantics' warning. Path 1 (Accept): The analyst acknowledges the system's warning about unverified units for `TransactionAmt`. They decide the aggregated averages are not trustworthy enough to use or present without clarification and accept the system's judgment to block the chart's generation. The system logs this action as 'Accepted' for AE-IEEE-001. Path 2 (Override): The analyst confirms through an internal document that all `TransactionAmt` values are in USD and can be treated with standard scaling. They choose to override the system's block. They select 'Domain knowledge assumption' as the justification, stating: 'Confirmed all `TransactionAmt` are USD with standard scaling via internal spec.' The system releases the chart but applies a 'Low Confidence' banner, and logs the override with the justification. Path 3 (Defer): The analyst is uncertain about the `TransactionAmt` currency and scaling and recognizes the high impact of this assumption. Lacking the authority to make a definitive declaration, they choose to defer the decision. The system keeps the chart generation blocked and escalates the task to a Senior Data Steward, logging the deferral and the escalation target.

```
# PHASE 3-C — TRUST CALIBRATION OVER TIME
```

Scenario: An analyst, currently in 'Calibrated' state, consistently demonstrates well-justified overrides and appropriate deferrals for critical issues over a sustained period. Observed Signals:

- For 10 instances where Rule JE-IEEE-001 (Join Error) was triggered and the analyst chose 'Override with Domain knowledge assumption', subsequent data validation confirmed their assumption was correct and the outcomes were positive.
- For 5 instances where Rule TE-IEEE-001 (Temporal Error) was triggered and the analyst chose 'Defer', the escalated decision by a Senior Data Steward aligned with the analyst's initial deferral, preventing a potential major error.
- In 15 instances, the analyst accepted the system's judgment for various rules (JE-IEEE-001, AE-IEEE-001, CE-IEEE-001), showing consistent alignment with system assessments when appropriate. Trust State Change: From 'Calibrated' to 'Trusted'. System Behavior Change:
- Fewer interruptive warnings will be presented to the analyst (e.g., less prominent notifications, consolidated summaries).
- Confidence downgrades for future analyses will be phrased more concisely, assuming a higher level of understanding from the analyst.
- Override friction will be slightly reduced (e.g., faster confirmation dialogs, pre-filled common justifications for minor issues), but the requirement for justification remains. Safeguards:
- Validation rules will still fire as normal, and all triggered events will be logged.
- Output blocks will still apply when triggered by critical violations, requiring explicit human intervention.
- There will be no silent bypasses of any validation checks or human intervention requirements.
- The analyst cannot auto-accept overrides; justification is still mandatory, albeit potentially streamlined.
- Confidence will not be increased beyond what the evidence supports for the underlying data; the system will not hide uncertainty.

Title: SentinelAI — Validation-First Analytical Decision Assistant

Problem

- Silent analytical errors
- Overconfident AI outputs

Core Idea

- Validation before answers
- Human retains final authority

What the system does

- Detects silent errors
- Blocks / downgrades outputs
- Forces explicit human decisions

What the system never does

- Never auto-fixes
- Never sounds confident without evidence

```
# PHASE 4 — RESEARCH PAPER STRUCTURE ( DONE )

# PHASE 4-B — EXPERIMENT BLUEPRINT (MINIMUM SUFFICIENT)
```

Experiment 1 - Step 3: Execution & Logging Setup

```
import pandas as pd

# Create a simple log table as a DataFrame
log_columns = ['Task', 'Condition', 'Output Executed (Y/N)', 'Looked
Valid (Y/N)', 'Silent Error? (Y/N)', 'Intercepted? (Y/N)', 'Notes']
experiment_log = pd.DataFrame(columns=log_columns)

print("Empty Experiment Log Table created. Please fill this manually
as you perform the tasks.")
display(experiment_log)
```

```
Empty Experiment Log Table created. Please fill this manually as you
perform the tasks.
```


```
{"repr_error":"Out of range float values are not JSON compliant:
nan","type":"dataframe","variable_name":"experiment_log"}
```


TASK C — AGGREGATION TASK: Baseline Execution

This cell attempts to aggregate `TransactionAmt` without explicit validation of its units or definitions, simulating a scenario where a silent aggregation error might occur.

```
print("Baseline Execution: Aggregating TransactionAmt by ProductCD.")
print("Attempting to calculate average TransactionAmt per ProductCD:")

# Performing a simple aggregation without prior formal declaration of
consistent units
# This operation will run without error, appearing 'valid' to the
analyst
# but silently misinterpreting TransactionAmt due to unvalidated
currency/scaling.
avg_amt_by_product = merged_df.groupby('ProductCD')
['TransactionAmt'].mean().reset_index()
display(avg_amt_by_product.head())
```

```
print("\nOutput Executed (Y), Looked Valid (Y), Silent Error (Y - unverified units/definitions), Intercepted (N)")
```

```
Baseline Execution: Aggregating TransactionAmt by ProductCD.
Attempting to calculate average TransactionAmt per ProductCD:
```


```
{"summary":"{\n  \"name\": \"print(\\\"\\\\nOutput Executed (Y), Looked Valid (Y), Silent Error (Y - unverified units/definitions), Intercepted (N)\\\")\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n  \"column\": \"ProductCD\",\n      \"properties\": {\n  \"dtype\": \"string\",\n        \"num_unique_values\": 5,\n  \"samples\": [\n          \"H\",\n          \"W\",\n          \"R\"\n  ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n  }\n    },\n    {\n      \"column\": \"TransactionAmt\",\n  \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 59.353571312731496,\n        \"min\": 44.504183924895514,\n  \"max\": 174.0778567780964,\n        \"num_unique_values\": 5,\n  \"samples\": [\n          71.30724426647251,\n  151.0000238169819,\n          174.0778567780964\n        ],\n  \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
Output Executed (Y), Looked Valid (Y), Silent Error (Y - unverified units/definitions), Intercepted (N)
```

TASK C — AGGREGATION TASK: SentinelAI Execution

This cell repeats the same analytical intent, but conceptually, the SentinelAI validation rule (AE-IEEE-001) would intercept the silent error related to unvalidated units or definitions in `TransactionAmt` aggregation. The code simulates the system's reaction (blocking/warning).

```
print("SentinelAI Execution: Aggregating TransactionAmt by ProductCD (same as Baseline).")

# Simulate SentinelAI's Aggregation Error validation rule (AE-IEEE-001)
# This check would happen before or during the execution of a function that assumes validated units/definitions.

print("\n--- SentinelAI ALERT ---\n")
print("Rule ID: AE-IEEE-001 - Aggregation Error Triggered!")
print("Trigger Condition: Analytical operation involving aggregation on TransactionAmt without explicit validation of underlying units or definitions.")
print("Validation Check: Aggregation of TransactionAmt detected without prior formal declaration of consistent units.")
print("System Action: Blocking result presentation. Output labeled 'Invalid Aggregation Semantics'.")
print("Human Involvement: Analyst must explicitly state the assumed
```

```
currency and scaling for TransactionAmt.")
print("--- End SentinelAI ALERT ---")

print("\nOutput Executed (Y), Looked Valid (Y - though analysis is
blocked), Silent Error (Y), Intercepted (Y)")
```

```
SentinelAI Execution: Aggregating TransactionAmt by ProductCD (same as
Baseline).

--- SentinelAI ALERT ---

Rule ID: AE-IEEE-001 - Aggregation Error Triggered!
Trigger Condition: Analytical operation involving aggregation on
TransactionAmt without explicit validation of underlying units or
definitions.
Validation Check: Aggregation of TransactionAmt detected without prior
formal declaration of consistent units.
System Action: Blocking result presentation. Output labeled 'Invalid
Aggregation Semantics'.
Human Involvement: Analyst must explicitly state the assumed currency
and scaling for TransactionAmt.
--- End SentinelAI ALERT ---

Output Executed (Y), Looked Valid (Y - though analysis is blocked),
Silent Error (Y), Intercepted (Y)
```

TASK B — TEMPORAL TASK: Baseline Execution

This cell attempts to use `TransactionDT` for time-based grouping and trend generation without explicit validation of its calendar semantics, simulating a scenario where a silent temporal error might occur.

```
print("Baseline Execution: Observing fraud trends over time using
TransactionDT.")
print("Attempting to group by TransactionDT and calculate fraud
rates:")

# Performing a simple downstream analysis that assumes calendar
semantics for TransactionDT
# This operation will run without error, appearing 'valid' to the
analyst
# but silently misinterpreting TransactionDT as a real timestamp.
fraud_trend = merged_df.groupby('TransactionDT')
['isFraud'].mean().reset_index()
display(fraud_trend.head())

print("\nOutput Executed (Y), Looked Valid (Y), Silent Error (Y -
TransactionDT treated as real timestamp), Intercepted (N)")
```

```
Baseline Execution: Observing fraud trends over time using
TransactionDT.
Attempting to group by TransactionDT and calculate fraud rates:
```


```
{"summary":"{\n  \"name\": \"print(\\\"\\\\nOutput Executed (Y),
Looked Valid (Y), Silent Error (Y - TransactionDT treated as real
timestamp), Intercepted (N)\\\")\",\n  \"rows\": 5,\n  \"fields\": [\n
{\n      \"column\": \"TransactionDT\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 51,\n        \"min\": 86400,\
n        \"max\": 86506,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          86401,\n          86506,\n          86469\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"isFraud\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
Output Executed (Y), Looked Valid (Y), Silent Error (Y - TransactionDT
treated as real timestamp), Intercepted (N)
```

TASK B — TEMPORAL TASK: SentinelAI Execution

This cell repeats the same analytical intent, but conceptually, the SentinelAI validation rule (TE-IEEE-001) would intercept the silent error related to the misinterpretation of `TransactionDT` as a real timestamp. The code simulates the system's reaction (blocking/warning).

```python
print("SentinelAI Execution: Observing fraud trends over time using
TransactionDT (same as Baseline).")

# Simulate SentinelAI's Temporal Error validation rule (TE-IEEE-001)
# This check would happen before or during the execution of a function
that assumes calendar semantics.
# For simulation, we assume any grouping by TransactionDT implies
calendar semantics if not explicitly clarified.

# Check for potential temporal misinterpretation
# In a real system, this would be more complex, involving static
analysis or runtime monitoring of function calls.
# For this experiment, we'll assume the intent to group by
TransactionDT triggers the rule.

# Since we're simulating, we'll assume the rule triggers if any
operation uses TransactionDT for grouping/trending without explicit
approval.
print("\n--- SentinelAI ALERT ---\n")
print("Rule ID: TE-IEEE-001 - Temporal Error Triggered!")
print("Trigger Condition: Analytical operation attempts to interpret
```

```
TransactionDT as a real timestamp.")
print("Validation Check: Grouping/trending operation detected on
TransactionDT assuming calendar semantics.")
print("System Action: Blocking result presentation. Output labeled
'Invalid Temporal Semantics'.")
print("Human Involvement: Analyst must explicitly state intended
temporal interpretation (e.g., relative ordering only vs. assumed
calendar mapping).")
print("--- End SentinelAI ALERT ---")

print("\nOutput Executed (Y), Looked Valid (Y - though analysis is
blocked), Silent Error (Y), Intercepted (Y)")
```

```
SentinelAI Execution: Observing fraud trends over time using
TransactionDT (same as Baseline).

--- SentinelAI ALERT ---

Rule ID: TE-IEEE-001 - Temporal Error Triggered!
Trigger Condition: Analytical operation attempts to interpret
TransactionDT as a real timestamp.
Validation Check: Grouping/trending operation detected on
TransactionDT assuming calendar semantics.
System Action: Blocking result presentation. Output labeled 'Invalid
Temporal Semantics'.
Human Involvement: Analyst must explicitly state intended temporal
interpretation (e.g., relative ordering only vs. assumed calendar
mapping).
--- End SentinelAI ALERT ---

Output Executed (Y), Looked Valid (Y - though analysis is blocked),
Silent Error (Y), Intercepted (Y)
```

TASK A — JOIN TASK: Baseline Execution

This cell performs the left join and then uses `DeviceType` and `DeviceInfo` in a downstream analysis (grouping and counting) without explicit null handling or validation, simulating a common baseline workflow where silent errors might pass unnoticed.

```
# Perform left join (merged_df already exists from earlier steps)
# Downstream use: Group by DeviceType and DeviceInfo and count entries
# This operation runs and appears valid, even with NaN values.

print("Baseline Execution: Analyzing fraud behavior by device
characteristics (DeviceType, DeviceInfo).")
print("Attempting to group by DeviceType and DeviceInfo and count
occurrences:")

# Performing a simple downstream analysis without handling nulls
explicitly
```

```
# This operation will run without error, appearing 'valid' to the
analyst
# but silently misinterpreting NaN as 'unknown device' if not handled.
fraud_by_device = merged_df.groupby(['DeviceType', 'DeviceInfo'])
['isFraud'].count().reset_index()
display(fraud_by_device.head())

print("\nOutput Executed (Y), Looked Valid (Y), Silent Error (Y -
nulls implicitly treated as a category), Intercepted (N)")
```

```
Baseline Execution: Analyzing fraud behavior by device characteristics
(DeviceType, DeviceInfo).
Attempting to group by DeviceType and DeviceInfo and count
occurrences:
```


```
{"summary":"{\n  \"name\": \"print(\\\"\\\\nOutput Executed (Y),
Looked Valid (Y), Silent Error (Y - nulls implicitly treated as a
category), Intercepted (N)\\\")\",\n  \"rows\": 5,\n  \"fields\": [\n
{\n      \"column\": \"DeviceType\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"desktop\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DeviceInfo\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"ATT-
IE11\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"isFraud\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 3,\n        \"min\": 1,\n        \"max\": 9,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
Output Executed (Y), Looked Valid (Y), Silent Error (Y - nulls
implicitly treated as a category), Intercepted (N)
```

TASK A — JOIN TASK: SentinelAI Execution

This cell repeats the same analytical intent, but conceptually, the SentinelAI validation rule (JE-IEEE-001) would intercept the silent error related to unclarified nulls in identity-related columns. The code simulates the system's reaction (blocking/warning).

```
print("SentinelAI Execution: Analyzing fraud behavior by device
characteristics (DeviceType, DeviceInfo).")
print("Attempting to group by DeviceType and DeviceInfo and count
occurrences (same as Baseline).")

# Simulate SentinelAI's Join Error validation rule (JE-IEEE-001)
# Check for nulls in DeviceType or DeviceInfo after the join
```

```
null_identity_records = merged_df[(merged_df['DeviceType'].isnull()) |
(merged_df['DeviceInfo'].isnull())]

if not null_identity_records.empty:
    print("\n--- SentinelAI ALERT ---\n")
    print("Rule ID: JE-IEEE-001 - Join Error Triggered!")
    print("Trigger Condition: Analytical operation attempts to
interpret DeviceType or DeviceInfo semantics after a left join where
null identity values are present.")
    print(f"Validation Check: Identified {len(null_identity_records)}
records where DeviceType or DeviceInfo is null.")
    print("System Action: Blocking downstream analysis. Semantic
meaning of these nulls is unclear.")
    print("Human Involvement: Analyst must clarify semantic meaning of
nulls before proceeding.")
    print("--- End SentinelAI ALERT ---")

    print("\nOutput Executed (Y), Looked Valid (Y - though analysis is
blocked), Silent Error (Y), Intercepted (Y)")
else:
    print("No null identity records found. Proceeding with analysis.")
    # In a real scenario, the downstream analysis would proceed if no
rule was triggered.
    fraud_by_device = merged_df.groupby(['DeviceType', 'DeviceInfo'])
['isFraud'].count().reset_index()
    display(fraud_by_device.head())
    print("\nOutput Executed (Y), Looked Valid (Y), Silent Error (N),
Intercepted (N)")
```

```
SentinelAI Execution: Analyzing fraud behavior by device
characteristics (DeviceType, DeviceInfo).
Attempting to group by DeviceType and DeviceInfo and count occurrences
(same as Baseline).

--- SentinelAI ALERT ---

Rule ID: JE-IEEE-001 - Join Error Triggered!
Trigger Condition: Analytical operation attempts to interpret
DeviceType or DeviceInfo semantics after a left join where null
identity values are present.
Validation Check: Identified 201525 records where DeviceType or
DeviceInfo is null.
System Action: Blocking downstream analysis. Semantic meaning of these
nulls is unclear.
Human Involvement: Analyst must clarify semantic meaning of nulls
before proceeding.
--- End SentinelAI ALERT ---

Output Executed (Y), Looked Valid (Y - though analysis is blocked),
Silent Error (Y), Intercepted (Y)
```

TASK C — AGGREGATION TASK: Baseline Execution

This cell attempts to aggregate `TransactionAmt` without explicit validation of its units or definitions, simulating a scenario where a silent aggregation error might occur.

```
print("Baseline Execution: Aggregating TransactionAmt by ProductCD.")
print("Attempting to calculate average TransactionAmt per ProductCD:")

# Performing a simple aggregation without prior formal declaration of
consistent units
# This operation will run without error, appearing 'valid' to the
analyst
# but silently misinterpreting TransactionAmt due to unvalidated
currency/scaling.
avg_amt_by_product = merged_df.groupby('ProductCD')
['TransactionAmt'].mean().reset_index()
display(avg_amt_by_product.head())

print("\nOutput Executed (Y), Looked Valid (Y), Silent Error (Y -
unverified units/definitions), Intercepted (N)")
```

```
Baseline Execution: Aggregating TransactionAmt by ProductCD.
Attempting to calculate average TransactionAmt per ProductCD:
```


```
{"summary":"{\n  \"name\": \"print(\\\"\\\\nOutput Executed (Y),
Looked Valid (Y), Silent Error (Y - unverified units/definitions),
Intercepted (N)\\\")\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n
\"column\": \"ProductCD\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \"H\",\n          \"W\",\n          \"R\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"TransactionAmt\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
59.353571312731496,\n        \"min\": 44.504183924895514,\n
\"max\": 174.0778567780964,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          71.30724426647251,\n
151.0000238169819,\n          174.0778567780964\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
Output Executed (Y), Looked Valid (Y), Silent Error (Y - unverified
units/definitions), Intercepted (N)
```

TASK C — AGGREGATION TASK: SentinelAI Execution

This cell repeats the same analytical intent, but conceptually, the SentinelAI validation rule (AE-IEEE-001) would intercept the silent error related to unvalidated units or definitions in `TransactionAmt` aggregation. The code simulates the system's reaction (blocking/warning).

```
print("SentinelAI Execution: Aggregating TransactionAmt by ProductCD (same as Baseline).")

# Simulate SentinelAI's Aggregation Error validation rule (AE-IEEE-001)
# This check would happen before or during the execution of a function that assumes validated units/definitions.

print("\n--- SentinelAI ALERT ---\n")
print("Rule ID: AE-IEEE-001 - Aggregation Error Triggered!")
print("Trigger Condition: Analytical operation involving aggregation on TransactionAmt without explicit validation of underlying units or definitions.")
print("Validation Check: Aggregation of TransactionAmt detected without prior formal declaration of consistent units.")
print("System Action: Blocking result presentation. Output labeled 'Invalid Aggregation Semantics'.")
print("Human Involvement: Analyst must explicitly state the assumed currency and scaling for TransactionAmt.")
print("--- End SentinelAI ALERT ---")

print("\nOutput Executed (Y), Looked Valid (Y - though analysis is blocked), Silent Error (Y), Intercepted (Y)")
```

```
SentinelAI Execution: Aggregating TransactionAmt by ProductCD (same as Baseline).

--- SentinelAI ALERT ---

Rule ID: AE-IEEE-001 - Aggregation Error Triggered!
Trigger Condition: Analytical operation involving aggregation on TransactionAmt without explicit validation of underlying units or definitions.
Validation Check: Aggregation of TransactionAmt detected without prior formal declaration of consistent units.
System Action: Blocking result presentation. Output labeled 'Invalid Aggregation Semantics'.
Human Involvement: Analyst must explicitly state the assumed currency and scaling for TransactionAmt.
--- End SentinelAI ALERT ---

Output Executed (Y), Looked Valid (Y - though analysis is blocked), Silent Error (Y), Intercepted (Y)
```

TASK B — TEMPORAL TASK: Baseline Execution

This cell attempts to use `TransactionDT` for time-based grouping and trend generation without explicit validation of its calendar semantics, simulating a scenario where a silent temporal error might occur.

```
print("Baseline Execution: Observing fraud trends over time using
TransactionDT.")
print("Attempting to group by TransactionDT and calculate fraud
rates:")

# Performing a simple downstream analysis that assumes calendar
semantics for TransactionDT
# This operation will run without error, appearing 'valid' to the
analyst
# but silently misinterpreting TransactionDT as a real timestamp.
fraud_trend = merged_df.groupby('TransactionDT')
['isFraud'].mean().reset_index()
display(fraud_trend.head())

print("\nOutput Executed (Y), Looked Valid (Y), Silent Error (Y -
TransactionDT treated as real timestamp), Intercepted (N)")
```

```
Baseline Execution: Observing fraud trends over time using
TransactionDT.
Attempting to group by TransactionDT and calculate fraud rates:
```


```
{"summary":"{\n  \"name\": \"print(\\\"\\\\nOutput Executed (Y),
Looked Valid (Y), Silent Error (Y - TransactionDT treated as real
timestamp), Intercepted (N)\\\")\",\n  \"rows\": 5,\n  \"fields\": [\n
{\n      \"column\": \"TransactionDT\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 51,\n        \"min\": 86400,\
n        \"max\": 86506,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          86401,\n          86506,\n          86469\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"isFraud\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
Output Executed (Y), Looked Valid (Y), Silent Error (Y - TransactionDT
treated as real timestamp), Intercepted (N)
```

TASK B — TEMPORAL TASK: SentinelAI Execution

This cell repeats the same analytical intent, but conceptually, the SentinelAI validation rule (TE-IEEE-001) would intercept the silent error related to the misinterpretation of `TransactionDT` as a real timestamp. The code simulates the system's reaction (blocking/warning).

```
print("SentinelAI Execution: Observing fraud trends over time using
TransactionDT (same as Baseline).")

# Simulate SentinelAI's Temporal Error validation rule (TE-IEEE-001)
# This check would happen before or during the execution of a function
that assumes calendar semantics.
# For simulation, we assume any grouping by TransactionDT implies
calendar semantics if not explicitly clarified.

# Check for potential temporal misinterpretation
# In a real system, this would be more complex, involving static
analysis or runtime monitoring of function calls.
# For this experiment, we'll assume the intent to group by
TransactionDT triggers the rule.

# Since we're simulating, we'll assume the rule triggers if any
operation uses TransactionDT for grouping/trending without explicit
approval.
print("\n--- SentinelAI ALERT ---\n")
print("Rule ID: TE-IEEE-001 - Temporal Error Triggered!")
print("Trigger Condition: Analytical operation attempts to interpret
TransactionDT as a real timestamp.")
print("Validation Check: Grouping/trending operation detected on
TransactionDT assuming calendar semantics.")
print("System Action: Blocking result presentation. Output labeled
'Invalid Temporal Semantics'.")
print("Human Involvement: Analyst must explicitly state intended
temporal interpretation (e.g., relative ordering only vs. assumed
calendar mapping).")
print("--- End SentinelAI ALERT ---")

print("\nOutput Executed (Y), Looked Valid (Y - though analysis is
blocked), Silent Error (Y), Intercepted (Y)")
```

```
SentinelAI Execution: Observing fraud trends over time using
TransactionDT (same as Baseline).

--- SentinelAI ALERT ---

Rule ID: TE-IEEE-001 - Temporal Error Triggered!
Trigger Condition: Analytical operation attempts to interpret
TransactionDT as a real timestamp.
Validation Check: Grouping/trending operation detected on
TransactionDT assuming calendar semantics.
```

```
System Action: Blocking result presentation. Output labeled 'Invalid
Temporal Semantics'.
Human Involvement: Analyst must explicitly state intended temporal
interpretation (e.g., relative ordering only vs. assumed calendar
mapping).
--- End SentinelAI ALERT ---

Output Executed (Y), Looked Valid (Y - though analysis is blocked),
Silent Error (Y), Intercepted (Y)
```

Table 1: Silent Error Interception

Task Type	Baseline Silent Errors	Intercepted by SentinelAI
Join	1	1
Temporal	1	1
Aggregate	1	1

```
# □ EXPERIMENT 2 — STEP 2 Execution & Logging (Override Governance)
```

Experiment 2 - Step 2: Execution & Logging Setup

```
import pandas as pd

# Create a simple log table as a DataFrame for Experiment 2
log_columns_exp2 = [
    'Task',
    'Condition',
    'Validation Triggered (Y/N)',
    'Override Used (Y/N)',
    'Justification Provided (Y/N)',
    'Justification Type',
    'Notes'
]
experiment_log_exp2 = pd.DataFrame(columns=log_columns_exp2)

print("Empty Experiment 2 Log Table created. Please fill this manually
as you perform the tasks.")
display(experiment_log_exp2)
```

```
Empty Experiment 2 Log Table created. Please fill this manually as you
perform the tasks.
```

```
{"repr_error":"Out of range float values are not JSON compliant:
nan","type":"dataframe","variable_name":"experiment_log_exp2"}
```

TASK A — JOIN TASK: SentinelAI Governance Enabled

This cell repeats the Join Task, but under SentinelAI governance, the validation rule (JE-IEEE-001) will trigger due to null identity values, requiring a formal human response (simulated as an override with justification).

```
print("SentinelAI Governance Enabled: Analyzing fraud behavior by device characteristics (DeviceType, DeviceInfo).")
print("Attempting to group by DeviceType and DeviceInfo and count occurrences.")

# Simulate SentinelAI's Join Error validation rule (JE-IEEE-001)
null_identity_records = merged_df[(merged_df['DeviceType'].isnull()) | (merged_df['DeviceInfo'].isnull())]

if not null_identity_records.empty:
    print("\n--- SentinelAI ALERT ---")
    print("Rule ID: JE-IEEE-001 - Join Error Triggered!")
    print(f"Validation Check: Identified {len(null_identity_records)} records where DeviceType or DeviceInfo is null. Blocking analysis.")
    print("Human Involvement: Analyst must clarify semantic meaning of nulls. Options: Accept, Override, Defer.")
    print("--- Simulating Override with Justification (Domain knowledge assumption) ---")
    # Simulate user choosing to override with justification
    print("Justification: 'Confirmed nulls mean no identity record, not unknown device, safe to proceed with analysis.'")

    # Proceed with the analysis after simulated override
    fraud_by_device_gov_A = merged_df.groupby(['DeviceType', 'DeviceInfo'])['isFraud'].count().reset_index()
    display(fraud_by_device_gov_A.head())

    print("\n--- Logging for Experiment 2 (SentinelAI, Task A) ---")
    print("Task: Join, Condition: SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification Type: Domain knowledge assumption, Notes: Overrode JE-IEEE-001, assuming null means no record.")
else:
    print("No null identity records found. Proceeding with analysis without validation trigger.")
    fraud_by_device_gov_A = merged_df.groupby(['DeviceType', 'DeviceInfo'])['isFraud'].count().reset_index()
    display(fraud_by_device_gov_A.head())
    print("\n--- Logging for Experiment 2 (SentinelAI, Task A) ---")
    print("Task: Join, Condition: SentinelAI, Validation Triggered (Y/N): N, Override Used (Y/N): N, Justification Provided (Y/N): NA, Justification Type: NA, Notes: No validation triggered.")
```

```
SentinelAI Governance Enabled: Analyzing fraud behavior by device
characteristics (DeviceType, DeviceInfo).
Attempting to group by DeviceType and DeviceInfo and count
occurrences.

--- SentinelAI ALERT ---
Rule ID: JE-IEEE-001 - Join Error Triggered!
Validation Check: Identified 201525 records where DeviceType or
DeviceInfo is null. Blocking analysis.
Human Involvement: Analyst must clarify semantic meaning of nulls.
Options: Accept, Override, Defer.
--- Simulating Override with Justification (Domain knowledge
assumption) ---
Justification: 'Confirmed nulls mean no identity record, not unknown
device, safe to proceed with analysis.'
```


```
{"summary":"{\n  \"name\": \"    print(\\\"Task: Join, Condition:
SentinelAI, Validation Triggered (Y/N): N, Override Used (Y/N): N,
Justification Provided (Y/N): NA, Justification Type: NA, Notes: No
validation triggered\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n
\"column\": \"DeviceType\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"desktop\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DeviceInfo\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"ATT-
IE11\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"isFraud\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 3,\n        \"min\": 1,\n        \"max\": 9,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
--- Logging for Experiment 2 (SentinelAI, Task A) ---
Task: Join, Condition: SentinelAI, Validation Triggered (Y/N): Y,
Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification
Type: Domain knowledge assumption, Notes: Overrode JE-IEEE-001,
assuming null means no record.
```

TASK B — TEMPORAL TASK: SentinelAI Governance Enabled

This cell repeats the Temporal Task, but under SentinelAI governance, the validation rule (TE-IEEE-001) will trigger due to ambiguous temporal semantics, requiring a formal human response (simulated as an override with justification).

```
print("SentinelAI Governance Enabled: Observing fraud trends over time using TransactionDT.")
print("Attempting to group by TransactionDT and calculate fraud rates.")

# Simulate SentinelAI's Temporal Error validation rule (TE-IEEE-001)
print("\n--- SentinelAI ALERT ---")
print("Rule ID: TE-IEEE-001 - Temporal Error Triggered!")
print("Validation Check: Grouping/trending on TransactionDT detected, assuming calendar semantics. Blocking result.")
print("Human Involvement: Analyst must explicitly state intended temporal interpretation. Options: Accept, Override, Defer.")
print("--- Simulating Override with Justification (External evidence reference) ---")
# Simulate user choosing to override with justification
print("Justification: 'Internal documentation confirms TransactionDT is a seconds offset from 2017-12-01 00:00:00 UTC, allowing relative trending.'")

# Proceed with the analysis after simulated override
fraud_trend_gov_B = merged_df.groupby('TransactionDT')['isFraud'].mean().reset_index()
display(fraud_trend_gov_B.head())

print("\n--- Logging for Experiment 2 (SentinelAI, Task B) ---")
print("Task: Temporal, Condition: SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification Type: External evidence reference, Notes: Overrode TE-IEEE-001, used external doc for temporal mapping.")
```

```
SentinelAI Governance Enabled: Observing fraud trends over time using TransactionDT.
Attempting to group by TransactionDT and calculate fraud rates.

--- SentinelAI ALERT ---
Rule ID: TE-IEEE-001 - Temporal Error Triggered!
Validation Check: Grouping/trending on TransactionDT detected, assuming calendar semantics. Blocking result.
Human Involvement: Analyst must explicitly state intended temporal interpretation. Options: Accept, Override, Defer.
--- Simulating Override with Justification (External evidence reference) ---
Justification: 'Internal documentation confirms TransactionDT is a seconds offset from 2017-12-01 00:00:00 UTC, allowing relative trending.'
```


```
{"summary":"{\n  \"name\": \"print(\\\"Task: Temporal, Condition: SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification Type: External evidence reference, Notes: Overrode TE-IEEE-001, used external doc for temporal
```

```
mapping\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n      \"column\":
\"TransactionDT\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 51,\n        \"min\": 86400,\n
\"max\": 86506,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          86401,\n          86506,\n          86469\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"isFraud\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
--- Logging for Experiment 2 (SentinelAI, Task B) ---
Task: Temporal, Condition: SentinelAI, Validation Triggered (Y/N): Y,
Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification
Type: External evidence reference, Notes: Overrode TE-IEEE-001, used
external doc for temporal mapping.
```

TASK C — AGGREGATION TASK: SentinelAI Governance Enabled

This cell repeats the Aggregation Task, but under SentinelAI governance, the validation rule (AE-IEEE-001) will trigger due to unvalidated units/definitions, requiring a formal human response (simulated as an override with justification).

```
print("SentinelAI Governance Enabled: Aggregating TransactionAmt by
ProductCD.")
print("Attempting to calculate average TransactionAmt per ProductCD.")

# Simulate SentinelAI's Aggregation Error validation rule (AE-IEEE-
001)
print("\n--- SentinelAI ALERT ---")
print("Rule ID: AE-IEEE-001 - Aggregation Error Triggered!")
print("Validation Check: Aggregation of TransactionAmt detected
without formal declaration of units. Blocking result.")
print("Human Involvement: Analyst must explicitly state assumed
currency and scaling. Options: Accept, Override, Defer.")
print("--- Simulating Override with Justification (Domain knowledge
assumption) ---")
# Simulate user choosing to override with justification
print("Justification: 'All TransactionAmt values are known to be USD,
with a consistent scaling factor of 1.'")

# Proceed with the analysis after simulated override
avg_amt_by_product_gov_C = merged_df.groupby('ProductCD')
['TransactionAmt'].mean().reset_index()
display(avg_amt_by_product_gov_C.head())

print("\n--- Logging for Experiment 2 (SentinelAI, Task C) ---")
```

```
print("Task: Aggregate, Condition: SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification Type: Domain knowledge assumption, Notes: Overrode AE-IEEE-001, confirmed USD units and scaling.")
```

```
SentinelAI Governance Enabled: Aggregating TransactionAmt by ProductCD.
Attempting to calculate average TransactionAmt per ProductCD.

--- SentinelAI ALERT ---
Rule ID: AE-IEEE-001 - Aggregation Error Triggered!
Validation Check: Aggregation of TransactionAmt detected without formal declaration of units. Blocking result.
Human Involvement: Analyst must explicitly state assumed currency and scaling. Options: Accept, Override, Defer.
--- Simulating Override with Justification (Domain knowledge assumption) ---
Justification: 'All TransactionAmt values are known to be USD, with a consistent scaling factor of 1.'
```


```
{"summary":"{\n  \"name\": \"print(\\\"Task: Aggregate, Condition: SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification Type: Domain knowledge assumption, Notes: Overrode AE-IEEE-001, confirmed USD units and scaling\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n      \"column\": \"ProductCD\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"H\",\n          \"W\",\n          \"R\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"TransactionAmt\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 59.353571312731496,\n        \"min\": 44.504183924895514,\n        \"max\": 174.0778567780964,\n        \"num_unique_values\": 5,\n        \"samples\": [\n          71.30724426647251,\n          151.0000238169819,\n          174.0778567780964\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
--- Logging for Experiment 2 (SentinelAI, Task C) ---
Task: Aggregate, Condition: SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification Type: Domain knowledge assumption, Notes: Overrode AE-IEEE-001, confirmed USD units and scaling.
```

TASK A — JOIN TASK: No Governance (Baseline)

This cell performs the join task similar to the baseline execution in Experiment 1, simulating a 'No Governance' workflow where an analyst might proceed despite a potential issue without formal justification.

```
print("No Governance (Baseline) Execution: Analyzing fraud behavior by device characteristics (DeviceType, DeviceInfo).")
print("Attempting to group by DeviceType and DeviceInfo and count occurrences:")

# Performing a simple downstream analysis without handling nulls explicitly
# This operation will run without error, appearing 'valid' to the analyst.
fraud_by_device_nogov_A = merged_df.groupby(['DeviceType', 'DeviceInfo'])['isFraud'].count().reset_index()
display(fraud_by_device_nogov_A.head())

print("\n--- Logging for Experiment 2 (No Governance, Task A) ---")
print("Task: Join, Condition: No Governance, Validation Triggered (Y/N): Y (conceptually), Override Used (Y/N): Y, Justification Provided (Y/N): N, Justification Type: NA, Notes: Proceeded with analysis despite nulls in identity columns.")
```

```
No Governance (Baseline) Execution: Analyzing fraud behavior by device characteristics (DeviceType, DeviceInfo).
Attempting to group by DeviceType and DeviceInfo and count occurrences:
```


```
{"summary":"{\n  \"name\": \"print(\\\"Task: Join, Condition: No Governance, Validation Triggered (Y/N): Y (conceptually), Override Used (Y/N): Y, Justification Provided (Y/N): N, Justification Type: NA, Notes: Proceeded with analysis despite nulls in identity columns\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n      \"column\": \"DeviceType\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 1,\n        \"samples\": [\n          \"desktop\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"DeviceInfo\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"ATT-IE11\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"isFraud\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 3,\n        \"min\": 1,\n        \"max\": 9,\n        \"num_unique_values\": 4,\n        \"samples\": [\n          2\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
--- Logging for Experiment 2 (No Governance, Task A) ---
Task: Join, Condition: No Governance, Validation Triggered (Y/N): Y (conceptually), Override Used (Y/N): Y, Justification Provided (Y/N): N, Justification Type: NA, Notes: Proceeded with analysis despite nulls in identity columns.
```

TASK B — TEMPORAL TASK: No Governance (Baseline)

This cell performs the temporal task similar to the baseline execution in Experiment 1, simulating a 'No Governance' workflow where an analyst might proceed despite a potential issue without formal justification.

```
print("No Governance (Baseline) Execution: Observing fraud trends over time using TransactionDT.")
print("Attempting to group by TransactionDT and calculate fraud rates:")

# Performing a simple downstream analysis that assumes calendar semantics for TransactionDT
fraud_trend_nogov_B = merged_df.groupby('TransactionDT')['isFraud'].mean().reset_index()
display(fraud_trend_nogov_B.head())

print("\n--- Logging for Experiment 2 (No Governance, Task B) ---")
print("Task: Temporal, Condition: No Governance, Validation Triggered (Y/N): Y (conceptually), Override Used (Y/N): Y, Justification Provided (Y/N): N, Justification Type: NA, Notes: Proceeded with analysis despite unverified temporal semantics.")
```

```
No Governance (Baseline) Execution: Observing fraud trends over time using TransactionDT.
Attempting to group by TransactionDT and calculate fraud rates:
```


```
{"summary":"{\n  \"name\": \"print(\\\"Task: Temporal, Condition: No Governance, Validation Triggered (Y/N): Y (conceptually), Override Used (Y/N): Y, Justification Provided (Y/N): N, Justification Type: NA, Notes: Proceeded with analysis despite unverified temporal semantics\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n      \"column\": \"TransactionDT\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 51,\n        \"min\": 86400,\n        \"max\": 86506,\n        \"num_unique_values\": 5,\n        \"samples\": [\n          86401,\n          86506,\n          86469\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"isFraud\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n        \"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
--- Logging for Experiment 2 (No Governance, Task B) ---
Task: Temporal, Condition: No Governance, Validation Triggered (Y/N): Y (conceptually), Override Used (Y/N): Y, Justification Provided (Y/N): N, Justification Type: NA, Notes: Proceeded with analysis despite unverified temporal semantics.
```

TASK C — AGGREGATION TASK: No Governance (Baseline)

This cell performs the aggregation task similar to the baseline execution in Experiment 1, simulating a 'No Governance' workflow where an analyst might proceed despite a potential issue without formal justification.

```
print("No Governance (Baseline) Execution: Aggregating TransactionAmt by ProductCD.")
print("Attempting to calculate average TransactionAmt per ProductCD:")

# Performing a simple aggregation without prior formal declaration of consistent units
avg_amt_by_product_nogov_C = merged_df.groupby('ProductCD')['TransactionAmt'].mean().reset_index()
display(avg_amt_by_product_nogov_C.head())

print("\n--- Logging for Experiment 2 (No Governance, Task C) ---")
print("Task: Aggregate, Condition: No Governance, Validation Triggered (Y/N): Y (conceptually), Override Used (Y/N): Y, Justification Provided (Y/N): N, Justification Type: NA, Notes: Proceeded with aggregation despite unverified units/definitions.")
```

```
No Governance (Baseline) Execution: Aggregating TransactionAmt by ProductCD.
Attempting to calculate average TransactionAmt per ProductCD:
```


```
{"summary":"{\n  \"name\": \"print(\\\"Task: Aggregate, Condition: No Governance, Validation Triggered (Y/N): Y (conceptually), Override Used (Y/N): Y, Justification Provided (Y/N): N, Justification Type: NA, Notes: Proceeded with aggregation despite unverified units/definitions\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n      \"column\": \"ProductCD\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"H\",\n          \"W\",\n          \"R\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"TransactionAmt\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 59.353571312731496,\n        \"min\": 44.504183924895514,\n        \"max\": 174.0778567780964,\n        \"num_unique_values\": 5,\n        \"samples\": [\n          71.30724426647251,\n          151.0000238169819,\n          174.0778567780964\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
--- Logging for Experiment 2 (No Governance, Task C) ---
Task: Aggregate, Condition: No Governance, Validation Triggered (Y/N): Y (conceptually), Override Used (Y/N): Y, Justification Provided (Y/N): N, Justification Type: NA, Notes: Proceeded with aggregation despite unverified units/definitions.
```

Table 2: Override Governance Impact

Condition	Total Overrides	Justified Overrides (%)
No Governance	3	0%
SentinelAI	3	100%

TASK A — JOIN TASK: SentinelAI Governance Enabled

This cell repeats the Join Task, but under SentinelAI governance, the validation rule (JE-IEEE-001) will trigger due to null identity values, requiring a formal human response (simulated as an override with justification).

```
print("SentinelAI Governance Enabled: Analyzing fraud behavior by device characteristics (DeviceType, DeviceInfo).")
print("Attempting to group by DeviceType and DeviceInfo and count occurrences.")

# Simulate SentinelAI's Join Error validation rule (JE-IEEE-001)
null_identity_records = merged_df[(merged_df['DeviceType'].isnull()) | (merged_df['DeviceInfo'].isnull())]

if not null_identity_records.empty:
    print("\n--- SentinelAI ALERT ---")
    print("Rule ID: JE-IEEE-001 - Join Error Triggered!")
    print(f"Validation Check: Identified {len(null_identity_records)} records where DeviceType or DeviceInfo is null. Blocking analysis.")
    print("Human Involvement: Analyst must clarify semantic meaning of nulls. Options: Accept, Override, Defer.")
    print("--- Simulating Override with Justification (Domain knowledge assumption) ---")
    # Simulate user choosing to override with justification
    print("Justification: 'Confirmed nulls mean no identity record, not unknown device, safe to proceed with analysis.'")

    # Proceed with the analysis after simulated override
    fraud_by_device_gov_A = merged_df.groupby(['DeviceType', 'DeviceInfo'])['isFraud'].count().reset_index()
    display(fraud_by_device_gov_A.head())

    print("\n--- Logging for Experiment 2 (SentinelAI, Task A) ---")
    print("Task: Join, Condition: SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification Type: Domain knowledge assumption, Notes: Overrode JE-IEEE-001, assuming null means no record.")
else:
    print("No null identity records found. Proceeding with analysis
```

```
without validation trigger.")
    fraud_by_device_gov_A = merged_df.groupby(['DeviceType',
'DeviceInfo'])['isFraud'].count().reset_index()
    display(fraud_by_device_gov_A.head())
    print("\n--- Logging for Experiment 2 (SentinelAI, Task A) ---")
    print("Task: Join, Condition: SentinelAI, Validation Triggered
(Y/N): N, Override Used (Y/N): N, Justification Provided (Y/N): NA,
Justification Type: NA, Notes: No validation triggered.")

SentinelAI Governance Enabled: Analyzing fraud behavior by device
characteristics (DeviceType, DeviceInfo).
Attempting to group by DeviceType and DeviceInfo and count
occurrences.

--- SentinelAI ALERT ---
Rule ID: JE-IEEE-001 - Join Error Triggered!
Validation Check: Identified 201525 records where DeviceType or
DeviceInfo is null. Blocking analysis.
Human Involvement: Analyst must clarify semantic meaning of nulls.
Options: Accept, Override, Defer.
--- Simulating Override with Justification (Domain knowledge
assumption) ---
Justification: 'Confirmed nulls mean no identity record, not unknown
device, safe to proceed with analysis.'

{"summary":"{\n  \"name\": \"    print(\\\"Task: Join, Condition:
SentinelAI, Validation Triggered (Y/N): N, Override Used (Y/N): N,
Justification Provided (Y/N): NA, Justification Type: NA, Notes: No
validation triggered\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n
\"column\": \"DeviceType\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"desktop\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DeviceInfo\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"ATT-
IE11\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"isFraud\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 3,\n        \"min\": 1,\n        \"max\": 9,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}


--- Logging for Experiment 2 (SentinelAI, Task A) ---
Task: Join, Condition: SentinelAI, Validation Triggered (Y/N): Y,
Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification
Type: Domain knowledge assumption, Notes: Overrode JE-IEEE-001,
assuming null means no record.
```

TASK B — TEMPORAL TASK: SentinelAI Governance Enabled

This cell repeats the Temporal Task, but under SentinelAI governance, the validation rule (TE-IEEE-001) will trigger due to ambiguous temporal semantics, requiring a formal human response (simulated as an override with justification).

```
print("SentinelAI Governance Enabled: Observing fraud trends over time using TransactionDT.")
print("Attempting to group by TransactionDT and calculate fraud rates.")

# Simulate SentinelAI's Temporal Error validation rule (TE-IEEE-001)
print("\n--- SentinelAI ALERT ---")
print("Rule ID: TE-IEEE-001 - Temporal Error Triggered!")
print("Validation Check: Grouping/trending on TransactionDT detected, assuming calendar semantics. Blocking result.")
print("Human Involvement: Analyst must explicitly state intended temporal interpretation. Options: Accept, Override, Defer.")
print("--- Simulating Override with Justification (External evidence reference) ---")
# Simulate user choosing to override with justification
print("Justification: 'Internal documentation confirms TransactionDT is a seconds offset from 2017-12-01 00:00:00 UTC, allowing relative trending.'")

# Proceed with the analysis after simulated override
fraud_trend_gov_B = merged_df.groupby('TransactionDT')['isFraud'].mean().reset_index()
display(fraud_trend_gov_B.head())

print("\n--- Logging for Experiment 2 (SentinelAI, Task B) ---")
print("Task: Temporal, Condition: SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification Type: External evidence reference, Notes: Overrode TE-IEEE-001, used external doc for temporal mapping.")
```

```
SentinelAI Governance Enabled: Observing fraud trends over time using TransactionDT.
Attempting to group by TransactionDT and calculate fraud rates.

--- SentinelAI ALERT ---
Rule ID: TE-IEEE-001 - Temporal Error Triggered!
Validation Check: Grouping/trending on TransactionDT detected, assuming calendar semantics. Blocking result.
Human Involvement: Analyst must explicitly state intended temporal interpretation. Options: Accept, Override, Defer.
--- Simulating Override with Justification (External evidence reference) ---
Justification: 'Internal documentation confirms TransactionDT is a
```

```
seconds offset from 2017-12-01 00:00:00 UTC, allowing relative
trending.'

{"summary":"{\n  \"name\": \"print(\\\"Task: Temporal, Condition:
SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y,
Justification Provided (Y/N): Y, Justification Type: External evidence
reference, Notes: Overrode TE-IEEE-001, used external doc for temporal
mapping\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n      \"column\":
\"TransactionDT\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 51,\n        \"min\": 86400,\n
\"max\": 86506,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          86401,\n          86506,\n          86469\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"isFraud\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}


--- Logging for Experiment 2 (SentinelAI, Task B) ---
Task: Temporal, Condition: SentinelAI, Validation Triggered (Y/N): Y,
Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification
Type: External evidence reference, Notes: Overrode TE-IEEE-001, used
external doc for temporal mapping.
```

TASK C — AGGREGATION TASK: SentinelAI Governance Enabled

This cell repeats the Aggregation Task, but under SentinelAI governance, the validation rule (AE-IEEE-001) will trigger due to unvalidated units/definitions, requiring a formal human response (simulated as an override with justification).

```
print("SentinelAI Governance Enabled: Aggregating TransactionAmt by
ProductCD.")
print("Attempting to calculate average TransactionAmt per ProductCD.")

# Simulate SentinelAI's Aggregation Error validation rule (AE-IEEE-
001)
print("\n--- SentinelAI ALERT ---")
print("Rule ID: AE-IEEE-001 - Aggregation Error Triggered!")
print("Validation Check: Aggregation of TransactionAmt detected
without formal declaration of units. Blocking result.")
print("Human Involvement: Analyst must explicitly state assumed
currency and scaling. Options: Accept, Override, Defer.")
print("--- Simulating Override with Justification (Domain knowledge
assumption) ---")
# Simulate user choosing to override with justification
print("Justification: 'All TransactionAmt values are known to be USD,
```

```
with a consistent scaling factor of 1.'")

# Proceed with the analysis after simulated override
avg_amt_by_product_gov_C = merged_df.groupby('ProductCD')
['TransactionAmt'].mean().reset_index()
display(avg_amt_by_product_gov_C.head())

print("\n--- Logging for Experiment 2 (SentinelAI, Task C) ---")
print("Task: Aggregate, Condition: SentinelAI, Validation Triggered
(Y/N): Y, Override Used (Y/N): Y, Justification Provided (Y/N): Y,
Justification Type: Domain knowledge assumption, Notes: Overrode AE-
IEEE-001, confirmed USD units and scaling.")
```

```
SentinelAI Governance Enabled: Aggregating TransactionAmt by
ProductCD.
Attempting to calculate average TransactionAmt per ProductCD.

--- SentinelAI ALERT ---
Rule ID: AE-IEEE-001 - Aggregation Error Triggered!
Validation Check: Aggregation of TransactionAmt detected without
formal declaration of units. Blocking result.
Human Involvement: Analyst must explicitly state assumed currency and
scaling. Options: Accept, Override, Defer.
--- Simulating Override with Justification (Domain knowledge
assumption) ---
Justification: 'All TransactionAmt values are known to be USD, with a
consistent scaling factor of 1.'
```


```
{"summary":"{\n  \"name\": \"print(\\\"Task: Aggregate, Condition:
SentinelAI, Validation Triggered (Y/N): Y, Override Used (Y/N): Y,
Justification Provided (Y/N): Y, Justification Type: Domain knowledge
assumption, Notes: Overrode AE-IEEE-001, confirmed USD units and
scaling\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n      \"column\":
\"ProductCD\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"H\",\n          \"W\",\n          \"R\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TransactionAmt\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
59.353571312731496,\n        \"min\": 44.504183924895514,\n
\"max\": 174.0778567780964,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          71.30724426647251,\n
151.0000238169819,\n          174.0778567780964\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
--- Logging for Experiment 2 (SentinelAI, Task C) ---
Task: Aggregate, Condition: SentinelAI, Validation Triggered (Y/N): Y,
Override Used (Y/N): Y, Justification Provided (Y/N): Y, Justification
```

```
Type: Domain knowledge assumption, Notes: Overrode AE-IEEE-001,
confirmed USD units and scaling.
```

Table 2: Override Governance Impact

Condition	Total Overrides	Justified Overrides (%)
No Governance	3	0%
SentinelAI	3	100%

Override Governance Effect. Table 2 reports analyst override behavior under unguided and governed conditions. In the absence of governance, all overrides proceeded without any recorded justification. When SentinelAI's override governance was enabled, overrides continued to occur, but each required an explicit justification aligned with documented assumptions or external evidence. These results indicate that structured governance does not eliminate analyst autonomy, but instead enforces accountability and traceability for high-risk analytical decisions.

```
# EXPERIMENT 3 — TRUST CALIBRATION OVER TIME


import pandas as pd

# Create a simple log table as a DataFrame for Experiment 3
log_columns_exp3 = [
    'Event',
    'Evidence Quality (High/Low)',
    'Confidence Tone (High/Low)',
    'Trust State',
    'Action Taken',
    'Notes'
]
experiment_log_exp3 = pd.DataFrame(columns=log_columns_exp3)

print("Empty Experiment 3 Log Table created. Please fill this manually
as you perform the tasks.")
display(experiment_log_exp3)
```

```
Empty Experiment 3 Log Table created. Please fill this manually as you
perform the tasks.
```


```
{"repr_error":"Out of range float values are not JSON compliant:
nan","type":"dataframe","variable_name":"experiment_log_exp3"}
```


Table 3: Confidence–Evidence Alignment Over Time

Trust State	High-Confidence on Sparse Data (Before)	After Calibration
Restricted	2	2

Trust State	High-Confidence on Sparse Data (Before)	After Calibration
Calibrated	1	0
Trusted	0	0

Trust Calibration Over Time. Table 3 summarizes the relationship between evidence quality, system confidence, and analyst trust state across observed interactions. Early interactions exhibited occasional high-confidence outputs despite reliance on sparse or high-risk evidence. As SentinelAI accumulated behavioral signals—such as justified overrides, correct deferrals, and alignment with system judgments—confidence expression became more conservative on weak evidence. Importantly, trust calibration did not suppress validation rules or block mechanisms; instead, it modulated interaction friction and confidence tone while preserving explicit uncertainty. These observations support the claim that trust calibration can align confidence expression with evidence quality over time without introducing automation bias.

```
# PROJECT PHASE P2 — MINIMAL EXECUTABLE PIPELINE ( in paper )

# PHASE P2 — MINIMAL EXECUTABLE PIPELINE
```

SentinelAI — Validation-First Analytical Decision System

This notebook implements a minimal executable pipeline of SentinelAI. It demonstrates how analytical intent is validated against explicit assumptions before results are presented, with governed human oversight.

No story. No motivation.

CELL 2 — DECISION-CRITICAL COLUMNS & INITIAL MERGED DATA

Identify 'decision-critical' columns for initial analysis.

Merge `train_transaction` and `train_identity` datasets based on these columns.

Create an 'Assumption Table' to document explicit assumptions for each selected column.

```
transaction_columns = ['TransactionID', 'TransactionDT',
'TransactionAmt', 'ProductCD', 'card1', 'addr1', 'isFraud']
train_txn_selected = train_txn[transaction_columns]

identity_columns = ['TransactionID', 'DeviceType', 'DeviceInfo']
train_id_selected = train_id[identity_columns]

merged_df = pd.merge(train_txn_selected, train_id_selected,
on='TransactionID', how='left')

print("First 5 rows of the merged DataFrame:")
display(merged_df.head())
print("\nSummary information of the merged DataFrame:")
merged_df.info()
```

```
First 5 rows of the merged DataFrame:
```


```
{"summary":"{\n  \"name\": \"merged_df\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"TransactionID\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 2987000,\n        \"max\": 2987004,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          2987001,\
n          2987004,\n          2987002\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TransactionDT\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
51,\n        \"min\": 86400,\n        \"max\": 86506,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          86401,\n
86506,\n          86469\n        ],\n        \"semantic_type\": \"\",\
```

```
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"TransactionAmt\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 14.627029773675856,\n
\"min\": 29.0,\n        \"max\": 68.5,\n        \"num_unique_values\":
4,\n        \"samples\": [\n          29.0,\n          50.0,\n
68.5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ProductCD\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"H\",\n          \"W\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"card1\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 6810,\n
\"min\": 2755,\n        \"max\": 18132,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          2755,\n
4497\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"addr1\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 71.30007012619272,\n        \"min\": 315.0,\n        \"max\":
476.0,\n        \"num_unique_values\": 5,\n        \"samples\": [\n
325.0,\n          420.0\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"isFraud\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 0,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"DeviceType\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"mobile\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"DeviceInfo\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"SAMSUNG SM-G892A Build/NRD90M\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
Summary information of the merged DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 272611 entries, 0 to 272610
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   TransactionID   272611 non-null  int64
 1   TransactionDT   272611 non-null  int64
 2   TransactionAmt  272611 non-null  float64
 3   ProductCD       272611 non-null  object
 4   card1           272611 non-null  int64
 5   addr1           242693 non-null  float64
```

```
 6   isFraud         272611 non-null  int64
 7   DeviceType      81599 non-null   object
 8   DeviceInfo      71107 non-null   object
dtypes: float64(2), int64(4), object(3)
memory usage: 18.7+ MB
```

```
assumption_table_data = [
    {'Column': 'TransactionID', 'Table':
'train_transaction/train_identity', 'Assumed Meaning': 'Unique
transaction identifier used for joins', 'Confidence': 'High', 'Risk if
Wrong': 'Incorrect joins, duplicated or missing records'},
    {'Column': 'TransactionDT', 'Table': 'train_transaction', 'Assumed
Meaning': 'Time offset (likely seconds) from reference point',
'Confidence': 'Low', 'Risk if Wrong': 'Temporal analysis invalid,
trends meaningless'},
    {'Column': 'TransactionAmt', 'Table': 'train_transaction',
'Assumed Meaning': 'Transaction amount (currency and scaling
unspecified)', 'Confidence': 'Medium', 'Risk if Wrong': 'Incorrect
monetary scaling leading to invalid fraud thresholds.'},
    {'Column': 'ProductCD', 'Table': 'train_transaction', 'Assumed
Meaning': 'Product code for the transaction', 'Confidence': 'Medium',
'Risk if Wrong': 'Misinterpretation of product categories affecting
fraud patterns'},
    {'Column': 'card1', 'Table': 'train_transaction', 'Assumed
Meaning': 'Masked credit card number (first block)', 'Confidence':
'Medium', 'Risk if Wrong': 'Misleading insights on card usage
patterns'},
    {'Column': 'addr1', 'Table': 'train_transaction', 'Assumed
Meaning': 'Billing address (masked)', 'Confidence': 'Medium', 'Risk if
Wrong': 'Incorrect geographical analysis'},
    {'Column': 'isFraud', 'Table': 'train_transaction', 'Assumed
Meaning': 'Target variable: 1 for fraud, 0 for legitimate',
'Confidence': 'High', 'Risk if Wrong': 'Incorrect model training and
evaluation'},
    {'Column': 'DeviceType', 'Table': 'train_identity', 'Assumed
Meaning': 'Type of device used for the transaction (e.g., mobile,
desktop)', 'Confidence': 'High', 'Risk if Wrong': 'Misunderstanding of
device-specific fraud behavior'},
    {'Column': 'DeviceInfo', 'Table': 'train_identity', 'Assumed
Meaning': 'Specific device information', 'Confidence': 'Medium', 'Risk
if Wrong': 'Loss of granular device insights, if any'}
]

assumption_df = pd.DataFrame(assumption_table_data)

print("Assumption Table for merged_df columns:")
display(assumption_df)
```

```
Assumption Table for merged_df columns:
```

{"summary":"{\n \"name\": \"assumption_df\",\n \"rows\": 9,\n \"fields\": [\n   {\n   \"column\": \"Column\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 9,\n   \"samples\": [\n   \"DeviceType\",\n   \"TransactionDT\",\n   \"addr1\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Table\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"train_transaction/train_identity\",\n   \"train_transaction\",\n   \"train_identity\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Assumed Meaning\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 9,\n   \"samples\": [\n   \"Type of device used for the transaction (e.g., mobile, desktop)\",\n   \"Time offset (likely seconds) from reference point\",\n   \"Billing address (masked)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Confidence\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"High\",\n   \"Low\",\n   \"Medium\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Risk if Wrong\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 9,\n   \"samples\": [\n   \"Misunderstanding of device-specific fraud behavior\",\n   \"Temporal analysis invalid, trends meaningless\",\n   \"Incorrect geographical analysis\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"assumption_df"}

CELL 1 — DATA LOADING & STRUCTURE (NO SEMANTICS)

Load datasets

Inspect schema, dtypes, null counts

Rules:

Only .info(), .head(), .isnull()

 No interpretation

 No cleaning

 No fixing

```
import pandas as pd

print("Loading train_transaction.csv...")
train_txn = pd.read_csv("train_transaction.csv")
print("train_transaction.csv head:")
display(train_txn.head())
print("train_transaction.csv info:")
train_txn.info()
print("train_transaction.csv null counts:")
display(train_txn.isnull().sum().sort_values(ascending=False).head(20)
) # Display top 20 null counts
```

```
Loading train_transaction.csv...
train_transaction.csv head:
```

```
{"type":"dataframe"}
```

```
train_transaction.csv info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 272611 entries, 0 to 272610
Columns: 394 entries, TransactionID to V339
dtypes: float64(376), int64(4), object(14)
memory usage: 819.5+ MB
train_transaction.csv null counts:
```

```
D7       255209
dist2    251556
D13      244726
D14      243035
D12      241590
D6       237412
D9       230358
D8       230358
V157     217442
V139     217442
V153     217442
V155     217442
V140     217442
V150     217442
V149     217442
V152     217442
V147     217442
V146     217442
V145     217442
V144     217442
dtype: int64
```

```
print("Loading train_identity.csv...")
train_id = pd.read_csv("train_identity.csv")
```

```python
print("train_identity.csv head:")
display(train_id.head())
print("train_identity.csv info:")
train_id.info()
print("train_identity.csv null counts:")
display(train_id.isnull().sum().sort_values(ascending=False).head(20))
# Display top 20 null counts
```

```
Loading train_identity.csv...
train_identity.csv head:
```

```
{"type":"dataframe"}
```

```
train_identity.csv info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144233 entries, 0 to 144232
Data columns (total 41 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   TransactionID  144233 non-null  int64
 1   id_01          144233 non-null  float64
 2   id_02          140872 non-null  float64
 3   id_03          66324 non-null   float64
 4   id_04          66324 non-null   float64
 5   id_05          136865 non-null  float64
 6   id_06          136865 non-null  float64
 7   id_07          5155 non-null    float64
 8   id_08          5155 non-null    float64
 9   id_09          74926 non-null   float64
 10  id_10          74926 non-null   float64
 11  id_11          140978 non-null  float64
 12  id_12          144233 non-null  object
 13  id_13          127320 non-null  float64
 14  id_14          80044 non-null   float64
 15  id_15          140985 non-null  object
 16  id_16          129340 non-null  object
 17  id_17          139369 non-null  float64
 18  id_18          45113 non-null   float64
 19  id_19          139318 non-null  float64
 20  id_20          139261 non-null  float64
 21  id_21          5159 non-null    float64
 22  id_22          5169 non-null    float64
 23  id_23          5169 non-null    object
 24  id_24          4747 non-null    float64
 25  id_25          5132 non-null    float64
 26  id_26          5163 non-null    float64
 27  id_27          5169 non-null    object
 28  id_28          140978 non-null  object
 29  id_29          140978 non-null  object
 30  id_30          77565 non-null   object
```

```
 31  id_31           140282 non-null  object
 32  id_32           77586 non-null   float64
 33  id_33           73289 non-null   object
 34  id_34           77805 non-null   object
 35  id_35           140985 non-null  object
 36  id_36           140985 non-null  object
 37  id_37           140985 non-null  object
 38  id_38           140985 non-null  object
 39  DeviceType      140810 non-null  object
 40  DeviceInfo      118666 non-null  object
dtypes: float64(23), int64(1), object(17)
memory usage: 45.1+ MB
train_identity.csv null counts:

id_24          139486
id_25          139101
id_07          139078
id_08          139078
id_21          139074
id_26          139070
id_23          139064
id_27          139064
id_22          139064
id_18           99120
id_04           77909
id_03           77909
id_33           70944
id_10           69307
id_09           69307
id_30           66668
id_32           66647
id_34           66428
id_14           64189
DeviceInfo      25567
dtype: int64
```

CELL 3 — VALIDATION RULES & HUMAN INTERACTION LAYER (SIMULATED)

This cell simulates the application of SentinelAI validation rules.

Each rule is triggered, and a human response (override with justification) is simulated.

The focus is on demonstrating the system's interception and governance capabilities.

Rules being simulated:

- **JE-IEEE-001 (Join Error)**: For nulls in identity-related columns.
- **TE-IEEE-001 (Temporal Error)**: For ambiguous `TransactionDT` semantics.
- **AE-IEEE-001 (Aggregation Error)**: For unvalidated units/definitions in `TransactionAmt` aggregations.

```
print("SentinelAI Governance Enabled: Analyzing fraud behavior by device characteristics (DeviceType, DeviceInfo).")
print("Attempting to group by DeviceType and DeviceInfo and count occurrences.")

# Simulate SentinelAI's Join Error validation rule (JE-IEEE-001)
null_identity_records = merged_df[(merged_df['DeviceType'].isnull()) | (merged_df['DeviceInfo'].isnull())]

if not null_identity_records.empty:
    print("\n--- SentinelAI ALERT ---")
    print("Rule ID: JE-IEEE-001 - Join Error Triggered!")
    print(f"Validation Check: Identified {len(null_identity_records)} records where DeviceType or DeviceInfo is null. Blocking analysis.")
    print("Human Involvement: Analyst must clarify semantic meaning of nulls. Options: Accept, Override, Defer.")
    print("--- Simulating Override with Justification (Domain knowledge assumption) ---")
    print("Justification: 'Confirmed nulls mean no identity record, not unknown device, safe to proceed with analysis.'")
```

```
    # Proceed with the analysis after simulated override
    fraud_by_device_gov_A = merged_df.groupby(['DeviceType',
'DeviceInfo'])['isFraud'].count().reset_index()
    display(fraud_by_device_gov_A.head())

    print("\nOutcome: Analysis proceeded after justified override.
SentinelAI logged the override.")
else:
    print("No null identity records found. Proceeding with analysis
without validation trigger.")
    fraud_by_device_gov_A = merged_df.groupby(['DeviceType',
'DeviceInfo'])['isFraud'].count().reset_index()
    display(fraud_by_device_gov_A.head())
    print("\nOutcome: Analysis proceeded, no validation triggered.")
```

```
SentinelAI Governance Enabled: Analyzing fraud behavior by device
characteristics (DeviceType, DeviceInfo).
Attempting to group by DeviceType and DeviceInfo and count
occurrences.

--- SentinelAI ALERT ---
Rule ID: JE-IEEE-001 - Join Error Triggered!
Validation Check: Identified 201525 records where DeviceType or
DeviceInfo is null. Blocking analysis.
Human Involvement: Analyst must clarify semantic meaning of nulls.
Options: Accept, Override, Defer.
--- Simulating Override with Justification (Domain knowledge
assumption) ---
Justification: 'Confirmed nulls mean no identity record, not unknown
device, safe to proceed with analysis.'
```


```
{"summary":"{\n  \"name\": \"    print(\\\"\\\\nOutcome: Analysis
proceeded, no validation triggered\",\n  \"rows\": 5,\n  \"fields\":
[\n    {\n      \"column\": \"DeviceType\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"desktop\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DeviceInfo\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"ATT-
IE11\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"isFraud\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 3,\n        \"min\": 1,\n        \"max\": 9,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```

```
Outcome: Analysis proceeded after justified override. SentinelAI
logged the override.
```

```
print("SentinelAI Governance Enabled: Observing fraud trends over time
using TransactionDT.")
print("Attempting to group by TransactionDT and calculate fraud
rates.")

# Simulate SentinelAI's Temporal Error validation rule (TE-IEEE-001)
print("\n--- SentinelAI ALERT ---")
print("Rule ID: TE-IEEE-001 - Temporal Error Triggered!")
print("Validation Check: Grouping/trending on TransactionDT detected,
assuming calendar semantics. Blocking result.")
print("Human Involvement: Analyst must explicitly state intended
temporal interpretation. Options: Accept, Override, Defer.")
print("--- Simulating Override with Justification (External evidence
reference) ---")
print("Justification: 'Internal documentation confirms TransactionDT
is a seconds offset from 2017-12-01 00:00:00 UTC, allowing relative
trending.'")

# Proceed with the analysis after simulated override
fraud_trend_gov_B = merged_df.groupby('TransactionDT')
['isFraud'].mean().reset_index()
display(fraud_trend_gov_B.head())

print("\nOutcome: Analysis proceeded after justified override.
SentinelAI logged the override.")
```

```
SentinelAI Governance Enabled: Observing fraud trends over time using
TransactionDT.
Attempting to group by TransactionDT and calculate fraud rates.

--- SentinelAI ALERT ---
Rule ID: TE-IEEE-001 - Temporal Error Triggered!
Validation Check: Grouping/trending on TransactionDT detected,
assuming calendar semantics. Blocking result.
Human Involvement: Analyst must explicitly state intended temporal
interpretation. Options: Accept, Override, Defer.
--- Simulating Override with Justification (External evidence
reference) ---
Justification: 'Internal documentation confirms TransactionDT is a
seconds offset from 2017-12-01 00:00:00 UTC, allowing relative
trending.'
```


```
{"summary":"{\n  \"name\": \"print(\\\"\\\\nOutcome: Analysis
proceeded after justified override\",\n  \"rows\": 5,\n  \"fields\":
[\n    {\n      \"column\": \"TransactionDT\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 51,\n
```

\"min\": 86400,\n   \"max\": 86506,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   86401,\n
86506,\n   86469\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"isFraud\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.0,\n   \"min\": 0.0,\n
\"max\": 0.0,\n   \"num_unique_values\": 1,\n   \"samples\":
[\n   0.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
Outcome: Analysis proceeded after justified override. SentinelAI
logged the override.
```

```
print("SentinelAI Governance Enabled: Aggregating TransactionAmt by
ProductCD.")
print("Attempting to calculate average TransactionAmt per ProductCD.")

# Simulate SentinelAI's Aggregation Error validation rule (AE-IEEE-
001)
print("\n--- SentinelAI ALERT ---")
print("Rule ID: AE-IEEE-001 - Aggregation Error Triggered!")
print("Validation Check: Aggregation of TransactionAmt detected
without formal declaration of units. Blocking result.")
print("Human Involvement: Analyst must explicitly state assumed
currency and scaling. Options: Accept, Override, Defer.")
print("--- Simulating Override with Justification (Domain knowledge
assumption) ---")
print("Justification: 'All TransactionAmt values are known to be USD,
with a consistent scaling factor of 1.'")

# Proceed with the analysis after simulated override
avg_amt_by_product_gov_C = merged_df.groupby('ProductCD')
['TransactionAmt'].mean().reset_index()
display(avg_amt_by_product_gov_C.head())

print("\nOutcome: Analysis proceeded after justified override.
SentinelAI logged the override.")
```

```
SentinelAI Governance Enabled: Aggregating TransactionAmt by
ProductCD.
Attempting to calculate average TransactionAmt per ProductCD.

--- SentinelAI ALERT ---
Rule ID: AE-IEEE-001 - Aggregation Error Triggered!
Validation Check: Aggregation of TransactionAmt detected without
formal declaration of units. Blocking result.
Human Involvement: Analyst must explicitly state assumed currency and
scaling. Options: Accept, Override, Defer.
--- Simulating Override with Justification (Domain knowledge
```

```
assumption) ---
Justification: 'All TransactionAmt values are known to be USD, with a
consistent scaling factor of 1.'

{"summary":"{\n  \"name\": \"print(\\\"\\\\nOutcome: Analysis
proceeded after justified override\",\n  \"rows\": 5,\n  \"fields\":
[\n    {\n      \"column\": \"ProductCD\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \"H\",\n          \"W\",\n          \"R\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"TransactionAmt\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
59.353571312731496,\n        \"min\": 44.504183924895514,\n
\"max\": 174.0778567780964,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          71.30724426647251,\n
151.0000238169819,\n          174.0778567780964\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}


Outcome: Analysis proceeded after justified override. SentinelAI
logged the override.
```

CELL 2 — ASSUMPTION DECLARATION (CRITICAL)

Markdown + Python dict.

ASSUMPTIONS = { "TransactionAmt_currency": { "claim": "TransactionAmt is expressed in a single currency", "confidence": "Low", "risk": "Critical" }, "TransactionDT_semantics": { "claim": "TransactionDT represents real calendar time", "confidence": "Low", "risk": "Major" } }

 Manual, explicit, frozen.

```
ASSUMPTIONS = {
  "TransactionAmt_currency": {
      "claim": "TransactionAmt is expressed in a single currency",
      "confidence": "Low",
      "risk": "Critical"
  },
  "TransactionDT_semantics": {
      "claim": "TransactionDT represents real calendar time",
      "confidence": "Low",
      "risk": "Major"
  }
}
```

CELL 3 — VALIDATION RULE FUNCTIONS

Define pure functions (no side effects).

```
def validate_aggregation(intent, assumptions):
    if intent["operation"] == "aggregate" and
assumptions["TransactionAmt_currency"]["confidence"] == "Low":
        return "block"
    return "pass"
```

CELL 4 — GOVERNANCE DECISION ENGINE

def sentinel_decision(intent, validations): if "block" in validations: return "BLOCK" if "warn" in validations: return "DOWNGRADE" return "ALLOW"

This is the system brain.

```
def sentinel_decision(intent, validations):
    if "block" in validations:
        return "BLOCK"
    if "warn" in validations:
        return "DOWNGRADE"
    return "ALLOW"
```

CELL 5 — HUMAN RESPONSE SIMULATION

def human_action(decision): if decision == "BLOCK": return {"action": "defer", "justification": None} return {"action": "accept", "justification": None}

Rule-based is fine. Goal = flow visible.

```
def human_action(decision):
    if decision == "BLOCK":
        return {"action": "defer", "justification": None}
    return {"action": "accept", "justification": None}
```

CELL 6 — TRUST STATE UPDATE

TRUST_STATE = "Calibrated"

def update_trust(trust, human_action): if human_action["action"] == "override": return trust return trust

Simple is OK. Logic > sophistication.

```
TRUST_STATE = "Calibrated"

def update_trust(trust, human_action):
    if human_action["action"] == "override":
        return trust
    return trust
```

CELL 7 — SINGLE END-TO-END RUN (MOST IMPORTANT)

This is the proof cell.

```
intent = {
  "operation": "aggregate",
  "field": "TransactionAmt",
  "group_by": "ProductCD"
}

validations = [
    validate_aggregation(intent, ASSUMPTIONS),
]

decision = sentinel_decision(intent, validations)
human = human_action(decision)
TRUST_STATE = update_trust(TRUST_STATE, human)

print("Decision:", decision)
print("Human action:", human)
print("Trust state:", TRUST_STATE)
```

```
Decision: BLOCK
Human action: {'action': 'defer', 'justification': None}
Trust state: Calibrated
```

CELL 8 — SYSTEM GUARANTEES (Markdown)

SentinelAI Guarantees

- Unvalidated analytical logic is never presented silently.
- Confidence is never inflated beyond evidence quality.
- Final decision authority always remains with the human.

SentinelAI Does NOT Guarantee

- Analytical correctness

- Optimal decisions
- Automated error correction

This cell is non-negotiable.