#**
****

***** *PANDAS NUMPY* *****

#**
****

INTRODUCTION

DECLARING NUMPY ARRAY

arr0 = np.array(24)

arr1 = np.array([1,2,3,4])

arr2 = np.array([[1,1,1],[1,2,1]])

NUMPY ARRAY MANIPULATION

np.add(arr1,arr2)

np.subtract(arr1,arr2)

np.multiply(arr1,arr2)

np.divide(arr1,arr2)

np.power(arr1,arr2)

np.median(arr1)

np.mean(arr1)

np.std(arr1)

np.var(arr1)

np.percentile(arr1,50)

STRING FUNCTIONS IN NUMPY

```
arr1 = np.array(['Hello','World'])

arr2 = np.array(['Welcome', 'Learners'])

str = "Hello How Are You"


np.char.add(arr1,arr2)

np.char.replace(str, 'Hello', 'Hi')

np.char.upper(str)

np.char.lower(str)
```

INDEXING NUMPY ARRAY - 2D, 3D, NEGATIVE INDEXING

MATRIX IN NUMPY

OPERATIONS ON NUMPY ARRAY

RESHAPING NUMPY ARRAY

ARITHMETIC OPERATIONS ON NUMPY ARRAY

LINEAR ALGEBRA IN NUMPY ARRAY

CREATE AN ARRAY (FROM A LIST) - 1D, 2D

CREATE AN ARRAY (FROM A TUPLE)

ACCESSING AN ARRAY INDEX

```
#*****************************************************************************************************
**********************************************
#                                                                          ***** PANDAS SERIES *****
#*****************************************************************************************************
**********************************************
```

DECLARING A SERIES

data = [1, 2, 3, 4, 5]

ser = pd.Series(data)	*# Creating a Pandas Series from a list*

index = ['a', 'b', 'c', 'd', 'e']

ser_with_ind = pd.Series(data, index=index)	*# Creating a Pandas Series with a specified index*

data_dict = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

ser_from_dict = pd.Series(data_dict)	*# Creating a Pandas Series from a dictionary*

BASIC INFO IN SERIES

first_n_rows = ser.head(n)	*# Return the first n rows*
last_n_rows = ser.tail(n)	*# Return the last n rows*
dimensions = ser.shape	*# Return dimensions (Rows, columns)*
stats = ser.describe()	*# Generate descriptive statistics*
unique_values = ser.unique()	*# Return unique values*
num_unique_values = ser.nunique()	*# Return the number of unique values*

OPERATIONS & TRANSFORMATIONS IN SERIES

result_series = ser + ser_with_ind	*# Element-wise addition*
squared_series = ser.apply(lambda x: x**2)	*# Apply a function to each element*
mapped_series = ser.map({1: 'one', 2: 'two', 3: 'three'})	*# Map values using a dictionary*
sorted_series = ser.sort_values()	*# Sort the Series by values*
missing_values = ser.isnull()	*# Check for missing values*
filled_series = ser.fillna(0)	*# Fill missing values with a specified value*

QUERYING A SERIES

data = {'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': 50}

series = pd.Series(data)

selected_greater_than_30 = series[series > 30]	*# Select elements greater than 30*
selected_equal_to_20 = series[series == 20]	*# Select elements equal to 20*
selected_not_equal_to_40 = series[series != 40]	*# Select elements not equal to 40*
selected_multiple_conditions = series[(series > 20) & (series < 50)]	*# Select elements based on multiple conditions*
selected_by_list = series[series.isin([20, 40, 60])]	*# Select elements based on a list of values*
selected_by_index_labels = series.loc[['a', 'c', 'e']]	*# Query based on index labels*

selected_by_numeric_position = series.iloc[1:4]	*# Query based on numeric position*

string_series = pd.Series(['apple', 'banana', 'cherry', 'date', 'elderberry'])

selected_by_string_method = string_series[string_series.str.startswith('b')] # Select elements using string methods (if applicable)

#**********

***** *PANDAS DATAFRAME* *****

#**********

DECLARING A DATAFRAME

data_dict = {'Name': ['Alice', 'Bob', 'Charlie'],'Age': [25, 30, 22],'Salary': [50000, 60000, 45000]}

df_dict = pd.DataFrame(data_dict)

data_list = [['Alice', 25, 50000], ['Bob', 30, 60000], ['Charlie', 22, 45000]]

df_list = pd.DataFrame(data_list, columns=columns)

data_array = np.array([['Alice', 25, 50000],['Bob', 30, 60000],['Charlie', 22, 45000]])

df_array = pd.DataFrame(data_array, columns=columns)

df_csv = pd.read_csv('HousePrices.csv')

df_excel = pd.read_excel('Iris.xlsx')

ACCESSING THE DATAFRAME

column_data = df['Column_Name']	*# Accessing a single column*
selected_columns = df[['Column1', 'Column2']]	*# Accessing multiple columns*
row_data = df.iloc[0]	*# Accessing a specific row by index*
filtered_rows = df[df['Column_Name'] > 10]	*# Accessing rows based on a condition*
value = df.at[0, 'Column_Name']	*# Accessing a single cell by label*
value = df.iat[0, 1]	*# Accessing a single cell by position*
selected_data = df.loc['Rown_Name', 'Column_Name']	*# Accessing data using .loc*
selected_data = df[df['Column_Name'] > 10]['Another_column']	*# Conditional Access*

BASIC INFO IN DATAFRAME

df.head(n)	*# Display the first row*
df.tail(n)	*# Display the last row*
df.info()	*# Provide a comprehensive summary of the DataFrame*
print(df.shape)	*# Return a tuple representing the dimensions of the DataFrame (Rows, columns)*

df.groupby(by, axis=0, sort=True)	
df.apply(func, axis=0)	
pd.merge(left, right, on=None, how='inner')	
df.plot(x=None, y=None, kind='line', ax=None)	
df.drop(index_label, axis=0, inplace=False)	

STATISTICAL OPERATIONS IN PANDAS

data = {'Numeric_column1': [5, 15, 8], 'Numeric_column2': [10, 20, 30], 'Numeric_column3': [100, 200, 300]}

df = pd.DataFrame(data)

print(df.describe())	*# Display descriptive statistics for numeric columns*
mean_value = df.mean()	*# Calculate mean, median, and standard deviation*
median_value = df.median()	*# Calculate mean, median, and standard deviation*
std_deviation = df.std()	*# Calculate mean, median, and standard deviation*
correlation_matrix = df.corr()	*# Compute correlation matrix*

data = {'Category': ['A', 'B', 'A', 'C', 'B', 'A', 'C', 'A', 'B', 'C']}

df = pd.DataFrame(data)

value_counts = df['Category'].value_counts()	*# Count occurrences of unique values in the category column*

STRING FUNCTIONS IN PANDAS

df = pd.DataFrame({'Column': ['Hello', 'World', 'Python', 'Data Science']})

substring = 'Data'

df['Length'] = df['Column'].str.len()	*# Calculates the length of each string*
df['Lowercase'] = df['Column'].str.lower()	*# Converts text to lowercase*
df['ContainsSubstring'] = df['Column'].str.contains(substring)	*# Checks if each string contains the specified substring*

SORTING A DATAFRAME

df.sort_values(by, axis=0, ascending=True, inplace=False)

df_sorted = df.sort_values(by='Age')	*# Sort DataFrame by the 'Age' column in ascending order*
df_sorted_multi = df.sort_values(by=['Age', 'Salary'], ascending=[True, False])	*# Sort DataFrame by 'Age' in ascending order, then by 'Salary' in descending order*
df_sorted_index = df.sort_index(ascending=False)	*# Sort DataFrame by index in descending order*

series_sorted = series.sort_values(ascending=False)	*# Sort Series in descending order*

PDD	Policy Detail Database	
QAS	Quote & Application System	
IRPM	Individual Risk Premium Modification	
CIM	Customer Information Management	
MDM	Master Data Management	
STG Billing	Strategic Technology Group	Majesco Billing System is referred as STG Billing
EDH	Erie Data Hub	
FOOD	File Of Our Dreams	
RS3	Reporting System 3	
DNB	Dun & Bradstreet	
DUNS	Data Universal Numbering System	
ECC	Erie Claims Center	
OOB	Out-of-Balance	

OOS	Out-of-Sequence	
OOSE	Out-of-Sequence Endorsement	OOSE is a policy change or endorsement that is dated earlier than a currently active transaction. It occurs out of chronological order, requiring the system to revert and reapply subsequent transactions to maintain consistency.
TRF	Term Roll Forward	TRF is a process that extends a policy term into a new period, typically at renewal. It involves carrying forward balances, coverages, and endorsements from the current term to the next.
TRX	Transaction	TRX refers to any recorded activity or event that affects a Policy, Billing, or Claims System.
BPP		
OSPAS		
TESL	Transaction Engine Service Layer	TESL is a Service-Oriented Architecture layer that exposes One Shield's core function as Web Service (SOAP/REST). It allows external systems (portals/mobile apps/third-party tools) to create, read, update, delete, and search insurance data. Key capabilities of TESL are - • Headless Processing: Decouples business logic from the UI, allowing backend services to be triggered independently. • API-Driven Architecture: Offers a rich set of prebuilt APIs for Policy, Billing, Claims, and Customer Data. • Metadata Services: Exposes product definitions (Coverages, Rules, Forms, etc.) to external systems.

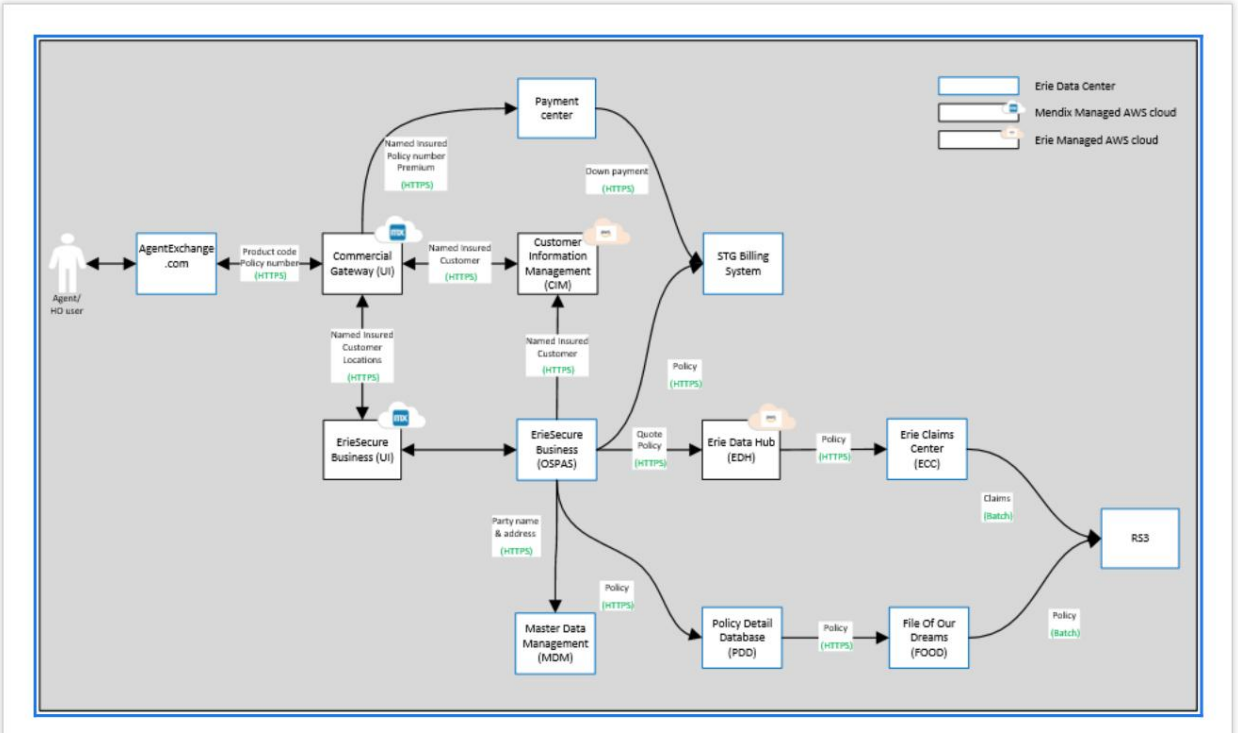
		• Portal Enablement: Powers self-service portals for Agents, Customers, and Partners. • Rapid Integration: Simplified and accelerates integration with CRM's, ERP's, and other insurance.
SOA	Service-Oriented Architecture	
FCRA	Fair Credit Reporting Act	
CLUE	Comprehensive Loss Underwriting Exchange	It shows the prior Auto and Property Claims history.
MVR	Motor Vehicle Records	

Calendar Effective Date vs Policy Effective Date

LexisNexis Service ?

ISO Bureau Content

ACORD Messaging System and London Exchange for RI?



LexisNexis Service	LexisNexis is used for risk assessment & underwriting, fraud detection, regulatory compliance, claims management, legal & court records access, and acts as data sharing among insurers. • Risk Assessment & Underwriting - LN can pull a detailed data on applicants about the prior claims history, driving records, property ownership, and credit-based insurance scores. It helps insurers assess risk profiles and determine policy eligibility, premium rates, coverage limits. • Fraud Detection - LN	The user enters application details (Name, Address, DOB) and the system calls the LexisNexis API. The API returns - • CLUE Auto & Property Reports • Prior Claims (Date, Types, and Amounts) • Carrier History The returned data will display on the UW dashboard.

/**********
**********/

SNOW

/**********
**********/

REQ0674446 - Full Access for SNOW.

Service Now - Production

Don't have full access.

PROD: https://erieprod.service-now.com/

PROD (Classic): https://erieprod.service-now.com/now/nav/ui/classic/params/target/%24pa_dashboard.do

TEST: https://erietest.service-now.com/

/**********
**********/

AGENCY PORTAL

/**********
**********/

TEST: https://testportal.erieinsurance.com/

/**
**/

URL & LINK

/**
**/

BA and DEV Assignment:

---

http://erieshare/sites/commlineprodsupport/_layouts/15/WopiFrame2.aspx?sourcedoc=%7B4CA666C7-2525-4B2E-ABD6-F2F83B289FE8%7D&file=BA%20and%20Dev%20assignment.xlsx&action=default

Commercial Lines - Production Support Team:

---

http://erieshare/sites/commlineprodsupport/_layouts/15/WopiFrame2.aspx?sourcedoc=%7BEB119C51-F833-4B1B-B77D-8AE46FF626AA%7D&file=Commercial%20Lines%20-%20Production%20Support%20Team.xlsx&action=default

- CUSTOMER
- PRODUCTS
- LOCATIONS - Building
- RATING DETAILS -
    - Coverage Type
        - Property
        - Liability
    - Coverage Details
        - Building
        - Business Personal Property

        - Income Protection
    - Insured's Operations (COP - Class of Operation, % Occupied, Annual Sales)
    - Building Details (Construction Type, Building Age, Alarms, Sprinklers)
- REPORTS
    - Calculate Building Valuation
    - Property Valuation Summary (CoreLogic - Commercial Valuation)
- COVERAGES
    - IRPM details
    - Policy Details - Liability Exclusions, Liability Limits, Property Deductions, Damages to the Property
    - Additional Coverages
    - Policy Forms

<< It shows the breakdown of the premium, limits, and rate at a coverage level>>

- INTERESTS - It contains the details about the other interested party.
- UNDERWRITING
- BILLING

***** QUESTIONS *****

Split Mod Policies || Split Experience Mod -

A split mod policy simply refers to how an insurance carrier calculates your workers' compensation EMR.

This is a mid-term policy.

Part of the policy is rated on one mod & part of the policy is rated on another mod.

State having Surcharge ?

Local government premium tax

What is a Surcharge ?

What is a Surcharge Policy ?

COP - Class of Operation

***** BUSINESS QUESTIONS IN AN MPP MEETING *****

- How many policies are affected?
- Is this issue for a single state policy, or multi-state policy (all state policy)?
- What is the fix going to be ?
- Is this a Calendar Date Fix (Retro Active Fix) or Policy Effective Fix ?
    - Is it effecting only the current policy period ?
    - Is this for a Single Term or Multiple Term ?
- Does this information go to downstream, and go to Bureau Reporting (EDI - Electronic Report) ?
- What is the Premium Impact - Overcharge or Undercharge ?
- Any Forms Impact ?
- Any Coverage Impact ?

***** TAEK AWAY ******

- BA 1.0 does not rate based on License Number - unlike BA2.0

***** PROBLEM MANAGEMENT STEPS *****

1. Analyze the Incident
    a. Replicate the issue in the production environment.
    b. Check the Splunk Logs for exact error message.
    c. Understand the level of impact - is it one entity or many entities => create PTASK for scan.
    d. Reproduce the issue in the lower environment.
    e. Tag a Problem to the incident and change the status to Awaiting Problem
2. Create a New Problem (if there is no older problem that exists).
3. Walkthrough the problem in the Problem Meeting or Analyst Session.
4. Get the problem schedule on triage meeting (occurs every Wednesday).
5. Handover the problem to Release BA (transition between Problem BA >> Release BA)
    a. Release Planning meetings occurs between limited audience (typically, Abby, Joshua)
6. The problem gets estimated, with the help of DEV/QA (T-Shirt Sizing)
7. The problem is scheduled on the release - start the Feature processing.
8. The problem is developed and tested (DEV + QA)
9. The problem will be promoted into production.
10. BA would validate the problem in the production.

***** BUSINESS QUESTIONS IN AN MPP MEETING *****

- How many policies are impacted?
- Is this issue for a single state policy, or multi-state policy (all state policy)?
- What is the fix going to be ?
- Is this a Calendar Date Fix (Retro Active Fix) or Policy Effective Fix ?
    - Is it effecting only the current policy period ?
    - Is this for a Single Term or Multiple Term ?
- Does this information go to downstream, and go to Bureau Reporting (EDI - Electronic Report) ?
- What is the Premium Impact - Overcharge or Undercharge ?
- Any Forms Impact ?
- Any Coverage Impact ?

***** TAEK AWAY ******

- BA 1.0 does not rate based on License Number - unlike BA2.0

You have provided a detailed outline of common challenges faced by a Product Owner (PO) or agile team, along with potential causes and solutions. This is a solid framework for addressing these scenarios.

Here is an organized and refined version of your points, which can serve as a comprehensive guide for handling these challenges:

1. Frequently Changing Priorities

The Challenge: Constant shifts in priorities make it difficult for the team to maintain focus, leading to inefficiency and potential burnout.

Strategies to Overcome:

Maintain Clear Vision (The "North Star"): A well-defined, consistent product vision guides the team when day-to-day priorities shift. This vision should be the ultimate reference point for all decision-making.

Implement Shorter Planning Cycles:

Reduce Sprint Length: Planning for 1-week sprints (or even daily/weekly goals instead of long-term roadmaps) increases adaptability to changing needs and provides more frequent opportunities to reprioritize.

Focus on Outcomes Over Outputs: Instead of tracking task completion, measure the delivery of business value. This ensures efforts are aligned with high-level goals regardless of tactical changes.

Build in Reserve Capacity: Allocate a percentage of the team's capacity (e.g., 10-20%) each sprint for ad-hoc requests, spikes, or sudden high-priority items.

Foster Tight Communication: Establish clear, rapid communication channels between the PO, stakeholders, and the development team to ensure everyone is aligned on the why behind a priority change.

Example (Spotify Agile Team): Spotify's model often treats "squads" like mini-startups, giving them autonomy within a clear mission (the "North Star") to pivot and adapt as needed to deliver value like the "Discover Weekly" feature.

2. Balancing New Features vs. Technical Debt

The Challenge: New features drive customer satisfaction and market competitiveness, while tech debt ensures long-term stability and avoids future system failures. The balance is crucial.

Strategies for a Balanced Approach:

Treat Tech Debt as a "First-Class Citizen":

Allocate Dedicated Time: Dedicate a consistent portion of every sprint (e.g., 15-20%) to addressing technical debt. Make it a non-negotiable part of planning.

Prioritize Based on Business Value (BV) and Risk Mitigation: Evaluate tech debt items just like new features.

Business Value: Does fixing this debt unlock a new market or dramatically increase developer velocity?

Risk: Does ignoring this debt pose a high risk of system failure, security breach, or increased long-term cost?

The Art of Trade-Offs:

Understand how neglecting technical debt impacts the team's velocity (e.g., a messy codebase slows down feature delivery). Be transparent about this impact with stakeholders.

Use metrics to show how velocity increases after a debt-reduction sprint.

Example (Facebook Mobile App): Early versions of the Facebook mobile app were built on an HTML codebase (a form of tech debt that provided speed of delivery but was slow). They eventually made the significant trade-off to rewrite it natively to ensure long-term performance and stability.

3. Handling a New Feature with a Low Adoption Rate

The Challenge: A feature is built and released, but users aren't engaging with it as expected.

Strategies for Identification and Action:

Phase 1: Data-Driven Analysis (Identify the "WHERE")

User Behavior Analytics: Use tools to track how users interact with the feature.

Funnel Analysis: Identify the exact bottlenecks where users drop off (e.g., they open the feature but don't complete the necessary steps).

Segmentation: Analyze adoption rates by user type (new vs. old customers, different demographics) to see if the issue is universal or specific.

Awareness vs. Engagement vs. Retention: Is the user aware of the feature? Are they using it frequently? Are they coming back to use it later?

Phase 2: Qualitative Feedback (Identify the "WHY")

Surveys & Interviews: Directly ask users why they aren't using the feature.

Customer Support Tickets: Look for patterns in complaints or confusion related to the feature.

Formulate Hypotheses: Based on data and feedback, hypothesize the core reasons (e.g., discoverability issues, too complex, misalignment with user needs, poor onboarding).

Phase 3: Strategy & Execution (Plan to Handle It)

Improve Discoverability/Onboarding: Use email campaigns, push notifications, in-product promotions, or an improved demo/tutorial experience.

Simplify the Feature: Use A/B testing to test simplified versions or remove confusing elements.

Leverage Power Users: Give early access to highly engaged users and gather feedback before a wider rollout.

Monitor and Adjust: This is an iterative process. Implement a change, measure the impact, and repeat.

Example (Slack Threads): Threads in Slack were a powerful feature but initially had low adoption. Slack used a combination of onboarding tweaks and communication nudges to increase awareness and usage.

4. Striking a Balance Between Innovation and Core Features

The Challenge: Ensuring the core product remains robust and valuable while simultaneously exploring new, innovative ideas that could be future growth drivers.

Strategies for Balance:

User-Centric Approach:

Understand E2E Customer Needs: Engage in continuous feedback loops to thoroughly understand user pain points.

Prioritize Features that Bring the Most Value: Both innovation and core improvements should solve a genuine user problem.

Impact vs. Effort & Risk Analysis: Use these frameworks to evaluate potential ideas:

Core Improvements: Often high impact, moderate effort, low risk (predictable value).

Innovation: Potentially very high impact, high effort, high risk (unpredictable value).

Balancing Short-Term Wins with Long-Term Missions:

The 70/20/10 Rule (Commonly used by Google): A structured way to allocate resources:

70% on core business/sustaining current success.

20% on related, innovative projects.

10% on completely new, risky ideas (pure innovation).

This ensures continuous delivery of core value while hedging bets on future growth opportunities.

https://www.geeksforgeeks.org/numpy/numpy-ndarray/

https://www.geeksforgeeks.org/python/python-numpy-practice-exercises-questions-and-solutions/

Why use Linear Algebra

Scalar/s & Vector/s

Vector Addition

Vector Multiplication - Dot Product & Cross Product

Magnitude or Length of a Vector

Matrix Operations - Addition, Subtraction, Multiplication (Scalar & Regular), Transpose

Determinant of Matrix (2X2, 3X3)

Identity Matrix

Minor of an Element

Matrix of Minors

Cofactor of an Element

Cofactor Matrix

Adjoint Matrix - transpose of a Cofactor Matrix

numpy.add(matrix1, matrix2)

numpy.subtract(matrix1, matrix2)

numpy.multiply(matrix1, matrix2)

numpy.divide(matrix1, matrix2)

numpy.dot(matrix1, matrix2)

numpy.transpose(matrix)

numpy.linalg.inv(matrix)

numpy.linalg.det(matrix)

numpy.linalg.eig(matrix)

numpy.linalg.matrix_rank(matrix)

CLASS DESCRIPTION

- Class Code
- Class Code Description
- Notes

CLASS EXPOSURE BASIS / RATING GROUP

- Class Code
- Auxiliary Rated
- Exposure Basis :- Insured Liability & Lessor Liability
- Rating Group :- Data Breach, Dependent Properties, and Liquor Liability

PROPERTY CLASS FACTOR

- Class Code
- Building :- Group I & Group II
- Business Personal Property :- Group I & Group II
- Income Protection :- Dollar Limit & Actual Loss Sustained
- Equiptment Breakdown :- Insured & Lessor

LIABILITY CLASS FACTOR

- Class Code
- Insured :- Premises, Products, DTCA
- Lessor :- Premises, Products, DTCA
- EPLI

MINIMUM PREMIUMS

- Class Code
- Property
- Liability
- Property and Liability

3010 BUFFALO RD, ERIE, PA 16510-1708

http://erieshare/sites/commlineprodsupport/_layouts/15/WopiFrame.aspx?sourcedoc={EFDDF47A-4A24-42EC-8851-9B6DD4F74E4D}&file=QAS%20BSD%20-

%20Coverages%20and%20Rating%20ESB%20v2.docx&action=default&DefaultItemOpen=1

ErieSecure Business(002986366)

10/7/2025-10/7/2026

Agent - GG7731

```
--- NUMPY ---

INTRODUCTION
        import numpy as np
        arr = np.array ([10,20,30,40,50])

DECLARING NUMPY ARRAY
        array_0d = np.array(24) # Create a 0D Array
        array_1d = np.array([1, 2, 3, 4, 5, 6]) # Create a 1D Array
        array_2d = np.array([[1, 2, 3], [4, 5, 6]]) # Create a 2D Array
        array_3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]]) # Create a 3D Array

NUMPY INDEXING
        print(array_1d[3]) # Print the value at index 3 of the 1D NumPy array
        print(array_2d[0, 2]) # Printing the third element in the first row of the 2D array
        print(array_3d[1, 0, 0]) # Print the first element of the first row of the second 2D
array within the 3D array

        print(array_1d[-3]) # Printing the fourth element from the end of the 1D array using
negative indexing
        print(array_2d[1, -1]) # Printing the last element in the second row of the 2D array
using negative indexing
        print(array_3d[1, 1, -1]) # Printing the last element in the last row of the last 2D
array within the 3D array using negative indexing

NUMPY ARRAY MANIPULATION
        print (type(arr1))
        result = np.add(arr1, arr2) # Perform element-wise addition of two arrays using the
'np.add' method
        result = np.subtract(arr1, arr2) # Perform element-wise subtraction of two arrays
using 'np.subtract' method
```

result = np.multiply(arr1, arr2) # Perform element-wise multiplication of two arrays using the 'np.multiply' method
result = np.divide(arr1, arr2) # Perform element-wise division of two arrays using the 'np.divide' method
result = np.power(arr1, arr2) # Perform element-wise power operation where each element in 'a' is raised to the corresponding element in 'b' using the 'np.power' method
median_result = np.median(arr1) # Median
mean_result = np.mean(arr1) # Mean
std_result = np.std(arr1) # Standard Deviation
var_result = np.var(arr1) # Variance
percentile_result = np.percentile(arr1, 50) # Find the value under the 50th percentile of the NumPy array.

STRING FUNCTIONS IN NUMPY
arr1 = np.array(['Hello','World'])
arr2 = np.array(['Welcome', 'Learners'])
str = "Hello How Are You"

concat_result = np.char.add(x,y) # Perform element-wise string concatenation for two arrays of string
resplace_result = np.char.replace (str, 'Hello', 'Hi') # Replacing the old substring with the new substring
upper_result = np.char.upper(str) # Converting all lowercase characters in a string to uppercase and vice-versa
lower_result = np.char.lower(str) # Converting all lowercase characters in a string to uppercase and vice-versa

INDEXING NUMPY ARRAY - 2D, 3D, NEGATIVE INDEXING

MATRIX IN NUMPY
OPERATIONS ON NUMPY ARRAY
RESHAPING NUMPY ARRAY
ARITHMETIC OPERATIONS ON NUMPY ARRAY
LINEAR ALGEBRA IN NUMPY ARRAY
CREATE AN ARRAY (FROM A LIST) - 1D, 2D
CREATE AN ARRAY (FROM A TUPLE)
ACCESSING AN ARRAY INDEX

==========

--PANDAS --

DECLARING A SERIES

```
data = [1, 2, 3, 4, 5]
ser = pd.Series(data) # Creating a Pandas Series from a list

index = ['a', 'b', 'c', 'd', 'e']
ser_with_ind = pd.Series(data, index=index) # Creating a Pandas Series with a specified index

data_dict = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
ser_from_dict = pd.Series(data_dict) # Creating a Pandas Series from a dictionary
```

BASIC INFO IN SERIES

```
first_n_rows = ser.head(n) # Return the first n rows
last_n_rows = ser.tail(n) # Return the last n rows
dimensions = ser.shape # Return dimensions (Rows, columns)
stats = ser.describe() # Generate descriptive statistics
unique_values = ser.unique() # Return unique values
num_unique_values = ser.nunique() # Return the number of unique values
```

OPERATIONS & TRANSFORMATIONS IN SERIES

```
result_series = ser + ser_with_ind # Element-wise addition
squared_series = ser.apply(lambda x: x**2) # Apply a function to each element
mapped_series = ser.map({1: 'one', 2: 'two', 3: 'three'}) # Map values using a dictionary
sorted_series = ser.sort_values() # Sort the Series by values
missing_values = ser.isnull() # Check for missing values
filled_series = ser.fillna(0) # Fill missing values with a specified value
```

QUERYING A SERIES

```
data = {'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': 50}
series = pd.Series(data)

selected_greater_than_30 = series[series > 30] # Select elements greater than 30
selected_equal_to_20 = series[series == 20] # Select elements equal to 20
selected_not_equal_to_40 = series[series != 40] # Select elements not equal to 40
selected_multiple_conditions = series[(series > 20) & (series < 50)] # Select elements based on multiple conditions
selected_by_list = series[series.isin([20, 40, 60])] # Select elements based on a list of values
selected_by_index_labels = series.loc[['a', 'c', 'e']] # Query based on index labels
selected_by_numeric_position = series.iloc[1:4] # Query based on numeric position
```

```
        string_series = pd.Series(['apple', 'banana', 'cherry', 'date', 'elderberry'])
        selected_by_string_method = string_series[string_series.str.startswith('b')] #
Select elements using string methods (if applicable)

--------------------------------------------------------------------------

DECLARING A DATAFRAME
        data_dict = {'Name': ['Alice', 'Bob', 'Charlie'],'Age': [25, 30, 22],'Salary': [50000,
60000, 45000]}
        df_dict = pd.DataFrame(data_dict)

        data_list = [['Alice', 25, 50000], ['Bob', 30, 60000], ['Charlie', 22, 45000]]
        df_list = pd.DataFrame(data_list, columns=columns)

        data_array = np.array([['Alice', 25, 50000],['Bob', 30, 60000],['Charlie', 22, 45000]])
        df_array = pd.DataFrame(data_array, columns=columns)

        df_csv = pd.read_csv('HousePrices.csv')

        df_excel = pd.read_excel('Iris.xlsx')

ACCESSING THE DATAFRAME
        column_data = df['Column_Name'] # Accessing a single column
        selected_columns = df[['Column1', 'Column2']] # Accessing multiple columns
        row_data = df.iloc[0] # Accessing a specific row by index
        filtered_rows = df[df['Column_Name'] > 10] # Accessing rows based on a
condition
        value = df.at[0, 'Column_Name'] # Accessing a single cell by label
        value = df.iat[0, 1] # Accessing a single cell by position
        selected_data = df.loc['Rown_Name', 'Column_Name'] # Accessing data using
.loc
        selected_data = df[df['Column_Name'] > 10]['Another_column'] # Conditional
Access

BASIC INFO IN DATAFRAME
        df.head(n) # Display the first row
        df.tail(n) # Display the last row
        df.info() # Provide a comprehensive summary of the DataFrame
        print(df.shape) # Return a tuple representing the dimensions of the DataFrame
(Rows, columns)

        df.groupby(by, axis=0, sort=True)
        df.apply(func, axis=0)
        pd.merge(left, right, on=None, how='inner')
```

```
        df.plot(x=None, y=None, kind='line', ax=None)
        df.drop(index_label, axis=0, inplace=False)

STATISTICAL OPERATIONS IN PANDAS
        data = {'Numeric_column1': [5, 15, 8], 'Numeric_column2': [10, 20, 30],
'Numeric_column3': [100, 200, 300]}
        df = pd.DataFrame(data)

        print(df.describe()) # Display descriptive statistics for numeric columns
        mean_value = df.mean()
        median_value = df.median()
        std_deviation = df.std()
        correlation_matrix = df.corr()

        data = {'Category': ['A', 'B', 'A', 'C', 'B', 'A', 'C', 'A', 'B', 'C']}
        df = pd.DataFrame(data)

        value_counts = df['Category'].value_counts() # Count occurrences of unique
values in the category column

STRING FUNCTIONS IN PANDAS
        df = pd.DataFrame({'Column': ['Hello', 'World', 'Python', 'Data Science']})
        substring = 'Data'

        df['Length'] = df['Column'].str.len()
        df['Lowercase'] = df['Column'].str.lower()
        df['ContainsSubstring'] = df['Column'].str.contains(substring)

SORTING A DATAFRAME
        df.sort_values(by, axis=0, ascending=True, inplace=False)

        df_sorted = df.sort_values(by='Age') # Sort DataFrame by the 'Age' column in
ascending order
        df_sorted_multi = df.sort_values(by=['Age', 'Salary'], ascending=[True, False]) #
Sort DataFrame by 'Age' in ascending order, then by 'Salary' in descending order
        df_sorted_index = df.sort_index(ascending=False) # Sort DataFrame by index in
descending order
        series_sorted = series.sort_values(ascending=False) # Sort Series in descending
order
```

Split Mod Policies -

- A split mod policy simply refers to how an insurance carrier calculates your workers' compensation Experience Modification Rate (EMR).

- It's not a different type of insurance policy itself, but a methodology used to determine your premium based on the frequency and severity of your past claims.

- In this system, each workers' comp claim is divided into two parts: a "primary loss" and an "excess loss".

How the split mod system works -

* Primary Loss: This is the amount of a claim that falls at or below a set dollar threshold, called the "split point". These losses are factored more heavily into the EMR calculation. This is because they are considered a better indicator of how frequently injuries occur at a company.

* Excess Loss: Any amount of a claim that exceeds the split point is considered an excess loss. These larger, less frequent losses are given less weight in the calculation because they are seen as less predictive of future claims severity.

The purpose of splitting losses -

* By weighting the costs differently, the split mod system encourages employers to focus on preventing frequent, small injuries.

* Frequent, small claims: Ten small claims that each fall under the split point will have a greater negative impact on a company's EMR than one single, large claim of the same total value. This incentivizes employers to reduce the frequency of accidents in the workplace.

* Single, large claims: A single, expensive claim is less impactful on the EMR once it crosses the split point threshold. The rest of the claim's value is counted as an excess loss and is discounted in the calculation.

The Business Auto Policy provides liability, collision and comprehensive coverage for retail, wholesale delivery, contractor or service industry vehicles.

The Commercial Auto Policy provides liability, collision and comprehensive coverage for retail, wholesale delivery, contractor or service industry vehicles. Garage Auto policies cover the legal liability of dealerships, repair garages and body shops for losses arising from their operations. This policy can also cover damage to Customers' and garage-owned autos.

Workers Compensation covers employees injured in the workplace or in the course of their employment. It pays for medical bills and lost wages as required by state law. The policy also provides employers' liability insurance for legal liability imposed on an employer to pay damages to an employee injured by the employer's negligence.

Business Umbrella provides additional liability coverage, serving as a financial cushion against a judgment beyond the limits of primary liability policies. It adds an additional $1 million or more above the protection provided by the underlying general liability, automobile liability, employers' liability and/or professional or errors and omissions liability coverages.

State(s): DC, IL, IN, KY, MD, NC, NY, OH, PA, TN, VA, WI, WV

ErieSecure Business (ESB)A package policy providing Property and Liability protection. It is referred to as a cafeteria policy because it allows for the selection of coverage to meet the insured's needs. Certain classifications require prior approval.

Garage - This contract is designed for service/repair shops; franchised and non-franchised dealerships. Dealerships or operations with spray painting should be discussed with Underwriting prior to binding.

Workers' Compensation (WC)

This policy provides Workers' Compensation and Employers' Liability protection. It must be written with supporting liability (ErieSecure Business) coverage to qualify for our program.

Business Catastrophe (BCL)

This is ERIE's Umbrella policy providing additional limits above what is written on the underlying contracts. ERIE must write all of the underlying policies (excluding WC) to qualify for our program.

---

1. Customer/Entity Details - Company Name, FEIN, Address, Contact Details, Audit Contact Details
2. Product Selection
3. Locations
4. Rating Details -
    - State of Operation - Bureau File Number, Experience Mod, Rating Effective Date
    - Employee Class of Operation & Payroll
    - Owner & Officer Details
5. Reports
6. Coverages
    - Employee Liability Limits
    - Additional Coverages
7. Interests
    - Add Other Interest
8. Underwriting
    - Underwriting Questionaire
    - Underwriting Referrals
9. Billing

Payment Method & Payment Plan

Authorized Representative

OSPAS - One Shield Policy Admin System

STG - Billing System

PDD - Policy Detail Database

---

1. Frequently changing priorities -

Cons -

Difficult for the team to maintain focus.

How to overcome -

1. Maintain shorter sprint cycles - don't plan for long targets, instead aim for short goals.

Ideally, plan for 1 week or 1 day sprint.

2. Have some reserve capacity.

3. Maintain clear Vision (North Star) - guides the team when priorities shift.

4. Tight communication between the team during frequent changes keeps the team together.

5. Focus on outcomes and not outputs. Don't focus on completing task, instead focus on delivering business value.

Example: Spotify Agile Team

Agile Squad => Mini Start-Ups

Discover Weekly

2. As a PO how do you balance the New Features vs Technical Debt.

New Features => It brings new business (Increases Market Competitivenss and Custoemr Satisfication)

Tech Debt => Long term product stability (Potential System Failures, Increase in Long Term Cost) - 1st Class Citizen

a. Understand the BV (Business Value)

b. Risk Mitigation

c. Art of Trade-Off (understadning how handlign technical debt impact the velocity)

Example:

Facebook Mobile App - HTML code base.

3. A new feature is having low adoption rate. Identify the reason and device a plan to handle it.

Data Driven Analysis - data speaks a lot.

User Behaviour Anlyatics - analyze how the user is using the features.

Where are the user gettign dropped out.

Time Spent on Feature vs Rest of the Product.

User Adoption Rate - at Segment Level (New Customer vs Old Customer, Male vs Female, Age (<25, 25 - 50, >50)

Funnel Analysis - identify the bottlenecks where the users are dropping

Awareness vs Engagement (is the feature getting used frequently) vs Retention (is the feature getting used frequently)

Survey, Interview, Customer Tickets.

Hypothesis (WHERE, WHY, WHAT REASONS) - discoverability issues (user not aware of the issues/benefits), complexity, onboarding and Education (not knowing how to use the feature), misalignment with user needs.

Strategy to increase the Engagement - Email Campaigns, Push Notifications, Product Promtions, improve Onboarding Experiecne (Demo), Simplify the Features, leverage power users (give early a ccess and make them use), A/B Testing, promotions to use the feature.

Monitor and Andjustment.

Example - Slack Threads

4. Striking balance between Innovation vs Core Feature

Understand the E2E about the Customer need (User Centric Approach) - User Pain Points, features that bring most value to the User.

Engage with Customer via Feedback Loops

Impact vs Effort

Risk Analysis - does it introduce new issues

Balancing Short Term Wins with Long term missions.

What SM does ?

perform Capacity Planning

Represent the team as facilitaor in PI Planning

Conduct Breakout sessions

Align with Sprint Goals to PI Objectives - to be in sync with ART

How to prepare and manage a PI Planning ?

Pre-PI Planning Phase -

M - Ensure Miro/Mural Board is updated for collobaration during PI Planning

E - Ensure the epics are aligned (speak and align with

S - All sprints are created in the Mural board and in JIRA.

C - Capacity Planning (Tentative)

O - Older Epics (What is Completed vs What is Not Completed - Is that required for Prioritization)

PI Planning Phase

List of Business Functionalities - what would be inculcated in the Quarter

Infrac Structure Stories >> discussed in ART - can be broken/decompossed into smaller stories + guestimate of SP

Non-Functional Stories >> discussed in ART - can be broken/decompossed into smaller stories + guestimate of SP

Breakout Sessions

Review from Management Team on the Breakout Sessions

Identify the Risks (ROAM Technique) - Resolved, Owned, Accepted, Mitigated

Resolved - Already taken care and no action required in the future.

Owned -

Accepted - Accpeted but no action being taken as the probabability of occurance is low.

Identify the Dependencies - call out on Mural/Miro board (mark as RAG)

When the Lagnadipathi is in Bala/Strength or when the 9th bhavadiapthi is in Bala/Strength THEN -

- the person will seek Independence.

- the person will seek a Nuclear Family. If in a Joint Family, the person will head the Family.

- the person will not work under others.

When there is 1-12 connection in the Horoscope, and is in Bala/Strength THEN it shows Parvata Yogam.

- the person will desire towards a Join Family.

- the person will desire towards a Sanga/Society and he/she will Head a Sanga.

- the person will work in Big Organization.

In a Su-Ju combination, and one of them attaining Bala/Strength => it shows a Joint Family

Su-Ju combination:

Su-Ju together,

Su in right after Ju,

Ju is right after Su,

Su aspecting Ju,

Ju aspecting Su,

Su in the house of Ju,

Ju in the house of Su

Teh day lord has a very-very vital role in givign an accurate and stampign prediction.

Make use of the day-lord for any kind of prediction - it has the real anwer.

Whosoever has a good time, he/she will come on a favourable day related to the bhava.

The matters corresponding to the bhava from the day lord will be good.

Day Lord is the support inorder to go about and predict.

The Dasha/Bhukti should not be ignored, by considering the Day Lord.

Don't only depend on the day lord to make the prediction. It should be looked along with the Dasha/Bhukti.

Example 1: If someone is planning to give Money to another person THEN check if the Panaphara Sthana from the day lord in the Horoscope should be good.

If the question was raised on Friday, check the 2/5/8/11 from Ve in the person's Horoscope.

Horoscope > Dasha/Bhukti > Gochara > Day Lord (this is the sequence to go)

Example 2: During Marriage matching, when both the 2nd & 7th bhavadipathi from the day lord is in Bala/Strength in both the Boy & Girls Horoscope THEN conclude both the Horoscopes match and recommend both the parties to proceed.

If in the Boy's Horoscope the 2nd and 7th bhavadipathi from the day lord is in Bala/Strength. But, from the Girl's Horoscope either the 2nd or 7th bhavadipathi from the day lord is not in Bala/Strength THEN the Boy's side are insteresed in the aliance, but the Girl's side are not interested

Example 3: If someone is planning to construct a House THEN check the 4th from the day lord in his/her Horoscope.

Example 4: If someone is planning related to a matter pertaining to the 3rd bhava and appraoched the Astrologer.

In the person's horoscope if the 3rd from the Day Lord is Strong, but the 3rd from Dashadipathi and Bhuktiadipathi is not Strong THEN ask the person to come back on the day which is represented by the Dashadipathi/Bhuktiadipathi

if he/she is looking for a Registeration in his mind.

When the Day Lord is saying something completely different from the Dasha/Bhukti THEN whatever suggestion is given to the native - he/she won't agree to the same.

So, recommend the person to come on a day that is shown by the Bhukti. On that given day, the native's mind will be balanced adn he/she will come to a conlcusion that the Astrologer is saying for his/her own benefit.

Example 5: In case of failed marriages, the